Machine Learning en Python

Comprobacion de un lista con Machine Learning

Alexandre Navarro Fernandez

Indice:

1.¿En que consistirá el trabajo?

Este trabajo consistirá en coger una base de datos llamada “creditcard.csv” donde se almacenan 284807 transacciones, catalogadas como “Legitimas” y “Fraudulentas”. De esta base de datos lo que intentaremos hacer es descubrir de manera automatica si los datos de fraude o legitimos son correctos. Para ello utilizaremos Machine Learning con la biblioteca Scikit-Learn

2.¿Que es Machine Learning?

Machine Learning (*o Aprendizaje Autónomo*) es una rama de la Inteligencia Artificial que tiene como objetivo crear sistemas capaces de aprender por ellos mismos a partir de un conjunto de datos (data set), sin ser programados de forma explícita.

2.1.¿Como funciona Machine Learning?

Machine Learning funciona mediante algoritmos matemáticos complejos que tienen la capacidad de identificar patrones en conjuntos de datos. Gracias a la identificación de patrones, los algoritmos de Machine Learning son capaces de sacar conclusiones a partir de nuevos datos para los que no han sido preparados, aplicando patrones similares a los previamente identificados.

Mediante este reconocimiento de patrones, los algoritmos de Machine Learning son aplicados a un sinfín de sistemas para la realización de análisis predictivos o la generación de respuestas inteligentes y automáticas.

El reconocimiento de patrones que realizan los algoritmos de Machine Learning no dejan de ser algo similar a lo que hacen las formulas estadísticas. Todo se reduce al análisis de enormes cantidades de datos

y a la aplicación de la probabilidad para calcular cuál es el resultado más factible.

2.2. ¿Para que sirve el Machine Learning?

Las aplicaciones de streaming de vídeo o musica como Netflix o Spotify usan algoritmos de Machine Learning para las recomendaciones personalizadas. Los asistentes virtuales capaces de responder a preguntas realizadas por humanos como Alexa o Siri, probablemente son el ejemplo más claro de Machine Learning. Sin embargo, esta tecnología también es empleada para la optimización de los resultados de motores de búsqueda como Google, para el funcionamiento de robots o vehículos autónomos, para la prevención de enfermedades o para la creación de antivirus que detectan softwares maliciosos.

3. Librerias necesarias

3.1. Numpy

3.1.1. ¿Que es Numpy?

NumPy es un módulo de Python. El nombre es un acrónimo de Python Numérico. Es una librería que consiste en objetos de matrices multidimensionales y una colección de rutinas para procesar esas matrices.

Es un módulo de extensión para Python, escrito en su mayor parte en C. Esto asegura que las funciones y funcionalidades matemáticas y

numéricas precompiladas de NumPy garantizan una gran velocidad de ejecución.

NumPy es un paquete de procesamiento de matrices de uso general. Proporciona un objeto de matriz multidimensional de alto rendimiento, y herramientas para trabajar con estas matrices. Es el paquete fundamental para la computación científica con Python. Además de sus obvios usos científicos, NumPy también puede ser usado como un eficiente contenedor multidimensional de datos genéricos.

3.1.2. ¿Para que nos servira Numpy en el proyecto?

Para acceder a la informacion que nos proporcionara el csv necesitamos Pandas, pero para poder desglosarla de la manera que necesitaremos mas adelante, sera necesario usar Numpy

3.2. Pandas

3.2.1. ¿Que es Pandas?

Pandas es una librería en Python que se especializa en el manejo, análisis y procesamiento de datos. Para ello, se basa en las estructuras de datos o arrays de la librería NumPy (por lo que representa una dependencia al momento de instalar Pandas), siendo tres las estructuras que tenemos disponibles en esta librería: Series, DataFrame y Panel.

3.2.2. ¿Para que nos servira Pandas en el proyceto?

En este caso, Pandas nos es muy útil, ya que tenemos que acceder al csv y a bastante informacion.

3.3. Scikit-Learn

3.3.1. ¿Que es Scikit-Learn?

Scikit-Learn (o ***sklearn***) es una libreria gratuita para Python. Cuenta con algoritmos de clasificación, regresión, clustering (*El clustering es una tarea que tiene como finalidad principal lograr el agrupamiento de conjuntos de objetos no etiquetados, para lograr construir subconjuntos de datos conocidos como Clusters*)y reducción de dimensionalidad. Además, presenta la compatibilidad con otras librerías de Python como Numpy y MatPlotLib.

La gran variedad de algoritmos y utilidades de Scikit-learn la convierten en la herramienta básica para empezar a programar y estructurar los sistemas de análisis de datos y modelado estadístico. Los algoritmos de Scikit-Learn se combinan y depuran con otras estructuras de datos y aplicaciones externas como Pandas o PyBrain.

3.3.2. Para que nos servira Scikit-Learn en el proyecto?

Es la parte primordial de este proyecto, Scikit-Learn nos permitira coger dos vertientes de datos(“Training data” y “Testing data”) y comprobar que dichos datos son correctos o no

4.Herramientas a utilizar

4.1.Jupiter Notebook

Jupyter Notebook es una aplicación web de código abierto que nos permite crear y compartir código y documentos. Es un entorno informático interactivo, que permite a los usuarios experimentar con el código y compartirlo.

4.2.Pagina web Kaggle

Es una pagina web de uso gratuito donde los usuarios colocan bases de datos que el resto puede usar.

5.Funcionamiento

En primer lugar deberemos instalar e importar las bibliotecas que vamos a necesitar, en este caso son: Numpy, Pandas y sklearn.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Una vez instalado lo ejecutamos para poder tenerlo en toda la pagina.

Debemos acceder ahora al csv “creditcard.csv”, para ello crearemos la variable “credit_card_datos” y utilizaremos panda para tenerlo.

```
1 #Cargamos la base de datos en pandas
2
3 credit_card_datos = pd.read_csv("Projecte/creditcard.csv")
4 credit_card_datos
```

Ahora podemos observas el csv y podemos ver los datos de la misma.

Vemos que esta la columna “Time” que refleja el tiempo transcurrido desde la primera transaccion en segundos hasta la ultima que son “172792.0” segundos (2 dias aprox desde la primera transaccion).

Los “V1”,”V2”,”V3”etc… equivalen a distintas caracteristicas dadas por un resultado ACP(analisis de componentes principales) debido a temas de confidencialidad solo se ven en esta forma.

Y la mas importante es “Class” donde solo tiene dos valores 0 y 1, donde 0 es una transaccion legitima y 1 es una fraudulenta.

Podemos utilizar las palabras reservadas “.head()” para ver los 5 primeros datos, o usar “.tail()” para revisar las 5 ultimas. Con todos estos datos no podemos hacer nada. Y para que nuestro programa funcione necesitamos ver que no hay ningun dato nulo. Para ello podemos optar por vertientes:

- Utilizando un “.info”(): Esta opcion nos enseña las columnas, su tipo y lo mas importante si hay algun valor nulo.

```
1 credit_card_datos.info()
```

- Utilizando “.isnull().sum()”: De esta manera se ve mas sencillamente si hay algun dato nulo o no.

```
1 credit_card_datos.isnull().sum()
```

En cualquiera de los dos casos vemos que no hay datos vacios, esto nos ayuda bastante, ya que sino tendriamos que utilizar algun metodo para rellenar esos vacios.

Lo siguiente que tendriamos que hacer es dividir del csv las clases que sean legitimas y las fraudulentas.

```
1 credit_card_datos['Class'].value_counts()
```

```
0    284315
1       492
Name: Class, dtype: int64
```

El numero de transacciones legitimas (0) es mucho mayor que las fraudulentas(1), por lo que si hicieramos una prediccion ahora con estos datos los datos saldrian mal ya que no hay una base solida.

Separaremos en dos variables llamadas “legitimo” y ”fraude” los datos anteriores.

```
1 legitimo = credit_card_datos[credit_card_datos.Class == 0]
2 fraude = credit_card_datos[credit_card_datos.Class == 1]
```

```
1 print (legitimo.shape)
2 print(fraude.shape)
```

```
(284315, 31)
(492, 31)
```

Necesitamos averiguar en que se basa un dato legitimo de uno fraudulento, por lo que usaremos “.Amount.describe()” en legitimo y fraude. A ver si podemos ver algo raro

```
1 legitimo.Amount.describe()
```

```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

```
1 fraude.Amount.describe()
```

```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

Al parecer la media de legitimo es menor que la de fraudulento. Por lo que deduciremos que eso es lo que los diferencia, pero para estar mas seguros hagamos una comparativa.

```
1 credit_card_datos.groupby("Class").mean()
```

Class	Time	V1	V2	V3	V4	V5	V6
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737

2 rows × 30 columns

Nuestra deduccion era correcta. Vemos que la cantidad movida en los datos legitimos ronda el 0,01 mientras que el fraudulento puede alcanzar el -7.

Como hemos dicho antes la cantidad de datos esta desbalanceada por lo que tendremos que hacer algo al respecto.

Sabemos que el numero de transacciones fraudulentas es 492, asi que lo que haremos sera conseguir 492 datos de transacciones legitimas y juntarlas con las 492 fraudulentas para poder hacer bien la estimacion. ".sample" elige de manera aleatoria los datos que vamos a darle, en este caso 492

```
1 legitimo_muestra = legitimo.sample(n=492)
```

Ahora juntaremos los dos datos, los 492 legitimos y los 492 fraudulentos

```
1 nuevo_dataset = pd.concat([legitimo_muestra, fraude],axis=0)
```

Comprobamos que los datos son completamente aleatorios.

```
1 nuevo_dataset.head()
```

	Time	V1	V2	V3	V4	V5	V6
9180	13128.0	0.961947	-1.271395	1.189481	-0.277193	-1.606945	0.261155
184332	126200.0	2.021865	0.561760	-2.486630	0.566489	0.840658	-1.186985
53239	45879.0	1.114807	0.041645	1.524413	1.460157	-1.167109	-0.484459
113612	73136.0	-0.956779	-0.027089	0.806662	-2.578570	-0.683281	-0.854688
40003	40053.0	-0.463442	0.927609	1.150332	0.919851	-0.092954	-0.651403

5 rows × 31 columns

Solo para estar seguros, vamos a comprobar que hay 492 de los dos datos.

```
1 nuevo_dataset["Class"].value_counts()
```

```
0    492
1    492
Name: Class, dtype: int64
```

Aqui podemos ver la media de los dos grupos de datos el fraudulento se mantiene y el legitimo a cambiado, tal como esperabamos.

```
1 nuevo_dataset.groupby("Class").mean()
```

Class	Time	V1	V2	V3	V4	V5	V6
0	93465.349593	0.030275	-0.098138	0.018524	-0.057107	0.148592	-0.090099
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737

2 rows × 30 columns

Una vez concluido esto toca la parte mas complicada, como norma general las variables nunca van en mayusculas, al parecer en cuestion de Machine Learning se tuiliza X e Y en mayuscula para dhacer ver que son matrices, si fueran las dos vectores serian en minusculas. Aparte de este pequeño cambio de paradigma crearemos las matrices “X” e “Y” donde “X” creara las columnas con los valores e “Y” dira si es fraudulento o no.

```python
X = nuevo_dataset.drop(columns="Class",axis=1)
Y = nuevo_dataset["Class"]
```

```python
print(X)
```

```
            Time        V1        V2        V3        V4        V5        V6
\
50087    44341.0 -0.099405 -0.281612  1.326792 -2.339705 -0.968656 -0.845396
268472  163228.0 -0.308791  0.193386  0.641245 -1.708744 -0.746617 -0.123684
60410    49313.0 -1.519004 -0.329718  1.476657  0.592387 -1.331166  0.306263
154327  101197.0  1.699514 -0.800349 -0.608646  0.270992  0.093692  1.431693
155154  104624.0 -2.341095 -0.842396  1.662913  4.921741  2.632680 -0.175099
...          ...       ...       ...       ...       ...       ...       ...
279863  169142.0 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.010494
280143  169347.0  1.378559  1.289381 -5.004247  1.411850  0.442581 -1.326536
280149  169351.0 -0.676143  1.126366 -2.213700  0.468308 -1.120541 -0.003346
281144  169966.0 -3.113832  0.585864 -5.399730  1.817092 -0.840618 -2.943548
281674  170348.0  1.991976  0.158476 -2.583441  0.408670  1.151147 -0.096695

              V7        V8        V9  ...       V20       V21       V22  \
50087  -0.304870 -0.019740 -2.136373  ... -0.298042 -0.118719  0.155577
268472 -0.099552  0.435856  0.837829  ...  0.047623  0.163372  0.389234
60410  -0.587518  0.535070 -0.125331  ... -0.019144  0.360706  1.027767
154327 -0.919691  0.422854  2.176877  ... -0.132582  0.174694  0.732779
155154  0.216154 -0.218014 -1.124101  ... -0.014834 -0.038600  0.855644
...          ...       ...       ...  ...       ...       ...       ...
279863 -0.882850  0.697211 -2.064945  ...  1.252967  0.778584 -0.319189
280143 -1.413170  0.248525 -1.127396  ...  0.226138  0.370612  0.028234
280149 -2.234739  1.210158 -0.652250  ...  0.247968  0.751826  0.834108
281144 -2.208002  1.058733 -1.632333  ...  0.306271  0.583276 -0.269209
281674  0.223050 -0.068384  0.577829  ... -0.017652 -0.164350 -0.295135

              V23       V24       V25       V26       V27       V28  Amount
50087    0.064860  0.371240 -0.325096 -0.397804  0.327785  0.104573    5.00
268472   0.185260 -0.652067 -0.639169  0.500633 -0.086430 -0.008693   88.61
60410   -0.062327  0.141663 -0.900968  1.256780 -0.108411  0.034388  156.40
154327   0.114535 -1.653269 -0.560037  0.602429 -0.029339 -0.064803  102.00
155154   1.328431 -0.400086  0.980239  0.714039  0.003183 -0.339099    0.00
...           ...       ...       ...       ...       ...       ...     ...
279863   0.639419 -0.294885  0.537503  0.788395  0.292680  0.147968  390.00
280143  -0.145640 -0.081049  0.521875  0.739467  0.389152  0.186637    0.76
280149   0.190944  0.032070 -0.739695  0.471111  0.385107  0.194361   77.89
281144  -0.456108 -0.183659 -0.328168  0.606116  0.884876 -0.253700  245.00
281674  -0.072173 -0.450261  0.313267 -0.289617  0.002988 -0.015309   42.53

[984 rows x 30 columns]
```

```
1 print(Y)
```

```
50087     0
268472    0
60410     0
154327    0
155154    0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

Ahora separaremos estos en "Training data" y "Testing data", para ello usaremos el “train_test_split” de “sklearn”, tenemos que crear 4 variables (en este caso son variables, no matrices pero mantiene la mayuscula).

```
1 X_train, X_test, Y_train,Y_test = train_test_split(X, Y,
2                                                    test_size=0.2,
3                                                    stratify=Y,
4                                                    random_state=2)
```

El “0.2” representa que el 20% de los datos iran a la parte de “Testing data” y el 80% ira al “Training data”.Straify ordenara de forma conjunta los datos.

```
1 print(X.shape, X_train.shape, X_test.shape)
```

```
(984, 30) (787, 30) (197, 30)
```

Aqui podemos ver que de los 984 datos, 787(80%) se ha ido a training y 197(20%) a test.

Toca crear un modelo para conseguir saber si los datos que hemos metido son ciertos y si puede distinguir los fraudulentos de los legitimos.

```
1 modelo = LogisticRegression()
```

Ahora lo vamos a entrenar con los "Training data". ".fit" cogera las variables de los "Training data"

```
1 modelo.fit(X_train, Y_train)
```

```
LogisticRegression()
```

Aunque no se vea nada, el entrenamiento ha finalizado, pero ahora lo importante, ¿como de acertado estan los datos de la base de datos?

Someteremos a nuestro modelo a una evaluacion tanto de entrenamiento como de Test.

```
1 X_train_prediccion = modelo.predict(X_train)
2 training_data_accuracy = accuracy_score(X_train_prediccion, Y_train)
```

```
1 print("Aciertos en el 'Training Data': ", training_data_accuracy)
```

```
Aciertos en el 'Training Data':  0.9085133418043202
```

```
1 X_test_prediccion = modelo.predict(X_test)
2 test_data_accuracy = accuracy_score(X_test_prediccion, Y_test)
```

```
1 print("Aciertos en el 'Test Data': ", test_data_accuracy)
```

```
Aciertos en el 'Test Data':  0.8984771573604061
```

Vemos que el resultado en "Training data" es de 0.90, y que el de "Test data" es 0.89. Si los porcentajes fueran mas diferentes significaria que algo a fallado y que no serian correctos los datos puesto, demostrando que la lista no es 100% fiable, no obstante el porcentaje de test y training se asemeja mucho, por lo que podemos determinar que los 984 datos que metimos aleatoriamente (492 fraudulentos/492legitimos) ha conseguido determinar que son ciertos, por lo que podemos deducir que la lista entera es correcta.