



**Cristian Camilo Chaparro Africano**

**20092020017**

**BigData**

**Universidad Distrital Francisco José de Caldas**

**Procesamiento de datos – Conjunto de datos del Titanic**

## Requerimientos técnicos sobre Mac OS

Lo primero que se debe tener en cuenta es instalar las siguientes dependencias:

- brew install python3
- pip3 install jupyter jupyter\_kernel\_gateway
- pip3 install matplotlib
- pip3 install pandas
- pip3 install sklearn

## Procesamiento del conjunto de datos.

Lo primero que se debe realizar es leer el dataset , para lo cual se empleara la función `read_csv` de pandas, la cual cargará el archive `train.csv`. Con la función `head` se podrán pre-visualizar algunas de las filas del conjunto de datos como se puede apreciar en la siguiente imagen.

The screenshot shows a Jupyter Notebook titled "titanic\_notebook" with a toolbar and a code cell. The code cell contains the following Python code:

```
In [1]: import pandas as pd
In [2]: import numpy as np
In [3]: import matplotlib.pyplot as plt
In [4]: %matplotlib inline
In [5]: training_set = pd.read_csv('./dataset/train.csv')
In [6]: training_set.head()
```

The output of the last cell is a table showing the first five rows of the dataset:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S



**Cristian Camilo Chaparro Africano**

**20092020017**

**BigData**

**Universidad Distrital Francisco José de Caldas**

**Procesamiento de datos – Conjunto de datos del Titanic**

## Información básica del conjunto de datos

La estructura `training_set` (`pandas.core.frame.DataFrame`) tiene varias propiedades entre ellas `shape`, la cual muestra el número filas y columnas. Por ejemplo la siguiente imagen indica que tiene 891 filas y 12 columnas.

```
In [7]: training_set.shape
```

```
Out[7]: (891, 12)
```

Si se quiere averiguar los títulos de las columnas se puede acceder a la propiedad `columns`

```
In [8]: training_set.columns
```

```
Out[8]: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',  
              'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],  
              dtype='object')
```

Si se quiere averiguar el número de valores nulos por columna se debe llamar a la función `isnull` y luego sumarizar todos los valores con la función `sum`

```
In [9]: training_set.isnull().sum()
```

```
Out[9]: 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
```

En la imagen anterior se puede apreciar que en la columna `Age` hay 177 registros nulos de la misma forma en la columna `Cabin` 687.

## Preparando el conjunto de datos

Se deben crear nuevas clases, en este caso se creará una nueva columna `Deck`, la cual nos indica si la columna



**Cristian Camilo Chaparro Africano**

**20092020017**

**BigData**

**Universidad Distrital Francisco José de Caldas**

**Procesamiento de datos – Conjunto de datos del Titanic**

```
In [17]: def assignDeckValue(CabinCode):  
         if pd.isnull(CabinCode):  
             category = 'Unknow'  
         else:  
             category = CabinCode[0]  
         return category  
  
deck = np.array([assignDeckValue(cabin) for cabin in training_set['Cabin'].values])  
training_set = training_set.assign(Deck= deck)
```

```
In [18]: training_set.head()
```

Out[18]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Deck
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	Unknow
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	Unknow
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	Unknow

Ahora se procede a crear la clase Family size, la cual indicará el número de hermanos y padres en conjunto.

```
In [19]: training_set['FamilySize'] = training_set['SibSp'] + training_set['Parch'] + 1
```

```
In [20]: training_set.head()
```

Out[20]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Deck	FamilySize
	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	Unknow	2
	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	C	2
	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	Unknow	1
	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	C	2
	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	Unknow	1



**Cristian Camilo Chaparro Africano**

**20092020017**

**BigData**

**Universidad Distrital Francisco José de Caldas**

**Procesamiento de datos – Conjunto de datos del Titanic**

Ahora se creará la clase Title la cual indicara si es Sr, Ms, entre otros.

```
In [23]: training_set.head()
```

Out[23]:

erId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Deck	FamilySize	Title
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	Unknow	2	Mr
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	C	2	Mrs
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	Unknow	1	Miss
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	C	2	Mrs
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	Unknow	1	Mr

Ahora que se proceso la información como se requiere se proce a borrar las columnas Cabin y Name

```
In [24]: training_set.drop(['Cabin', 'Name'], 1, inplace=True)
```

```
In [25]: training_set.head()
```

Out[25]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Deck	FamilySize	Title
0	1	0	3	male	22.0	1	0	A/5 21171	7.2500	S	Unknow	2	Mr
1	2	1	1	female	38.0	1	0	PC 17599	71.2833	C	C	2	Mrs
2	3	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	S	Unknow	1	Miss
3	4	1	1	female	35.0	1	0	113803	53.1000	S	C	2	Mrs
4	5	0	3	male	35.0	0	0	373450	8.0500	S	Unknow	1	Mr

## Controlando valores Faltantes

Para la clase Embarked, faltan algunos valores, los valores que puede tomar esta son C=Cherbourg, Q=Queenstown, S = Southampton, en este caso se procede para los valores



**Cristian Camilo Chaparro Africano**

**20092020017**

**BigData**

**Universidad Distrital Francisco José de Caldas**

**Procesamiento de datos – Conjunto de datos del Titanic**

faltantes asignar S que es el mas comun. Luego se realiza una contabilidad para verificar que no hay valores nulos para esta columna.

```
In [26]: # Returns count for each category
training_set['Embarked'].value_counts()

# Fills null values with 'S'-most common occurrence
common = 'S'
training_set['Embarked'] = training_set['Embarked'].fillna('S')

# Checking the no of null values now
training_set['Embarked'].isnull().sum()
```

Out[26]: 0

Para la clase Age, se realiza un procedimiento similar, estableciendo el valor medio a partir del titulo.

```
n [27]: means = training_set.groupby('Title')['Age'].mean()

title_list = ['Master', 'Miss', 'Mr', 'Mrs', 'Others']
def age_missing_replace(means, dframe, title_list):
    for title in title_list:
        temp = dframe['Title'] == title
        dframe.loc[temp, 'Age'] = dframe.loc[temp, 'Age'].fillna(means[title])

age_missing_replace(means, training_set, title_list)
```

## Codificación de características categóricas.

Muchos algoritmos de aprendizaje automático no pueden admitir valores categóricos sin convertirse en valores numéricos. Para lo cual se encarga de usar la función map la cual asignara un valor numerico a cada columna dependiendo cómo se haya definido.



**Cristian Camilo Chaparro Africano**

**20092020017**

**BigData**

**Universidad Distrital Francisco José de Caldas**

**Procesamiento de datos – Conjunto de datos del Titanic**

```
In [29]: training_set['Embarked'] = training_set['Embarked'].map({'C':0, 'Q':1, 'S':2})
training_set['Sex'] = training_set['Sex'].map({'male':0, 'female':1})
training_set['Title'] = training_set['Title'].map({'Master':0, 'Miss':1, 'Mr':2, 'Mrs':3, 'Others':
```

```
In [30]: training_set.head()
```

Out[30]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Deck	FamilySize	Title
0	1	0	3	0	22.0	1	0	A/5 21171	7.2500	2	Unknow	2	2
1	2	1	1	1	38.0	1	0	PC 17599	71.2833	0	C	2	3
2	3	1	3	1	26.0	0	0	STON/O2. 3101282	7.9250	2	Unknow	1	1
3	4	1	1	1	35.0	1	0	113803	53.1000	2	C	2	3
4	5	0	3	0	35.0	0	0	373450	8.0500	2	Unknow	1	2

Asignar valores de forma manual, no es la mejor opción ya que se puede dejar datos de forma inconsistente y el tiempo empleado puede ser alto depende del volumen de datos a procesar. En este caso se empleara LabelEncoder de sklear para realizar la conversión de la columna Deck. En la siguiente imagen se aprecia la asignación de valores numéricos por categoría de forma aleatoria a la columna Deck.



**Cristian Camilo Chaparro Africano**

**20092020017**

**BigData**

**Universidad Distrital Francisco José de Caldas**

**Procesamiento de datos – Conjunto de datos del Titanic**

```
In [30]: training_set.head()
```

Out[30]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Deck	FamilySize	Title
0	1	0	3	0	22.0	1	0	A/5 21171	7.2500	2	Unknow	2	2
1	2	1	1	1	38.0	1	0	PC 17599	71.2833	0	C	2	3
2	3	1	3	1	26.0	0	0	STON/O2. 3101282	7.9250	2	Unknow	1	1
3	4	1	1	1	35.0	1	0	113803	53.1000	2	C	2	3
4	5	0	3	0	35.0	0	0	373450	8.0500	2	Unknow	1	2

```
In [31]: from sklearn import preprocessing
```

```
In [32]: le = preprocessing.LabelEncoder()
```

```
In [33]: training_set['Deck'] = le.fit_transform(training_set['Deck'])
```

```
In [34]: training_set.head()
```

Out[34]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Deck	FamilySize	Title
0	1	0	3	0	22.0	1	0	A/5 21171	7.2500	2	8	2	2
1	2	1	1	1	38.0	1	0	PC 17599	71.2833	0	2	2	3
2	3	1	3	1	26.0	0	0	STON/O2. 3101282	7.9250	2	8	1	1
3	4	1	1	1	35.0	1	0	113803	53.1000	2	2	2	3
4	5	0	3	0	35.0	0	0	373450	8.0500	2	8	1	2

Después de realizar este paso ya se encuentra lista la información sin ningún valor faltante.

```
In [36]: training_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Sex            891 non-null int64
Age           891 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Embarked       891 non-null int64
Deck           891 non-null int64
FamilySize     891 non-null int64
Title          891 non-null int64
dtypes: float64(2), int64(10), object(1)
memory usage: 90.6+ KB
```