Etapa 5 – Resultado de análisis de datos

Lili Yohana López Barrera

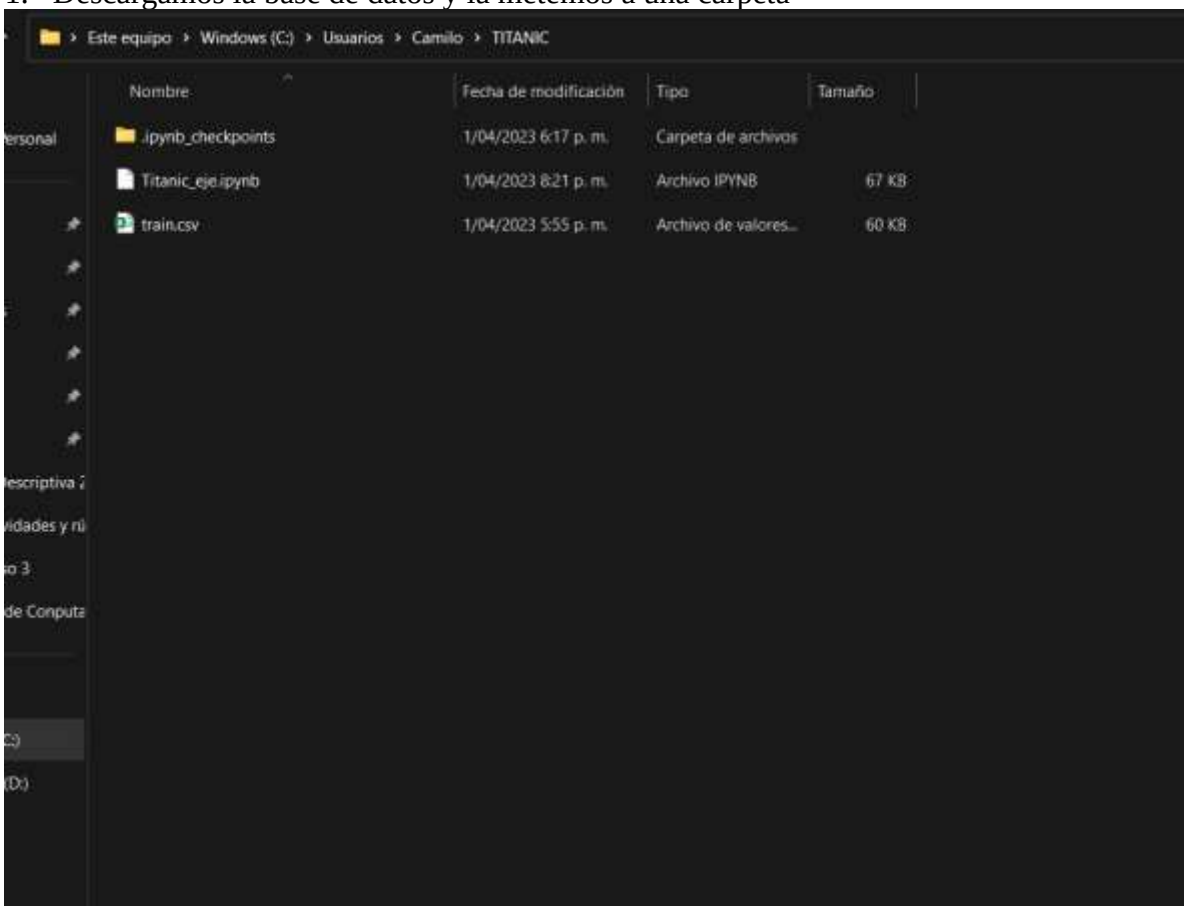
Grupo: 202016908_33

Tutor
Gloria Alejandra Rubio

Universidad Nacional Abierta y a Distancia-UNAD
Facultad de Ingeniería
Ingeniería de Sistemas
Sogamoso – 2023

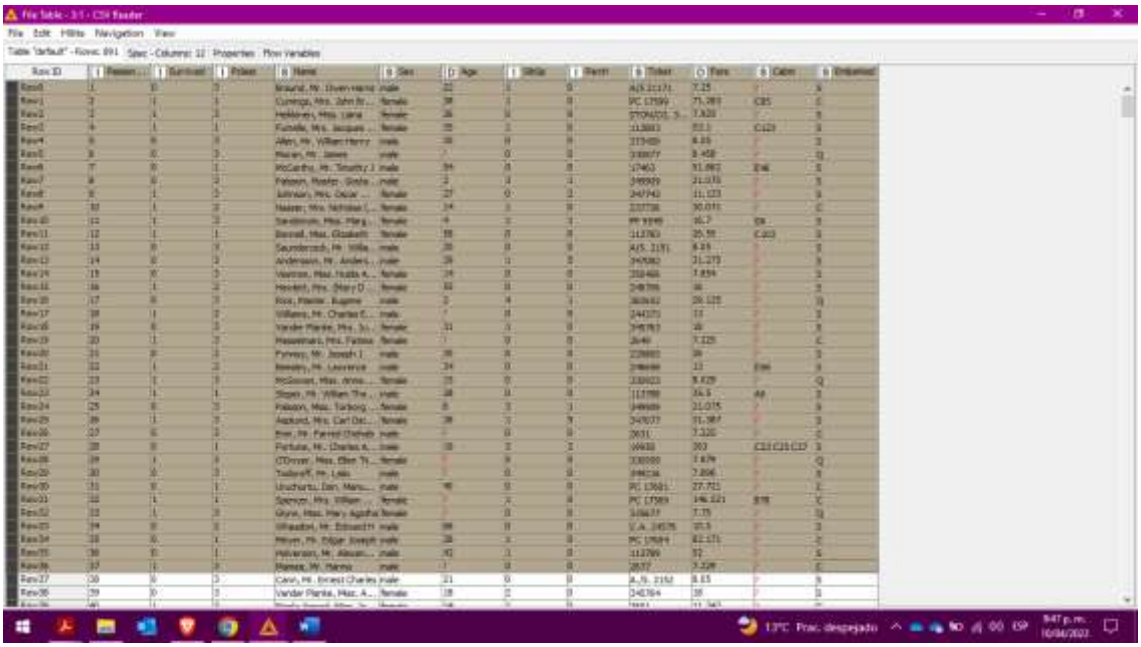
En la fase de análisis de datos, se cargó el conjunto de datos desde un archivo CSV llamado "train.csv". Se realizaron algunas transformaciones en los datos, como la eliminación de columnas innecesarias y el filtrado de filas con valores específicos. Estas acciones se llevaron a cabo para asegurar que el conjunto de datos sea adecuado y coherente con los requisitos del modelo.
Pasos para hacer el ejercicio

1. Descargamos la base de datos y la metemos a una carpeta



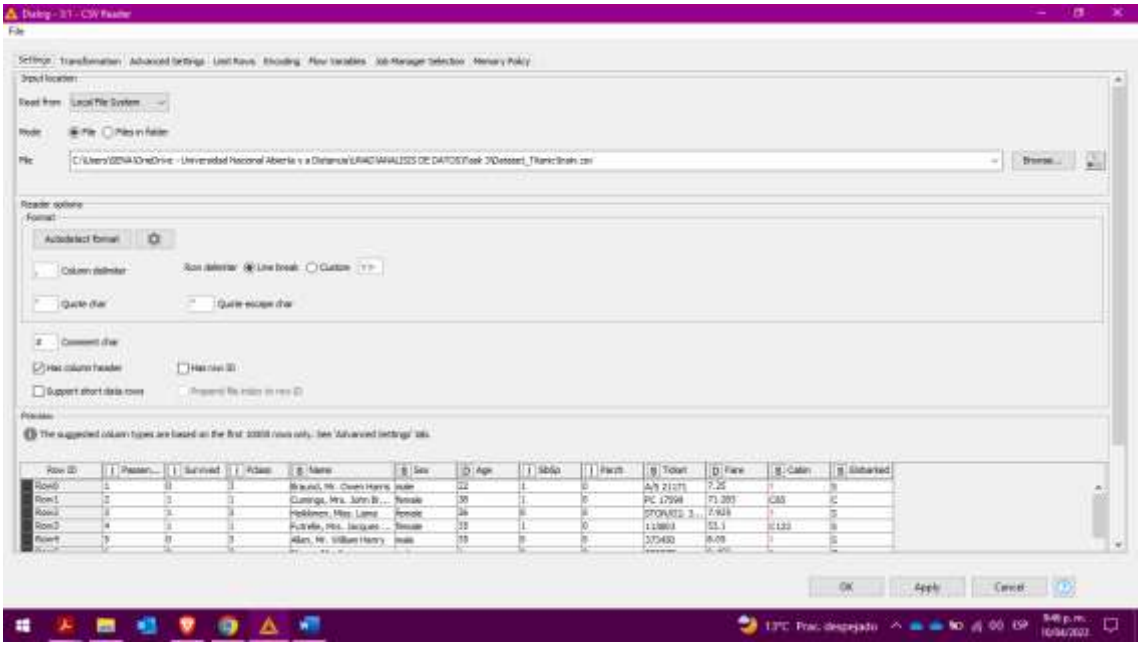
Preprocesamiento de Datos

Una vez cargados los datos, se procedió a dividir el conjunto en variables independientes (características) y la variable dependiente (resultado de titanic). A continuación, se realizó una división adicional en conjuntos de entrenamiento y prueba para evaluar el rendimiento del modelo. La división se hizo con una proporción del 70% para entrenamiento y 30% para prueba.

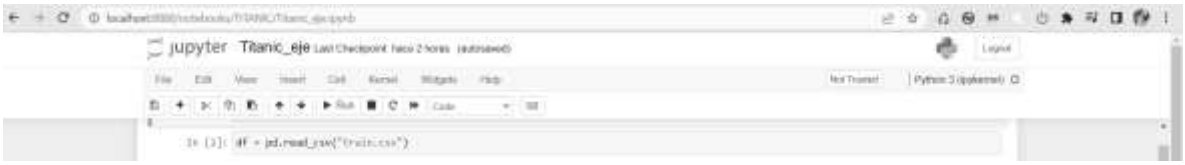


Entrenamiento del Modelo

El modelo de Naive Bayes, en su variante Gaussiana, se utilizó para entrenar el modelo de predicción. Se ajustó el modelo utilizando los datos de entrenamiento, lo que permitió capturar las relaciones probabilísticas entre las características y los resultados de TITANIC. Este enfoque se basa en la suposición de independencia condicional de las características dadas las clases.



En la segunda celda nos traemos o leemos la base de datos que está en el archivo csv.



En la tercera celda con el head imprimimos los 5 primeros registros del archivo

```
In [3]: df.head()
```

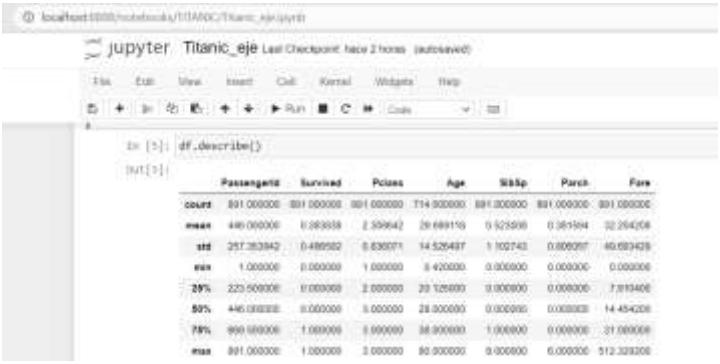
Out[3]:

	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

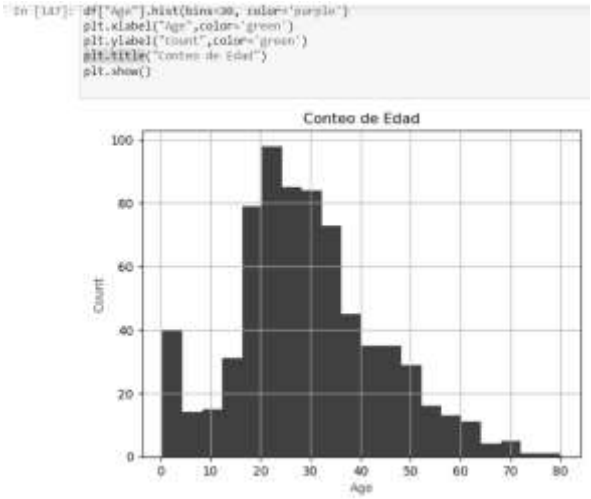
En la cuarta celda con el df.info() imprimimos el ddl de la tabla para saber la cantidad de campos, el nombre de los campos, si puede ser nulo, el conteo de los campos, el tipo del dato y el peso de toda la información.

```
In [4]: df.info()
<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: 83.7+ KB
```

En la quinta celda con el df.describe() lo que hacemos es que nos muestra unos resultados de unos cálculos como lo son el *count* que hace un conteo de cada campo, el *min* que muestra el valor más pequeño que hay en cada campo, y el *max* que me muestra el valor más grande de cada campo.

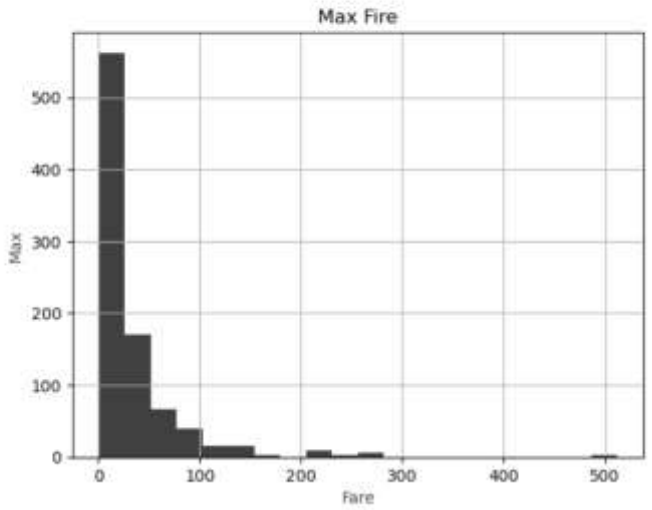


En la sexta celda lo que hacemos es graficar con el *hist* que es igual a histograma y le asignamos los valores de *x* con *xlabel* y de *y* con *ylabel* en nuestro caso *xlabel le asignamos el valor de Age* y a *ylabel* le asignamos el valor del *count* y luego imprimimos esto con el *show*, Se le agrego en esta parte colores y títulos para para que la gráfica se evidenciara más bonita para este caso se agregó a utilizar plt.title para el titulo y color para agregar colores.



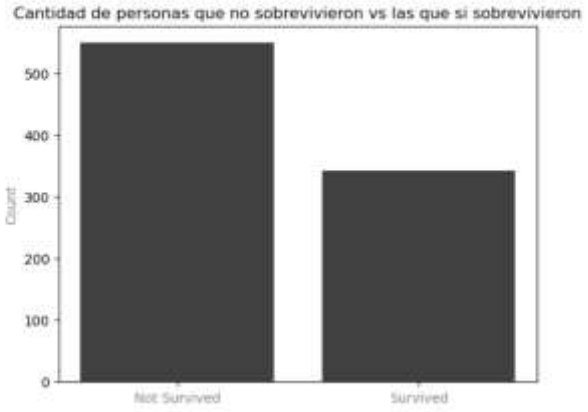
Es esta parte se creó un histograma el cual muestra el max Fire decorado con colores y poniendo le un título a la gráfica.

```
In [148]: df["Fare"].hist(bins=20, color='green')
          plt.xlabel("Fare", color='purple')
          plt.ylabel("Max", color='purple')
          plt.title("Max Fire")
          plt.show()
```



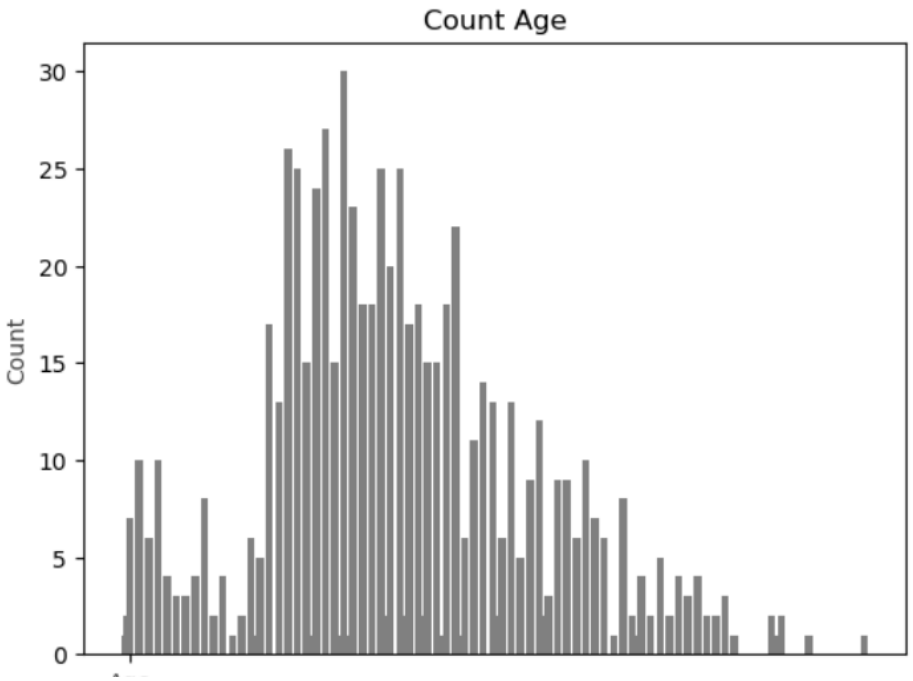
En la siguiente grafica generamos un conteo de las personas que no sobrevivieron versus las personas que si sobrevivieron dándonos como resultado la siguiente grafica. Se agrega colores y titulo para hacerla más presentable.

```
In [152]: survived_counts = df["Survived"].value_counts()
          plt.bar(survived_counts.index, survived_counts.values, color='green')
          plt.xticks([0, 1],["Not Survived", "Survived"], color='red')
          plt.ylabel("Count", color='red')
          plt.title("Cantidad de personas que no sobrevivieron vs las que si sobrevivieron")
          plt.show()
```



Se creo la siguiente grafica en la cual se hace un conteo uno por una de las edades se le agrega título y colores.

In [153]:
```
conteo_sobrevivientes = df["Age"].value_counts()
plt.bar(conteo_sobrevivientes.index, conteo_sobrevivientes.values, color='orange')
plt.xticks([1],["Age"], color='green')
plt.ylabel("Count", color='green')
plt.title("Count Age")
plt.show()
```



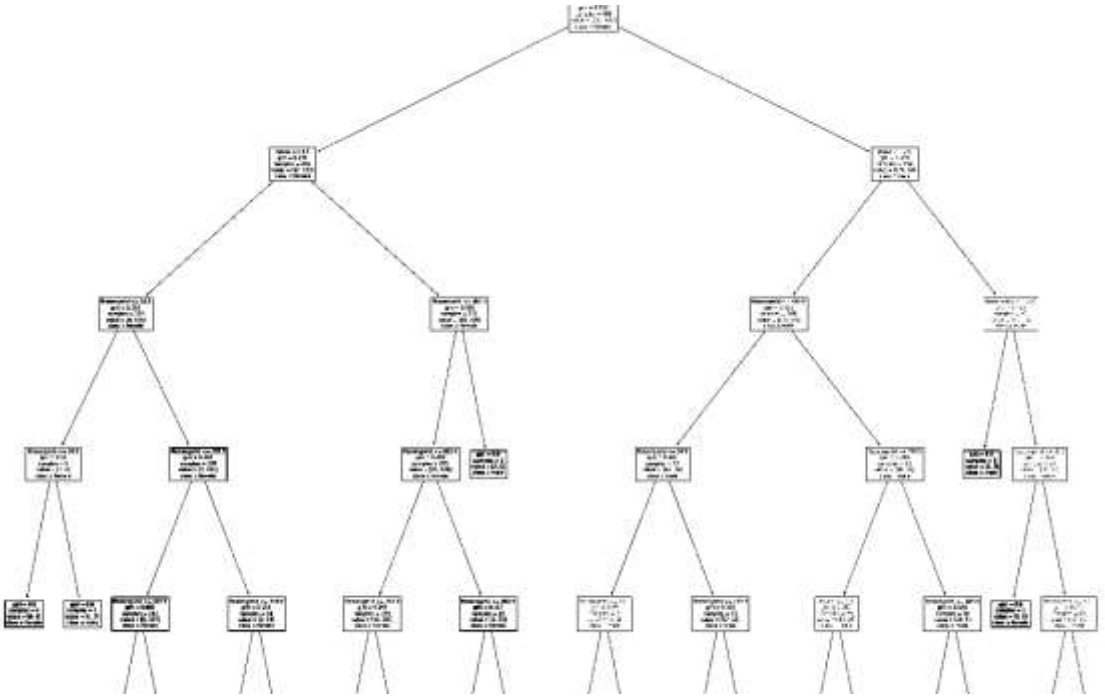
Se hace un árbol de decisiones para esta actividad guiándose de uno echo en grupo para una actividad anterior donde logramos comprender mejor el funcionamiento del código.

```
In [80]: X   df.iloc[:,0:3]

         Y   df.iloc[:,4]

         X.head()
```

Out[80]:

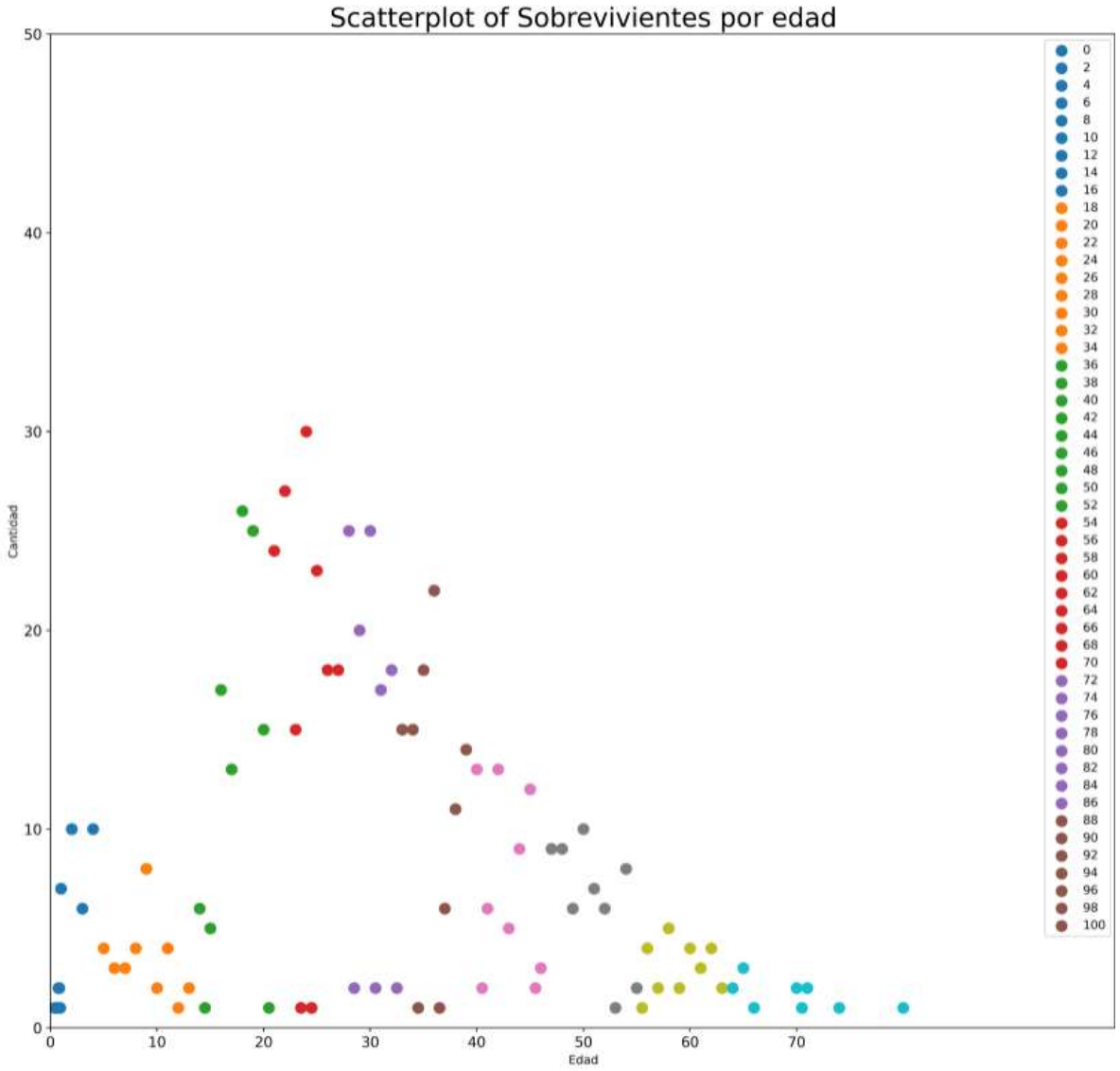
	PassengerId	Survived	Pclass
0	1	0	3
1	2	1	1
2	3	1	3
3	4	1	1
4	5	0	3

```
In [81]: from sklearn.model_selection import train_test_split

         X_train, X_test, Y_train, Y_test   train_test_split(X, Y, train_size 0.75, random_state 0)

In [82]: X_train.info()

         <class 'pandas.core.frame.DataFrame'>
         Int64Index: 668 entries, 105 to 684
         Data columns (total 3 columns):
          #   Column       Non-Null Count  Dtype
         ---  ------       --------------  -----
          0   PassengerId  668 non-null    int64
          1   Survived     668 non-null    int64
          2   Pclass       668 non-null    int64
         dtypes: int64(3)
         memory usage: 20.9 KB

In [83]: from sklearn.tree import DecisionTreeClassifier


         arbol   DecisionTreeClassifier(max_depth 5)

         arbol_bmedina   arbol.fit(X_train, Y_train)

In [84]: import matplotlib.pyplot as plt
         from sklearn import tree

         fig   plt.figure(figsize (25,20))

         tree.plot_tree(arbol_bmedina, feature_names list(X.columns.values),
                        class_names list(Y.values), filled True)
         plt.show()
```



Para no quedarnos con los conocimientos ya previamente adquiridos se realiza la investigación y se crea un Scatterplot de sobrevivientes por edad para visualizar la información de una manera diferente y aprendiendo a utilizar fors y iteración de la información adicional manejar nuevas cosas como plt.figure, unique(),df.loc, plt.legend entre otras, con este conocimiento adquirido podemos llegar a graficar la información.

```
In [146]: edades = np.unique(df["Age"])

colors = [plt.cm.tab10(i/float(len(edades)-1)) for i in range(len(edades))]



color2 = []

for color in colors:
    color2.append(mat.colors.to_hex(color,True))
#color2

plt.figure(dpi= 500, facecolor='w', edgecolor='k',figsize=(16, 15))
for i, age in enumerate(edades):

    plt.scatter(df.loc[df["Age"]==age, :]["Age"].unique(),df.loc[df["Age"]==age]["Age"].value_counts(),
                s=80, c=color2[i], label=str(age))
plt.gca().set(xlim=(0.0, 100), ylim=(0, 50),xlabel='Edad', ylabel='Cantidad')

plt.xticks(fontsize=12); plt.yticks(fontsize=12)
plt.xticks(np.arange(0, max(edades), 10))
plt.title("Scatterplot of Sobrevivientes por edad", fontsize=22)
plt.legend(fontsize=12)
plt.legend([i*2 for i in range(51)])
plt.show()
```



Enlace [github]

https://github.com/yohanal86/Analisis_de_datos