Proyecto 3 Supervivencia del Titanic

Thursday, April 30, 2020 3:41 PM

Este proyecto es un clasificador para identificar la supervivencia del titanic, para obtener la información de los DataSets es necesario contar con una cuenta en Kaggle y descargar los csv de la siguiente página: https://www.kaggle.com/c/titanic/data

En está ocasión se cuenta con 2 archivos CSV uno de datos de entrenamiento "train.csv"

	A	B	C	D	E	F	G	H	I	J	K	L
1	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr. (	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, Mr	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen, N	female	26	0	0	STON/O2. 31	7.925		S
5	4	1	1	Futrelle, Mrs	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr. W	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, Mr. J	male		0	0	330877	8.4583		Q
8	7	0	1	McCarthy, M	male	54	0	0	17463	51.8625	E46	S
9	8	0	3	Palsson, Mas	male	2	3	1	349909	21.075		S
10	9	1	3	Johnson, Mr	female	27	0	2	347742	11.1333		S
11	10	1	2	Nasser, Mrs.	female	14	1	0	237736	30.0708		C

Así se cuenta con otro archivo que tiene los datos de prueba "test.csv"

	A	B	C	D	E	F	G	H	I	J	K
1	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	892	3	Kelly, Mr. Ja	male	34.5	0	0	330911	7.8292		Q
3	893	3	Wilkes, Mrs.	female	47	1	0	363272	7		S
4	894	2	Myles, Mr. T	male	62	0	0	240276	9.6875		Q
5	895	3	Wirz, Mr. Alb	male	27	0	0	315154	8.6625		S
6	896	3	Hirvonen, M	female	22	1	1	3101298	12.2875		S
7	897	3	Svensson, M	male	14	0	0	7538	9.225		S
8	898	3	Connolly, Mi	female	30	0	0	330972	7.6292		Q
9	899	2	Caldwell, Mr	male	26	1	1	248738	29		S
10	900	3	Abrahim, Mr	female	18	0	0	2657	7.2292		C
11	901	3	Davies, Mr. J	male	21	2	0	A/4 48871	24.15		S

Los datos importantes entre ambos archivos son los siguientes:

- Se cuenta con un Id de pasajero
- En el archivo "test.csv" no cuenta con la columna "Survived"
- La columna respetiva a sexo son cadenas de texto
- La columna respecto a embarque tiene las letras
  - Q=>Queenstown
  - S=>Southampton
  - C=>Cherbourg

Para iniciar con la programación se debe contar con las siguientes 4 librerías instaladas

- Scikit-Learn
- Pandas
- Seaborn
- Matplotlib

Estás últimas 2 son para realizar gráficos

```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
```

También se hará uso de 4 modelos de machine Learning para comprobar el mejor para el siguiente problema, los modelos a usar son:

- Regresión Logística
- Máquina de Vectores de Soporte
- Vecinos más cercanos
- Árbol de Decisiones

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
```

Comenzamos a leer ambos archivos CSV e imprimir las primeras 5 líneas y así iniciar el acomodo de la información

```
train=pd.read_csv('train.csv')
print(train.head())
```

```
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
1            2         1       1  ...  71.2833   C85         C
2            3         1       3  ...   7.9250   NaN         S
3            4         1       1  ...  53.1000  C123         S
4            5         0       3  ...   8.0500   NaN         S

[5 rows x 12 columns]
```

```
test=pd.read_csv('test.csv')
print(test.head())
```

```
   PassengerId  Pclass  ... Cabin Embarked
0          892       3  ...   NaN        Q
1          893       3  ...   NaN        S
2          894       2  ...   NaN        Q
3          895       3  ...   NaN        S
4          896       3  ...   NaN        S

[5 rows x 11 columns]
```

Como se demuestra en el archivo de entrenamiento se tienen un total de 12 columnas y en el de prueba se cuenta con 11 columnas.

Ahora se tiene que verificar la cantidad de datos en los archivos

```
#Verificar la cantidad de datos en el dataset
print("\nCantidad de datos: ")
print(train.shape)
print(test.shape)
```

```
Cantidad de datos:
(891, 12)
(418, 11)
```

En el archivo de entrenamiento se tiene un total de 891 filas con 12 columnas, mientras que en el de prueba se cuenta con 418 filas y 11 columnas, se tiene que verificar que tipo de datos se cuenta en ambos archivos, por lo cual accederemos a esta información con el siguiente comando

```
#Verificamos la información que hay en el Dataset
print("\nTipos de datos: ")
print(train.info())
print(test.info())
```

La información de train.csv es:

```
Tipos de datos:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 66.2+ KB
None
```

La información en test.csv

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Pclass       418 non-null    int64
 2   Name         418 non-null    object
 3   Sex          418 non-null    object
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 27.8+ KB
None
```

En ambos Datasets se identifica que la mayoria de las columnas son de tipo numerico como interger y flotante, a excepción del Nombre, Sexo, Ticket, Cabin y Embarked

Se debe verificar la cantidad de valores faltantes por columna, para hacer esto se usa el siguiente comando

```
#Verificamos si hay algún dato faltante
print("\nDatos faltantes: ")
print(pd.isnull(train).sum())
print(pd.isnull(test).sum())
```

Del archivo train.csv

```
Datos faltantes:
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

De este Dataset se cuenta con un total de 177 datos perdidos en la edad, 687 en Cabin y 2 en Embarke

Del archivo test.csv

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

Cuenta con 86 datos faltantes en la edad, 1 en Fare y 327 en Cabin

Por último se imprimirá una descripción de las características estadísticas de ambos archivos

```
#Descripción del data set
print("\nEstadisticas del Dataset: ")
print(train.describe())
print(test.describe())
```

De train.csv

```
Estadisticas del Dataset:
       PassengerId    Survived      Pclass  ...       SibSp       Parch        Fare
count   891.000000  891.000000  891.000000  ...  891.000000  891.000000  891.000000
mean    446.000000    0.383838    2.308642  ...    0.523008    0.381594   32.204208
std     257.353842    0.486592    0.836071  ...    1.102743    0.806057   49.693429
min       1.000000    0.000000    1.000000  ...    0.000000    0.000000    0.000000
25%     223.500000    0.000000    2.000000  ...    0.000000    0.000000    7.910400
50%     446.000000    0.000000    3.000000  ...    0.000000    0.000000   14.454200
75%     668.500000    1.000000    3.000000  ...    1.000000    0.000000   31.000000
max     891.000000    1.000000    3.000000  ...    8.000000    6.000000  512.329200

[8 rows x 7 columns]
```

De test.csv

```
       PassengerId      Pclass         Age       SibSp       Parch        Fare
count   418.000000  418.000000  332.000000  418.000000  418.000000  417.000000
mean   1100.500000    2.265550   30.272590    0.447368    0.392344   35.627188
std     120.810458    0.841838   14.181209    0.896760    0.981429   55.907576
min     892.000000    1.000000    0.170000    0.000000    0.000000    0.000000
25%     996.250000    1.000000   21.000000    0.000000    0.000000    7.895800
50%    1100.500000    3.000000   27.000000    0.000000    0.000000   14.454200
75%    1204.750000    3.000000   39.000000    1.000000    0.000000   31.500000
max    1309.000000    3.000000   76.000000    8.000000    9.000000  512.329200
```

A partir de este momento se modificaran los datos de nuestra data tanto de entrenamiento como de prueba, en la columna "Sex" se modificara los objetos female y male por 0 y 1 de la siguiente manera

```
#Cambiar el sexo de los pasajeros
train['Sex'].replace(['female','male'],[0,1],inplace=True)
test['Sex'].replace(['female','male'],[0,1],inplace=True)
```

Continuamos al modificar la información de la columna "Embarked" que contiene los objetos Q, S y C por 0, 1 y 2 de la misma manera que sucedió con la columna "Sex"

```
#Cambiamos los datos de embarque por numeros
train['Embarked'].replace(['Q','S','C'],['0','1','2'],inplace=True)
test['Embarked'].replace(['Q','S','C'],['0','1','2'],inplace=True)
```

Recordemos que la Columna "Age" cuenta con muchos datos nulos, por lo tanto hay que llenarlos con el valor medio, para eso escribimos el comando

```
#Se cambian los valores de la edad faltantes por la media
print(train["Age"].mean())
print(test["Age"].mean())
```

Se presentan 2 valores, el primero es la media de edad en el archivo "train" y el segundo es la media de edad en el archivo "test"

```
29.69911764705882
30.272590361445783
```

Se usara un valor general para ambos archivos, es este caso se va a proponer una media de 30 de Edad y se modificarán todos aquellos valores nulos por este nuevo valor usando el comando de replace

```
media=30
train["Age"]=train["Age"].replace(np.nan,media)
test["Age"]=test["Age"].replace(np.nan,media)
```

Al analizar la data se encontraron algunas filas con datos perdidos, por lo cual deben ser eliminadas con el siguiente comando

```
#Se elimia las filas con datos perdidos
train.dropna(axis=0,how='any',inplace=True)
test.dropna(axis=0,how='any',inplace=True)
```

En la data existen columnas que son la información del pasajero como su Id, nombre, ticket y cabin, pero para el modelo son irrelevantes, por lo tanto se eliminaran para este análisis

```
#Se elimina las columnas PassengerId, Name, Ticket & Cabin
train=train.drop(["PassengerId","Name","Ticket","Cabin"],axis=1)
test=test.drop(["Name","Ticket","Cabin"],axis=1)
```

Para facilitar el análisis en la edad de los pasajeros se formaran 7 grupos en los que se pondrán un rango de edades que van de 0 a 8, 9 a 15, 16 a 18, 19 a 25, 26 a 40, 41 a 60 y 61 a 100 y estos se modificaran con la función "cut"

```
#Se crean varios grupos segun la edad
#Bandas 0-8,9-15,16-18,19-25,26-40,41-60,61-100
bins=[0,8,15,18,25,40,60,100]
names=['1','2','3','4','5','6','7']
train['Age']=pd.cut(train["Age"],bins,labels=names)
test['Age']=pd.cut(test["Age"],bins,labels=names)
```

Antes de implementar los algoritmos de machine learning, se verificaran los datos para que todo se encuentre en buenas condiciones

```
#Verificamos los datos
print(pd.isnull(train).sum())
print(pd.isnull(test).sum())
```

No se cuenta con ninguna inf[illegible]hivo de entrenamiento y prueba

```
Survived    0
Pclass      0
Sex         0
Age         0
```

```
PassengerId    0
Pclass         0
Sex            0
Age            0
SibSp          0
```

```
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

```
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

La cantidad de datos se muestra con el atributo shape

```
print(train.shape)
print(train.shape)
```

Son 202 filas con 8 columnas para ambos archivos

```
(202, 8)
(202, 8)
```

Por último se imprimen las primeras 5 filas de nuestros archivos

```
print(train.head())
print(test.head())
```

Train

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
1	1	1	0	5	1	0	71.2833	2
3	1	1	0	5	1	0	53.1000	1
6	0	1	1	6	0	0	51.8625	1
10	1	3	0	1	1	1	16.7000	1
11	1	1	0	6	0	0	26.5500	1

Test

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
12	904	1	0	4	1	0	82.2667	1
14	906	1	0	6	1	0	61.1750	1
24	916	1	0	6	1	3	262.3750	2
26	918	1	0	4	0	1	61.9792	2
28	920	1	1	6	0	0	30.5000	1

A partir de este momento se inicia el proceso de machine Learning y lo importante es dividir la información en X e Y, por lo tanto Y serán los pasajeros que sobrevivieron y X las variables restantes, el test será un 80% de train y el 20% test

```
X=np.array(train.drop(['Survived'],1))
y=np.array(train['Survived'])

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

El primer modelo a utilizar él es de Regresión logística

```
#-------------------
#Regresión Logística
#-------------------
logreg=LogisticRegression()
logreg.fit(X_train,y_train)
Y_pred=logreg.predict(X_test)
print("Presición de la regresión logistica: ",logreg.score(X_train,y_train))
```

Seguidamente del modelo de Maquinas de Vectores de Soporte

```
#------------------------------
#Maquinas de Vectores de Soporte
#------------------------------
MSV=SVC()
MSV.fit(X_train,y_train)
Y_pred_MSV=MSV.predict(X_test)
print('Presición del algoritmo Maquinas de Vectores de Soporte es: {}'.format(MSV.score(X_train,y_train)))
```

Continuamos con el modelo de vecinos más cercanos con un total de 3 vecinos

```
#-------------------
#Vecinos más Cercanos
#-------------------
VC=KNeighborsClassifier(n_neighbors=3)
VC.fit(X_train,y_train)
Y_pred_VC=VC.predict(X_test)
print('Presición del algoritmo Vecinos más Cercanos es: {}'.format(VC.score(X_train,y_train)))
```

Y para finalizar el modelo de Árbol de decisiones

```
#-------------------
#Arbol de Decisiones
#-------------------
AD=DecisionTreeClassifier()
AD.fit(X_train,y_train)
Y_pred_AD=AD.predict(X_test)
print('Presición del algoritmo Árbol de Decisiones es: {}'.format(AD.score(X_train,y_train)))
```

La precisión de cada modelo se presenta a continuación

```
Presición de la regresión logistica:  0.8198757763975155
Presición del algoritmo Maquinas de Vectores de Soporte es: 0.6770186335403726
Presición del algoritmo Vecinos más Cercanos es: 0.8136645962732919
Presición del algoritmo Árbol de Decisiones es: 0.9813664596273292
```

Como se puede apreciar en el resultado, el mejor algoritmo es el de árbol de decisiones con un 98% de exactitud en los resultados, ahora que se tienen los modelos entrenamos, se hará la predicción con el archivo test.csv, para lo cual se crea una variable llamada "ids" la cual va a contar con la información de la columna "PassengerId" y otra "sex" que tendrá la información de la columna "Sex"

```
#Ya contamos con los modelos entrenados, ahora usamos el csv de prueba
ids=test['PassengerId']
sex=test['Sex']
```

En ese momento se pone a prueba que modelo tiene más supervivientes en el titanic, iniciamos con la Regresión logistica, está predicción se hará con la data del test.csv pero eliminando la primera columna, luego se creará un nuevo dataframe en el cual solo tendrá 3 columnas, la primera será el "PassengerId" con la información de ids, la segunda de "survived" con la predicción y la tercera será el sexo del pasajero.

Además se modificarán los resultados, en "Survived" se cambiará el 0 por N y el 1 por Y, mientras que en la columna "Sex" se cambiará el 0 por F y el 1 por M

```
#-------------------
#Regresión Logística
#-------------------
prediccion_logreg=logreg.predict(test.drop('PassengerId',axis=1))
out_logreg=pd.DataFrame({'PassengerId':ids,'Survived':prediccion_logreg,'Sex':sex})
out_logreg['Sex'].replace([0,1],['F','M'],inplace=True)
out_logreg['Survived'].replace([0,1],['N','Y'],inplace=True)
print("Predicción de la regresión logistica: \n",out_logreg.head())
```

Seguidamente se realiza el mismo proceso con el algoritmo Maquinas de vectores de soporte

```
#-------------------------------
#Maquinas de Vectores de Soporte
#-------------------------------
prediccion_MSV=MSV.predict(test.drop('PassengerId',axis=1))
out_MSV=pd.DataFrame({'PassengerId':ids,'Survived':prediccion_MSV,'Sex':sex})
out_MSV['Sex'].replace([0,1],['F','M'],inplace=True)
out_MSV['Survived'].replace([0,1],['N','Y'],inplace=True)
print("Predicción de mauina de vectores de soporte: \n",out_MSV.head())
```

Damos paso al 3 algoritmo Vecinos más cercanos

```
#--------------------
#Vecinos más Cercanos
#--------------------
prediccion_VC=VC.predict(test.drop('PassengerId',axis=1))
out_VC=pd.DataFrame({'PassengerId':ids,'Survived':prediccion_VC,'Sex':sex})
out_VC['Sex'].replace([0,1],['F','M'],inplace=True)
out_VC['Survived'].replace([0,1],['N','Y'],inplace=True)
print("Predicción de Vecinos más Cercanos: \n",out_VC.head())
```

Finalizando con el algoritmo de Árbol de decisiones

```
#------------------
#Arbol de Decisiones
#------------------
prediccion_AD=AD.predict(test.drop('PassengerId',axis=1))
out_AD=pd.DataFrame({'PassengerId':ids,'Survived':prediccion_AD,'Sex':sex})
out_AD['Sex'].replace([0,1],['F','M'],inplace=True)
out_AD['Survived'].replace([0,1],['N','Y'],inplace=True)
print("Predicción del Arbol de Decisiones: \n",out_AD.head())
```

Estos son los resultados al imprimir las primeras 5 filas de cada dataframe creado

```
Predicción de la regresión logistica:
     PassengerId  Survived
12           904         1
14           906         1
24           916         1
26           918         1
28           920         0
Predicción de mauina de vectores de soporte:
     PassengerId  Survived
12           904         1
14           906         1
24           916         1
26           918         1
28           920         1
Predicción de Vecinos más Cercanos:
     PassengerId  Survived
12           904         1
14           906         1
24           916         1
26           918         1
28           920         0
Predicción del Arbol de Decisiones:
     PassengerId  Survived
12           904         1
14           906         1
24           916         1
26           918         1
28           920         0
```

A pesar de que el algoritmo de Árbol de decisiones mostro una aproximación con mejor rendimiento, el segundo modelo presenta un mayor número de personas sobrevivientes en las primeras cinco líneas.
Por lo que en este momento se graficara el resultado de la simulación por el sexo de las personas y sobrevivientes.

En este código se considera el nuevo Dataframe de cada método de Machine Learning, estos se agruparán según su sexo y después si es un superviviente o no, el grafico se dividirá en 4 partes cada una correspondiente al modelo

```
fig,axs=plt.subplots(2,2)
out_logreg.groupby(['Sex','Survived']).size().unstack().plot(kind='bar',ax=axs[0,0])
out_VC.groupby(['Sex','Survived']).size().unstack().plot(kind='bar',ax=axs[0,1])
out_AD.groupby(['Sex','Survived']).size().unstack().plot(kind='bar',ax=axs[1,0])
out_MSV.groupby(['Sex','Survived']).size().unstack().plot(kind='bar',ax=axs[1,1])
for ax in axs.flat:
    ax.set(xlabel='Sexo',ylabel='Numero de personas')
axs[0,0].set_title('Regresión Logística')
axs[0,1].set_title('Maquina de vectores de soporte')
axs[1,0].set_title('Vecinos más cercanos')
axs[1,1].set_title('Árbol de Decisiones')
plt.show()
```

El gráfico se observa de la siguiente manera