Universidad Carlos III de Madrid
Máster de Ciencias Actuariales y Financieras

Comparación de Tarificación de Seguro de Auto por GLM y Redes Neurales

Autor

Ying Li

Tutor/es

José Miguel Rodríguez Pardo del Castillo

Jesús Ramón Simón del Potro

Índice de contenido

1. Motivación 7

2. La naturaleza del Seguro 9

3. Resumen de los métodos de tarificación de seguros 10

3.1. GLM 10

3.1.1 The exponencial family distribution 12

3.1.2 GLM y función de vinculación 13

3.2. Desde lineal a No lineal---Redes neurales 15

3.2.1 Introducción de diferentes neutral networks 15

3.2.2 Principios de algoritmo de MLP 16

3.2.4 Is GLM a single layer neural network? 20

4. Case Study 21

4.1 Introducción de base de dato 21

4.2 Exploratory Data Analisis (R) 23

4.2.1 Uni-variable 23

4.2.2 Multivariable 31

4.3 Proceso de Modelización de GLM (R) 43

4.3.1 Mejor modelo de frecuencia 43

4.2.2 Mejor modelo de severidad 47

4.4 Proceso de Modelización de Redes Neurales (Python) 49

4.4.1 Pasos de la formación y conceptos básicos 51

4.4.2 Mejor modelo de frecuencia. 55

4.4.3 Mejor modelo de severidad 56

4.5 Comparación de resultados predicción de GLM y redes neurales 57

5. Memorias de modelado Y retroalimentación sobre el uso de R y Python 58

Bibliografía 61

Anexo (Godigo) 64

Anexo (Python Code) 83

Índice de Figuras

Figure 1: *Interpretability vs. Flexibility* 7

Figure *2*： *Que es Machine Learning?* 8

Figure *3*： *Problemas frecuentes sobre el aprendizaje automático* 8

Figure 4: *R vs. Python* 9

Figure *5*: *Neural Network* 15

Figure *6*: *Convolutional Neural Networks* 16

Figure 7： *Neurona* 17

Figure 8:*Imágenes de funciones de activación comunes en el aprendizaje automático* 18

Figure 9: *Estructura de Redes Neuronales* 18

Figure *10*: *Gradiente Boosting* 19

Figure *11*: *Un Red neuronale Real* 21

Figure 12: *Distribucion de ClaimNb y Exposure* 24

Figure 13: *Distribución de VehPower y VehAge* 24

Figure 14: *Distribución de DrivAge y BonusMalus* 25

Figure 15: *Distribución de VehBrand y Gas* 26

Figure 16: *Distribución de Area y Density* 26

Figure 17: *Distribución de Region* 27

Figure 18: *Distribución de ClaimAmount* 28

Figure 19: *Distribucion de VehPower y VehAge* 29

Figure 20: *Distribución de DrivAge y BonusMalus* 29

Figure 21: *Distribución de Density y VehBrand* 30

Figure 22: *Distribución de Gas y Area* 30

Figure 23: *Distribución de Region* 31

Figure 24: *Frecuencia Empirical de VehPower* 32

Figure 25: *Frecuencia Empirical de VehAge* 32

Figure 26: *Frecuencia Empirica de DrivAge* 33

Figure 27: *Frcuencia empirica de BonusMalus* 33

Figure 28: *Frecuencia Empirica de VehBrand* 34

Figure 29: *Frecuencia Empirica de Gas* 34

Figure 30: *Frcuencia Empirica de Area* 34

Figure 31: *Frcuencia Empirica de Density* 35

Figure 32: *Frecuencia Empirica de Rigion* 35

Figure 33: *Correlación de Variables Numéricas* 36

Figure 34: *Correlación de Variables Categóricas* 37

Figure 35: *Severidad Empirica de VehPower* 37

Figure 36: *Severidad Empirica de VehAge* 38

Figure 37: *Severidad Empirica de DrivAge* 38

Figure 38: *Severidad Empírica de BonusMalus* 39

Figure 39: *Severidad Empírica de VehBrand* 39

Figure 40: *Severidad Empírica de Area* 39

Figure 41: *Severidad Empírica de Density* 40

Figure 42: *Severidad Empírica de Gas* 40

Figure 43: *Severidad Empirica de Region* 41

Figure 44: *Correlación de variables numéricos de ClaimAmount* 42

Figure 45: *Correlación de variables categóricas de Claim Amount* 42

Figure 46: *Cheatsheet de Algorithm de Sklearn* 50

Figure 47: *Flujo de trabajo típico y módulos importantes asociados a cada paso* 51

Figure 48: *One hot Encoding* 53

Figure 49: *Dummy encoding* 54

Figure 50: *Drop out* 54

Figure 51: *MSE de Frecuencia de GLM* 57

Figure 52: *MSE de Frecuencia de Redes Neurales* 57

Figure 53:*MSE de Severidad de GLM* 58

Figure 54: *MSE de Severidad de Redes Neurales* 58

Figure 55: *Predicho negativo* 59

Figure 56: *Source code de MPLRegresor* 59

Índice de Tablas

Table 1: *Link function de GLM* 14

Table 2: *Datos de freMTPL2freq* 22

Table 3: *Datos de freMTPL2sev* 23

Table 4: *Estadisticas Descripitivas de freMTPL2freq* 24

Table 5: *Observaciones(num.) por grupo de VehAge* 25

Table 6: *Observaciones por grupo de BonusMalus* 25

Table 7: *Observaciones por grupo de Density* 27

Table 8: *Cuantil de ClaimAmount* 28

Table 9: *Estadisticas Descripitivas de freMTPL2seve* 28

Table 10: *correlación de Pearson para datos de frecuencias* 36

Table 11: *coeficiente de V-Cramer para datos de frecuencias* 36

Table 12: *correlación de Pearson para datos de severidad* 41

Table 13: *coeficiente de V-Cramer para datos de severidad* 42

Table 14: *Region luego de simplificaciones* 43

Table 15: *Modelo Inicial de Frecuencia de GLM* 45

Table 16: *Mejor Modelo de Frecuencia de GLM* 46

Table 17: *Modelo Inicial de Severidad de GLM* 48

Table 18: *Mejor Modelo de Severidad de GLM* 49

Table 19: *Mejor Modelo de Frecuencia de Redes Neurales* 56

Table 20: *MSE del Redes Neurales de Frecuencia* 56

Table 21:*Mejor Modelo de Severidad de Redes Neurales* 56

Table 22: *MSE del Redes Neurales de Severidad* 56

Table 23: *Métricas de evaluación del aprendizaje automático* 57

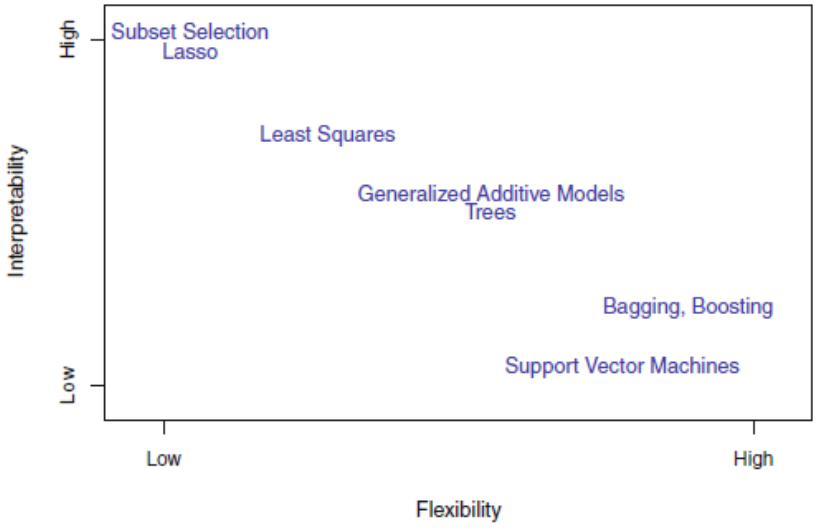
1. Motivación

Nadie duda de que la tarificación es el núcleo del negocio de los seguros, porque hay tres preocupaciones principales de los actuarios y los profesionales de los seguros: la fijación de las primas, la fijación de las reservas y el estudio de la probabilidad de insolvencia. La determinación de las primas es el aspecto más fundamental de la actividad aseguradora, y los otros dos están estrechamente relacionados con él. Para el asegurado, una póliza es una mercancía y la buena relación calidad-precio siempre ha sido la primera opción del cliente, por lo que el precio es un factor clave a tener en cuenta a la hora de elegir un producto de seguros.

En la industria tradicional de seguro, la manera más utilizada para hacer tarificación es el modelo GLM (General Lineal Model), que es muy interpretable pero menos flexible. Con el desarrollo de FINTECH, el sector de los seguros ha entrado en la era del machine learning.

Indudable, machine learning tiene más potencia en predecir, pero menos interpretable.

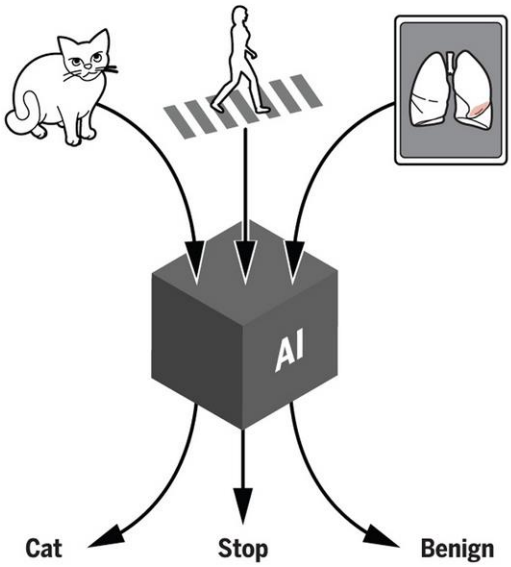
Figure 1: *Interpretability vs. Flexibility*



Fuente: *An Introduction to Statistical Learning with Applications in R (8th ed.).*

También, en el área de machine learning, algunos modelos carecen de apoyo teórico, por lo que tenemos que lidiar con algunos problemas de caja negra.

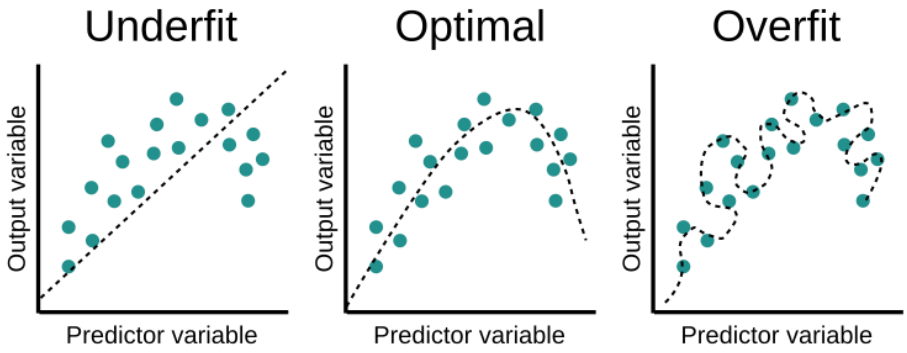
Figure 2： *Que es Machine Learning?*



Fuente: *Por internet*

Además, A veces, en la búsqueda de un mayor poder predictivo tenemos que enfrentarnos a algunos problemas de over-fitting.

Figure 3： *Problemas frecuentes sobre el aprendizaje automático*



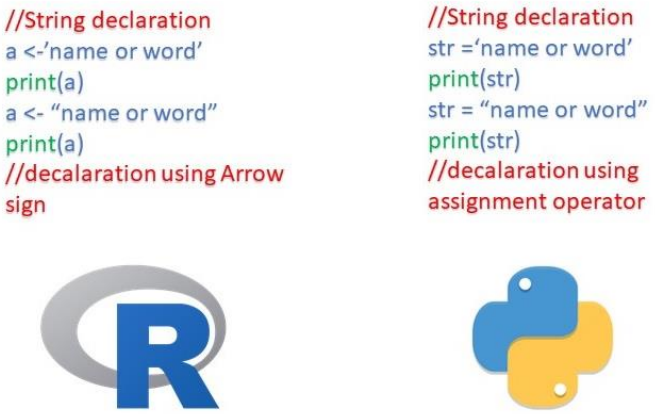
Fuente: *Summary of <<Learning from Data>>*

Por eso, la idea aquí es aprovechar la potencia de predecir del modelo Redes Neurales y la interpretad del modelo clásico GLM, y comparar las ventajas y desventajas entre sí.

En este trabajo se utilizan dos lenguajes, R y Python, dos de los más utilizados en la ciencia de datos. R es un software esencial para el aprendizaje de la estadística y la investigación. En el campo del aprendizaje automático, especialmente para los algoritmos de reconocimiento de imágenes, Python tiene más recursos, como scikit-learn, Pytorch, y los más sofisticados Tensor Flow, Keras... Además, al tratarse de lenguajes de programación de código abierto, cualquier persona puede descargarlos fácilmente y acceder a ellos de forma gratuita, y hay muchas similitudes entre los dos lenguajes de programación, que ambos funcionan super bien. Así que el debate entre r y Python ha sido continuo también en el campo de la tarificación de seguros. Otro

objetivo de este trabajo es comparar la experiencia de estos dos lenguajes en términos de computación.

Figure 4: *R vs. Python*



Fuente: *R vs Python Para El Aprendizaje Automático*

2. **La naturaleza del Seguro**

La industria del seguro se diferencia de manera fundamental de la mayoría de las industrias de la economía basadas en el esquema típico de producción de un bien y servicio a partir de unos factores productivos que se combinan para finalmente ofrecer un producto ya elaborado al consumidor.

En el caso de las aseguradoras, no se ofrece un bien o servicio directamente, sino que se hace una promesa (se establece un compromiso) de ofrecer un servicio en el futuro en el caso de que un hecho incierto se materialice sobre la persona o la propiedad del asegurado (cliente).

La implicación fundamental de este hecho es que en el momento de la venta, la aseguradora no conocerá los costes que derivarán de su servicio, dada la incertidumbre material y temporal a la que está sujeto el bien asegurado. Por ello, el precio a cobrar (la prima) se establece en base a una estimación de lo que podría pasar y del alcance de dichos hechos, no estando determinado en el presente.

De este ciclo inverso de explotación surgen todos los riesgos potenciales que deberán ser identificados, medidos, modelizados, y gestionados mediante coberturas, reservas apropiadas y dotaciones de capital, de modo que la información empírica (experiencia) tendrán un valor crucial para que, mediante las técnicas cuantitativas/estadísticas apropiadas, se pueda garantizar la estabilidad y viabilidad del negocio asegurador a lo largo del tiempo.

La ley de los grandes números es uno de los principios fundamentales sobre los que se sustenta el negocio asegurador y la técnica actuarial. La idea fundamental de este principio es que el riesgo relativo (Coeficiente de Variación) disminuye a medida que

aumenta el tamaño de negocio entendido como el número de unidades de riesgo o contratos en la cartera de la compañía.

Este efecto es debido a que el principio de diversificación elimina los riesgos idiosincráticos, los cuales se contrarrestan entre los distintos contratos siempre y cuando los riesgos entre estos sean independientes. Los riesgos no diversificables en cambio implican una limitación en la aplicación de la ley de los grandes números a causa de la simultaneidad en el acaecimiento de un riesgo o grupo de riesgos entre las distintas pólizas de la cartera.

En el área de seguro no vida, utilizamos la igualdad de

$$Prima\ pura = severidad * frecuencia$$

dentro de cual,

$$Frecuencia = \frac{numero\ de\ siniestro}{exposicion}$$

$$Severidad = \frac{Coste\ total}{numero\ de\ siniestro}$$

3. Resumen de los métodos de tarificación de seguros

3.1. GLM

El modelo lineal (**ordinary linear model**) puede representarse mediante la siguiente ecuación:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_{p-1} x_{p-1} + \epsilon \quad (3.1)$$

Aquí $\beta_i, i = 1, \dots, p-1$ se denomina parámetro desconocido, el β_0 se se denomina término intercepto.

Los principales supuestos del modelo lineal ordinario son los siguientes:

1. Normalidad de la variable dependiente Y y del término de error ϵ: variable dependiente y el término de error siguen una distribución normal, y el ϵ es un proceso de ruido blanco y, por tanto, tiene media cero y homocedasticidad.
2. No aleatoriedad de la variable explicativa x_i y del parámetro desconocido β_i: la variable explicativa es no aleatorios, medibles y libres de error de medición; los parámetros desconocidos se consideran constantes desconocidas, pero no aleatorias, y cabe destacar que las estimaciones de los parámetros desconocidos resueltas por métodos de mínimos cuadrados o de máxima verosimilitud entonces tiene normalidad.
3. Objeto de estudio: Como se ha mencionado anteriormente el término de salida del modelo lineal ordinario es la variable aleatoria Y.

4. Función de vinculación: Bajo los supuestos anteriores, tomando la expectativa matemática para ambos lados de (3.1), obtenemos:
$$E[Y] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_{p-1} x_{p-1} \ (3.2)$$
Como se ve en la ecuación (3.2), en el modelo lineal ordinario, la media de la variable dependiente y la combinación lineal de las variables explicativas por medio de una unión constante (identity), que por supuesto también puede pensarse como una unión por medio de una ecuación de la forma la función de enlace que une a los dos.

El modelo lineal generalizado (**generalized linear model**) es un modelo de regresión basado en el modelo lineal generalizado, que extiende los cuatro supuestos del modelo mencionados anteriormente a una gama más amplia de aplicaciones y es más práctico.

1. La distribución de la variable de respuesta se extiende a la familia de dispersión exponencial: por ejemplo, normal, Poisson, binomial, binomial negativa, gamma, gaussiana inversa. la definición detallada de la familia de dispersión exponencial no se describe en detalle aquí por razones de espacio.
2. No aleatoriedad de la variable explicativa x_i y del parámetro desconocido β_i: se sigue suponiendo que la variable explicativa no es aleatoria, es medible y está libre de errores de medición; el parámetro desconocido se considera una constante desconocida y no aleatoria.
3. Objeto de estudio: El principal objeto de estudio del modelo lineal generalizado sigue siendo la media de la variable dependiente $E[Y]$.
4. Función de vinculación: La función de vinculación utilizada en el modelo lineal generalizado puede ser teóricamente arbitraria, y ya no se limita a $f(x) = x$. Por supuesto, la elección de la función de vinculación debe adaptarse al caso concreto que se estudie. Al mismo tiempo, existen funciones de vinculación que corresponden a las distribuciones mencionadas en el supuesto 1, denominadas enlace canónico o enlace estándar, como la distribución normal correspondiente a la ecuación constante, la distribución de Poisson correspondiente a la función logaritmo natural, etc. La derivación de la función de enlace estándar y las ventajas de su aplicación implican la definición estandarizada de la familia de dispersión exponencial, que no se discutirá en detalle aquí.

Ahora vamos a hacer uno ejemplo que es adecuado más a modelo GLM:

- Se vendieron un total de 12 productos y un total de 6 personas devolvieron el producto durante el periodo de garantía. Cuando se hace una regresión de la tasa de retorno utilizando las características de estos 12 clientes, como los ingresos, la ocupación, la edad, etc., como variables predictoras, el modelo lineal normal deja de ser aplicable, ya que el evento de retorno ya no sigue una distribución normal, sino una **distribución binomial** (por supuesto, si el número de observaciones es lo suficientemente grande, por ejemplo, superior a 30, una distribución normal es una buena aproximación). El modelo lineal generalizado

(con una distribución binomial para la distribución objetivo) puede entonces utilizarse para modelar el evento.

- Las variables de resultado pueden ser del tipo de recuento (por ejemplo, número de accidentes de tráfico en una semana, número de bebidas consumidas al día). Dichas variables son todas valores finitos no negativos y sus medias y varianzas suelen estar correlacionadas (no es el caso de las variables con distribución normal, que son independientes entre sí), en este caso podemos utilizar el **Regresión Poisson**.

3.1.1 The exponencial family distribution

El modelo GLM se basa en la técnica de las distribuciones de la familia exponencial, por lo que primero introducimos la forma estándar de las distribuciones de probabilidad de la familia exponencial.

Una familia exponencial (**exponential family**) es esencialmente una generalización de una clase de funciones de densidad de probabilidad o distribuciones con dichas funciones de densidad que tienen la forma:

$$p(y \mid \theta) = \exp\{\theta^T T(y) - A(\theta) + S(y)\} \quad (3.3)$$

donde θ se denomina parámetro natural o parámetro canónico, que representa todos los parámetros desconocidos del modelo. Normalmente, una distribución de la familia exponencial tendrá dos parámetros, uno que representa la ubicación (location) y otro la escala (scale). El parámetro de localización está relacionado con la expectativa de la distribución, y el parámetro de escala está relacionado con la varianza de la distribución.

El modelo lineal generalizado que discutimos en este capítulo no utiliza la forma anterior de la familia exponencial, sino un subconjunto de la familia exponencial, **la familia exponencial natural**, que es la familia de exponentes que satisface $T(y) = y$, es decir:

$$p(y \mid \theta) = \exp\{\theta^T y - A(\theta) + S(y)\} \quad (3.4)$$

Esta forma de la familia exponencial se conoce como la forma natural (natural form) o la forma canónica (canonical form). Aunque la mayoría de las distribuciones de la familia exponencial pueden escribirse en esta forma natural, hay algunas distribuciones que forman parte de la familia exponencial pero que no pueden escribirse en esta forma natural, como la distribución LogNormal.

La familia de distribuciones exponenciales tiene algunas distribuciones con un parámetro y otras con dos parámetros, el parámetro canónico θ contiene todos los parámetros originales de la distribución. Cuando la distribución tiene un solo parámetro, θ es un parámetro escalar, y cuando la distribución tiene dos parámetros, θ es un parámetro vectorial binario. Los dos parámetros de la familia de distribuciones exponenciales están relacionados con la expectativa y la varianza de la distribución, que representan la localización y la escala, respectivamente.

Existe un mapeo uno a uno entre el parámetro canónico θ y el parámetro original de la familia de distribuciones exponenciales, donde el parámetro canónico θ puede ser un parámetro escalar o un vector que contiene dos parámetros. θ es una función de μ. Para una distribución de la familia exponencial de dos parámetros, los parámetros originales suelen ser la expectativa μ y la varianza σ^2 de la distribución, donde θ es un vector con dos parámetros y θ es una función de la expectativa μ y σ^2 en función de

La forma canónica de la familia de índices (Ecuación (3.4)) El parámetro canónico θ contiene todos los parámetros, lo cual no es conveniente de manejar. Por lo tanto, dividimos los parámetros e introducimos un parámetro ϕ que representa la escala sobre la forma canónica.

$$p(y \mid \theta) = \exp\left\{\frac{\theta y - b(\theta)}{a(\phi)} + c(y, \phi)\right\} \quad (3.5)$$

Esta forma de familia exponencial suele denominarse familia de dispersión exponencial (**exponential dispersion family**, EDF). $a(\phi)$ se conoce como función de dispersión (**dispersion function**) y lo sabemos. ϕ se denomina parámetro de dispersión (**dispersion parameter**). θ se sigue denominando parámetro natural o parámetro canónico.

La familia exponencial en la forma de la ecuación (6.1.3) es, de hecho, un desdoblamiento del parámetro θ, separando el parámetro de la expectativa del parámetro de la varianza. de modo que el parámetro natural θ sólo está relacionado con la expectativa μ y el parámetro de dispersión ϕ está relacionado con el parámetro de varianza de la distribución. Después de la división, el parámetro canónico θ sólo está relacionado con el parámetro de expectativa μ de la distribución y existe un mapeo uno a uno entre θ y μ. En otras palabras, θ y μ pueden transformarse el uno en el otro.

$$\theta = f(\mu) \quad (3.6)$$

$$\mu = f^{-1}(\theta) \quad (3.7)$$

3.1.2 GLM y función de vinculación

En el marco del **GLM**, suponemos que la variable de respuesta Y sigue una distribución de la familia exponencial, nuestro objetivo es predecir el valor de la variable de respuesta Y a partir de la variable de entrada X, y el MLG es un modelo lineal. Es decir, la combinación lineal de las variables de entrada X de Y se predice mediante una combinación lineal de las variables de entrada X.

Dominamos la ecuación siguiente predictores lineal.

$$\eta = \beta^T x + b$$

Recordemos que nuestra intención original era utilizar el resultado lineal η de la variable de entrada X para predecir el valor de la variable de salida Y, que es una familia exponencial de variables aleatorias, cuyo valor puede ser cualquiera de los valores del

espacio de su dominio de valores, pero cada valor puede tener una probabilidad diferente (aunque, por supuesto, la probabilidad de cada valor es la misma para una distribución uniforme). Sin embargo, esperamos un valor específico de Y, y obviamente la esperanza μ de la variable aleatoria Y es la mejor opción.

Ahora tenemos que obtener μ a partir de η, y luego utilizar μ como la salida del modelo, es decir, el valor predicho por el modelo de Y valor del modelo. ¿Cómo podemos hacer esto? Obviamente, se puede definir una función que relacione ambas cosas.

$$\eta = g(\mu)$$

$$\mu = g^{-1}(\eta)$$

La función $g(\cdot)$ suele denominarse función de vinculación (link function), y la función de vinculación $g(\cdot)$ se utiliza para conectar el predictor lineal η con la media μ. La inversa de la función de conexión $g^{-1}(\cdot)$ puede llamarse función de respuesta (**response function**), o función de activación (**active function**), y hay muchas opciones para la función de conexión. En un modelo lineal gaussiano (un modelo de regresión lineal tradicional), la función de conexión es la función constante $\eta = g(\mu) = \mu$. En una distribución de Poisson, la media μ debe ser positiva, por lo que $\eta = \mu$ ya no se aplica, ya que η toma valores sobre todo el dominio de los números reales. Para la distribución de Poisson, la función de conexión puede elegirse la función logarítmica $\eta = log(\mu)$, cuando $\mu = e^{\eta}$garantiza que μ es un número positivo. **La función de conexión es, en esencia, una transformación de η en el rango del dominio real a un espacio de valores μ que son legales para una distribución particular.**

La siguiente tabla muestra las distribuciones más utilizadas y sus correspondientes funciones de enlace.

Table 1: *Link function de GLM*

Common distributions with typical uses and canonical link functions

Distribution	Support of distribution	Typical uses	Link name	Link function, $\mathbf{X}\boldsymbol{\beta} = g(\mu)$	Mean function
Normal	real: $(-\infty, +\infty)$	Linear-response data	Identity	$\mathbf{X}\boldsymbol{\beta} = \mu$	$\mu = \mathbf{X}\boldsymbol{\beta}$
Exponential Gamma	real: $(0, +\infty)$	Exponential-response data, scale parameters	Inverse	$\mathbf{X}\boldsymbol{\beta} = \mu^{-1}$	$\mu = (\mathbf{X}\boldsymbol{\beta})^{-1}$
Inverse Gaussian	real: $(0, +\infty)$		Inverse squared	$\mathbf{X}\boldsymbol{\beta} = \mu^{-2}$	$\mu = (\mathbf{X}\boldsymbol{\beta})^{-1/2}$
Poisson	integer: $0, 1, 2, \ldots$	count of occurrences in fixed amount of time/space	Log	$\mathbf{X}\boldsymbol{\beta} = \ln(\mu)$	$\mu = \exp(\mathbf{X}\boldsymbol{\beta})$
Bernoulli	integer: $\{0, 1\}$	outcome of single yes/no occurrence	Logit	$\mathbf{X}\boldsymbol{\beta} = \ln\left(\frac{\mu}{1-\mu}\right)$	$\mu = \frac{\exp(\mathbf{X}\boldsymbol{\beta})}{1+\exp(\mathbf{X}\boldsymbol{\beta})} = \frac{1}{1+\exp(-\mathbf{X}\boldsymbol{\beta})}$
Binomial	integer: $0, 1, \ldots, N$	count of # of "yes" occurrences out of N yes/no occurrences			
Categorical	integer: $[0, K)$ K-vector of integer: $[0, 1]$, where exactly one element in the vector has the value 1	outcome of single K-way occurrence			
Multinomial	*K*-vector of integer: $[0, N]$	count of occurrences of different types (1 .. *K*) out of *N* total *K*-way occurrences			

Fuente: *wikipedia*

3.2. Desde lineal a No lineal---Redes neurales

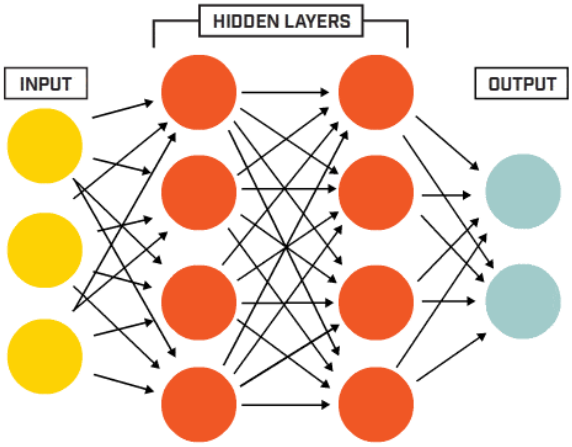
3.2.1 Introducción de diferentes neutral networks

- Multilayer Perceptron (MLP)

Multilayer Perceptron es la red neuronal más clásica. Consta de una o varias capas de neuronas. Los datos se introducen en la capa de entrada (**Input Layer**), una o varias capas ocultas (**Hidden Layer**) que pueden existir para proporcionar un nivel de abstracción, y las predicciones se realizan en la capa de salida (**Output Layer**).

El MLP es aplicable a los problemas de predicción de **clasificación**, en los que se asignan clases o etiquetas a las entradas. También es aplicable a los problemas de predicción por **regresión**, en los que se predice una cantidad de valor real, dado un conjunto de entradas. En la próxima capitulo vamos a introducir el algoritmo más detalladamente, por lo tanto, no nos extenderemos aquí.

Figure 5: *Neural Network*



Fuente: *What is ANN. TIBCO.*

- Convolucional Neural Networks (CNN)

Una red neuronal convolucional consta de una o más capas convolucionales y una capa superior totalmente conectada (correspondiente a una red neuronal clásica), pero también incluye pesos asociativos y una capa de agrupación. Esta estructura permite a las redes neuronales convolucionales explotar la estructura bidimensional de los datos de entrada.

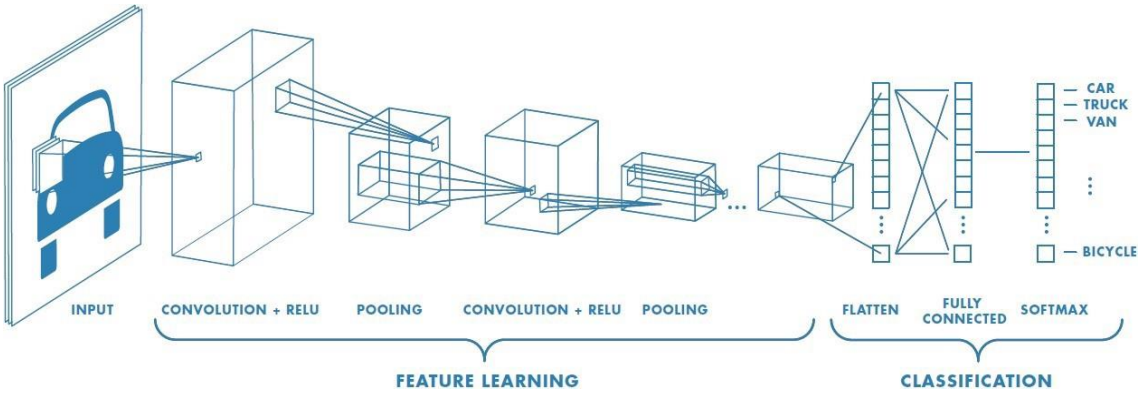
La CNN es la que mejor procesa las imágenes. Se inspira en el sistema nervioso visual humana.

La CNN tiene dos características principales:

a. Puede reducir eficazmente la dimensionalidad de las imágenes de gran volumen de datos en un volumen de datos pequeño.

b. Puede preservar eficazmente las características de la imagen, lo que está en consonancia con el principio del procesamiento de imágenes

Figure 6: *Convolutional Neural Networks*



Fuente: *Components of a Convolutional Neural Network and How the Convolutional Layers Work in an Image? Zhihu*

- Recursive Neural Network (RNN)

Las redes neuronales recurrentes (RNN) están diseñadas para tratar problemas de predicción de secuencias. Los problemas de predicción de secuencias pueden adoptar muchas formas y se describen mejor en términos de los tipos de entradas y salidas admitidos. Las aplicaciones de las redes neuronales recurrentes se encuentran en el análisis sintáctico del lenguaje y las escenas naturales, por ejemplo, la mayoría de los programas de traducción utilizan esta tecnología y consiguen resultados excelentes, comparables a los de la traducción humana.

3.2.2 Principios de algoritmo de MLP

En este documento utilizaremos el MPL, por lo que a continuación presentamos una breve introducción a los principios estructurales de este modelo.

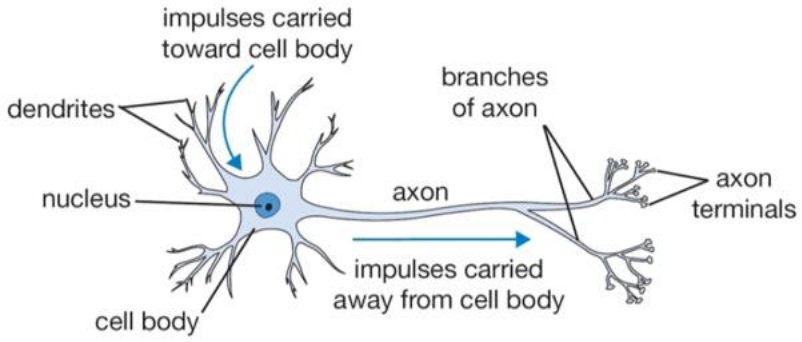
- Inspiración neuronal

Como su nombre indica, los algoritmos de las redes neuronales proceden de las neuronas biológicas.

Las neuronas biológicas están conectadas capa por capa, y cuando una señal neuronal alcanza una determinada condición, esta neurona se activa y sigue transmitiendo información.

Para seguir utilizando las redes neuronales para resolver este problema de no tener diferenciabilidad lineal, se toman más neuronas para insertarlas entre la entrada y la salida de la red neuronal.

Figure 7： *Neurona*



Fuente: *What is ANN. TOBIC*

Podemos obtener la estructura básica de un perceptrón multicapa MLP basándonos en el modelo neuronal biológico. El MLP más típico consta de tres capas: una capa de entrada, una capa oculta y una capa de salida, y las diferentes capas de la red neuronal MLP están totalmente conectadas entre sí (totalmente conectadas significa: cualquier neurona de la capa superior está conectada a todas las neuronas de la capa inferior).

- Introducción al principio del Perceptrón Multicapa (MLP)

Se puede ver que las redes neuronales tienen tres elementos básicos principales: pesos(**weights**), sesgos(**bais**) y funciones de activación (**active function**)

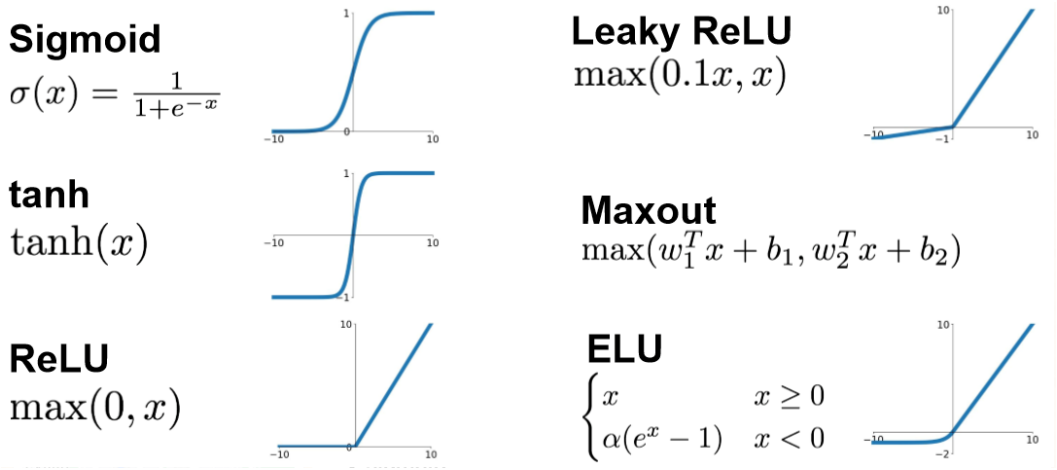
- Pesos w_{ji}, la fuerza de las conexiones entre las neuronas está representada por los pesos, cuyo tamaño indica la magnitud de la probabilidad
- Sesgo b_i, el sesgo se establece para clasificar correctamente las muestras y es un parámetro importante en el modelo, es decir, garantiza que los valores de salida calculados a partir de las entradas no puedan activarse al azar.
- Función de activación$f(\cdot)$, actúa como un mapeo no lineal, que limita la amplitud de salida de la neurona a un determinado rango.

La función de activación es un componente importante de un modelo de red neuronal. Presentamos brevemente las características de las funciones de activación y las funciones de activación más utilizadas.

La función de activación debe satisfacer al menos los siguientes puntos.

- Diferenciabilidad: dado que el método de optimización se basa en el gradiente, esta propiedad es imprescindible.
- Monotonicidad: Cuando la función de activación es monótona, es capaz de garantizar que la red monocapa es una función convexa.
- Rango de valores de salida: El rango de valores de salida de la función de activación puede ser finito o infinito. Cuando el valor de salida es finito, la optimización basada en el gradiente es más estable porque la representación de las características se ve afectada de forma más significativa por los pesos finitos; cuando el valor de salida es infinito, el modelo se entrena de forma más eficiente, aunque en este caso se requiere generalmente una tasa de apren

Figure 8:*Imágenes de funciones de activación comunes en el aprendizaje automático*



Fuente: *funciones de activación comunes en el aprendizaje automático. CSND*

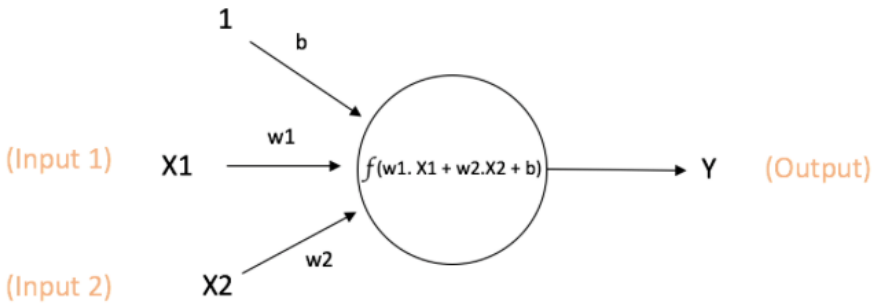
Además, como hablamos en la introducción, MLP consta de una o varias capas de neuronas: Capa de entrada (**Input Layer**), una o varias capas ocultas (**Hidden Layer**) ***H***, que pueden existir para proporcionar un nivel de abstracción, y las predicciones se realizan en la capa de salida (**Output Layer**) ***O***.

$$H = f(XW_h + b_h)$$

$$O = HW_o + b_o$$

Hacemos un ejemplo más sencillo, el abajo es una estructura más plana de redes neurales: en la capa entrada, tiene dos variables, no tiene capa oculta.

Figure 9: *Estructura de Redes Neuronales*



Fuente: *Structure of Neural Network. CSND*

Su fórmula es:

$$\boldsymbol{O}: Y = f(w1.X1 + w2.X2 + b)$$

No sé si esto le resulta familiar?

Con esta estructura, podemos ampliar nuestras redes neurales: más capas ocultas, diferentes funciones de activación, etc…

- Gradient Descent Optimization back propagation

Ya sea el aprendizaje automático o el aprendizaje profundo. No hay forma de evitar el Gradient Descent Optimization. En nuestra introducción al aprendizaje profundo, hemos cubierto los pasos generales del aprendizaje profundo, que son

1. Construir una red neuronal
2. Ajuste de los datos
3. Seleccionar el mejor modelo

El mejor modelo se selecciona mediante un algoritmo de descenso de gradiente（**Gradient Descent**）para seleccionar el que tenga la función de pérdida más pequeña. En un modelo estadístico tradicional, esto es relativamente fácil de hacer, basta con hacer los cálculos. En el aprendizaje profundo, sin embargo, hay capas de entrada, capas ocultas y capas de salida, y se desconoce la profundidad de las capas ocultas, por lo que el cálculo es más complicado.

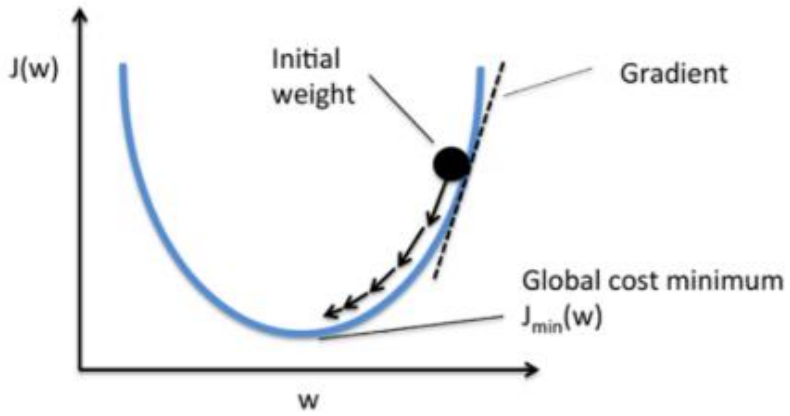
A continuación, ofrecemos una breve introducción al descenso de gradiente.

El principio del algoritmo de descenso de gradiente: el gradiente de la función objetivo $J(\theta)$con respecto a los parámetros θ será la dirección en la que la función de pérdida (función de pérdida) suba más rápido. Para minimizar la pérdida, simplemente avanzamos los parámetros un paso en la dirección opuesta al gradiente para conseguir una disminución de la función objetivo (función de pérdida). Este paso [ecuación] también se conoce como tasa de aprendizaje. La fórmula de actualización de los parámetros es la siguiente.

$$\theta \leftarrow \theta - \eta \cdot \nabla J(\theta)$$

donde $\nabla J(\theta)$ es el gradiente del parámetro.

Figure 10: *Gradiente Boosting*



Fuente: *Gradiente Boosting. CSND*

Si la salida de datos en la capa de salida difiere significativamente del objetivo y de los criterios que hemos establecido, entonces se requiere la propagación hacia atrás. Utilizando la back propagation, las derivadas parciales de la función objetivo con respecto a los pesos de cada neurona se encuentran capa por capa, formando el gradiente de la función objetivo con respecto al vector de pesos. La razón para calcular esto es proporcionar una base para la optimización de los pesos, y cuando los pesos han sido optimizados, entonces cambiar a la propagación hacia adelante. El algoritmo termina cuando la salida alcanza el criterio establecido.

3.2.4 Is GLM a single layer neural network?

En la última subsección, hemos intentado escribir una expresión sencilla para un modelo redes neurales y te hemos preguntado si te resultaba familiar. Estoy seguro de que después de leer sobre la estructura de las redes neuronales tendrá una sensación de déjà vu(familiar). En esta sección estableceremos una analogía entre las redes neuronales y los modelos lineales generalizados. **De hecho, las redes neuronales no son más que GLM recursivos.**

Sin embargo, en la sección anterior hemos dado ejemplos de aplicaciones de los GLM, y en todos ellos hemos asumido que tenemos métodos establecidos y estamos bastante seguros del modelo paramétrico de los GLM que estamos ajustando. Por ejemplo, supongamos que tenemos una relación de entrada y salida con n características y la salida es una distribución Bernoulli. En este caso, elegiríamos ajustar un modelo de regresión logística especificando (n+1) coeficientes (1 término de intercepción y un coeficiente para cada una de las n características) sin ninguna transformación de las características ni términos de interacción conjuntos. el modelo de regresión logística devolvería sólo los resultados e inferencias "correctos" porque la especificación del modelo del problema es correcta. En realidad, se trata de supuestos bastante estrictos.

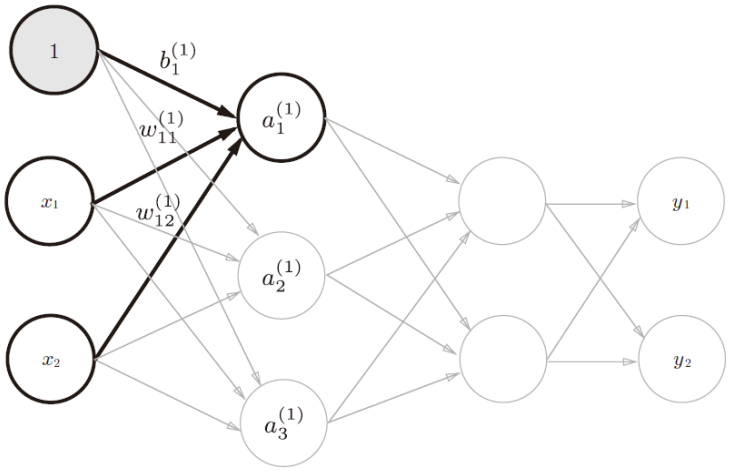
Sin embargo, supongamos ahora que no estamos dispuestos a hacer suposiciones paramétricas tan estrictas sobre nuestra comprensión de la especificación del modelo. De hecho, supongamos que no estamos dispuestos a hacer ninguna suposición sobre la especificación del modelo en absoluto. Nos gustaría utilizar un enfoque que relaja en gran medida la necesidad de los supuestos anteriores. Hay una clase de métodos conocidos como "aproximadores genéricos", y las redes neuronales están entre ellos.

Supongamos que tenemos un modelo estadístico como éste.

- En lugar de especificar un único GLM, especificamos varios **GLM paralelos** entre sí. Estas agrupaciones paralelas de GLM las llamamos "capas".
- En lugar de un modelo compuesto por una sola "capa", hay varias capas. La salida de la capa anterior de GLMs se convierte en las características de entrada de la siguiente capa de GLMs.

Supongamos que tenemos la siguiente red:

Figure 11: *Un Red neuronale Real*



Fuente: Design of Three-Layer Neural Networks. Icode.

ahora nos centramos en la parte superior izquierda de este modelo (la parte oscura), la expresión para esta parte es la siguiente.

$$a_1^{(1)} = f\left(w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + b_1^{(1)}\right)$$

Si sustituimos la función de activación externa por la 'indentity', tenemos una **regresión lineal**. Si lo sustituimos por sigmiod, es una **regresión logística**.

Como puede ver, el modelo descrito anteriormente es, en realidad, una red neuronal de avance estándar. Llamamos a la primera capa de la red "capa de entrada", a la última capa "capa de salida" y a todas las capas intermedias "capas ocultas".

Por eso decimos: **las redes neuronales no son más que GLM recursivos.**

Esta parte también podemos derivarla a partir de fórmulas matemáticas, pero no es el objetivo de esta tesis, por lo que no nos extenderemos en ella.

4. Case Study

A continuación, haremos modelos de tarificación por el modelo tradicional GLM y un modelo de Machine learning---Nueral netwrok con un conjunto de datos disponibles públicamente y compararemos los resultados.

4.1 Introducción de base de dato

El conjunto de datos originalmente para el libro 'Computational Actuarial Science with R'. Vamos a utilizar el dato en la sección freMTPL como "French Motor Third-Part Liability datasets". Especialmente, los siguiente dos son datos que vamos a utilizar para modelización.

- freMTPL2freq

- freMTPL2sev

En los dos conjuntos de datos freMTPL2freq, freMTPL2sev, se recogen las características de riesgo de 677.991 pólizas de responsabilidad civil de automóviles (observadas principalmente en un año). Además, disponemos de los números de siniestros por póliza, así como de los correspondientes importes de siniestros. freMTPL2freq contiene las características de riesgo y el número de siniestro, mientras que freMTPL2sev contiene el importe de siniestro y el correspondiente ID de póliza.

Los datos están ordenados por número de póliza. Hay 8 características en los datos.

freMTPL2freq contiene 11 columnas:

- IDpol: El ID de la póliza (utilizado para enlazar con el conjunto de datos de reclamaciones).
- ClaimNB: Número de siniestros durante el periodo de exposición.
- Exposure: El periodo de exposición de una póliza, en años.
- VehPower: La potencia del coche (ordenada categóricamente).
- VehAge: La edad del vehículo, en años.
- DrivAge: La edad del conductor, en años (en Francia, se puede conducir un coche a los 18 años).
- VehBrand: La marca del coche (categorías desconocidas).
- VehGas: La gasolina del coche, Diesel o Regular.
- Area: El valor de la densidad de la comunidad de la ciudad en la que vive el conductor del coche: desde "A" para la zona rural hasta "F" para el centro urbano.
- Density: La densidad de habitantes (número de habitantes por kilómetro cuadrado) de la ciudad en la que vive el conductor del coche.
- Region: La región política en Francia (basada en la clasificación de 1970-2015).
- BonusMalus: entre 50 y 350. <100 significa bonificación, >100 significa malus en Francia.

Table 2: *Datos de freMTPL2freq*

IDpol	ClaimNb	Exposure	VehPower	VehAge	DrivAge	BonusMalus	VehBrand	VehGas	Area	Density	Region
1	1	0.10	5	0	55	50	B12	Regular	D	1217	Rhone-Alpes
3	1	0.77	5	0	55	50	B12	Regular	D	1217	Rhone-Alpes
5	1	0.75	6	2	52	50	B12	Diesel	B	54	Picardie
10	1	0.09	7	0	46	50	B12	Diesel	B	76	Aquitaine
11	1	0.84	7	0	46	50	B12	Diesel	B	76	Aquitaine
13	1	0.52	6	2	38	50	B12	Regular	E	3003	Nord-Pas-de-Calais

Fuente: *Elaboración Propia*

freMTPL2sev contiene 2 columnas:

- IDpol: La fecha de ocurrencia (utilizada para enlazar con el conjunto de datos del contrato).

- ClaimAmount: El coste de la reclamación, visto en una fecha reciente.

Table 3: *Datos de freMTPL2sev*

IDpol	ClaimAmount
1552	995.20
1010996	1128.12
4024277	1851.11
4007252	1204.00
4046424	1204.00
4073956	1204.00

Fuente: *Elaboración Propia*

4.2 Exploratory Data Analisis (R)

A continuación, vamos a utilizar hacer Exploratory Data Analisis por visualización y estadísticas descriptivas.

En el caso de industria, es un paso muy necesario, por el EDA podemos tener una idea general de conjunto de dato: la distribución, si existe valores erróneos y outliers. Además, en este paso, podemos tener ideas de base de dato, y también proporciona una buena base para futuros currículos de ingeniería de características (feature engineering).

En esta tesis, el EDA tiene 3 secciones principales: estadística descriptiva, análisis/visualización univariante, análisis multivariantes.

Además, antes de realizar el EDA deberíamos realizar la detección de valores perdidos, y la eliminación de valores atípicos, pero estos datos son muy completos y no tienen valores perdidos, así que no volveré a hablar de esto. Además, eliminamos los registros que son mayores de 6 veces, ya que es una situación muy poco común en la vida real.

Además, cuando hagamos la modelización de GLM de severidad, utilizaremos la regresión Gamma, que se diferencia de la regresión de Poisson en que es más probable que reciba valores extremos. En las operaciones de la vida real, además, modelaremos estos extremos en lo que se conoce como modelo de catástrofe. Por eso vamos a quitar los valores extremos luego hacer EDA sobre el conjunto de dato freMTPL2sev.

4.2.1 Uni-variable

En esta sección, tenemos 2 partes:

a) estadística descriptiva para variable numérica
b) visualización de tanta variable predictiva como variable explicativa.

Frecuencia

a) Estadística descriptiva

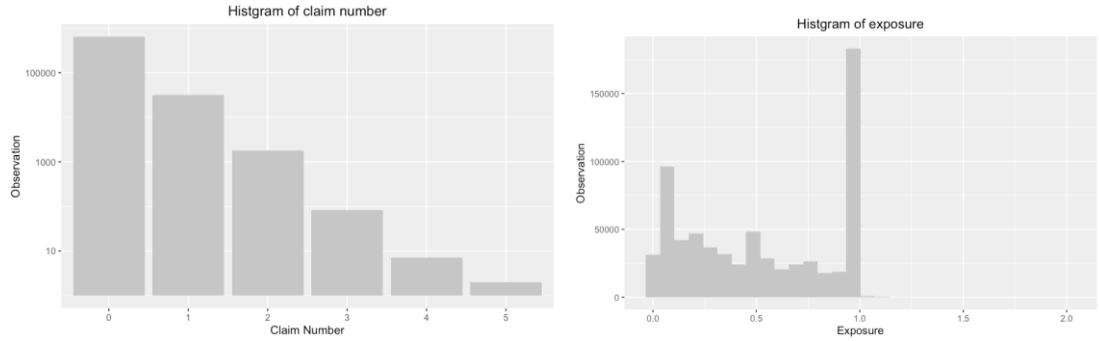
Table 4: *Estadisticas Descripitivas de freMTPL2freq*

	ClaimNb	Exposure	VehPower	VehAge	DrivAge	BonusMalus	Density
Min. :	0,00	0,00	4,00	0,00	18,00	50,00	1,00
1st Qu.:	0,00	0,18	5,00	2,00	34,00	50,00	92,00
Median :	0,00	0,49	6,00	6,00	44,00	55,00	392,00
Mean :	0,05	0,53	6,46	7,04	45,50	59,76	1972,00
3rd Qu.:	0,00	0,99	7,00	11,00	55,00	64,00	1658,00
Max. :	5,00	2,01	15,00	100,00	100,00	230,00	27000,00

Fuente: *Elaboración Propia*

b) Visualización

Figure 12: *Distribucion de ClaimNb y Exposure*

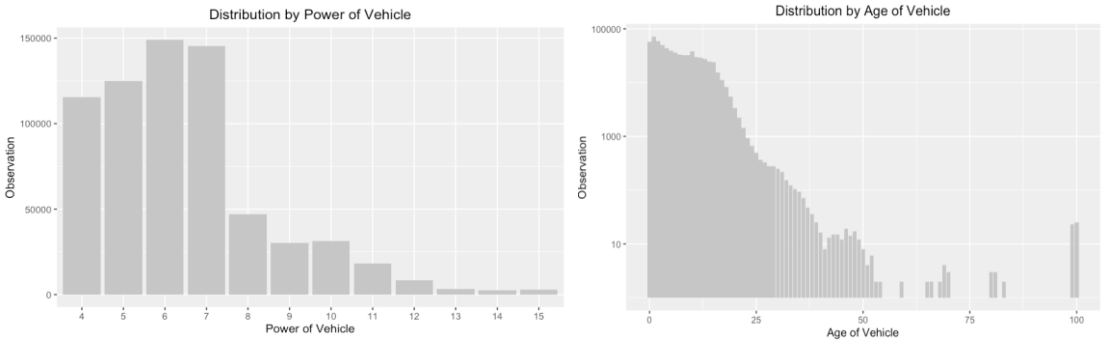


Fuente: *Elaboración Propia*

En este conjunto de datos, 63465 pólizas no han tenido ningún accidente, el 94% del total de los datos, por lo que el 0 aparece con mayor frecuencia. Para una mejor visualización de los datos, hemos utilizado aquí coordenadas del logaritmo.

Se puede ver que la mayoría de las pólizas tiene exposición de un año. y muy pocos tiene que más que un año.

Figure 13: *Distribución de VehPower y VehAge*



Fuente: *Elaboración Propia*

Podemos ver que esta variable presenta un fuerte imbalance en los datos de VehPower, ya que las potencias 4 a 12 ocupan la mayor parte del cuadro general, mientras que las imágenes de los niveles 13 a 15 apenas son visibles.

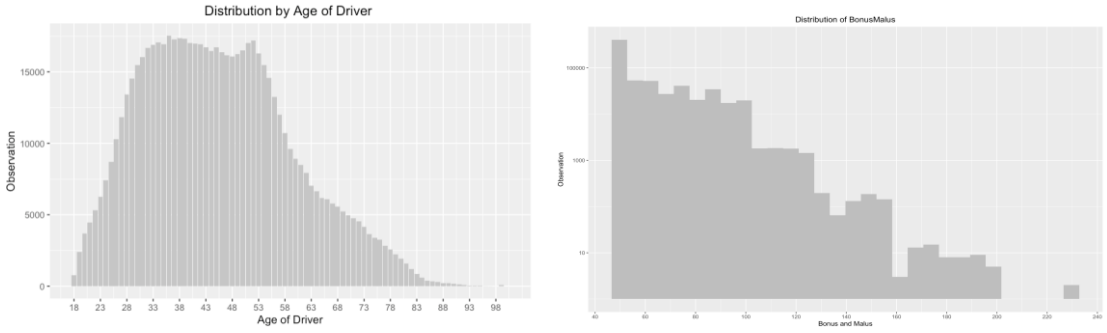
En el gráfico de distribución de VehAge, estamos utilizando el log10 axis para ver mejores la distribución. Para saber bien de distribución de la variable VehAge.

Table 5: *Observaciones(num.) por grupo de VehAge*

(0,10]	(10,20]	(20,30]	(30,40]	(40,50]	(50,60]	(60,70]	(70,80]	(80,90]	(90,100]
434491	177454	7206	886	133	18	16	7	8	48

La edad de vehículo (VehAge) también tiene un claro efecto de cola larga, por lo que he utilizado el eje logarítmico en las imágenes. Con una edad mínima de 0 años y una edad máxima de 100 años, podemos ver que la mayoría de los vehículos asegurados se concentran antes de los 50 años, mientras que menos de un centenar de vehículos están asegurados después de los 50 años.

Figure 14: *Distribución de DrivAge y BonusMalus*



Fuente: *Elaboración Propia*

En comparación, la distribución de la edad de conductores (DrivAge) es relativamente estable, y aunque hay muchos conductores de edad avanzada, son relativamente pocos. El conductor más joven tiene dieciocho años y el más viejo cien, y la mayoría se concentra entre los treinta y los cincuenta años.

Lo mismos que la variable VehAge, por el log10 axis, pongo unas tablas de observaciones para la variable BonusMalus.

Table 6: *Observaciones por grupo de BonusMalus*

(50,60]	(60,70]	(70,80]	(80,90]	(90,100]	(100,110]	(110,120]
94446	56929	61826	35725	37137	1989	3565
(120,130]	(130,140]	(140,150]	(150,160]	(160,170]	(170,180]	(180,190]
1458	374	199	144	16	17	17
(190,200]	(200,210]	(210,220]	(220,Inf]			
11	1	1	2			

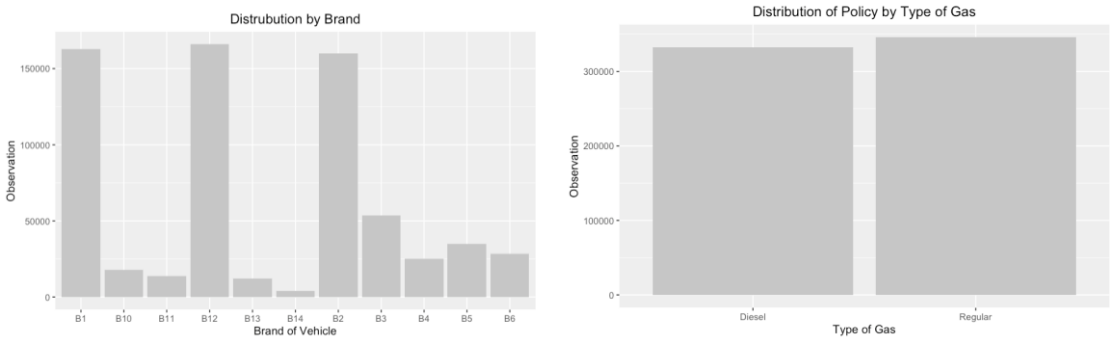
Fuente: *Elaboración Propia*

Se trata de un ramo importante del seguro no-vida y en muchos países es incluso el mayor ramo en términos de ingresos totales por primas. Una característica de los seguros de automóviles es que, normalmente para satisfacción de todos, las primas que se cobran dependen en gran medida de los siniestros pasados de la póliza.

En nuestro caso, el rango de Bonus Malus es de 50 a 350, siendo Bonus para menos de 100 y Malus para más de 100, y está claro que hay más gente con un buen historial de conducción que con uno malo.

- VehBrand

Figure 15: *Distribución de VehBrand y Gas*

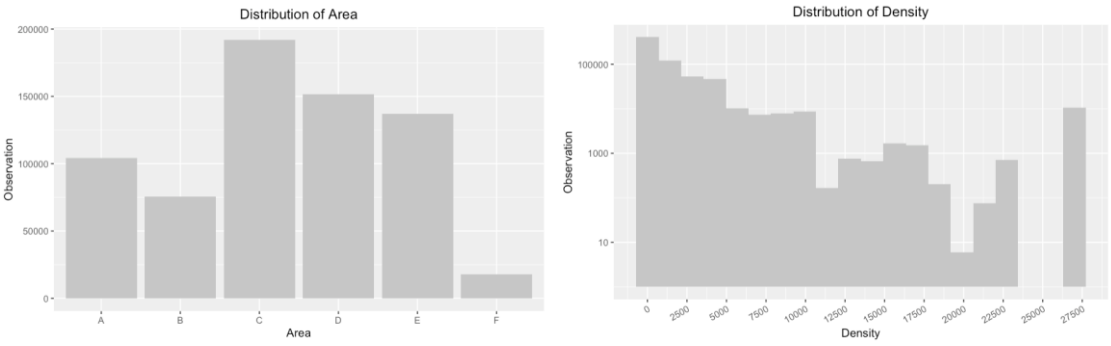


Fuente: *Elaboración Propia*

En la variable VehBrand, también se observa que los datos están desequilibrados, dominando las marcas B1, B12 y B2.

La proporción de clientes que utilizan Gas en la variable de Diesel es casi igual a la proporción de clientes que utilizan Regular.

Figure 16: *Distribución de Area y Density*



Fuente: *Elaboración Propia*

En la variable Area, podemos ver que los clientes que viven en el área F (urban centre) son los menos.

Ten cuidado, usamos el log10 scale para y axis. Como el anterior pongo números concretos de observaciones de cada grupo

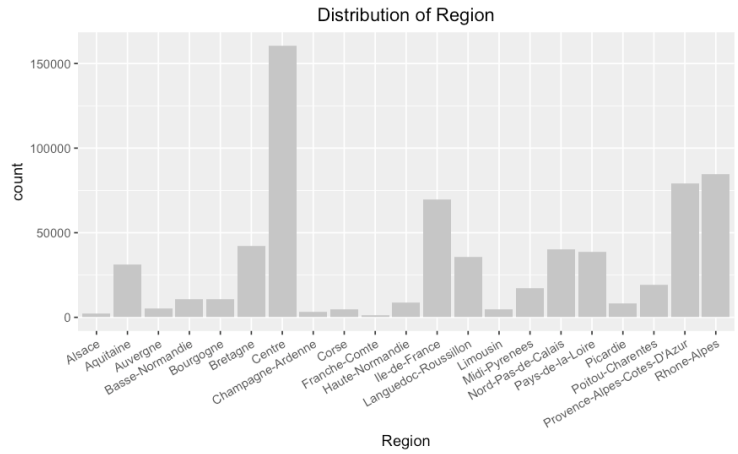
Table 7: *Observaciones por grupo de Density*

(0,2500]	(2500,5000]	(5000,7500]	(7500,10000]
536801	91821	15896	15534
(10000,12500]	(12500,15000]	(15000,17500]	(17500,20000]
2208	1050	3188	206
(20000,22500]	(22500,25000]	(25000,27500]	(27500,Inf]
76	711	10515	0

Fuente: *Elaboración Propia*

La densidad tiene un efecto de cola larga. Por eso hemos utilizado también el eje logarítmico.

Figure 17: *Distribución de Region*



Fuente: *Elaboración Propia*

En 2014, Francia aprobó una reforma de sus divisiones administrativas, que entró en vigor en 2016. Los datos para esta tesis se recogieron antes de 2014, por lo que las divisiones administrativas son diferentes a las actuales en Francia.

En la variable Región, los datos siguen siendo desiguales. La provincia con más clientes es el Centro, y en los siguientes trabajos de características fusionaremos las provincias con menos usuarios.

Severidad

Como hemos explicado antes, en el proceso de modelización gamma, que es más ser afectada por el valor extremos. Por eso, vamos a quitar los valores extremos, luego hacemos el EDA.

a) Quitar valores extremos

En el mundo real, pocos siniestros van a tener coste mayor que 10,000 euros, por eso vamos a quitar las observaciones con coste mayor que esta masa. Podemos ver que sinestros con costes mayores que 10,000 euros son sobre 500 registros, que representa aproximadamente el 2% de los datos globales. Tras eliminar estos datos extremos, nos quedan 25966 observaciones.

Table 8: *Cuantil de ClaimAmount*

	Quantil <fctr>	Claim Amount <dbl>	Obervations <dbl>
1	98%	9742.239	529
2	98.1%	10000.000	502
3	98.2%	10057.076	476
4	98.3%	10407.131	450
5	98.4%	11005.666	423
6	98.5%	11704.593	397

Fuente: *Elaboración Propia*

b) Estadística Descriptiva

El siguiente cuadro es lo de estadísticos descriptivos.

Lo que me llama atención es que la media es mayor que mediana. Según la definición tradicional, una media mayor que la mediana se denomina sesgada a la derecha, lo que también puede interpretarse como que la cola larga está a la derecha.

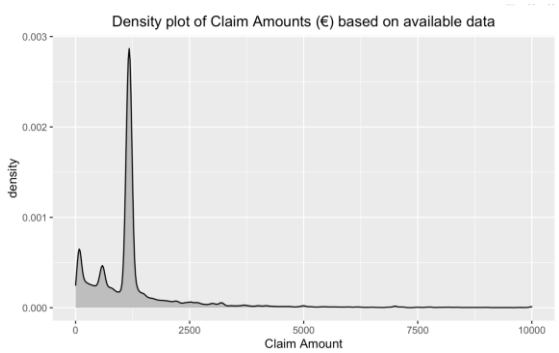
Table 9: *Estadisticas Descripitivas de freMTPL2seve*

	ClaimAmount	VehPower	VehAge	DrivAge	BonusMalus	Density
Min. :	1,00	4,00	0,00	18,00	50,00	2,00
1st Qu.:	665,10	5,00	3,00	34,00	50,00	115,00
Median :	1172,00	6,00	7,00	45,00	55,00	523,00
Mean :	1326,10	6,46	7,36	45,14	65,19	2022,00
3rd Qu.:	1204,00	7,00	11,00	54,00	776,00	2252,00
Max. :	10000,00	15,00	99,00	99,00	228,00	27000,00

Fuente: *Elaboración Propia*

c) Visualización
   - ClaimAmount

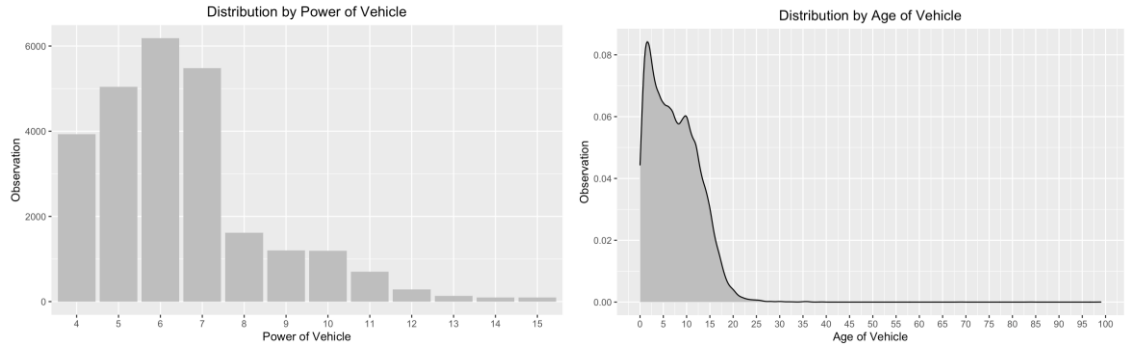
Figure 18: *Distribución de ClaimAmount*



Fuente: *Elaboración Propia*

Lugo de quitar los valores extremos, aun podemos observar un claro efecto de cola larga. La mayoría de los costes de siniestros se centran en 1000 euros.

Figure 19: *Distribucion de VehPower y VehAge*

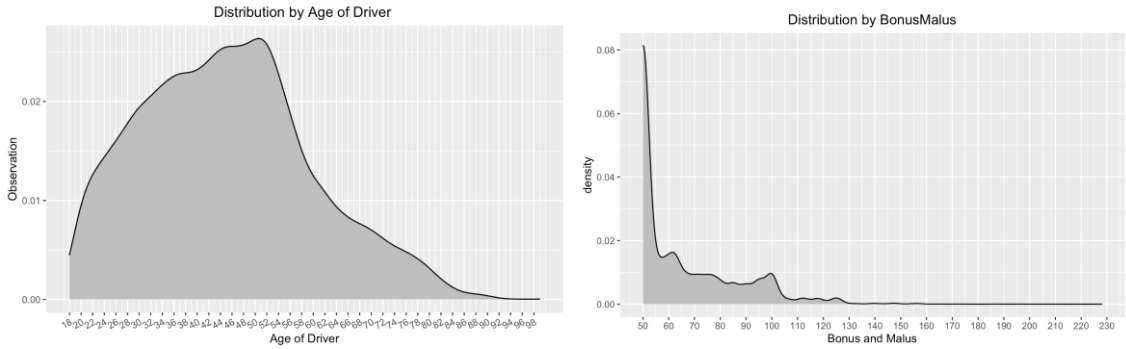


Fuente: *Elaboración Propia*

La mayoría observaciones están en la Potencia 4-7. Lo que tiene más observaciones es la potencia 6.

Podemos ver que dentro que los clientes que tuvieron accidente. Los coches nuevos constituyen la mayoría de la muestra, los coches luego de 25 años de antigüedad casi no tienen accidentes.

Figure 20: *Distribución de DrivAge y BonusMalus*

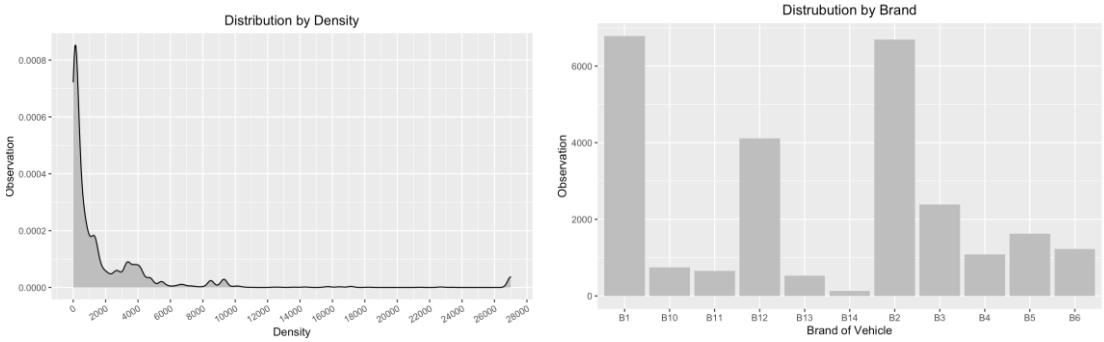


Fuente: *Elaboración Propia*

Dentro de los clientes que han tenido accidentes, las personas de mediana edad (40-50 años) son las más numerosas,

El bonus entre 50 y 350: <100 significa bonificación, >100 significa malus en Francia. Obviamente, dentro de la gente que ha tenido accidente, lo ha tenido bonus es mayor que lo ha tenido bonus.

Figure 21: *Distribución de Density y VehBrand*

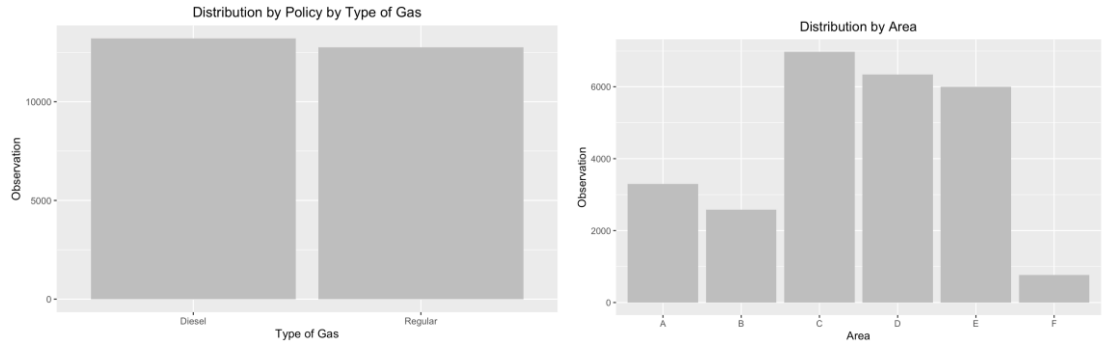


Fuente: *Elaboración Propia*

La mayoría viven en ciudad con densidad menos que 6000 habitantes por km^2.

La distribución de la gente ha tenido accidente es similar a la distribución de la cartera de polizas. Las mayorías de los siniestros se centran en B1, B2 y B12.

Figure 22: *Distribución de Gas y Area*

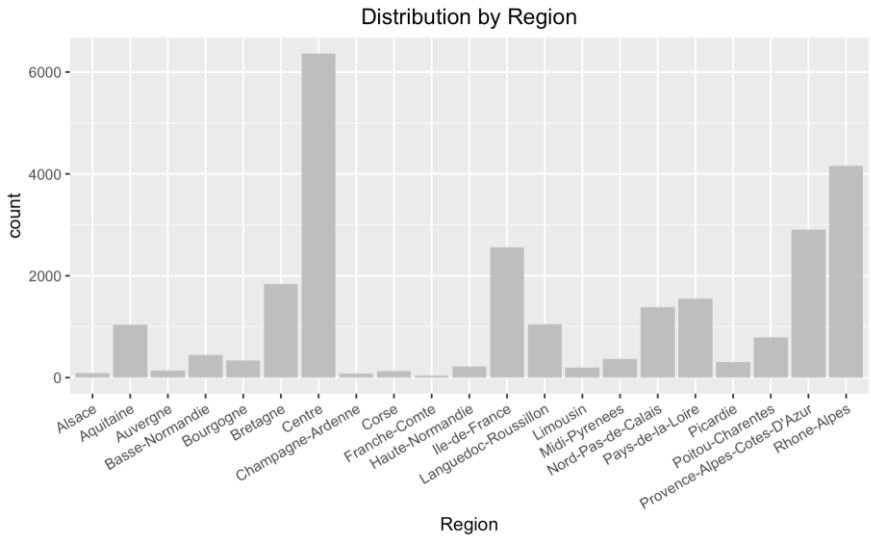


Fuente: *Elaboración Propia*

Con respecto a la variable Gas, La proporción de personas accidentadas que usan gas Regular y las que usan Diesel es la misma

Con respecto con la variable Area, podemos ver que la mayoría gente viven en Area C, D y E.

Figure 23: *Distribución de Region*



Fuente: *Elaboración Propia*

En los gráficos de las personas clientes que ha tenido accidentes, podemos ver que la mayoría gráficos tiene apariencias similares, por ejemplo, Gas, Region, VehPower, etc., algunos no por ejemplo la variable DrivAge. Por lo tanto, no los repetiremos aquí.

4.2.2 Multivariable

En esta sección, vamos a hacer el análisis univariante. En concreto, vamos a hacer el análisis por

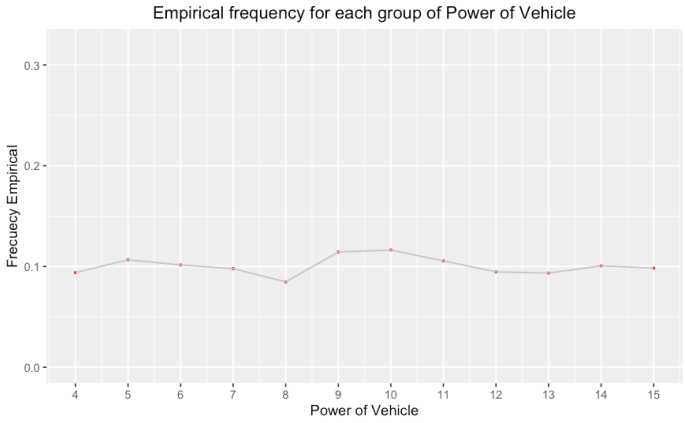
a) pintar el grafico entre variables contra sus frecuencias empírico
b) pintar y calcular correlación entre variables explicativos (ex: Coeficiente de correlación de Pearson para variables numérico y V- Cramer para variables categóricas)

Los gráficos de frecuencia empírico nos ayudan saber anticipadamente cuáles variables pueden ofrecer información en nuestro modelo. La correlación entre variables puede evitar el efecto de colinealidad.

Frecuencia

a) Frecuencias empíricas

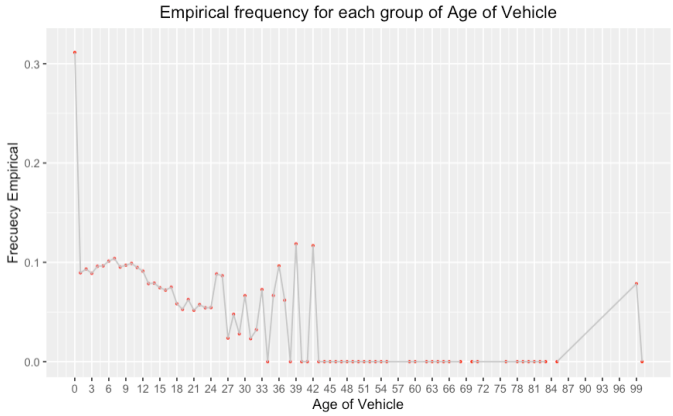
Figure 24: *Frecuencia Empirical de VehPower*



Fuente: *Elaboración Propia*

Podemos ver que la frecuencia de cada categoría de VehPower está por 10%. Veo que el riesgo de esta variable es más o menos homogéneo. La potencia 8 tiene riesgo más bajo entre todo.

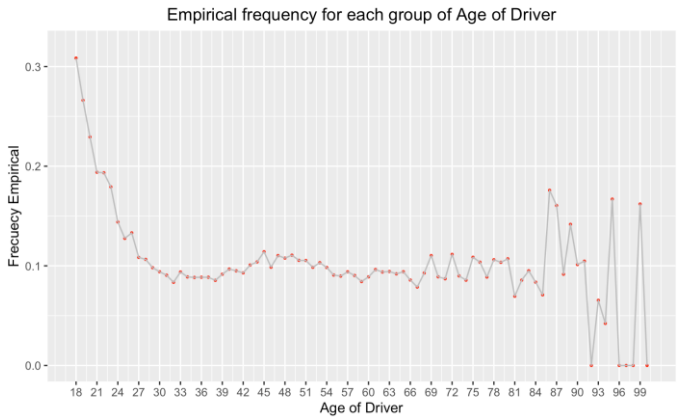
Figure 25: *Frecuencia Empirical de VehAge*



Fuente: *Elaboración Propia*

Podemos ver que el auto de 0 año de antigüedad tiene la mayor probabilidad de tener accidente. A medida que el coche envejece, las posibilidades de sufrir un accidente de tráfico disminuyen.

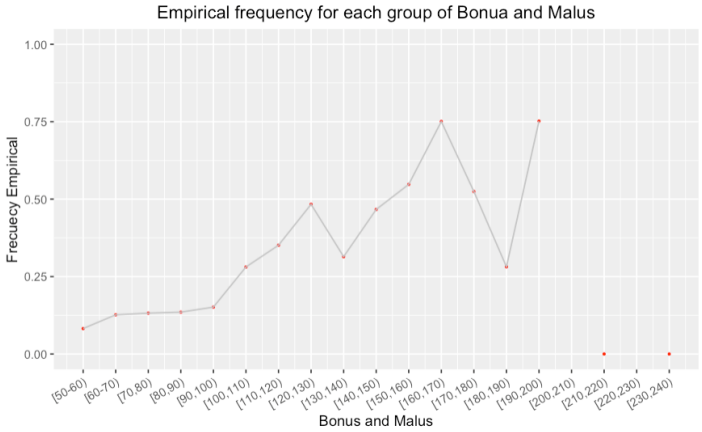
Figure 26: *Frecuencia Empirica de DrivAge*



Fuente: *Elaboración Propia*

Con respecto a la variable DrivAge, el conductor de edad 18-27 son los que son más posible tener accidente. Especialmente el conductor de 18 años tiene probabilidad mas alta de tener accidente.

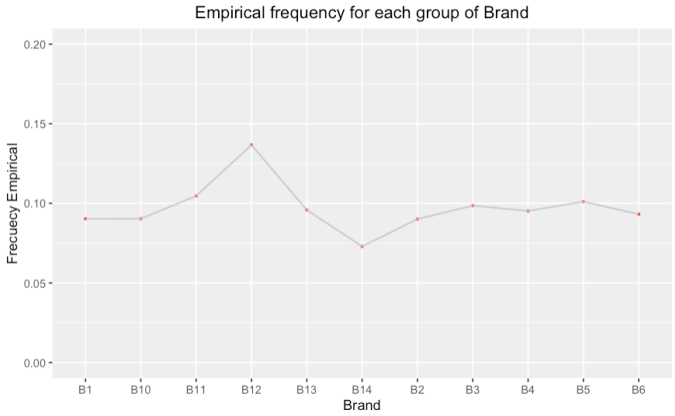
Figure 27: *Frcuencia empirica de BonusMalus*



Fuente: *Elaboración Propia*

Como hemos hablado antes, luego de el numero menos que 100 represente Bonus y mayor que 100 represente Malus. Esta muy clara que la gente tiene malus tiene mas probabilidad de tener accidentes que la gente tiene Bonus.

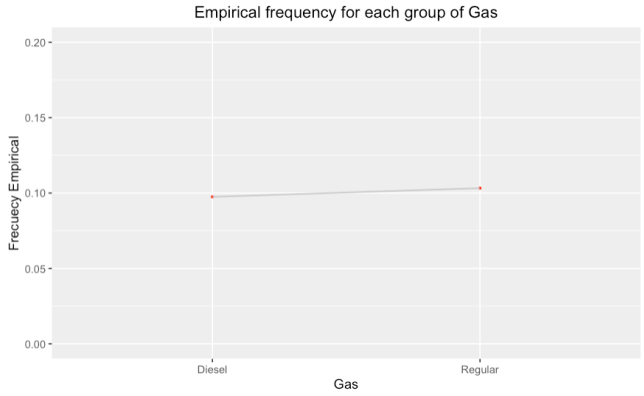
Figure 28: *Frecuencia Empirica de VehBrand*



Fuente: *Elaboración Propia*

Por mi parte, la variable VehBrand tiene riesgo más o menos homogéneo. Solo la categoría B12 tiene un poco más alta que otras.

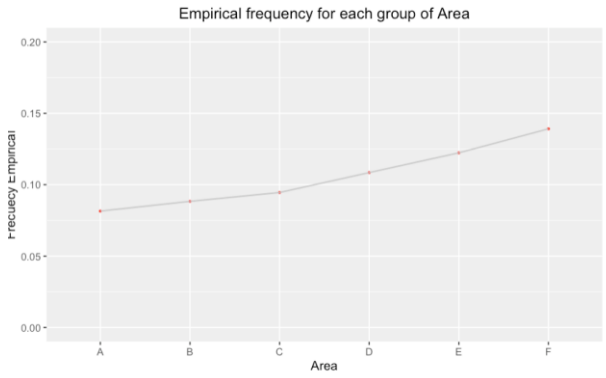
Figure 29: *Frecuencia Empirica de Gas*



Fuente: *Elaboración Propia*

Los niveles de frecuencia de Diesel y Regular están casi lo mismo.

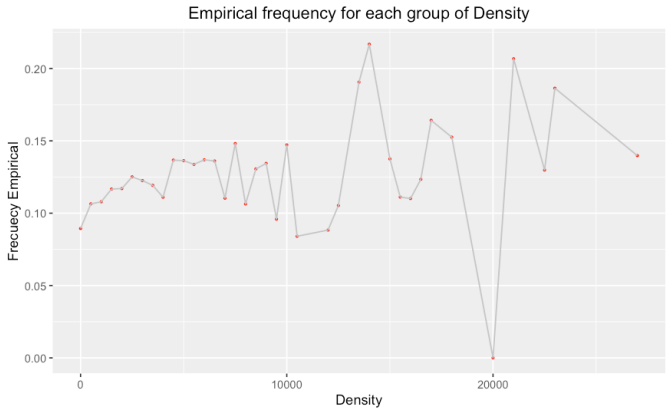
Figure 30: *Frcuencia Empirica de Area*



Fuente: *Elaboración Propia*

Obviamente, la Area menos urbano es menos posible tiene accidente.

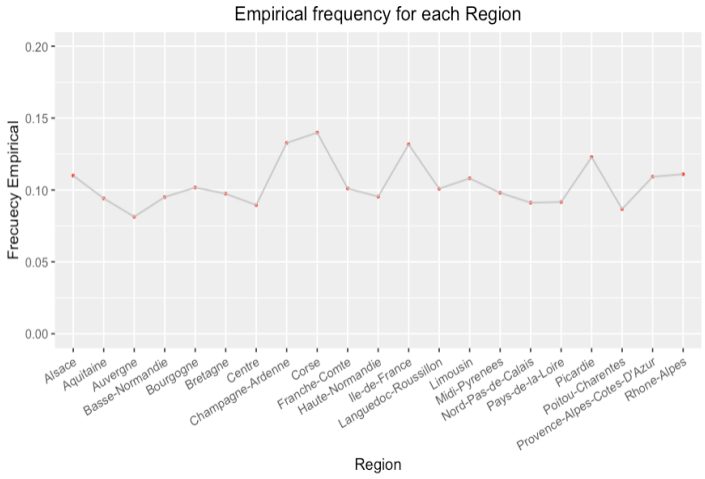
Figure 31: *Frcuencia Empirica de Density*



Fuente: *Elaboración Propia*

El patrón de esta variable es un poco complicado. Generalmente la mayor densidad es, lo más posible tiene accidente.

Figure 32: *Frecuencia Empirica de Rigion*



Fuente: *Elaboración Propia*

Los riesgos de cada región son más o menos similares. Solo la región Champagne-Ardenne, Corse y Ile-de-France son un poco más altos.

Por el grafico, veo que algunas variables tienen riesgos más o menos homogéneos, por ejemplo: **VehPower, VehBrand, VehGas**. Mientras que otras variables tienen un efecto significativo sobre la variable predictiva, por ejemplo: **VehAge, Region, Density, Area, BonusMalus.**

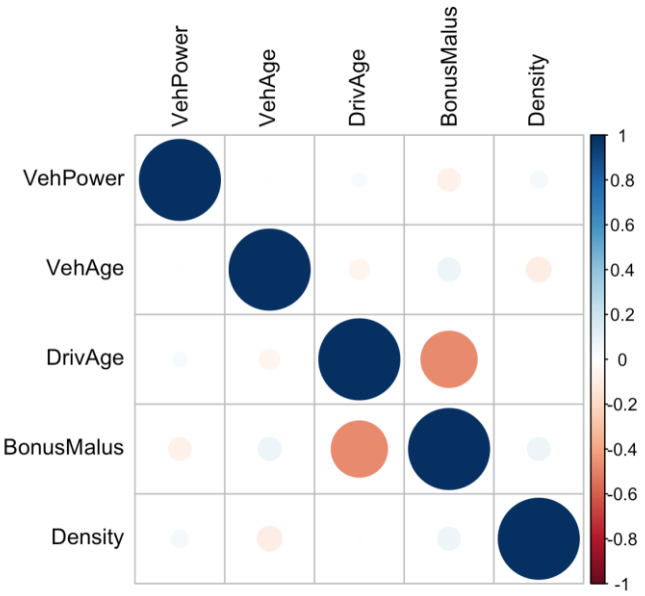
b) Correlación

- Variable Numérica (Coeficiente de correlación de Pearson)

Table 10: *correlación de Pearson para datos de frecuencias*

	VehPower	VehAge	DrivAge	BonusMalus	Density
VehPower	1	-0,005991856	0,030114417	-0,07590261	0,04289793
VehAge	-0,005991856	1	-0,059217402	0,07992853	-0,0904263
DrivAge	0,030114417	-0,059217402	1	-0,47996423	-0,0046978
BonusMalus	-0,075902615	0,079928531	-0,479964234	1	0,07771521
Density	0,042897925	-0,090426322	-0,004697845	0,07771521	1

Fuente: *Elaboración Propia*
Figure 33: *Correlación de Variables Numéricas*



Fuente: *Elaboración Propia*

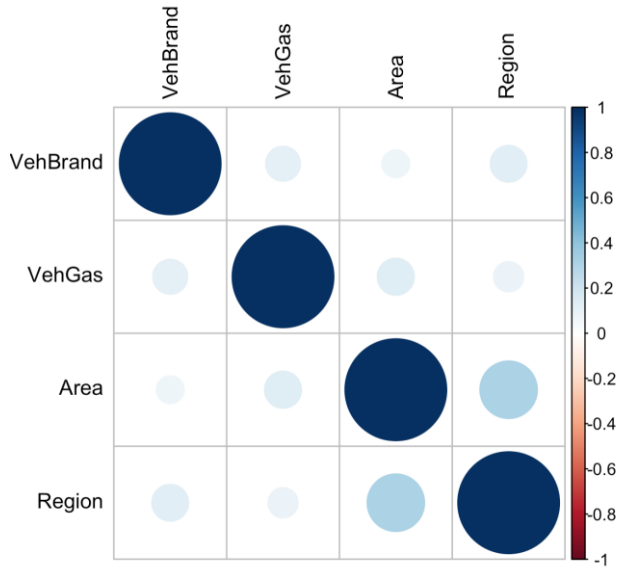
- Variable Categórica (V-Cramer)

Table 11: *coeficiente de V-Cramer para datos de frecuencias*

	VehBrand	VehGas	Area	Region
VehBrand	1	0,1165098	0,0732716	0,1297065
VehGas	0,1165098	1	0,1314338	0,0872388
Area	0,0732716	0,1314338	1	0,3184404
Region	0,1297065	0,0872388	0,3184404	1

Fuente: *Elaboración Propia*

Figure 34: *Correlación de Variables Categóricas*



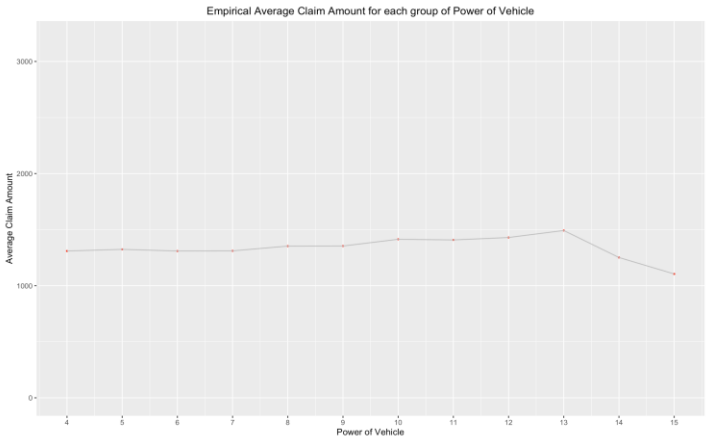
Fuente: *Elaboración Propia*

Con el resultado de medidas de correlación, podemos ver que las variables tienen poca correlación entre sí.

Severidad

a) severidades empíricas

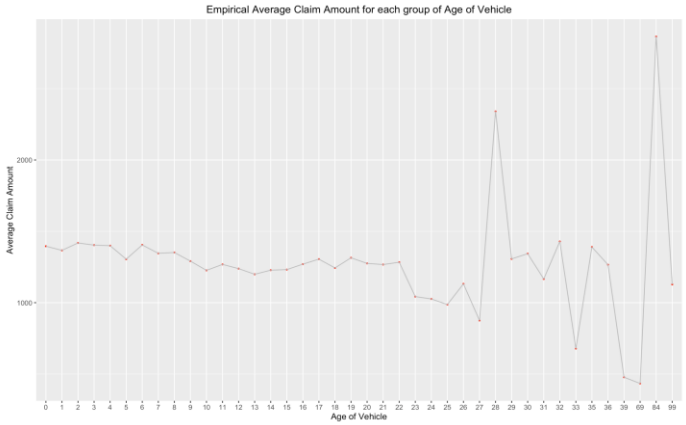
Figure 35: *Severidad Empirica de VehPower*



Fuente: *Elaboración Propia*

Podemos ver que desde 4 hasta 13, la severidad crecen. Pero no tiene muchas diferenicas entre sí.

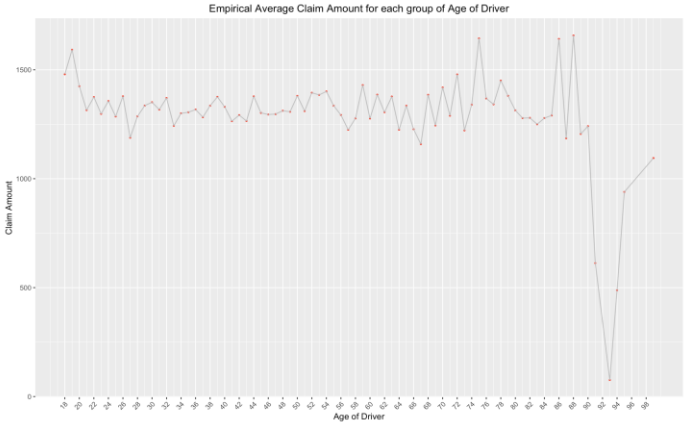
Figure 36: *Severidad Empirica de VehAge*



Fuente: *Elaboración Propia*

Parece que el VehAge mas antigua tiene una severidad mejor, pero luego de VehAge 27, la tendencia se queda mas complicada. Esto es por a nosotros nos falta observaciones.

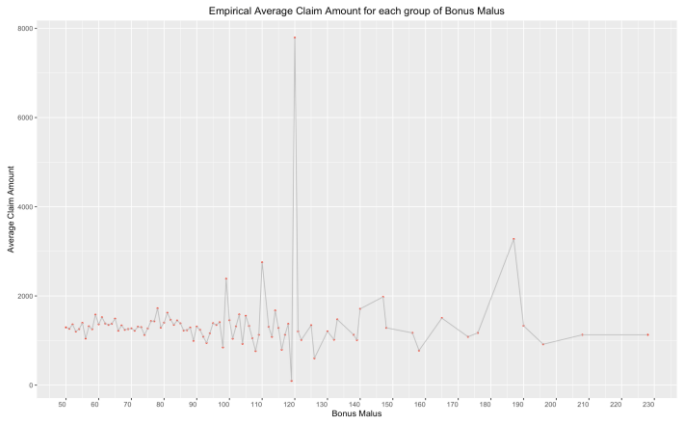
Figure 37: *Severidad Empirica de DrivAge*



Fuente: *Elaboración Propia*

Parece que el conductor más joven, tiene severidad mayor. Especialmente los conductores de 18-23 años.

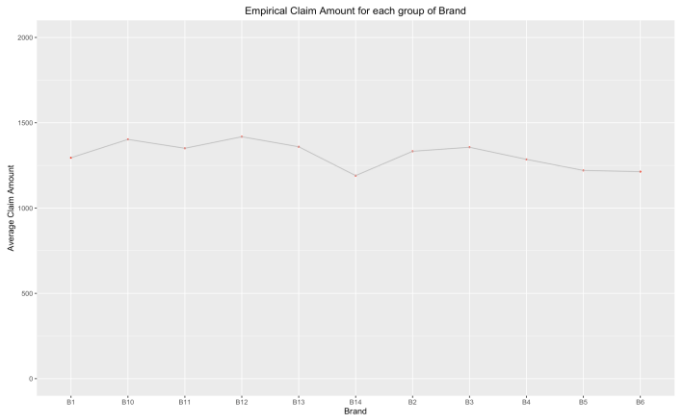
Figure 38: *Severidad Empírica de BonusMalus*



Fuente: *Elaboración Propia*

Las severidades es diferentes BonusMalus son casi mismos. No he visto ningunas tendencias.

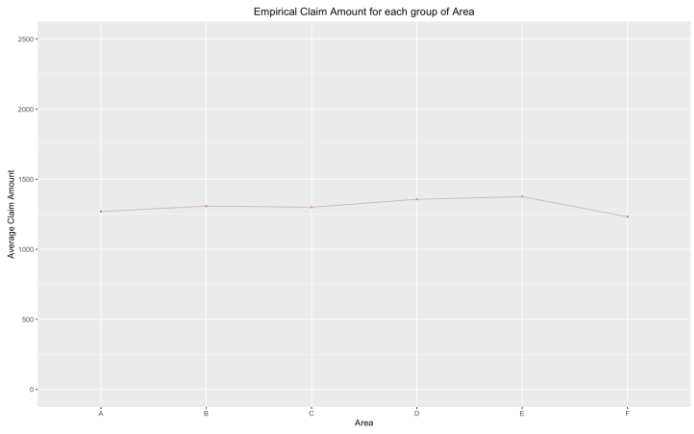
Figure 39: *Severidad Empírica de VehBrand*



Fuente: *Elaboración Propia*

Podemos que el B14 tiene mejor severidad, pero no tiene muchas diferencias entre sí.

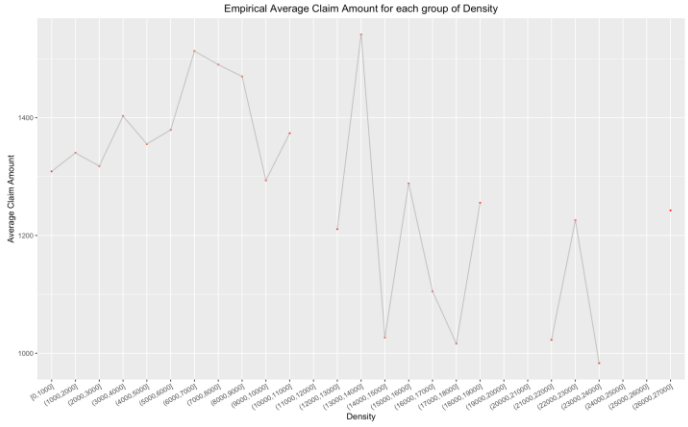
Figure 40: *Severidad Empírica de Area*



Fuente: *Elaboración Propia*

Las severidades de diferentes Areas son casi mismos.

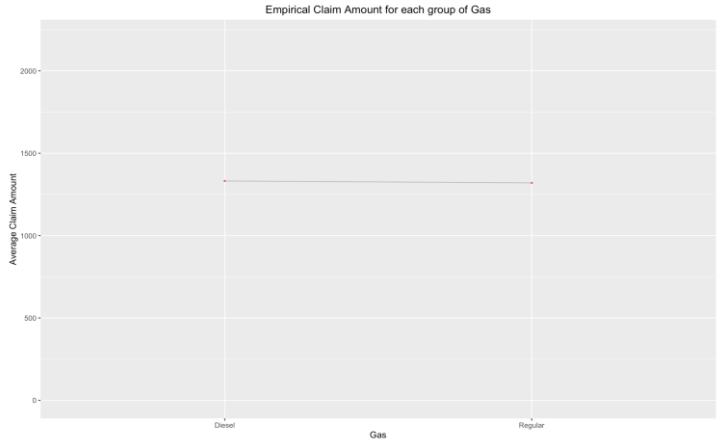
Figure 41: *Severidad Empírica de Density*



Fuente: *Elaboración Propia*

Los conductores viven de distritos con densidad desde 5000 hasta 9000 tiene severidad más altos.

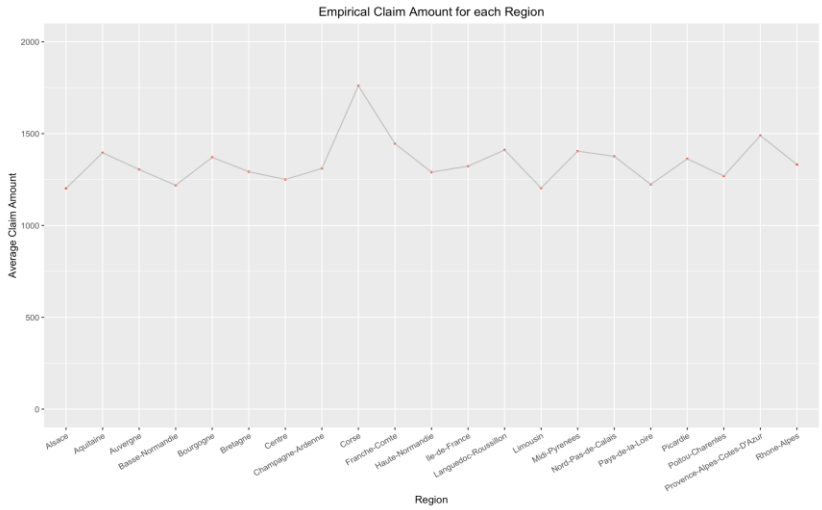
Figure 42: *Severidad Empírica de Gas*



Fuente: *Elaboración Propia*

Los conductores quienes usan Diesel tienen misma severidad que los usan Regular.

Figure 43: *Severidad Empirica de Region*



Fuente: *Elaboración Propia*

Podemos ver que la provincia Centre tiene mayor severidad que otras provincias.

Los importes de las reclamaciones también tienen dependencias marginales entre las características, y aquí representaremos todas de ellas. Pero, en comparación a la frecuencia empírica, la severidad empírica es menos heterogénea.

a) Correlación

A continuación, calculamos la Coeficiente de correlación de Pearson.

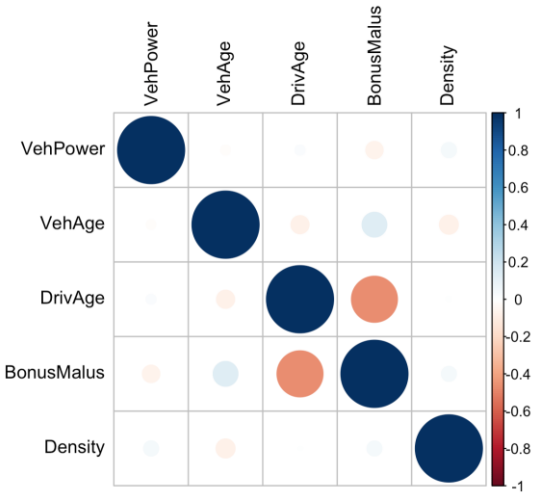
- Variable Numérica (Coeficiente de correlación de Pearson)

Table 12: *correlación de Pearson para datos de severidad*

	VehPower	VehAge	DrivAge	BonusMalus	Density
VehPower	1	-0,019574	0,021304	-0,066462	0,049858
VehAge	-0,019574	1	-0,072028	0,135412	-0,079448
DrivAge	0,021304	-0,072028	1	-0,468757	0,004064
BonusMalus	-0,066462	0,135412	-0,468757	1	0,048575
Density	0,049858	-0,079448	0,004064	0,048575	1

Fuente: *Elaboración Propia*

Figure 44: *Correlación de variables numéricos de ClaimAmount*



Fuente: *Elaboración Propia*

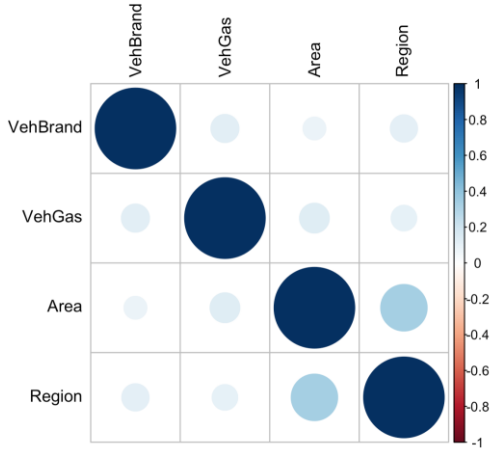
Para las variables numéricos, no tiene correlación entre sí.

- Variable Categórica (V-Cramer)

Table 13: *coeficiente de V-Cramer para datos de severidad*

	VehBrand	VehGas	Area	Region
VehBrand	1	0,121245	0,080007	0,114095
VehGas	0,121245	1	0,136304	0,100636
Area	0,080007	0,136304	1	0,331664
Region	0,114095	0,100636	0,331664	1

Fuente: *Elaboración Propia*
Figure 45: *Correlación de variables categóricas de Claim Amount*



Fuente: *Elaboración Propia*

Para las variables categóricas, tampoco tiene correlación entre sí.

4.3 Proceso de Modelización de GLM (R)

En esta sección, vamos a empezar la parte más importante de esta tesis, el proceso de modelización.

4.3.1 Mejor modelo de frecuencia

- Feature engineer

Para equilibrar la complejidad y la interpretabilidad del modelo, debemos realizar una ingeniería de rasgos para transformar las variables, es decir, combinar las variables para las que hay muy pocos datos teniendo en cuenta la homogeneidad del riesgo. La variable Gas no se modifica, los restos realizamos las siguientes transformaciones:

- VehPower: Separamos en Categoría [4,5], (5,8], (8, Inf].
- VehAge: Separamos en Categoría [0,1], (1,10], (10,20], (20, Inf].
- DrivAge: Separamos en Categoría [18,21], (21,26], (26,33], (33,40], (40,55], (55,70], (70, Inf].
- BonusMalus: Separamos en Categoría [50,100], (100,140], (140, Inf].
- Density: Separamos en Categoría [0,1500], (1500,10000], (10000, Inf].
- VehBrand: Agrupamos B11 y B12 como bran_a, los restos como bran_b.
- Region: Considerando balance de dato y riesgo homogéneo, agrupamos en los siguientes 3 ramos.

Table 14: *Region luego de simplificaciones*

region_a	Lorraine Ile-de-France Picardie Champagne-Ardenne Corse
region_b	Alsace Limousin Provence-Alpes-Cotes-D'Azur Rhone-Alpes Languedoc-Roussillon Bourgogne Franche-Comte
region_c	Midi-Pyrenees Bretagne Haute-Normandie Basse-Normandie Aquitaine Pays-de-la-Loire Nord-Pas-de-Calais Centre Poitou-Charentes Auvergne

Fuente: *Elaboración Propia*

- Modelización

En esta sección, comenzaremos por ajustar nuestro primer GLM para frecuencias utilizando la función GLM(...) del paquete básico de R. Concretamente, ajustaremos una regresión de Poisson.

Además, como hemos visto en la parte de EDA, podemos que ver que algunas variables cuyos riesgos no es muy heterogéneo. En este caso, meter variables así no es necesarios, a causa de que estas variables no solo no otorgan ninguno informaciones, sino también hacen el modelo más complejo.

Nuestro principal objetivo será seleccionar un modelo en el que

- todos los parámetros cumplan el siguiente nivel de significación

$$(\mathrm{P} > |\mathrm{Z}|) < 0.05$$

- mientras ofrecen más información y mientras no es demasiado complejo, vamos a medir por AIC:

$$AIC = 2p - 2l$$

Dentro del cual

$$p = \text{número de parámetros del modelo}$$
$$l = log - verosimilitud\ del\ modelo$$

Ahora vamos a empezar partiendo de un modelo solo con dos variables:

$$\frac{ClaimNb}{Exposure} \sim Car\ Age\ +\ Driver\ Age$$

y luego vamos a agregar diferentes combinaciones de variables hasta llegar los criterios hemos descrito arribas.

El resultado del modelo arriba es siguiente:

Table 15: *Modelo Inicial de Frecuencia de GLM*

```
Call:
glm(formula = ClaimNb ~ vehA_glm1 + dirA_glm1, family = poisson(),
    data = learn, offset = log(Exposure))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.8952  -0.4016  -0.2970  -0.1625   7.5194

Coefficients:
                  Estimate Std. Error z value            Pr(>|z|)
(Intercept)       -0.91463    0.04058  -22.54 <0.0000000000000002 ***
vehA_glm1(1,10]   -0.55734    0.01580  -35.28 <0.0000000000000002 ***
vehA_glm1(10,20]  -0.75262    0.01871  -40.22 <0.0000000000000002 ***
vehA_glm1(20,Inf] -1.13455    0.07395  -15.34 <0.0000000000000002 ***
dirA_glm1(21,26]  -0.46643    0.04530  -10.30 <0.0000000000000002 ***
dirA_glm1(26,33]  -0.90120    0.04214  -21.39 <0.0000000000000002 ***
dirA_glm1(33,40]  -0.98316    0.04163  -23.61 <0.0000000000000002 ***
dirA_glm1(40,55]  -0.85042    0.03967  -21.44 <0.0000000000000002 ***
dirA_glm1(55,70]  -1.00503    0.04127  -24.35 <0.0000000000000002 ***
dirA_glm1(70,Inf] -0.90075    0.04493  -20.05 <0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 156611  on 474603  degrees of freedom
Residual deviance: 154162  on 474594  degrees of freedom
AIC: 202538
```

Fuente: *Elaboración Propia*

Podemos ver que cada variable tiene una significancia bien fuerte. Esto significa que su información es muy fiable y la dependencia de ClaimNb/Exposrue con CarAge y DrivAge está bien fundada.

Ahora debemos agregar diferente combinación de variables en el modelo hasta llegar el menor AIC.

El mejor modelo de frecuencias es el siguiente:

Table 16: *Mejor Modelo de Frecuencia de GLM*

```
Call:
glm(formula = ClaimNb ~ vehA_glm1 + vehP_glm1 + dirA_glm1 + Bm_glm1 +
    bran_glm1 + Den_glm1 + region_glm1 + are_glm1, family = poisson(),
    data = learn, offset = log(Exposure))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5002  -0.3911  -0.2978  -0.1629   7.4860

Coefficients:
                    Estimate Std. Error z value             Pr(>|z|)
(Intercept)          -1.18880    0.04977 -23.884 < 0.0000000000000002 ***
vehA_glm1(1,10]      -0.49062    0.01692 -28.996 < 0.0000000000000002 ***
vehA_glm1(10,20]     -0.65399    0.02066 -31.660 < 0.0000000000000002 ***
vehA_glm1(20,Inf]    -1.02535    0.07467 -13.732 < 0.0000000000000002 ***
vehP_glm1(5,8]        0.03211    0.01392   2.306             0.021088 *
vehP_glm1(8,Inf]      0.12760    0.02001   6.377       0.000000000181 ***
dirA_glm1(21,26]     -0.43456    0.04539  -9.574 < 0.0000000000000002 ***
dirA_glm1(26,33]     -0.78218    0.04273 -18.306 < 0.0000000000000002 ***
dirA_glm1(33,40]     -0.84718    0.04245 -19.957 < 0.0000000000000002 ***
dirA_glm1(40,55]     -0.70154    0.04059 -17.283 < 0.0000000000000002 ***
dirA_glm1(55,70]     -0.85381    0.04218 -20.240 < 0.0000000000000002 ***
dirA_glm1(70,Inf]    -0.72145    0.04570 -15.788 < 0.0000000000000002 ***
Bm_glm1(100,140]      1.15774    0.03586  32.284 < 0.0000000000000002 ***
Bm_glm1(140,Inf]      1.57180    0.11849  13.265 < 0.0000000000000002 ***
bran_glm1bran_b      -0.10262    0.01661  -6.180       0.000000000641 ***
Den_glm1(1500,1e+04]  0.08799    0.03534   2.490             0.012780 *
Den_glm1(1e+04,Inf]   0.11604    0.05481   2.117             0.034246 *
region_glm1region_b  -0.02773    0.02322  -1.194             0.232441
region_glm1region_c  -0.08690    0.02331  -3.728             0.000193 ***
are_glm1B             0.06229    0.02587   2.408             0.016052 *
are_glm1C             0.11929    0.02097   5.688       0.000000012836 ***
are_glm1D             0.20715    0.02249   9.211 < 0.0000000000000002 ***
are_glm1EF            0.19147    0.04183   4.577       0.000004715744 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 156611  on 474603  degrees of freedom
Residual deviance: 152773  on 474581  degrees of freedom
AIC: 201175
```

Fuente: *Elaboración Propia*

Para más modelos del proceso de selección del mejor modelo, pueden ver el Anexo. Salvo el features engineer mencionado arriba, puede intentar más medidas tratar las variables. En este documento se han probado muchas otras combinaciones diferentes de métodos de caja dividida y el modelo anterior es el mejor que se puede derivar.

Además, con diferentes criterios puede tener diferentes mejores modelos.

4.2.2 Mejor modelo de severidad

En este capítulo, vamos a empezar el proceso de modelización de severidad.

En comparación a modelos de Frecuencia, es más difícil modelizar ‘severidad’. Este proceso de modelado es similar al del capítulo anterior, por lo que no lo repasaremos.

- Feature engineer

- VehPower: Separamos en Categoría [0,8], (8, Inf].
- VehAge: Separamos en Categoría [0,8], (8, Inf].
- DrivAge: Separamos en Categoría [18,23], (23, Inf].
- Density: Separamos en Categoría [0,3000], (3000,9000], (10000, Inf].
- VehBrand: Agrupamos B5, B6 y B12 como bran_a, los restos como bran_b.
- Region: Agrupamos "Provence-Alpes-Cotes-D'Azur", "Corse" como “region_h”, y las otras provincias como “region_l”.
- Area: Agrupamos el área E y D como “ED”.

- Modelización

Ahora empezamos el proceso de modelización de severidad.

Empezamos desde el modelo

$$ClaimAmount \sim Driver\ Age$$

El siguiente es el resultado de programa R.

Table 17: *Modelo Inicial de Severidad de GLM*

```
Call:
glm(formula = ClaimAmount ~ dirA_glm3, family = Gamma(link = "log"),
    data = learn)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.5172  -0.6096  -0.1523  -0.0898   3.0188

Coefficients:
                  Estimate Std. Error t value            Pr(>|t|)
(Intercept)        7.25240    0.02854 254.108 <0.0000000000000002 ***
dirA_glm3(23,Inf] -0.06787    0.02950  -2.301              0.0214 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Gamma family taken to be 0.9457102)

    Null deviance: 14605  on 18175  degrees of freedom
Residual deviance: 14600  on 18174  degrees of freedom
AIC: 296723

Number of Fisher Scoring iterations: 5
```

Fuente: *Elaboración Propia*

Como hemos analizado en el capítulo EDA, el DrivAge tiene afecto a severidad. Luego agregamos diferente variable desde el modelo básico hasta llegar el mejor resultado AIC con todas variables significativos.

El mejor modelo de severidad es el siguiente:

$$ClaimAmount \sim Brand + Area + Region + DrivAge$$

Table 18: *Mejor Modelo de Severidad de GLM*

```
Call:
glm(formula = ClaimAmount ~ bran_glm1 + are_glm1 + region_glm1 +
    dirA_glm3, family = Gamma(link = "log"), data = learn)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.5074  -0.6105  -0.1236  -0.0406   3.2511

Coefficients:
                    Estimate Std. Error t value             Pr(>|t|)
(Intercept)          7.29169    0.04555 160.069 < 0.0000000000000002 ***
bran_glm1bran_b      0.08091    0.02214   3.654             0.000259 ***
are_glm1B            0.02714    0.03044   0.891             0.372685
are_glm1C            0.01113    0.02452   0.454             0.649886
are_glm1DE           0.04257    0.02292   1.858             0.063247 .
are_glm1F           -0.04811    0.04545  -1.059             0.289804
region_glm1region_l -0.14551    0.02256  -6.449       0.000000000116 ***
dirA_glm3(23,Inf]   -0.07697    0.02920  -2.636             0.008395 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Gamma family taken to be 0.9247776)

    Null deviance: 14605  on 18175  degrees of freedom
Residual deviance: 14532  on 18168  degrees of freedom
AIC: 296640

Number of Fisher Scoring iterations: 5
```

Fuente: *Elaboración Propia*

Finalmente podemos que solo el VehBrand, Area, Region, y DrivAge tiene afecto a la severidad.

4.4 Proceso de Modelización de Redes Neurales (Python)

Ahora empezamos modelización de Redes Neuroles.

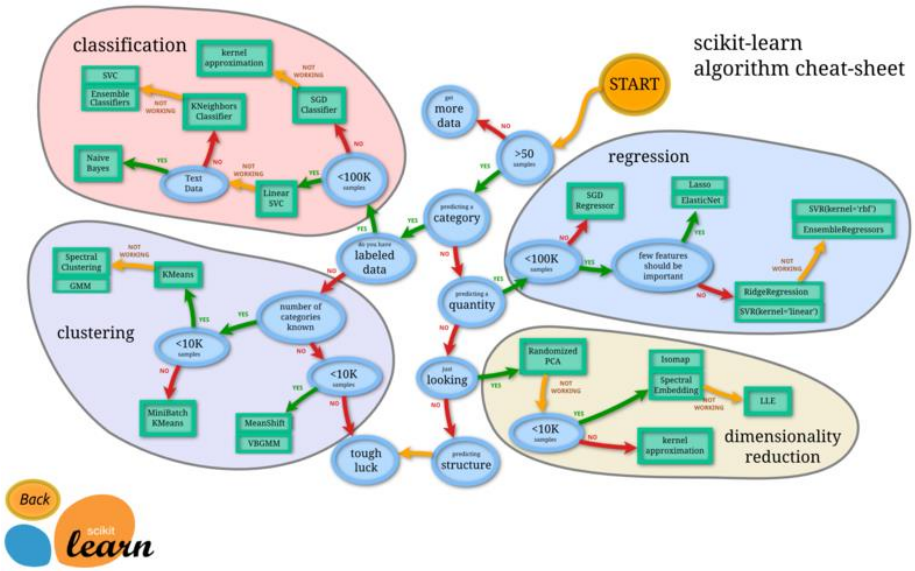
Esta parte vamos a utilizar el lenguaje Python, en concreto, vamos a utilizar el API **scikit-learn** y también el **Pytorch.**

El scikit-learn, también conocido como sklearn, es un conjunto de herramientas de aprendizaje automático de código abierto basado en el lenguaje python. Permite aplicaciones algorítmicas eficientes a través de bibliotecas de python para el cálculo numérico como NumPy, SciPy y Matplotlib, y cubre casi todos los principales algoritmos de aprendizaje automático.

Los módulos más utilizados en Sklearn son la clasificación, la regresión, la agrupación(cluster), la reducción de la dimensionalidad, la selección del modelo y el

preprocesamiento. La imagen siguiente ofrece un breve resumen de los principales paquetes de sklearn para diversas aplicaciones.

Figure 46: *Cheatsheet de Algorithm de Sklearn*



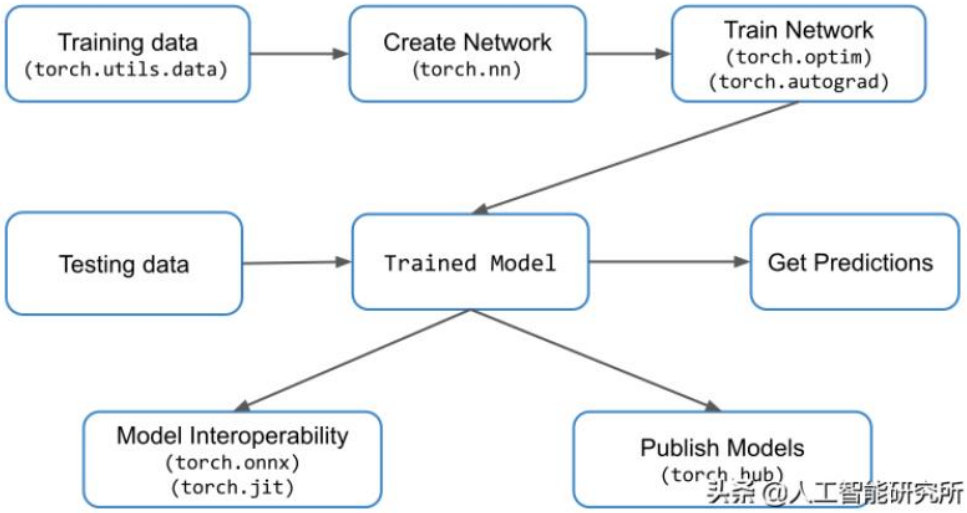
Fuente: *Scikit-learn*

PyTorch es un marco de aprendizaje automático de código abierto basado en la Torch, utilizado para aplicaciones como la visión por ordenador y el procesamiento del lenguaje natural (**natural language processing, NLP**), desarrollado principalmente por Meta AI. Es un software gratuito y de código abierto publicado bajo la licencia BSD modificada. Aunque la interfaz de Python está más pulida y es el principal foco de desarrollo, PyTorch también tiene una interfaz de C++.

Un número de piezas de software de aprendizaje profundo se construyen sobre PyTorch, incluyendo Tesla Autopilot, Uber's Pyro, Hugging Face's Transformers, PyTorch Lightning, y Catalyst.

PyTorch es particularmente popular entre la comunidad de investigadores debido a su naturaleza pitónica y a su facilidad de extensión (**es decir, la implementación de tipos de capas personalizadas, arquitecturas de red, etc.**) PyTorch utiliza Tensor como su estructura de datos principal, similar a una matriz Numpy. Con el software y el hardware adecuados, los tensores pueden acelerar una gran variedad de operaciones matemáticas. A la hora de realizar estas operaciones en grandes cantidades en el aprendizaje profundo, la velocidad marca una gran diferencia. pyTorch, al igual que Python, se centra en la facilidad de uso y permite que incluso usuarios con conocimientos muy básicos de programación puedan utilizar el aprendizaje profundo en sus proyectos.

Figure 47: *Flujo de trabajo típico y módulos importantes asociados a cada paso*



Fuente: *Pytorch*

4.4.1 Pasos de la formación y conceptos básicos

Como principiante en el aprendizaje automático, uno podría pensar que el aprendizaje automático es similar a los modelos estadísticos. Esto es normal, ya que el aprendizaje automático ha evolucionado a partir de la estadística. Pero también hay algunos conceptos únicos que son específicos del aprendizaje automático. Aquí haremos una breve introducción con respecto a training de Redes Neurales y proceso de training de un modelo de machine learning.

Proceso general y pasos necesarios para las tareas de aprendizaje automático son los siguientes:

- 1, Recogida de datos
- 2, Preprocesamiento de datos e ingeniería de características
  - 2.1 Limpieza de datos
  - 2.2 Integración de datos
  - 2.3 Estatuto de los datos
  - 2.4 Transformación de datos
- 3, Selección y entrenamiento de modelos
- 4, Evaluación y optimización del modelo

El proceso de entrenamiento es muy similar al de los modelos estadísticos, pero hay varios conceptos que son específicos del aprendizaje automático, y ofrecemos una breve introducción a estos conceptos.

Hiperparámetros： Los hiperparámetros son relativos a los parámetros normales. En este trabajo, el número de capas ocultas y el número de neuronas por capa son los hiperparámetros, mientras que los pesos entrenados, las desviaciones son los parámetros.

Los hiperparámetros no pueden obtenerse a partir del modelo de entrenamiento, es decir, los datos de entrenamiento no contienen información sobre los hiperparámetros, por lo que la selección y el ajuste de los hiperparámetros se convierten en el centro de la investigación.

En la optimización de hiperparámetros, los principales métodos son: grid search, también conocida como exploración de parámetros, que es el método tradicional de optimización de hiperparámetros, y simplemente realiza una búsqueda exhaustiva de un subconjunto del modelo especificado manualmente; la búsqueda aleatoria(random search), que es una búsqueda aleatoria que simplemente muestrea un número fijo de ajustes de parámetros en un espacio de alta dimensión; y la optimización bayesiana, que es un método ruidoso y sin ruido. La optimización bayesiana (Bayesian optimizatio), que es un método de optimización global de funciones ruidosas de caja negra; y así sucesivamente.

Estandarización y normalización de los datos: Tenemos que hacer Transformación de datos en el paso 2, Preprocesamiento de datos e ingeniería de características. En un sistema de evaluación multiindicador, debido a la diferente naturaleza de cada indicador de evaluación, suelen tener diferentes niveles y órdenes de magnitud. Cuando el nivel de cada indicador es muy diferente, si el análisis se realiza directamente con los valores brutos de los indicadores, destacará el papel de los indicadores con valores más altos en el análisis global y debilitará relativamente el papel de los indicadores con niveles de valor más bajos. Por lo tanto, para garantizar la fiabilidad de los resultados, es necesario normalizar los datos brutos de los indicadores.

La normalización de los datos consiste en escalarlos para que entren en un intervalo pequeño y específico. Se suele utilizar en el tratamiento de ciertos indicadores comparativos y evaluativos para eliminar las restricciones de unidades de los datos y convertirlos en un valor puro y sin dimensiones que permita comparar y ponderar indicadores de diferentes unidades o magnitudes.

Los siguientes son algunos de los métodos más comunes：

- min-max normalization

$$x^* = \frac{x - \min}{\max - \min}$$

- zero-mena normalization

$$x^* = \frac{x - \bar{x}}{\sigma}$$

Encoding: En machine learning, la característica de categoría es un tipo de variable muy común. En el paso 2, hay que hacer encoding para la variable categórica.

- Label Encoding

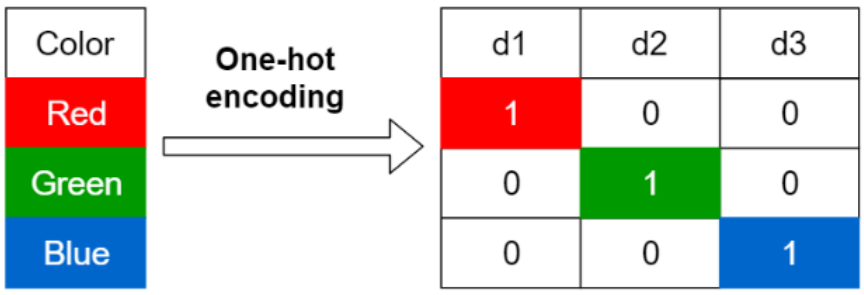
La llamada codificación dura significa que las características de la categoría se asignan directamente a los valores, con tantas búsquedas de categorías como valores. Este tipo de codificación dura es simple, brutal y conveniente. Sin embargo, sólo es bueno cuando los valores tomados dentro de las características de la categoría están ordenados, es decir, hay una relación secuencial obvia entre los valores tomados por las características de la categoría, por ejemplo, si las características de la educación son la escuela secundaria, la licenciatura, el máster y el doctorado, hay una relación secuencial obvia entre cada educación.

- One-hot encoding

One-hot encoding es probablemente la forma más utilizada para codificar las características de la categoría. Suponiendo que una variable de categoría tiene m valores de categoría, podemos convertirlo en m variables binarias mediante la codificación One-hot, cada uno de los cuales debe tomar el valor de categoría.

La codificación en caliente es útil cuando no hay un orden intrínseco obvio de los valores dentro de las características de la categoría, es decir, cuando la hard encoding no es aplicable. Sin embargo, cuando las características de la categoría toman demasiados valores, la codificación One-hot puede causar fácilmente desastres dimensionales, especialmente para las características basadas en el texto, que son básicamente un mar de ceros si se codifican utilizando la codificación One-hot. Por lo tanto, el método One-hot puede utilizarse para codificar categorías cuando los rasgos de la categoría no están ordenados y el número de rasgos tomados es inferior a cinco.

Figure 48: *One hot Encoding*

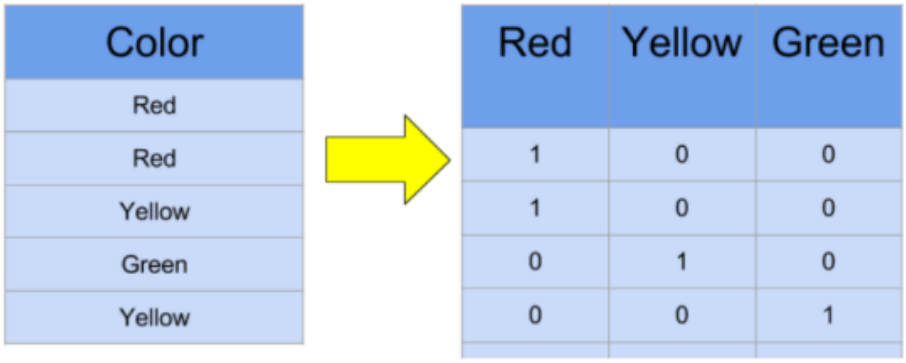


Fuente: *How to Handle Variables in Components. Zhihu*

- Dummy encoding

Ten cuidado, one hot encoding no es lo mismo que dummy encoding. Cuando una variable tiene k valores de atributos, la codificación ficticia tiene k-1 características binarias, la categoría base será ignorada, y si la categoría base no se elige razonablemente, todavía hay covarianza, y la categoría del plural se sugiere como la categoría base.

Figure 49: *Dummy encoding*



Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

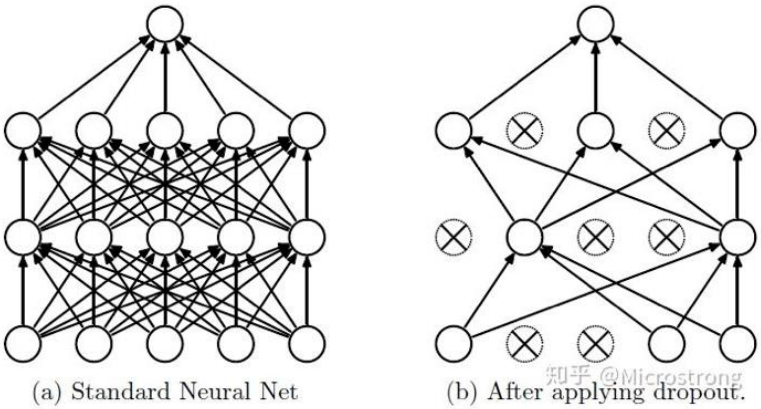
Fuente: *How to Handle Variables in Components. Zhihu*

En la sección GLM de este artículo no se realiza ningún proceso de codificación adicional, ya que la función puede ser codificada automáticamente. Pero en la red neuronal tenemos que codificarla nosotros mismos, así que utilizamos OneHotEncoderde sklearn.preprocessing modulo.

Dropout: El dropout es simplemente esto: dejamos que el valor de activación de una determinada neurona deje de funcionar con una determinada probabilidad p durante la propagación hacia delante, lo que hace que el modelo sea más generalizable porque no depende demasiado de determinadas características locales.

El dropout puede utilizarse como una alternativa para el entrenamiento de redes neuronales profundas. El sobreajuste puede reducirse significativamente ignorando la mitad de los detectores de características en cada lote de entrenamiento (dejando la mitad de los nodos de la capa oculta con un valor de 0). Este enfoque reduce las interacciones entre los detectores de características (nodos de la capa oculta), donde las interacciones de los detectores son aquellas en las que unos detectores dependen de otros para funcionar.

Figure 50: *Drop out*



Fuente: *Explanation of Dropout Principle in Deep Learning. Zhihu.*

Epoch, Batch Size y Iterations: En primer lugar, dejemos claro que sólo utilizamos estos términos cuando el conjunto de datos es tan grande que no podemos meter todos los datos en el ordenador de una vez para aprender. Cuando el conjunto de datos es muy grande, no podemos transferir todos los datos al ordenador a la vez, así que para superar este problema tenemos que dividir el conjunto de datos en subconjuntos de menor tamaño y dar estos subconjuntos al ordenador para que los calcule uno tras otro. Una vez calculado cada subconjunto, el ordenador actualiza los pesos de la red neuronal en consecuencia y permite ajustar el modelo al conjunto de datos. El número de iteraciones(**iteration)** es el número de batch que hay que calcular para completar una época. Iteraciones.

Tenga en cuenta que el número de lotes es el número de iteraciones para 1 época.

Una época (**Epoch**) = el proceso de actualización de los parámetros después de todos los datos de entrenamiento hacia adelante+hacia atrás.

Una iteración (**Iteration**) = [**Batch Size]** de datos de entrenamiento hacia adelante y hacia atrás y lego actualizar los parámetros.

Por ejemplo, supongamos que tenemos un conjunto de datos de 2000 muestras de entrenamiento. Podemos dividir estas 2000 muestras en 4 lotes, cada uno de ellos con 500 muestras. Cuando terminamos de entrenar 4 lteraciones, hemos completado 1 epoch de entrenamiento.

Aquí, el tamaño del batch (**batch size**) es 500, las iteraciones/batch son 4 y la época es 1.

4.4.2 Mejor modelo de frecuencia.

En este trabajo, los datos utilizados en R y los números aleatorios muestreados de los datos generados en R se exportan a Python para mantener la consistencia en los datos de entrenamiento.

En este trabajo, nos centramos en el número de neuronas de la capa oculta y descubrimos que el número de capas tiene muy poco efecto en la eficacia del modelo durante el proceso de entrenamiento. Además, se comprobó que más de 100 neuronas tendrían un efecto negativo en el modelo, y los mejores resultados se obtuvieron con un recuento de neuronas de alrededor de 30.

Además, hay que tener en cuenta que tanto la frecuencia como la cantidad de pérdida deben ser positivas, por lo que se aplicó una función softplus a la última capa del modelo como función de activación.

Desgraciadamente, debido a la gran cantidad de datos de entrenamiento y al limitado rendimiento del ordenador, se abandonó GridSearch basada en el principio de validación cruzada en favor de una búsqueda manual.

Nuestros mejores resultados se muestran en la siguiente tabla, un red con 2 hidden layers, cada capa se aplica un Relu función.

Table 19: *Mejor Modelo de Frecuencia de Redes Neurales*

num de hidden layer	Nodes de Hidden layer	activition	Dropout	Batchnorrn
2	n_hidden1 = 20 n_hidden2 = 10	Relu Relu	si	si

Fuente: *Elaboración Propia*

Además, el modelo se evalúa por la función/medida MSEloss, es decir mean squared error.

Table 20: *MSE del Redes Neurales de Frecuencia*

```
print('Last iteration loss value: '+str(loss.item()))
Last iteration loss value: 0.21333862841129303
```

Fuente: *Elaboración Propia*

Para mejorar la velocidad de entrenamiento, hemos puesto 64 bathches, learning_rate 0,001

4.4.3 Mejor modelo de severidad

Dado que tenemos menos datos en importes de sinestros, el proceso de entrenamiento de severidad es más rápido. Pero como hemos visto en el EDA de datos de severidad, solo una variable tiene riesgos no homogéneos, la modelización no es nada fácil. Luego de cambiar varias combinaciones de capas y diferentes cantidades nodos en el hidden layers. El mejor modelo es el siguiente:

En comparación al modelo de frecuencia, lo de

Table 21:*Mejor Modelo de Severidad de Redes Neurales*

num de hidden layer	Nodes de Hidden layer	activition	Dropout	Batchnorrn
2	n_hidden1 = 17 n_hidden2 = 40	Relu Relu	si	si

Fuente: *Elaboración Propia*

Y su MSELoss es

Table 22: *MSE del Redes Neurales de Severidad*

```
print('Last iteration loss value: '+str(loss.item()))
Last iteration loss value: 5639313.5
```

Fuente: *Elaboración Propia*

4.5 Comparación de resultados predicción de GLM y redes neurales

Recordemos ahora todo el proceso. Primero seleccionamos el mejor glm con aic y luego la mejor red neuronal con mse, y ahora comparamos los resultados de predicción del glm y de la red neuronal.

Hay muchas formas de hacerlo en la etapa de comparación de modelos, como se muestra en la tabla siguiente. Pero la que elegimos es MSE (mean squared error).

Table 23: *Métricas de evaluación del aprendizaje automático*

Regression	Classification	Recommender System
• Mean Absolute Error (MAE) • Root Mean Squared Error (RMSE) • R-Squared and Adjusted R-Squared	• Recall • Precision • F1-Score • Accuracy • Area Under the Curve (AUC)	• Mean Reciprocal Rank • Root Mean Squared Error (RMSE)

Fuente: *tricas de evaluación del aprendizaje automático. Zhihu*

Aquí tiene la expresión matemática de MSE (mean squared error)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2$$

Frecuencia

Figure 51: *MSE de Frecuencia de GLM*

```
```{r}
mean((test$ClaimNb - pred)^2)
```

[1] 0.05627427
```

Fuente: *Elaboración Propia*

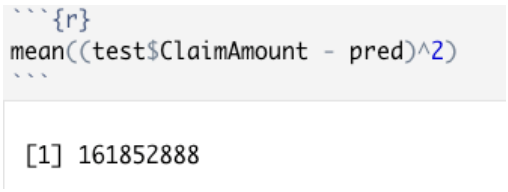
Figure 52: *MSE de Frecuencia de Redes Neurales*

```
metrics.mean_squared_error(pre_exposure, ori_exposure)
0.07573979291353194
```

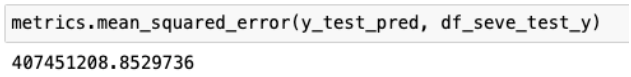
Fuente: *Elaboración Propia*

Severidad

Figure 53:*MSE de Severidad de GLM*



Fuente: *Elaboración Propia*
Figure 54: *MSE de Severidad de Redes Neurales*



Fuente: *Elaboración Propia*

Es muy lamentable que nuestro modelo de red neuronal no prediga tan bien como el modelo GLM tradicional tanto en los componentes de frecuencia como de valor de pérdida.

Las razones de esto podrían ser que no realizamos una validación cruzada durante nuestro proceso de entrenamiento para evitar problemas de sobreajuste. En particular, en el modelo de severidad podemos ver que el MSE de la parte de entrenamiento es superior.

5. Memorias de modelado Y retroalimentación sobre el uso de R y Python

En la parte de redes neuronales del proceso de modelización de esta tesis, sólo se presentan los resultados de la parte de Pytorch y se comparan con los resultados de GLM. Sin embargo, en el proceso real construí el modelo tres veces, y Pytorch fue el único que era factible y predecible. Los otros dos fallos se describirán aquí, espero que con algunas pistas para los que hayan leído este documento, y el código completo se puede encontrar en el apéndice.

Los otros dos procesos y lenguajes de modelización son:

- Python: scikit-learn, MPLegresor()

- R: neuralnet, nueralnet ()

A continuación, voy a explicar las causas de los fallos.

- Python, Scikit-learn: incapaz de restringir los valores de salida.

Antes de empezar el trabajo, habría pensado que modelar una red neuronal sería un proceso maravillosamente suave, y lo fue al principio, con el modelo construido en

sklearn devolviendo un mejor MSE que glm, hasta que descubrí que mi mejor modelo con MPLRegresor y GridSearchCV devolvía una predicción negativa. Después de comprobar la configuración de los hiperparámetros, no he encontrado ningún error en la configuración. Las funciones ReLu y SoftPlus no deberían devolver algunos resultados negativos.

Figure 55: *Predicho negativo*

```
[168] predictions = mlp_reg.predict(df_freq_test_x)
```

```
predictions
```

```
array([-0.00500885, -0.00500885, -0.00500885, ..., -0.00500885,
       -0.00500885, -0.00500885])
```

Fuente: *Elaboración Propia*

Después de releer la documentación de sklearn y buscar, he encontrado que la salida por defecto de esta función es la 'identity' y no hay ningún parámetro que cambiar. La siguiente imagen muestra la parte del código fuente de la configuración.

Figure 56: *Source code de MPLRegresor*

```
327         # Compute the number of layers
328         self.n_layers_ = len(layer_units)
329
330         # Output for regression
331         if not is_classifier(self):
332             self.out_activation_ = "identity"
333         # Output for multi class
334         elif self._label_binarizer.y_type_ == "multiclass":
335             self.out_activation_ = "softmax"
336         # Output for binary class and multi-label
337         else:
```

Fuente: *Source code de sklearn*

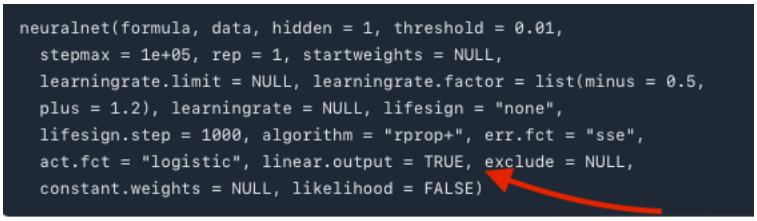
Solución:

1. Método 1: Modificar el código fuente
2. Método 2: Mapear el valor predicho a todo el campo real utilizando la función de retorno de la función de activación que se desea utilizar y luego hacer una predicción. Sin embargo, este método causará algunas pérdidas de datos, por ejemplo, producirá un infinito positivo o negativo, como $log(0)$. Claro puede modificar por método matemática, como agregar 0 un valor super pequeño para evitar infinito mapeo, o eliminar datos.

Por lo tanto, no creo que Sklearn.MPLRegresor() sea adecuado para utilizarlo en la modelización de los precios de los seguros.

- R, muy largo proceso de entrenamiento

La función neuralnet () en r tiene parámetros que se pueden establecer. Hice la construcción de la red neuronal en r, pero con el fondo de 400.000 datos de entrenamiento y el ordenador funcionando a altas frecuencias, no hubo manera de llegar a un modelo en absoluto, a pesar de numerosos intentos.



Fuente: *Paquete R*

Aunque es posible predecir en Pytorch, el proceso de entrenamiento también es extremadamente difícil. El proceso de entrenamiento dura más de 10 minutos cada vez. Esto hace que el proceso de ajuste de parámetros sea doloroso.

Hay tantos rumores maravillosos sobre los algoritmos de aprendizaje automático: que no necesitan hacer ingeniería de características, que algunos algoritmos no son sensibles a valores outliers, por lo que no es necesario examinar los valores atípicos, y que si se tiene la paciencia de hacerlo unas cuantas veces más, se obtendrán buenos resultados.

Por eso tuve una idea maravillosa sobre el modelado de seguros. Utilice los métodos tradicionales para determinar la importancia de los parámetros y utilice los métodos de aprendizaje automático para obtener predicciones más precisas. Pero al final del modelado tuve una idea diferente.

El aprendizaje automático no es necesariamente aplicable a todas las situaciones y no tiene necesariamente una capacidad de capa de pescado mejor que nuestras soluciones existentes. Imponer el aprendizaje automático en áreas en las que algunos modelos estadísticos tradicionales ya están muy bien establecidos requiere un ordenador suficientemente potente con cierta base matemática y una buena comprensión del dominio al que se aplica en primer lugar. De lo contrario, producirá modelos que, aunque tengan una alta puntuación, carecen de sentido, como hemos encontrado en Sklearn.

Al mismo tiempo, creo que el aprendizaje automático tiene un largo camino que recorrer antes de generalizarse, porque todos los procesos actuales de ajuste de parámetros son ciegos y no estructurados, sin apoyo teórico. Sin un conocimiento suficiente de los datos, los modelos resultantes son demasiado buenos para saber por qué son correctos.

Bibliografía

Ahlgren, M. (2018). Claims Reserving using Gradient Boosting and Generalized Linear Models. KTH, Matematisk statistik.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An Introduction to Statistical Learning with Applications in R* (8th ed.). Springer New York Heidelberg Dordrecht London.

Chambers, J. M., Hothorn, T., Lan, D. temple, & Wickham, H. (2015). *Computational Actuarial Science with R* (1st ed.). University of Québec at Montreal Canada.

Denuit, M., Hainaut, D., & Trufin, J. (2019). *Effective Statistical Learning Methods for Actuaries I*. Springer International Publishing AG.

Noll, A., Salzmann, R., & Wuthrich, M. V. (2015*). Case Study: French Motor Third-Party Liability Claims*. SSRN Electronic Journal

Martínez de lizarduy kostornichenko, V. (2021, June 14). *Comparative Performance Analysis between Grandient Boosting Models and GLMs for Non-Life Pricing*. Bibliotecta UC3M. https://e-archivo.uc3m.es/handle/10016/33732

Zhang, Z. (2018). *Genaralized Lineal Model*. Github. https://zhangzhenhu.github.io/blog/glm/source/index.html

N1k31t4. (2018, May 23). *MLPRegressor Output Range*. StackExchange. https://datascience.stackexchange.com/questions/31957/mlpregressor-output-range

Laradji, I. H., Mueller, A., & Qian, J. (2022, February 19). *Source Code: Scikit-Learn/Sklearn/Neural_network/_multilayer_perceptron.Py*. Github. https://github.com/scikit-learn/scikit-learn/blob/baf0ea25d/sklearn/neural_network/_multilayer_perceptron.py#L1266

Nanfangchouchangke. (2021, June 30). *Deep Learning---Backpropation*. CSDN. https://blog.csdn.net/johnny_love_1968/article/details/117598649?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216572703481678224852596 7%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=165727034816782248525967&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-117598649-null-null.142%5Ev32%5Epc_rank_34,185%5Ev2%5Etag_show&utm_term=%E5%8F%8D%E5%90%91%E4%BC%A0%E6%92%AD&spm=1018.2226.3001.41

Rothman, A. (2021, August 19). *GLMs Part III: Deep Neural Networks as Recursive Generalized Linear Models*. Towardsdatascience. https://towardsdatascience.com/glms-part-iii-deep-neural-networks-as-recursive-generalized-linear-models-ccb02817c9b5

Provincia(Francia). (2019, January 19). Wikipedia. https://zh.wikipedia.org/wiki/%E5%A4%A7%E5%8C%BA_(%E6%B3%95%E5%9B%BD)

Package 'CASdatasets.' (2020, December 11). UQAM. http://cas.uqam.ca/pub/web/CASdatasets-manual.pdf

Samuel1500. (2019, April 16). *R vs Python Para El Aprendizaje Automático*. Wordpress. https://stips.wordpress.com/2019/04/16/r-vs-python-para-el-aprendizaje-automatico/

Quincylk. (2019, January 9). *Example of Building a Simple Neural Network on PyTorch to Implement Regression and Classification*. CSDN. https://blog.csdn.net/Quincylk/article/details/86183752

Quincylk. (2019, January 9). *Example of Building a Simple Neural Network on PyTorch to Implement Regression and Classification*. CSDN. https://blog.csdn.net/Quincylk/article/details/86183752

Hong, L. (2020, June 25). *Summary of <Learning from Data>*. http://column.hongliangjie.com/读书笔记,机器学习,教材/2020/06/25/learning-from-data/

What Is ANN? (n.d.). TIBCO. https://www.tibco.com/zh-hans/reference-center/what-is-a-neural-network

bbuf. (2019, December 6). *Convolutional Neural Network Learning Route (I) | Components of a Convolutional Neural Network and How the Convolutional Layers Work in an Image?* 知乎. https://zhuanlan.zhihu.com/p/95758175

Li. (2022, June 3). *[Fundamentals of Deep Learning] Design of Three-Layer Neural Networks*. Icode. https://icode.best/i/26506248114056

Li, W. (2020, April 15). *How to Handle Variables in Components [with Python Code]*. Zhihu. https://zhuanlan.zhihu.com/p/130809720

Li, M. (2018, June 19). *Explanation of Dropout Principle in Deep Learning*. Zhihu. https://zhuanlan.zhihu.com/p/38200980

Natiedaxia. (2020, August 9). *Pytorch BatchNorm Parameters in Detail, the Calculation Process*. CSDN. https://blog.csdn.net/weixin_39228381/article/details/107896863

Zhaijiadexiaowei. (2022, June 5). *Pytorch: Fully Connected Neural Network - MLP Regression*. CSDN. https://blog.csdn.net/weixin_44979150/article/details/122778521

Zhao, X. (2022, June 5). *PyTorch Notes 2: Neural Networks for Regression Fitting to Data*. TOP. http://zhaoxuhui.top/blog/2018/09/11/PyTorchNote2.html

Jiqixuexirukengzhe. (2019, October 16). *Linear Regression Model with Pytorch Implementation*. Zhihu. https://zhuanlan.zhihu.com/p/86982616

Zhijiantingxi. (2021, March 3). *[Pytorch] Unsqueeze() and Squeeze() Explained*. Zhihu. https://zhuanlan.zhihu.com/p/354177617

Anexo (Godigo)

#install.packages("CASdatasets", repos = "http://cas.uqam.ca/pub/", type="source")

```
1. ############################
2. ##GLM frecuencia##########
3. ############################
4. 
5. #Intrducemos los paquetes
6. library(xts)
7. library(zoo)
8. library(CASdatasets) # paquete de datos
9. library(readxl)
10.library(dplyr)
11.library(knitr)
12.library(ggplot2)
13.library(magrittr)
14.library(sqldf)
15.library(corrplot)
16.library(DescTools)
17.data(freMTPL2freq)
18.data(freMTPL2sev)
19.freq<-freMTPL2freq  # siniestros
20.claim<-freMTPL2sev
21.
22.# dimensiones
23.(dim(freq))
24.(dim(claim))
25.
26.## Fase1: EDA de datos de Frecuencias
27.
28.### 1, Tratar los datos erróneos / faltan datos (missing valu
   es)
29.sum(is.na(freq)) # no hay erroneos
30.### 2, Detectar los outliers (datos atípicos)
31.#by expert judgement
32.#eliminate that with more than 5 claims(7 observationes).
33.freq<-subset(freq,freq$ClaimNb<6)
34.### 3, Análisis Univariante(alalisis de feature)
35.#Para saber si tiene imbalancia de datos
36.#### a) ESTADISTICA DESCRIPATIVA
37.des_table<-freq[,-which(colnames(freq)%in% c("IDpol"))]
38.des_table$ClaimNb<-as.numeric(des_table$ClaimNb)
39.estad_desp<-summary(select_if(des_table,is.numeric))
40.kable(estad_desp,caption="Estaditica Descriptiva")
41.####  b) visualizacion
42.#  Histgramma de num_sini
43.options(scipen=1000000)
44.ggplot(data = freq,
45.       mapping = aes(x=as.factor(ClaimNb)))+
46.  geom_bar(fill="grey")+
```

```
  scale_y_log10()+
  labs(title = "Histgram of claim number",y="Observation",x="
Claim Number")+
  theme(plot.title = element_text(hjust = 0.5))
#exposure
ggplot(data = freq,
       mapping = aes(x=Exposure))+
  geom_histogram(fill="grey")+
  labs(title = "Histgram of exposure",y="Observation")+ # por
   defecto el ancho es 30
  theme(plot.title = element_text(hjust = 0.5))

#ClaimAmout
options(scipen=1000000)
ggplot(claim,aes(x=ClaimAmount))+
  geom_density(fill="grey")+
  #scale_y_log10()+
  labs(title="Density plot of Claim Amounts (€) based on avai
lable data ")+
  xlim(0,5000)

#VehPower
options(scipen=1000000)
ggplot(data = freq,
       mapping = aes(x=as.factor(VehPower)))+
  geom_bar(fill="grey")+
  labs(title = "Distribution by Power of Vehicle",y="Observat
ion",x="Power of Vehicle")+
  theme(plot.title = element_text(hjust = 0.5))

#VehAge
ggplot(data = freq,
       mapping = aes(x=VehAge))+
  geom_bar(fill="grey")+
  scale_y_log10()+
  labs(title = "Distribution by Age of Vehicle",y="Observatio
n",x="Age of Vehicle")+
  theme(plot.title = element_text(hjust = 0.5))

#DrivAge
gplot(data=freq,aes(x=DrivAge))+
  geom_bar(fill="grey")+
  scale_x_continuous(n.break=10)+
  #scale_y_log10()+
  scale_x_continuous(breaks = seq(min(freq$DrivAge),max(freq$
DrivAge),5))+
  labs(title="Distribution by Age of Driver ",y="Observation"
,x="Age of Driver")+
  theme(plot.title = element_text(hjust = 0.5))

#BonusMalus
```

```
ggplot(data=freq,aes(x=BonusMalus))+
  geom_histogram(fill="grey")+
  scale_y_log10()+
  scale_x_continuous(n.break=10)+
  labs(title = "Distribution of BonusMalus",y="Observation",x
="Bonus and Malus")+
  theme(plot.title = element_text(hjust = 0.5))

#VehBrand
ggplot(data = freq,aes(x=as.factor(VehBrand)))+
        geom_bar(fill="grey")+
        labs(title = "Distrubution by Brand",y="Observation",
x="Brand of Vehicle")+
        theme(plot.title = element_text(hjust = 0.5))

      #VehGas
      options(scipen=1000000)
      ggplot(freq,aes(x=VehGas))+
        geom_bar(fill="grey")+
        labs(x="Type of Gas",title = "Distribution of Policy
by Type of Gas",y="Observation")+
        theme(plot.title = element_text(hjust = 0.5))
      # ylim(0,0.32)+ #if i dont adjust the ylim, it seems li
ke this variable change alot

      #Area
      ggplot(data = freq,aes(x=Area))+
        geom_bar(fill="grey")+
        labs(title = "Distribution of Area",y="Observation")+
        theme(plot.title = element_text(hjust = 0.5))

      #Density
      ggplot(data = freq,aes(x=Density))+
        geom_histogram(fill="grey",bins = 20)+ #bins 代表分成
了20份
        scale_y_log10()+
        scale_x_continuous(n.breaks = 14)+
        labs(title = "Distribution of Density",y="Observation
")+
        theme(plot.title = element_text(hjust = 0.5),axis.tex
t.x = element_text(hjust=1,angle =30 ))

      #Region
      ggplot(data = freq,aes(x=Region))+
        geom_bar(fill="grey")+
        #scale_y_log10()+
        #scale_x_continuous(n.breaks = 14)+
        labs(title = "Distribution of Region")+
        theme(plot.title = element_text(hjust = 0.5),axis.tex
t.x = element_text(hjust=1,angle =30 ))

```

```r
### 4, Análisis multivariante.

#a) feature contra frequencia

#VehPower
#options(scipen=1000000)
summary_veh<-sqldf("select VehPower,sum(ClaimNB)/sum(Exposure) as freq_veh from freq group by VehPower")
summary_veh%>%ggplot(mapping = aes(x=VehPower,y=freq_veh))+
  geom_point(size=0.5,colour="red")+
  geom_line(colour="grey")+
  scale_x_continuous(breaks = seq(min(freq$VehPower),max(freq$VehPower),1))+
  labs(y="Frecuecy Empirical",x="Power of Vehicle",title = "Empirical frequency for each group of Power of Vehicle")+
  ylim(0,0.32)+ #if i dont adjust the ylim, it seems like this variable change alot
  theme(plot.title = element_text(hjust = 0.5))

#VehAge
#options(scipen=1000000)
summary_age<-sqldf("select VehAge,(sum(ClaimNB)/sum(Exposure)) as freq_age from freq group by VehAge")
summary_age%>%ggplot(mapping = aes(x=VehAge,y=freq_age))+
  geom_point(size=0.5,colour="red")+
  geom_line(colour="grey")+
  scale_x_continuous(breaks = seq(min(freq$VehAge),max(freq$VehAge),3))+
  labs(y="Frecuecy Empirical",x="Age of Vehicle",title = "Empirical frequency for each group of Age of Vehicle")+
  ylim(0,0.32)+ #if i dont adjust the ylim, it seems like this variable change alot
  theme(plot.title = element_text(hjust = 0.5))

#DrivAge
summary_dage<-sqldf("select DrivAge,(sum(ClaimNB)/sum(Exposure)) as freq_dage from freq group by DrivAge")
summary_dage%>%ggplot(mapping = aes(x=DrivAge,y=freq_dage))+
  geom_point(size=0.5,colour="red")+
  geom_line(colour="grey")+
  scale_x_continuous(breaks = seq(min(freq$DrivAge),max(freq$DrivAge),3))+
  labs(y="Frecuecy Empirical",x="Age of Driver",title = "Empirical frequency for each group of Age of Driver")+
```

```
        ylim(0,0.32)+ #if i dont adjust the ylim, it seems like this variable change alot
        theme(plot.title = element_text(hjust = 0.5))

    #BonusMalus
    BM<-floor(freq$BonusMalus/10)*10#注意区间
    sst_bm<-cbind(subset(freq,select = c(ClaimNb,Exposure)),BM)
    summary_bm<-sqldf("select BM,(sum(ClaimNB)/sum(Exposure)) as freq_bm from sst_bm group by BM")
    summary_bm%>%ggplot(mapping = aes(x=BM,y=freq_bm))+
      geom_point(size=0.5,colour="red")+
      geom_line(colour="grey")+
      labs(y="Frecuecy Empirical",x="Bonus and Malus",title = "Empirical frequency for each group of Bonua and Malus")+
      xlim(50,220)+
      ylim(0,1)+
      scale_x_continuous(breaks=seq(50,230,10),labels=c("[50-60)","[60-70)","[70,80)","[80,90)","[90,100)","[100,110)","[110,120)","[120,130)","[130,140)","[140,150)","[150,160)","[160,170)","[170,180)","[180,190)","[190,200)","[200,210)","[210,220)","[220,230)","[230,240)"))+
      theme(plot.title = element_text(hjust = 0.5),axis.text.x = element_text(hjust=1,angle =30 ))

    #VehBrand
    summary_bd<-sqldf("select VehBrand,(sum(ClaimNB)/sum(Exposure)) as freq_bd from freq group by VehBrand")
    summary_bd%>%ggplot(mapping = aes(x=VehBrand,y=freq_bd,group = 1))+ # group = 1
      geom_point(size=0.5,colour="red")+
      geom_line(colour="grey")+
      #scale_x_continuous(breaks = seq(min(freq$DrivAge),max(freq$DrivAge),3))+
      labs(y="Frecuecy Empirical",x="Brand",title = "Empirical frequency for each group of Brand")+
      ylim(0,0.2)+ #if i dont adjust the ylim, it seems like this variable change alot
      theme(plot.title = element_text(hjust = 0.5))

    #VehGas
    summary_gs<-sqldf("select VehGas,(sum(ClaimNB)/sum(Exposure)) as freq_gs from freq group by VehGas")
    summary_gs%>%ggplot(mapping = aes(x=VehGas,y=freq_gs,group = 1))+ # group = 1
      geom_point(size=0.5,colour="red")+
      geom_line(colour="grey")+
```

```
        #scale_x_continuous(breaks = seq(min(freq$DrivAge),max(freq$DrivAge),3))+
        labs(y="Frecuecy Empirical",x="Gas",title = "Empirical frequency for each group of Gas")+
        ylim(0,0.2)+ #if i dont adjust the ylim, it seems like this variable change alot
        theme(plot.title = element_text(hjust = 0.5))

    #Are
    summary_ar<-sqldf("select Area,(sum(ClaimNB)/sum(Exposure)) as freq_ar from freq group by Area")
    summary_ar%>%ggplot(mapping = aes(x=Area,y=freq_ar,group = 1))+ # group = 1
        geom_point(size=0.5,colour="red")+
        geom_line(colour="grey")+
        #scale_x_continuous(breaks = seq(min(freq$DrivAge),max(freq$DrivAge),3))+
        labs(y="Frecuecy Empirical",x="Area",title = "Empirical frequency for each group of Area")+
        ylim(0,0.2)+ #if i dont adjust the ylim, it seems like this variable change alot
        theme(plot.title = element_text(hjust = 0.5))


    #Density
    DE<-floor(freq$Density/500)*500#ten cuidado del rango

    sst_de<-cbind(subset(freq,select = c(ClaimNb,Exposure)),DE)

    summary_de<-sqldf("select DE,(sum(ClaimNB)/sum(Exposure)) as freq_de from sst_de group by DE")
    summary_de%>%ggplot(mapping = aes(x=DE,y=freq_de))+
        geom_point(size=0.5,colour="red",group=1)+
        geom_line(colour="grey")+
        labs(y="Frecuecy Empirical",x="Density",title = "Empirical frequency for each group of Density")+
        theme(plot.title = element_text(hjust = 0.5))

    #xlim(50,200)+
    #ylim(0,0.8)+ #if i dont adjust the ylim, it seems like this variable change alot
    #+theme(plot.title = element_text(hjust = 0.5))

    #Region
    summary_re<-sqldf("select Region,(sum(ClaimNB)/sum(Exposure)) as freq_re from freq group by Region")
```

```
232.      summary_re%>%ggplot(mapping = aes(x=Region,y=freq_re,gr
   oup = 1))+ # group = 1
233.        geom_point(size=0.5,colour="red")+
234.        geom_line(colour="grey")+
235.        #scale_x_continuous(breaks = seq(min(freq$DrivAge),ma
   x(freq$DrivAge),3))+
236.        labs(y="Frecuecy Empirical",x="Region",title = "Empir
   ical frequency for each Region")+
237.        ylim(0,0.2)+ #if i dont adjust the ylim, it seems lik
   e this variable change alot
238.        theme(plot.title = element_text(hjust = 0.5))+
239.        theme(axis.text.x = element_text(hjust=1,angle =30 ))

240.
241.
242.      #b)Correlacion
243.
244.      ##b.1) Correlación entre variables numéricas (e.g. Coef
   iciente de correlación de Pearson).
245.
246.      freq_num<-select_if(freq,is.numeric)
247.      freq_num<-select(freq_num,-
   c("IDpol","ClaimNb","Exposure"))
248.      (M_n<-cor(as.matrix(freq_num)))
249.      corrplot(M_n,tl.col="black")
250.
251.      ##b.2) Correlación entre variables carácter (V-Cramer)
252.
253.      PairApply(select(freq,VehBrand,VehGas,Area,Region), FUN
    = CramerV,
254.                symmetric=TRUE)
255.      corrplot::corrplot(DescTools::PairApply(select(freq,Veh
   Brand,VehGas,Area,Region) , DescTools::CramerV),tl.col="black
   ")
256.
257.      ## GLM de Frecuencias
258.
259.      #part 1, feature engineer
260.
261.      # region
262.      regionM<-
   read_excel("/Users/liying/Desktop/glm_region.xlsx") # need to
    change path.
263.
264.      freq_glm<-
   merge(freq,regionM,all.x = TRUE,all.y = FALSE)
265.      freq_glm$region_glm1<-as.factor(freq_glm$region_glm1)
266.
267.      # area
268.      table(freq$Area)/dim(freq)[1]
269.      #
```

```
freq_glm$are_glm1<-as.character(freq_glm$Area)
freq_glm$are_glm1<-replace(freq_glm$are_glm1,freq_glm$are_glm1=="E"|freq_glm$are_glm1=="F","EF")

freq_glm$are_glm1<-as.factor(freq_glm$are_glm1)

freq_glm$bran_glm1<-as.character(freq_glm$VehBrand)
freq_glm$bran_glm1<-ifelse(freq_glm$bran_glm1 %in% c("B11", "B12"), "bran_a","bran_b")

freq_glm$bran_glm1<-as.factor(freq_glm$bran_glm1)

#VehPower
freq_glm$vehP_glm1<-cut(freq_glm$VehPower,c(4,5,8,Inf),include.lowest = TRUE)
#VehAge
freq_glm$vehA_glm1<-cut(freq_glm$VehAge,c(0,1,10,20,Inf),include.lowest = TRUE)
#DrivAge
freq_glm$dirA_glm1<-cut(freq_glm$DrivAge,c(18,21,26,33,40,55,70,Inf),include.lowest = TRUE)
#BonusMalus
freq_glm$Bm_glm1<-cut(freq_glm$BonusMalus,c(50,100,140,Inf),include.lowest = TRUE)
#Density
freq_glm$Den_glm1<-cut(freq_glm$Density,c(0,1500,10000,Inf),include.lowest = TRUE,dig.lab = 4)

#part 2, sampling training and testing

set.seed (100)

num_glm <- sample (c(1: nrow( freq_glm )), round (0.7*nrow( freq_glm )), replace = FALSE)
learn <- freq_glm [num_glm ,]
test <- freq_glm [-num_glm ,]

#model 0
#freq_glm1_1<-select(freq,-c ("IDpol"))
fre_glm0<-glm( formula = ClaimNb ~ vehA_glm1 + dirA_glm1 , family = poisson (), data = learn , offset = log( Exposure ))
summary(fre_glm0)

### modelo(3 feature)
```

```r
fre_glm<-
   glm( formula = ClaimNb ~ vehA_glm1 + dirA_glm1+vehP_glm1 , fa
   mily = poisson (), data = learn , offset = log( Exposure ))
summary(fre_glm)

### modelo(3 feature)
fre_glm<-
   glm( formula = ClaimNb ~ vehA_glm1 + dirA_glm1+bran_glm1 , fa
   mily = poisson (), data = learn , offset = log( Exposure ))
summary(fre_glm)

### modelo(4 feature)
fre_glm<-
   glm( formula = ClaimNb ~ vehA_glm1 + dirA_glm1+bran_glm1+vehA
   _glm1 , family = poisson (), data = learn , offset = log( Exp
   osure ))
summary(fre_glm)

### modelo(8 feature)
#freq_glm1_1<-select(freq,-c ("IDpol"))
fre_glm1b<-
   glm( formula = ClaimNb ~ vehA_glm1 + vehP_glm1 + dirA_glm1 +
   Bm_glm1 + bran_glm1+ Den_glm1 + region_glm1 + are_glm1, famil
   y = poisson (), data = learn , offset = log( Exposure ))
summary(fre_glm1b)

### modelo(7 features)
#freq_glm1_1<-select(freq,-c ("IDpol"))
fre_glm1a<-
   glm( formula = ClaimNb ~ vehA_glm1 + vehP_glm1 + dirA_glm1 +
   Bm_glm1 + bran_glm1+  region_glm1 + are_glm1 , family = poiss
   on (), data = learn , offset = log( Exposure ))
summary(fre_glm1a)

### modelo(sartuno)[the one with best aic]
#freq_glm1_1<-select(freq,-c ("IDpol"))
fre_glm1<-
   glm( formula = ClaimNb ~ vehA_glm1 + vehP_glm1 + dirA_glm1 +
   Bm_glm1 + bran_glm1+ Den_glm1 + region_glm1 + are_glm1+VehGas
    , family = poisson (link = log), data = learn , offset = log
   ( Exposure ))
summary(fre_glm1)

#Predecir y sacar su MSE

pred<-predict(fre_glm1,test,type="response")
mean((test$ClaimNb - pred)^2)

############################
##GLM Severidad############
############################
```

```r

    # introducir los paquetes necesarios y datos
    library(CASdatasets) # paquete de datos
    library(xts)
    library(readxl)
    library(zoo)
    library(dplyr)
    library(knitr)
    library(ggplot2)
    library(magrittr)
    library(sqldf)
    library(corrplot)
    library(DescTools)
    library(corrplot)

    #introducer datas
    data(freMTPL2sev)
    data("freMTPL2freq")
    dim(freMTPL2sev)
    head(freMTPL2sev)

    #chequear si tiene toda la informacion segun IDpol
    sum(freMTPL2sev$IDpol %in% freMTPL2freq$IDpol)
    # We can see that not all the `freMTPL2sev$IDpol` are i
  ncluded in the `freMTPL2freq$IDpol`,
    #so I would just get the intersection.

    #merge
    claim<-merge(freMTPL2sev,freMTPL2freq,all=FALSE)
    claim<-select(claim,-
  c("IDpol","ClaimNb","Exposure")) #Delete the columns we don't
   need,


    ## Fase1, EDA of the datas of severity

    ### 1, Errones
    sum(is.na(claim))

    ## 2, Detect outliers

    #Given that gamma is very sensitve to the extreme value
  , so I will remove based on his effect on the average.

    summary(claim$ClaimAmount)
    options(scipen=1000000)
    plot(claim$ClaimAmount)
    catas<-head(subset(claim,claim$ClaimAmount>800000))
    head(catas)
    #quantil
```

```
386.      x1<-
   as.data.frame(as.table(quantile(sort(claim$ClaimAmount),seq(0
   .98,1,0.001))))
387.      x2<-as.data.frame(round((1-
   seq(0.98,1,0.001))*dim(claim)[1],digits = 0))
388.      x3<-cbind(x1,x2)
389.      colnames(x3)<-c("Quantil","Claim Amount","Obervations")
390.      head(x3)
391.      #we can see that we that most values are distributed be
   low than 1500. Next we will analisis the affect of these extr
   ame values.
392.
393.      #For better concentration, I would just keep `ClaimAmou
   nt` lesss than 10,000.
394.      tapply(claim$ClaimAmount, claim$VehAge, mean)
395.      claim<-subset(claim,ClaimAmount<=10000)
396.
397.      ### 3, Univariate Analysis
398.      #### a) DESCRIPTIVE STATISTICS
399.
400.      summary(claim)
401.      #### b) Visualization(analisis of the observation to ma
   ke segmentation)
402.      sta_table <- round(table(cut(claim$ClaimAmount,c(seq(0,
   5000,500),Inf),dig.lab = 10000))/dim(claim)[1],digits = 3)
403.      sta_table
404.
405.
406.      b.1)
407.
408.      #ClaimAmount
409.      options(scipen=1000000)
410.      ggplot(data = claim,
411.             mapping = aes(x=ClaimAmount))+
412.        geom_density(fill="grey")+
413.        #scale_y_log10()+
414.        #xlim(0,5000)+
415.        labs(title = "Density plot of Claim Amounts (€) based
    on available data",x="Claim Amount")+
416.        theme(plot.title = element_text(hjust = 0.5))
417.
418.      #VehPower
419.      ggplot(data = claim,
420.             mapping = aes(x=as.factor(VehPower)))+
421.        geom_bar(fill="grey")+
422.        labs(title = "Distribution by Power of Vehicle",y="Ob
   servation",x="Power of Vehicle")+
423.        theme(plot.title = element_text(hjust = 0.5))
424.
425.      #VehAge
426.      ggplot(data = claim,
```

```
427.             mapping = aes(x=VehAge))+
428.       geom_density(fill="grey")+
429.       #scale_y_log10()+
430.       #xlim(0,50)+
431.       scale_x_continuous(n.breaks = 33)+
432.       labs(title = "Distribution by Age of Vehicle",y="Obse
   rvation",x="Age of Vehicle")+
433.       theme(plot.title = element_text(hjust = 0.5))
434.
435.     #DrivAge
436.     ggplot(data=claim,aes(x=DrivAge))+
437.       geom_density(fill="grey")+
438.       scale_x_continuous(n.break=20)+
439.       #xlim(0,80)+
440.       scale_x_continuous(breaks = seq(min(claim$DrivAge),ma
   x(claim$DrivAge),2))+
441.       labs(title="Distribution by Age of Driver ",y="Observ
   ation",x="Age of Driver")+
442.       theme(plot.title = element_text(hjust = 0.5),axis.tex
   t.x = element_text(hjust=1,angle =30))
443.
444.     #BonusMalus
445.     ggplot(data=claim,aes(x=BonusMalus))+
446.       geom_density(fill="grey")+
447.       labs(title = "Distribution by BonusMalus",x="Bonus an
   d Malus")+
448.       scale_x_continuous(n.break=30)+
449.       theme(plot.title = element_text(hjust = 0.5))
450.
451.     #VehBrand
452.     ggplot(data = claim,aes(x=as.factor(VehBrand)))+
453.       geom_bar(fill="grey")+
454.       labs(title = "Distrubution by Brand",y="Observation",
   x="Brand of Vehicle")+
455.       theme(plot.title = element_text(hjust = 0.5))
456.
457.     #VehGas
458.     ggplot(claim,aes(x=VehGas))+
459.       geom_bar(fill="grey")+
460.       labs(x="Type of Gas",title = "Distribution by Policy
   by Type of Gas",y="Observation")+
461.       theme(plot.title = element_text(hjust = 0.5))
462.
463.     #Area
464.     ggplot(data = claim,aes(x=Area))+
465.       geom_bar(fill="grey")+
466.       labs(title = "Distribution by Area",y="Observation")+
467.       theme(plot.title = element_text(hjust = 0.5))
468.
469.     #Density
470.     ggplot(data = claim,aes(x=Density))+
```

```
471.        geom_density(fill="grey")+ #bins 代表分成了20份
472.        #`scale_y_log10()+
473.        scale_x_continuous(n.breaks = 20)+
474.        labs(title = "Distribution by Density",y="Observation
    ")+
475.        theme(plot.title = element_text(hjust = 0.5),axis.tex
    t.x = element_text(hjust=1,angle =30 ))
476.
477.      #Region
478.      ggplot(data = claim,aes(x=Region))+
479.        geom_bar(fill="grey")+
480.        labs(title = "Distribution by Region")+
481.        theme(plot.title = element_text(hjust = 0.5),axis.tex
    t.x = element_text(hjust=1,angle =30 ))
482.
483.      ### 4, Análisis multivariante.
484.
485.      #a)feature vs medium claim amount
486.      #1.VehPower
487.      summary_vehc<-
    sqldf("select VehPower, AVG(ClaimAmount) as cs_veh from claim
     group by VehPower")
488.      summary_vehc%>%ggplot(mapping = aes(x=VehPower,y=cs_veh
    ))+
489.        geom_point(size=0.5,colour="red")+
490.        geom_line(colour="grey")+
491.        ylim(0,3200)+
492.        scale_x_continuous(breaks = seq(min(claim$VehPower),m
    ax(claim$VehPower),1))+
493.        labs(y="Average Claim Amount",x="Power of Vehicle",ti
    tle = "Empirical Average Claim Amount for each group of Power
     of Vehicle")+
494.        theme(plot.title = element_text(hjust = 0.5))
495.
496.
497.      #******this is the best way i can think, for group cifr
    a
498.      df<-
    as.data.frame(as.table(tapply(claim$ClaimAmount,claim$VehAge,
    mean)))
499.      colnames(df)<-c("VehAge","Ave_cost")
500.      df%>%ggplot(mapping = aes(x=VehAge,y=Ave_cost,group = 1
    ))+
501.        geom_point(size=0.5,colour="red")+
502.        geom_line(colour="grey")+
503.        labs(y="Average Claim Amount",x="Age of Vehicle",titl
    e = "Empirical Average Claim Amount for each group of Age of
    Vehicle")+
504.        theme(plot.title = element_text(hjust = 0.5))
505.
506.      #DrivAge
```

```r
    summary_dage<-sqldf("select DrivAge, AVG(ClaimAmount) as cs_vehd from claim group by DrivAge")
    summary_dage%>%ggplot(mapping = aes(x=DrivAge,y=cs_vehd))+
      geom_point(size=0.5,colour="red")+
      geom_line(colour="grey")+
      scale_x_continuous(breaks = seq(min(claim$DrivAge),max(claim$DrivAge),2))+
      labs(y="Claim Amount",x="Age of Driver",title = "Empirical Average Claim Amount for each group of Age of Driver")+
      theme(plot.title = element_text(hjust = 0.5),axis.text.x =element_text(hjust=1,angle =45 ) )

    # BonusMalus

    #summary(claim$BonusMalus)
    #df<-as.data.frame(as.table(tapply(claim$ClaimAmount,cut(claim$BonusMalus,c(seq(50,230,10)),include.lowest = TRUE,dig.lab = 10000),mean)))
    df<-as.data.frame(as.table(tapply(claim$ClaimAmount,claim$BonusMalus,mean)))
    colnames(df)<-c("BM","Ave_cost")
    df$BM<-as.integer(as.character(df$BM)) # when transform number-like factor to number, we need to turn to character first
    df%>%ggplot(mapping = aes(x=BM,y=Ave_cost,group = 1))+
      geom_point(size=0.5,colour="red")+
      geom_line(colour="grey")+
      scale_x_continuous(n.breaks = 30)+
      labs(y="Average Claim Amount",x="Bonus Malus",title = "Empirical Average Claim Amount for each group of Bonus Malus")+
      theme(plot.title = element_text(hjust = 0.5),axis.text.x = element_text(hjust=1))

    # VehBrand

    summary_bd<-sqldf("select VehBrand, AVG(ClaimAmount) as cs_vehb from claim group by VehBrand")
    summary_bd%>%ggplot(mapping = aes(x=VehBrand,y=cs_vehb,group = 1))+ # group = 1
      geom_point(size=0.5,colour="red")+
      geom_line(colour="grey")+
      ylim(0,2000)+
      labs(y="Average Claim Amount",x="Brand",title = "Empirical Claim Amount for each group of Brand")+
```

```
538.        theme(plot.title = element_text(hjust = 0.5))
539.
540.      #VehGas
541.
542.      summary_gs<-
   sqldf("select VehGas,AVG(ClaimAmount) as cs_vehb from claim g
   roup by VehGas")
543.      summary_gs%>%ggplot(mapping = aes(x=VehGas,y=cs_vehb,gr
   oup = 1))+ # group = 1
544.        geom_point(size=0.5,colour="red")+
545.        geom_line(colour="grey")+
546.        #scale_x_continuous(breaks = seq(min(freq$DrivAge),ma
   x(freq$DrivAge),3))+
547.        labs(y="Average Claim Amount",x="Gas",title = "Empiri
   cal Claim Amount for each group of Gas")+
548.        ylim(0,2200)+
549.        theme(plot.title = element_text(hjust = 0.5))
550.
551.
552.      #Area
553.      summary_ar<-
   sqldf("select Area,AVG(ClaimAmount) as cs_ar from claim group
    by Area")
554.      summary_ar%>%ggplot(mapping = aes(x=Area,y=cs_ar ,group
    = 1))+ # group = 1
555.        geom_point(size=0.5,colour="red")+
556.        geom_line(colour="grey")+
557.        #scale_x_continuous(breaks = seq(min(freq$DrivAge),ma
   x(freq$DrivAge),3))+
558.        labs(y="Average Claim Amount",x="Area",title = "Empir
   ical Claim Amount for each group of Area")+
559.        ylim(0,2500)+ #if i dont adjust the ylim, it seems li
   ke this variable change alot
560.        theme(plot.title = element_text(hjust = 0.5))
561.
562.      # Density
563.
564.      #summary(claim$Density)
565.      df<-
   as.data.frame(as.table(tapply(claim$ClaimAmount,cut(claim$Den
   sity,c(seq(0,27000,1000)),include.lowest = TRUE,dig.lab = 100
   00),mean)))
566.      colnames(df)<-c("Density","Ave_cost")
567.      df%>%ggplot(mapping = aes(x=Density,y=Ave_cost,group =
   1))+
568.        geom_point(size=0.5,colour="red")+
569.        geom_line(colour="grey")+
570.        labs(y="Average Claim Amount",x="Density",title = "Em
   pirical Average Claim Amount for each group of Density")+
571.        theme(plot.title = element_text(hjust = 0.5)) +
572.        theme(axis.text.x = element_text(hjust=1,angle =30 ))
```

```
573.
574.      # Region
575.
576.      summary_re<-
   sqldf("select Region,AVG(ClaimAmount) as cs_re from claim gro
   up by Region")
577.      summary_re%>%ggplot(mapping = aes(x=Region,y=cs_re,grou
   p = 1))+ # group = 1
578.        geom_point(size=0.5,colour="red")+
579.        geom_line(colour="grey")+
580.        #scale_x_continuous(breaks = seq(min(freq$DrivAge),ma
   x(freq$DrivAge),3))+
581.        labs(y="Average Claim Amount",x="Region",title = "Emp
   irical Claim Amount for each Region")+
582.        ylim(0,2000)+ #if i dont adjust the ylim, it seems li
   ke this variable change alot
583.        theme(plot.title = element_text(hjust = 0.5))+
584.        theme(axis.text.x = element_text(hjust=1,angle =30 ))

585.
586.      #b.1) Correlación entre variables numéricas (e.g. Coefi
   ciente de correlación de Pearson).
587.      seve_num<-select_if(claim,is.numeric)
588.      seve_num<-select(seve_num,-c("ClaimAmount"))
589.      (S_n<-cor(as.matrix(seve_num)))
590.      corrplot(S_n,tl.col="black")
591.
592.      #b.2) Correlación entre variables carácter (V-Cramer)
593.      PairApply(select(claim,VehBrand,VehGas,Area,Region), FU
   N = CramerV,
594.                symmetric=TRUE)
595.      corrplot::corrplot(DescTools::PairApply(select(claim,Ve
   hBrand,VehGas,Area,Region) , DescTools::CramerV),tl.col="blac
   k")
596.
597.
598.      ### GLM of severity
599.
600.      #part 1, feature engineer
601.
602.      sev_glm<-claim
603.      #VehBrand
604.      sev_glm$bran_glm1<-as.character(sev_glm$VehBrand)
605.      sev_glm$bran_glm1<-
   ifelse(sev_glm$bran_glm1 %in% c("B14", "B5","B6"), "bran_a","
   bran_b") #bran_a refers to brands with average claim less tha
   n 1500, otherwise bran b
606.      sev_glm$bran_glm1<-as.factor(sev_glm$bran_glm1)
607.
608.      #Area
609.      sev_glm$are_glm1<-as.character(sev_glm$Area)
```

```
610.      sev_glm$are_glm1<-
   replace(sev_glm$are_glm1,sev_glm$are_glm1=="D"|sev_glm$are_gl
   m1=="E","DE")
611.      sev_glm$are_glm1<-as.factor(sev_glm$are_glm1)
612.
613.      #Region
614.      sev_glm$region_glm1<-as.character(sev_glm$Region)
615.      sev_glm$region_glm1<-
   ifelse(sev_glm$region_glm1%in% c("Provence-Alpes-Cotes-
   D'Azur", "Corse"), "region_h","region_l")
616.      sev_glm$region_glm1<-as.factor(sev_glm$region_glm1)
617.
618.      #VehPower
619.      sev_glm$vehP_glm1<-
   cut(sev_glm$VehPower,c(0,8,Inf),include.lowest = TRUE)
620.      #VehAge
621.      sev_glm$vehA_glm1<-
   cut(sev_glm$VehAge,c(0,8,Inf),include.lowest = TRUE)
622.      #table(tapply(claim$ClaimAmount,cut(claim$VehAge,c(0,8,
   Inf),include.lowest = TRUE),mean))
623.      sev_glm$vehA_glm2<-
   cut(sev_glm$VehAge,c(0,1,4,15,Inf),include.lowest = TRUE)
624.
625.
626.
627.      #DirvAge
628.      sev_glm$dirA_glm1<-
   cut(sev_glm$DrivAge,c(18,22,79,Inf),include.lowest = TRUE)
629.      sev_glm$dirA_glm2<-
   cut(sev_glm$DrivAge,c(18,22,69,Inf),include.lowest = TRUE)
630.      sev_glm$dirA_glm3<-
   cut(sev_glm$DrivAge,c(18,23,Inf),include.lowest = TRUE)
631.      #Density
632.      sev_glm$Den_glm1<-
   cut(sev_glm$Density,c(0,3000,9000,Inf),include.lowest = TRUE)
633.
634.      #part 2, sampling training and testing
635.
636.
637.      set.seed (100)
638.
639.      num_glm <- sample (c(1: nrow( sev_glm )), round (0.7* n
   row( sev_glm )), replace = FALSE)
640.      learn <- sev_glm [num_glm ,]
641.      test <- sev_glm [-num_glm ,]
642.
643.
644.      ### modelo 0
645.      seve_glm1<-
   glm( formula = ClaimAmount ~ bran_glm1+are_glm1+region_glm1
```

```
                      , family = Gamma(link="log"), data = le
  arn )
    summary(seve_glm1)

    seve_glm1<-glm( formula = ClaimAmount ~ dirA_glm2
                      , family = Gamma(link="log"), data = le
  arn )
    summary(seve_glm1)
    seve_glm1<-glm( formula = ClaimAmount ~ vehA_glm2
                      , family = Gamma(link="log"), data = le
  arn )
    summary(seve_glm1)
    seve_glm1<-
  glm( formula = ClaimAmount ~ bran_glm1+are_glm1
                      , family = Gamma(link="log"), data = le
  arn )
    summary(seve_glm1)

    ###Modelo 1
    seve_glm1<-
  glm( formula = ClaimAmount ~ vehA_glm1 + vehP_glm1 + dirA_glm
  1 + bran_glm1+ Den_glm1 + region_glm1 + are_glm1, family = Ga
  mma(link="log"), data = learn )
    summary(seve_glm1)
    #Modelo 2
    seve_glm1<-
  glm( formula = ClaimAmount ~ bran_glm1+are_glm1+region_glm1+D
  en_glm1
                      , family = Gamma(link="log"), data = le
  arn )
    summary(seve_glm1)
    #Modelo 3

    seve_glm1<-
  glm( formula = ClaimAmount ~ vehA_glm1 + vehP_glm1 + dirA_glm
  1 + bran_glm1+ Den_glm1 + region_glm1 + are_glm1+VehGas, fami
  ly = Gamma(link="log"), data = learn )
    summary(seve_glm1)


    #### the best one
    seve_glm1<-
  glm( formula = ClaimAmount ~ bran_glm1+are_glm1+region_glm1+d
  irA_glm3
                      , family = Gamma(link="log"), data = le
  arn )
    summary(seve_glm1)

    #Un modelo no tiene ninguon sentido
    seve_glm1<-glm( formula = ClaimAmount ~ BonusMalus
```

```
679.                     , family = Gamma(link="log"), data = le
   arn )
680.      summary(seve_glm1)
681.
682.      #Predecir y sacar MSE
683.      pred<-predict(seve_glm1,test,type="response")
684.      mean((test$ClaimAmount - pred)^2)
685.
686.
687.
688.      ############################
689.      ##NNT Frecuencia##########
690.      ############################
691.
692.
693.      #One-hot encoding
694.      ngas<-as.data.frame(model.matrix(~VehGas-1,freq_glm ) )
695.      nbran<-as.data.frame(model.matrix(~bran_glm1-
   1,freq_glm ) )
696.      narea<-as.data.frame(model.matrix(~are_glm1-
   1,freq_glm ) )
697.      nregion<-as.data.frame(model.matrix(~region_glm1-
   1,freq_glm ))
698.      nnum_feature<-
   select(freq_glm,VehPower,VehAge,DrivAge,BonusMalus,Density)
699.      standard4<-preProcess(nnum_feature,method = 'range')
700.      freq_num_s<- predict(standard4, nnum_feature)
701.      nfreq<-
   as.data.frame(freq_glm$ClaimNb/freq_glm$Exposure)
702.      ndata<-cbind(freq_num_s,ngas,narea,nregion,nbran,nfreq)
703.
704.      #Samping
705.      num_glm <- sample (c(1: nrow( freq_glm )), round (0.7*
   nrow( freq_glm )), replace = FALSE)
706.      ntlearn <- ndata [num_glm ,]
707.      nttest <- ndata[-num_glm ,]
708.
709.      #modelizacion
710.      neuralnet(Freq~VehPower+VehAge+DrivAge+BonusMalus+Densi
   ty+VehGasDiesel+VehGasRegular+are_glm1A+are_glm1B+are_glm1C+a
   re_glm1D+are_glm1EF+region_glm1region_a+region_glm1region_b+r
   egion_glm1region_c+bran_glm1bran_a+bran_glm1bran_b,
711.                 data=ntlearn,hidden = c(30),threshold = 0.01,
   algorithm = "rprop+",err.fct = "sse",linear.output = FALSE,ac
   t.fct = "tanh")
```

Anexo (Python Code)

- PyTorch

```
1. ####################freq pytorch#################
2. from sklearn.preprocessing import StandardScaler
3. #from sklearn.model_selection import train_test_split
4. from sklearn.model_selection import GridSearchCV
5. from sklearn.neural_network import MLPRegressor
6. from sklearn.metrics import mean_squared_error
7. import pandas as pd
8. from sklearn.preprocessing import OneHotEncoder
9.
10.#sample index(to have the same sampling with R)
11.ll = pd.read_csv('/content/sample-李颖.csv',sep=";")
12.ll=ll.x #transforme to serie
13.ll_f=ll-1
14.
15.#read datas
16.data = pd.read_csv('/content/data_freq-李
   颖.csv',sep=";",decimal = ',')
17.data=data.drop(['IDpol'],axis=1)
18.data['freq']=data['ClaimNb']/data['Exposure']
19.data = data.drop(['ClaimNb'],axis=1)
20.
21.#standerlization learn
22.data_n=data[["VehPower","DrivAge","VehAge","DrivAge","BonusMa
   lus","Density"]]
23.scaler = StandardScaler()
24.scaled_dataset = scaler.fit_transform(data_n)
25.
26.data_s=pd.DataFrame(scaled_dataset,index=None,columns = ["Veh
   Power","DrivAge","VehAge","DrivAge","BonusMalus","Density"])
27.df_freq=pd.concat([data_s,data[["freq","VehBrand","VehGas","A
   rea","Region","brandS","areaS","regionS"]]],axis=1)
28.
29.
30.df_freq.info()
31.
32.# Check how many unique labels for every categorical variable
33.for col in data[["VehBrand","VehGas","Area","Region","brandS"
   ,"areaS","regionS"]]:
34.  print(col, ':', len(data[col].unique()), 'labels')
35.
36.#encoding the simplified categorical variables
37.
38.#gas
39.ohe = OneHotEncoder()
40.gas_ohe = ohe.fit_transform(df_freq[["VehGas"]]).toarray()
```

```
gas_ohe_df = pd.DataFrame(gas_ohe, columns= ohe.get_feature_names_out())
df_freq=pd.concat([df_freq,gas_ohe_df],axis = 1)

#areaS
ohe = OneHotEncoder()
areaS_ohe = ohe.fit_transform(df_freq[["areaS"]]).toarray()
areaS_ohe_df = pd.DataFrame(areaS_ohe, columns= ohe.get_feature_names_out())
df_freq=pd.concat([df_freq,areaS_ohe_df],axis = 1)

#brandS
ohe = OneHotEncoder()
brandS_ohe = ohe.fit_transform(df_freq[["brandS"]]).toarray()
brandS_ohe_df = pd.DataFrame(brandS_ohe, columns= ohe.get_feature_names_out())
df_freq=pd.concat([df_freq,brandS_ohe_df],axis = 1)

#regionS
ohe = OneHotEncoder()
regionS_ohe = ohe.fit_transform(df_freq[["regionS"]]).toarray()
regionS_ohe_df = pd.DataFrame(regionS_ohe, columns= ohe.get_feature_names_out())
df_freq=pd.concat([df_freq,regionS_ohe_df],axis = 1)


df_freq_code=df_freq.drop(['VehGas','Area','Region','VehBrand','brandS','areaS','regionS'],axis=1)

#to keep the same train data and test data, i would use the random number of R.
import numpy as np

df_freq_trainval=df_freq_code.iloc[ll_f]

df_freq_trainval_x= np.array(df_freq_trainval.drop(['freq'],axis=1))
df_freq_trainval_y=np.array(df_freq_trainval['freq'])

df_freq_test=df_freq_code.drop(ll_f,axis=0)
df_freq_test_x=np.array(df_freq_test.drop(['freq'],axis=1))
df_freq_test_y=np.array(df_freq_test['freq'])

import torch
from torch.utils.data import TensorDataset, DataLoader
from torchvision import datasets
from torchvision.transforms import ToTensor
import torch.nn as nn

X_train = torch.from_numpy(df_freq_trainval_x).float()
```

```
y_train = torch.from_numpy(df_freq_trainval_y).float()
X_test = torch.from_numpy(df_freq_test_x).float()
y_test = torch.from_numpy(df_freq_test_y).float()

##Define a batch size ,
bs = 64

#Both x_train and y_train can be combined in a single TensorDataset, which will be easier to iterate over and slice
y_train_tensor = y_train.unsqueeze(1)
train_ds = TensorDataset(X_train, y_train_tensor)
#Pytorch's DataLoader is responsible for managing batches.
#You can create a DataLoader from any Dataset. DataLoader makes it easier to iterate over batches
train_dl = DataLoader(train_ds, batch_size=bs)

#y_test2_tensor = y_test2.unsqueeze(1)
test_ds = TensorDataset(X_test, y_test)
    test_dl = DataLoader(test_ds, batch_size=32)

    n_input_dim = X_train.shape[1]

    #Layer size
    n_hidden1 = 20  # Number of hidden nodes
    n_hidden2 = 10
    n_output =  1   # Number of output nodes = for binary classifier


    class ChurnModel(nn.Module):
        def __init__(self):
            super(ChurnModel, self).__init__()
            self.layer_1 = nn.Linear(n_input_dim, n_hidden1)
            self.layer_2 = nn.Linear(n_hidden1, n_hidden2)
            self.layer_out = nn.Linear(n_hidden2, n_output)


            self.relu = nn.ReLU()
            self.softplus =  nn.Softplus()
            self.dropout = nn.Dropout(p=0.1)
            self.batchnorm1 = nn.BatchNorm1d(n_hidden1)
            self.batchnorm2 = nn.BatchNorm1d(n_hidden2)


        def forward(self, inputs):
            x = self.relu(self.layer_1(inputs))
            x = self.batchnorm1(x)
            x = self.relu(self.layer_2(x))
```

```
            x = self.batchnorm2(x)
            x = self.dropout(x)
            x = self.softplus(self.layer_out(x))

            return x

    model = ChurnModel()
    print(model)

    from sklearn import metrics
    #Loss Computation
    loss_function = nn.MSELoss()
    #Optimizer
    learning_rate = 0.001
    optimizer = torch.optim.Adam(model.parameters(), lr=lea
rning_rate)
    epochs = 20

    model.train()
    train_loss = []
    for epoch in range(epochs):
        #Within each epoch run the subsets of data = batch
sizes.
        for xb, yb in train_dl:
            y_pred = model(xb)            # Forward Propaga
tion
            loss = loss_function(y_pred, yb)  # Loss Comput
ation
            optimizer.zero_grad()         # Clearing all pr
evious gradients, setting to zero
            loss.backward()               # Back Propagatio
n
            optimizer.step()              # Updating the pa
rameters
        #print("Loss in iteration :"+str(epoch)+" is: "+str
(loss.item()))
        train_loss.append(loss.item())
    print('Last iteration loss value: '+str(loss.item()))

    import itertools

    y_pred_list = []
    model.eval()
    #Since we don't need model to back propagate the gradie
nts in test set we use torch.no_grad()
    # reduces memory usage and speeds up computation
    with torch.no_grad():
        for xb_test, yb_test in test_dl:
            y_test_pred = model(xb_test)
            y_pred_list.append(y_test_pred.detach().numpy()
)
```

```
171.
172.       #Takes arrays and makes them list of list for each batc
   h
173.       y_pred_list = [a.squeeze().tolist() for a in y_pred_lis
   t]
174.       #flattens the lists in sequence
175.       y_test_pred = list(itertools.chain.from_iterable(y_pred
   _list))
176.
177.       pre_exposure = y_test_pred * data['Exposure'].drop(ll_f
   ,axis=0)
178.       ori_exposure=df_freq_test['freq']*data['Exposure'].drop
   (ll_f,axis=0) # turn it back to compare to result of R
179.       metrics.mean_squared_error(pre_exposure, ori_exposure)
180.       y_test_pred[1:15] # we don't have negative value at all
181.
182.
183.
184.
185.
186.       ######################severidad PyTorch#################
   ########################
187.       from sklearn.preprocessing import StandardScaler
188.       #from sklearn.model_selection import train_test_split
189.       from sklearn.model_selection import GridSearchCV
190.       from sklearn.neural_network import MLPRegressor
191.       from sklearn.metrics import mean_squared_error
192.       import pandas as pd
193.       from sklearn.preprocessing import OneHotEncoder
194.
195.       #sample index(to have the same sampling with R)
196.       ll = pd.read_csv('/content/sample1.csv',sep=";")
197.       ll=ll.x #transforme to serie
198.       ll_f=ll-1
199.
200.       #read datas
201.       data = pd.read_csv('/content/data_seve_without.csv',sep
   =";",decimal = ',')
202.
203.       #standerlization learn
204.       data_n=data[["VehPower","DrivAge","VehAge","DrivAge","B
   onusMalus","Density"]]
205.       scaler = StandardScaler()
206.       scaled_dataset = scaler.fit_transform(data_n)
207.
208.       data_s=pd.DataFrame(scaled_dataset,index=None,columns =
    ["VehPower","DrivAge","VehAge","DrivAge","BonusMalus","Densi
   ty"])
209.       df_seve=pd.concat([data_s,data[["ClaimAmount","VehBrand
   ","VehGas","Area","Region","brandS","areaS","regionS"]]],axis
   =1)
```

```
210.
211.
212.      df_seve.info()
213.
214.      # Check how many unique labels for every categorical va
   riable
215.      for col in data[["VehBrand","VehGas","Area","Region","b
   randS","areaS","regionS"]]:
216.        print(col, ':', len(data[col].unique()), 'labels')
217.
218.
219.
220.      #encoding
221.
222.      #gas
223.      ohe = OneHotEncoder()
224.      gas_ohe = ohe.fit_transform(df_seve[["VehGas"]]).toarra
   y()
225.      gas_ohe_df = pd.DataFrame(gas_ohe, columns= ohe.get_fea
   ture_names_out())
226.      df_seve=pd.concat([df_seve,gas_ohe_df],axis = 1)
227.
228.      #areaS
229.      ohe = OneHotEncoder()
230.      areaS_ohe = ohe.fit_transform(df_seve[["areaS"]]).toarr
   ay()
231.      areaS_ohe_df = pd.DataFrame(areaS_ohe, columns= ohe.get
   _feature_names_out())
232.      df_seve=pd.concat([df_seve,areaS_ohe_df],axis = 1)
233.
234.      #brandS
235.      ohe = OneHotEncoder()
236.      brandS_ohe = ohe.fit_transform(df_seve[["brandS"]]).toa
   rray()
237.      brandS_ohe_df = pd.DataFrame(brandS_ohe, columns= ohe.g
   et_feature_names_out())
238.      df_seve=pd.concat([df_seve,brandS_ohe_df],axis = 1)
239.
240.      #region
241.      ohe = OneHotEncoder()
242.      regionS_ohe = ohe.fit_transform(df_seve[["regionS"]]).t
   oarray()
243.      regionS_ohe_df = pd.DataFrame(regionS_ohe, columns= ohe
   .get_feature_names_out())
244.      df_seve=pd.concat([df_seve,regionS_ohe_df],axis = 1)
245.
246.
247.      df_seve_code=df_seve.drop(['VehGas','Area','Region','Ve
   hBrand','brandS','areaS','regionS'],axis=1)
248.
```

```
    #to keep the same train data and test data, i would use the random number of R.

    import numpy as np

    df_seve_trainval=df_seve_code.iloc[ll_f]

    df_seve_trainval_x=np.array(df_seve_trainval.drop(['ClaimAmount'],axis=1))
    df_seve_trainval_y=np.array(df_seve_trainval['ClaimAmount'])

    df_seve_test=df_seve_code.drop(ll_f,axis=0)

    df_seve_test_x=np.array(df_seve_test.drop(['ClaimAmount'],axis=1))
    df_seve_test_y=np.array(df_seve_test['ClaimAmount'])


    import torch
    from torch.utils.data import TensorDataset, DataLoader
    from torchvision import datasets
    from torchvision.transforms import ToTensor
    import torch.nn as nn

    X_train = torch.from_numpy(df_seve_trainval_x).float()
    y_train = torch.from_numpy(df_seve_trainval_y).float()
    X_test = torch.from_numpy(df_seve_test_x).float()
    y_test = torch.from_numpy(df_seve_test_y).float()

    ##Define a batch size ,
    bs = 64

    #Both x_train and y_train can be combined in a single TensorDataset, which will be easier to iterate over and slice
    y_train_tensor = y_train.unsqueeze(1)
    train_ds = TensorDataset(X_train, y_train_tensor)
    #Pytorch’s DataLoader is responsible for managing batches.
    #You can create a DataLoader from any Dataset. DataLoader makes it easier to iterate over batches
    train_dl = DataLoader(train_ds, batch_size=bs)

    #y_test2_tensor = y_test2.unsqueeze(1)
    test_ds = TensorDataset(X_test, y_test)
    test_dl = DataLoader(test_ds, batch_size=32)

    n_input_dim = X_train.shape[1]

    #Layer size
    n_hidden1 = 40  # Number of hidden nodes
```

```
293.      n_hidden2 = 20
294.      n_output =  1   # Number of output nodes = for binary c
   lassifier
295.
296.
297.      class ChurnModel(nn.Module):
298.          def __init__(self):
299.              super(ChurnModel, self).__init__()
300.              self.layer_1 = nn.Linear(n_input_dim, n_hidden1
   )
301.              self.layer_2 = nn.Linear(n_hidden1, n_hidden2)
302.              self.layer_out = nn.Linear(n_hidden2, n_output)

303.
304.
305.              self.relu = nn.ReLU()
306.              self.softplus =  nn.Softplus()
307.              self.dropout = nn.Dropout(p=0.1)
308.              self.batchnorm1 = nn.BatchNorm1d(n_hidden1)
309.              self.batchnorm2 = nn.BatchNorm1d(n_hidden2)
310.
311.
312.
313.          def forward(self, inputs):
314.              x = self.relu(self.layer_1(inputs))
315.              x = self.batchnorm1(x)
316.              x = self.relu(self.layer_2(x))
317.              x = self.batchnorm2(x)
318.              x = self.dropout(x)
319.              x = self.softplus(self.layer_out(x))
320.
321.              return x
322.
323.      model = ChurnModel()
324.      print(model)
325.
326.      from sklearn import metrics
327.      #Loss Computation
328.      loss_function = nn.MSELoss()
329.      #Optimizer
330.      learning_rate = 0.001
331.      optimizer = torch.optim.Adam(model.parameters(), lr=lea
   rning_rate)
332.      epochs = 50
333.
334.      model.train()
335.      train_loss = []
336.      for epoch in range(epochs):
337.          #Within each epoch run the subsets of data = batch
   sizes.
338.          for xb, yb in train_dl:
```

```
            y_pred = model(xb)            # Forward Propagation
            loss = loss_function(y_pred, yb)  # Loss Computation
            optimizer.zero_grad()         # Clearing all previous gradients, setting to zero
            loss.backward()               # Back Propagation
            optimizer.step()              # Updating the parameters
        #print("Loss in iteration :"+str(epoch)+" is: "+str(loss.item()))
        train_loss.append(loss.item())
    print('Last iteration loss value: '+str(loss.item()))

    import itertools

    y_pred_list = []
    model.eval()
    #Since we don't need model to back propagate the gradients in test set we use torch.no_grad()
    # reduces memory usage and speeds up computation
    with torch.no_grad():
        for xb_test, yb_test in test_dl:
            y_test_pred = model(xb_test)
            y_pred_list.append(y_test_pred.detach().numpy())

    #Takes arrays and makes them list of list for each batch
    y_pred_list = [a.squeeze().tolist() for a in y_pred_list]
    #flattens the lists in sequence
    y_test_pred = list(itertools.chain.from_iterable(y_pred_list))

    metrics.mean_squared_error(y_test_pred, df_seve_test_y)
    #have a glips of predicted, there is no negative
    y_test_pred[1:15]
```

- sklearn

```
1. ###########scikit learn##############
2.
3.
4. ############freq######################
5. from sklearn.preprocessing import StandardScaler
6. #from sklearn.model_selection import train_test_split
7. from sklearn.model_selection import GridSearchCV
8. from sklearn.neural_network import MLPRegressor
9. from sklearn.metrics import mean_squared_error
10.import pandas as pd
11.from sklearn.preprocessing import OneHotEncoder
12.
13.#sample index(to have the same sampling with R)
14.ll = pd.read_csv('/Users/liying/Desktop/sample1.csv',sep=";")
15.ll=ll.x #transforme to serie
16.ll_f=ll-1
17.
18.
19.#read datas
20.data = pd.read_csv('/Users/liying/Desktop/data_freq.csv',sep=
   ";",decimal = ',')
21.data=data.drop(['IDpol'],axis=1)
22.data['freq']=data['ClaimNb']/data['Exposure']
23.data=data.drop(['ClaimNb','Exposure'],axis=1)
24.
25.
26.#standerlization learn
27.data_n=data[["freq","VehPower","DrivAge","VehAge","DrivAge","
   BonusMalus","Density"]]
28.scaler = StandardScaler()
29.scaled_dataset = scaler.fit_transform(data_n)
30.
31.data_s=pd.DataFrame(scaled_dataset,index=None,columns = ["fre
   q","VehPower","DrivAge","VehAge","DrivAge","BonusMalus","Dens
   ity"])
32.df_freq=pd.concat([data_s,data[["VehBrand","VehGas","Area","R
   egion","brandS","areaS","regionS"]]],axis=1)
33.
34.
35.df_freq.info()
36.#encoding
37.
38.#gas
39.ohe = OneHotEncoder()
40.gas_ohe = ohe.fit_transform(df_freq[["VehGas"]]).toarray()
41.gas_ohe_df = pd.DataFrame(gas_ohe, columns= ohe.get_feature_n
   ames())
42.df_freq=pd.concat([df_freq,gas_ohe_df],axis = 1)
43.
```

```
#areaS
ohe = OneHotEncoder()
areaS_ohe = ohe.fit_transform(df_freq[["areaS"]]).toarray()
areaS_ohe_df = pd.DataFrame(areaS_ohe, columns= ohe.get_feature_names())
df_freq=pd.concat([df_freq,areaS_ohe_df],axis = 1)

#brandS
ohe = OneHotEncoder()
brandS_ohe = ohe.fit_transform(df_freq[["brandS"]]).toarray()
brandS_ohe_df = pd.DataFrame(brandS_ohe, columns= ohe.get_feature_names())
df_freq=pd.concat([df_freq,brandS_ohe_df],axis = 1)

#region
ohe = OneHotEncoder()
regionS_ohe = ohe.fit_transform(df_freq[["regionS"]]).toarray()
regionS_ohe_df = pd.DataFrame(regionS_ohe, columns= ohe.get_feature_names())
df_freq=pd.concat([df_freq,regionS_ohe_df],axis = 1)


df_freq_code=df_freq.drop(['VehGas','Area','Region','VehBrand','brandS','areaS','regionS'],axis=1)

#to keep the same train data and test data, i would use the random number of R.
df_freq_trainval=df_freq_code.iloc[ll_f]

df_freq_trainval_x=df_freq_trainval.drop(['freq'],axis=1)
df_freq_trainval_y=df_freq_trainval['freq']

df_freq_test=df_freq_code.drop(ll_f,axis=0)
df_freq_test_y=df_freq_test

#gonna use GridSearchCV


#X_train ,X_val,y_train,y_val = train_test_split(df_freq_trainval_x,df_freq_trainval_y,random_state=1)


mlpr = MLPRegressor(max_iter=10000)
param_list = {"hidden_layer_sizes": [(20,30),(10,10)], "activation": ["identity", "tanh", "relu"], "solver": ["adam"], "alpha": [0.0001]}
gridCV = GridSearchCV(estimator=mlpr, param_grid=param_list,cv=2,scoring='neg_mean_squared_error',verbose=10)


```

```
gridCV.fit(df_freq_trainval_x, df_freq_trainval_y)

"""

######severidad#################


#sample index(to have the same sampling with R)
ll = pd.read_csv('/Users/liying/Desktop/sample1.csv',sep=";")
ll=ll.x #transforme to serie
ll_f=ll-1


#read datas
data = pd.read_csv('/Users/liying/Desktop/data_seve_without.csv',sep=";",decimal = ',')
    #data=data.drop(['IDpol'],axis=1)
    #data['fre']=data['ClaimNb']/data['Exposure']
    #data=data.drop(['ClaimNb','Exposure'],axis=1)




    #standerlization learn
    data_n=data[["ClaimAmount","VehPower","DrivAge","VehAge","DrivAge","BonusMalus","Density"]]
    scaler = StandardScaler()
    scaled_dataset = scaler.fit_transform(data_n)

    data_s=pd.DataFrame(scaled_dataset,index=None,columns = ["ClaimAmount","VehPower","DrivAge","VehAge","DrivAge","BonusMalus","Density"])
    df_seve=pd.concat([data_s,data[["VehBrand","VehGas","Area","Region","brandS","areaS","regionS"]]],axis=1)


    df_seve.info()
    #encoding

    #gas
    ohe = OneHotEncoder()
    gas_ohe = ohe.fit_transform(df_seve[["VehGas"]]).toarray()
    gas_ohe_df = pd.DataFrame(gas_ohe, columns= ohe.get_feature_names())
    df_seve=pd.concat([df_seve,gas_ohe_df],axis = 1)

    #areaS
    ohe = OneHotEncoder()
```

```
    areaS_ohe = ohe.fit_transform(df_seve[["areaS"]]).toarray()
    areaS_ohe_df = pd.DataFrame(areaS_ohe, columns= ohe.get_feature_names())
    df_seve=pd.concat([df_seve,areaS_ohe_df],axis = 1)

    #brandS
    ohe = OneHotEncoder()
    brandS_ohe = ohe.fit_transform(df_seve[["brandS"]]).toarray()
    brandS_ohe_df = pd.DataFrame(brandS_ohe, columns= ohe.get_feature_names())
    df_seve=pd.concat([df_seve,brandS_ohe_df],axis = 1)

    #region
    ohe = OneHotEncoder()
    regionS_ohe = ohe.fit_transform(df_seve[["regionS"]]).toarray()
    regionS_ohe_df = pd.DataFrame(regionS_ohe, columns= ohe.get_feature_names())
    df_seve=pd.concat([df_seve,regionS_ohe_df],axis = 1)


    df_seve_code=df_seve.drop(['VehGas','Area','Region','VehBrand','brandS','areaS','regionS'],axis=1)

    #to keep the same train data and test data, i would use the random number of R.
    df_seve_trainval=df_seve_code.iloc[ll_f]

    df_seve_trainval_x=df_seve_trainval.drop(['ClaimAmount'],axis=1)
    df_seve_trainval_y=df_seve_trainval['ClaimAmount']

    df_seve_test=df_seve_code.drop(ll_f,axis=0)
    df_seve_test_x=df_seve_test.drop(['ClaimAmount'],axis=1)



    df_seve_test_y=df_seve_test['ClaimAmount']

    #gonna use GridSearchCV


    #X_train ,X_val,y_train,y_val = train_test_split(df_freq_trainval_x,df_freq_trainval_y,random_state=1)


    mlpr = MLPRegressor(max_iter=10000)
```

```
167.      param_list = {"hidden_layer_sizes": [(20,30),(10,10)],
   "activation": ["identity", "tanh", "relu"], "solver": ["adam"
   ], "alpha": [0.0001]}
168.      gridCV = GridSearchCV(estimator=mlpr, param_grid=param_
   list,cv=2,scoring='neg_mean_squared_error',verbose=10)
169.
170.
171.      gridCV.fit(df_seve_trainval_x, df_seve_trainval_y)
```