Take any Dataset of your choice ,perform EDA(Exploratory Data Analysis)

(Include classifier, regressor , clusterer and accuracy of the model)

MAJOR PROJECT

Sub line: AI-MAJOR-AUGUST-AI-08-SPB1

TEACHNOOK

SUBMITTED BY

EHILL SMILA.J

Introduction:

Data are growing very faster in today's world. It is not so easy to process the data manually. Data analysis and visualization programs allow for reaching even deeper understanding. The programming language Python, with its English commands and easy-to-follow syntax, offers an amazingly powerful (and free!) open-source alternative to traditional techniques and applications.

Data analytics allow businesses to understand their efficiency and performance, and ultimately helps the business make more informed decisions. For example, an e-commerce company might be interested in analyzing customer attributes in order to display targeted ads for improving sales.

Data analysis can be applied to almost any aspect of a business if one understands the tools available to process information. The ecommerce companies are analyzing the reviews of customer by using proper visualization method.

Exploratory Data Analysis (EDA) is an approach to summarize the data by taking their main characteristics and visualize it with proper representations. EDA focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, and handling missing values and making transformations of variables as needed. EDA encompasses IDA.

EDA quickly describes the data sets number of rows/columns, missing data, data types and preview. Clean corrupted data; handle missing data, invalid data types.

Exploratory Data Analysis (EDA)

- *EDA is applied to investigate the data and summarize the key insights.*
- *It will give you the basic understanding of your data, it's distribution, null values and much more.*
- *You can either explore data using graphs or through some python functions.*
- *There will be two type of analysis. Univariate and Bivariate. In the univariate, you will be analyzing a single attribute. But in the bivariate, you will be analyzing an attribute with the target attribute.*
- *In the non-graphical approach, you will be using functions such as shape, summary, describe, isnull, info, datatypes and more.*
- *In the graphical approach, you will be using plots such as scatter, box, bar, density and correlation plots.*

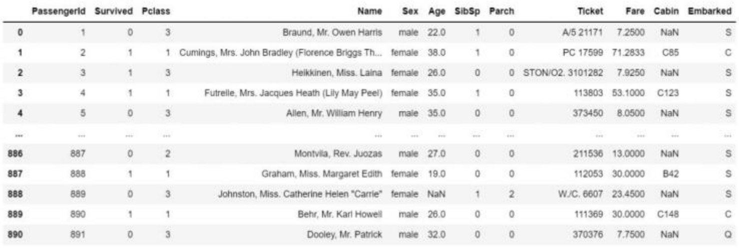
LOAD THE DATA:

We will load the titanic dataset into python to perform EDA.

*:::::::::::::(‘***titanic.csv***’):::::::::::::::*

```
#Load the required libraries

import pandas as pd

import numpy as np

import seaborn as sns

#Load the data

df = pd.read_csv('titanic.csv')

#View the data

df.head()
```

<u>Output:</u>



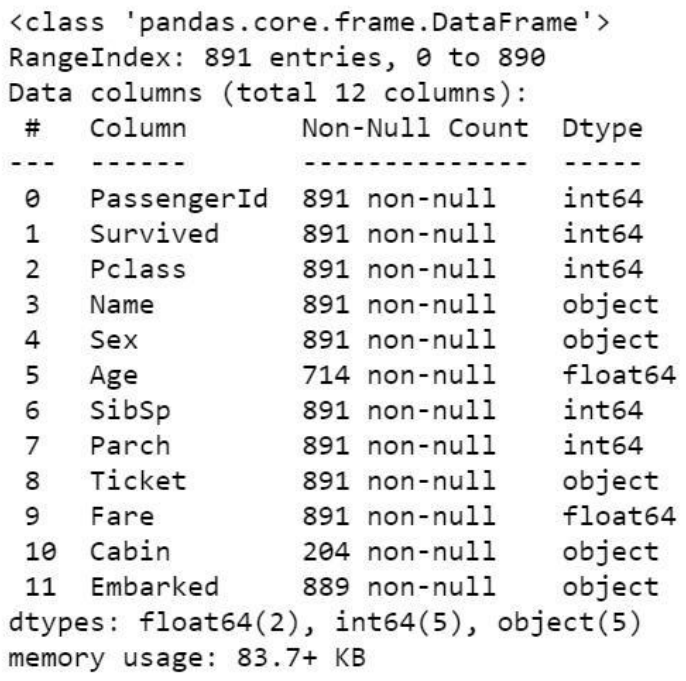
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 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
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

1.Basic information about data – <u>EDA</u>

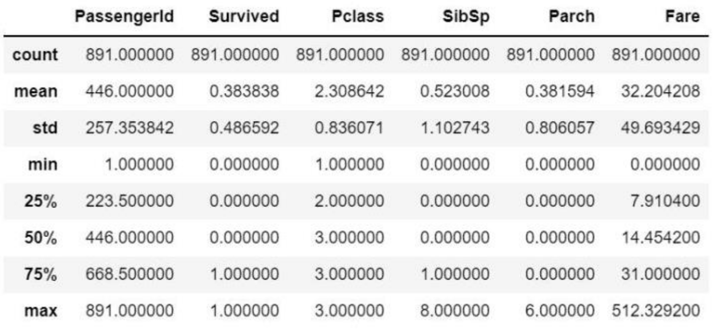
The df.info() function will give us the information about the dataset. For any data, it is good to start by knowing its information. Let's see how it works with our data.

```
#Basic information

df.info()

#Describe the data

df.describe()
```

<u>Output:</u>



```
<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
```



	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

Using this function, you can see the number of null values, datatypes, and memory usage as shown in the above outputs along with descriptive statistics.

2. Duplicate values

You can use the df.duplicate.sum() function to the sum of duplicate value present if any. It will show the number of duplicate values if they are present in the data.

```
#Find the duplicates

df.duplicated().sum()
```

Output:

0

Well, the function returned ‘0’. This means, there is not a single duplicate value present in our dataset and it is a very good thing to know.

3. Unique values in the data

You can find the number of unique values in the particular column using unique() function in python. The unique() function has returned the unique values which are present in the data and it is pretty much cool.

```
df[‘Pclass’].unique() #unique values

df[‘Survived’].unique()

df[‘Sex’].unique()
```

Output:

```
Array([3, 1, 2], dtype=int64)

Array([0, 1], dtype=int64)

Array([‘male’, ‘female’], dtype=object)
```

```
df['pclass'].nunique()

df['Survived'].nunique()

df['Sex'].nunique()
```

Output:

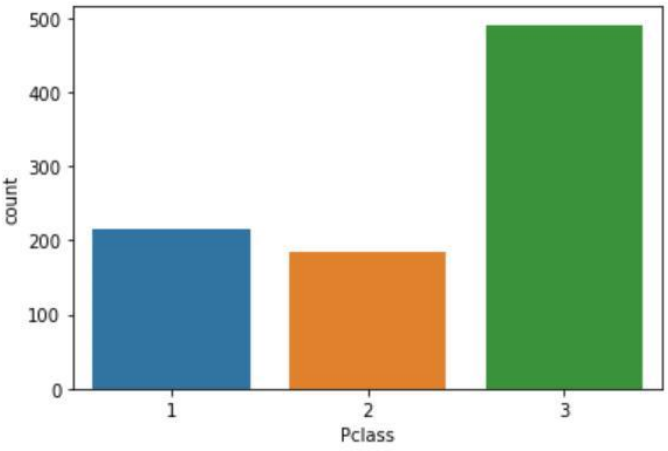
```
 3

 2

2
```

4. Visualize the Unique counts

Yes, you can visualize the unique values present in the data. For this, we will be using the seaborn library. You have to call the sns.countplot() function and specify the variable to plot the count plot.Though EDA has two approaches, a blend of graphical and non-graphical will give you the bigger picture altogether.

```
#Plot the unique values

sns.countplot(df['Pclass']).unique()
```

Output:



5. Find the Null values and replace null values

Finding the null values is the most important step in the EDA. As I told many a time, ensuring the quality of data is paramount. So, let's see how we can find the null value.

```
#Find null values

df.isnull().sum()
```

Output:

```
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
```

6.Replace the values:

We have some null values in the ‘Age’ and ‘Cabin’ variables.we got a replace() function to replace all the null values with a specific data. It is too good!

```
#Replace null values
df.replace(np.nan,’0’,inplace = True)
#Check the changes now
df.isnull().sum()
```

I have used 0 to replace null values.

<u>Output:</u>

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
Dtype: int64
```

7. Know the datatypes

Knowing the datatypes which you are exploring is very important and an easy process too.

```
#Datatypes

df.dtypes
```

Output:

```
PassengerId     int64

Survived        int64

Pclass          int64

Name            object

Sex             object

Age             object

SibSp           int64

Parch           int64

Ticket          object

Fare            float64

Cabin           object
```

Embarked object

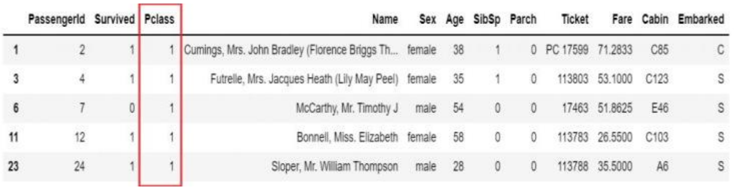
Dtype: object

8. Filter the Data

Yes, you can filter the data based on some logic.

```
#Filter data

df[df[‘Pclass’]==1].head()
```

<u>Output:</u>



	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	C85	C
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000	C123	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783	26.5500	C103	S
23	24	1	1	Sloper, Mr. William Thompson	male	28	0	0	113788	35.5000	A6	S

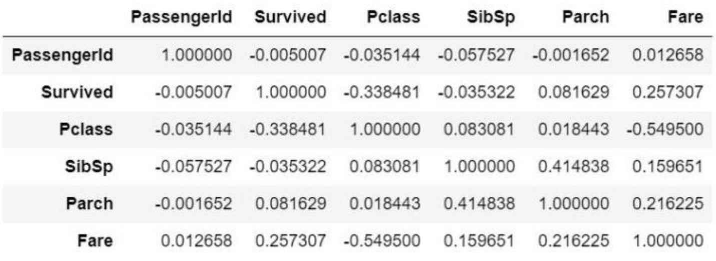
You *can see that the above code has returned only data values that belong to class 1.*

<u>9. Correlation Plot – EDA</u>

Finally, to find the correlation among the variables, we can make use of the correlation function. This will give you a fair idea of the correlation strength between different variables.

```
#Correlation

df.corr()
```

Output:

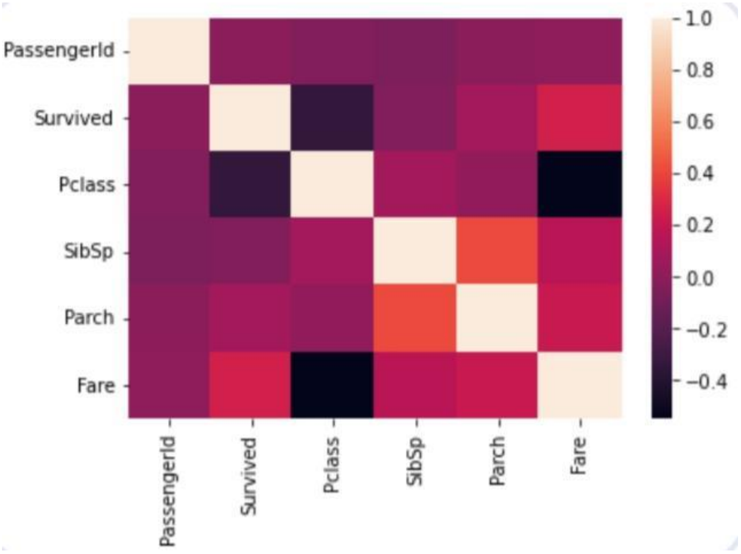


	PassengerId	Survived	Pclass	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	0.083081	0.018443	-0.549500
SibSp	-0.057527	-0.035322	0.083081	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.159651	0.216225	1.000000

This is the correlation matrix with the range from +1 to -1 where +1 is highly and positively correlated and -1 will be highly negatively correlated.You can even visualize the correlation matrix using seaborn library .

```
#Correlation plot
sns.heatmap(df.corr())
```

Output:

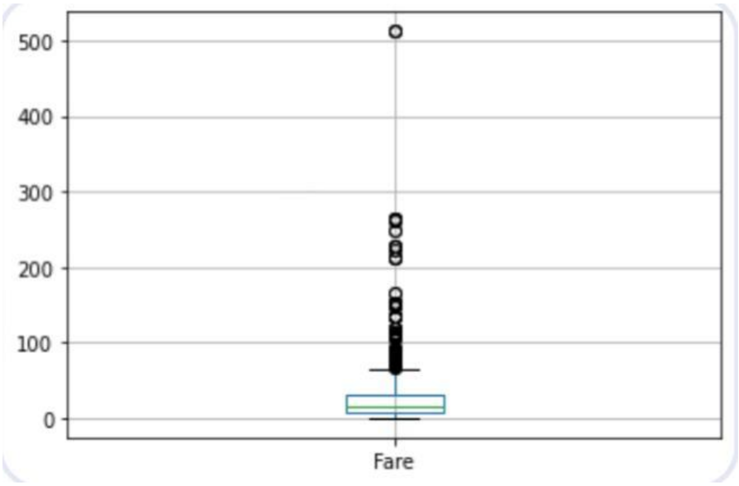


10. A quick box plot

You can create a box plot for any numerical column using a single line of code.

```
#Boxplot
df[['Fare']].boxplot()
```

<u>Output:</u>



Divide the data into input and output:

```
x = data.iloc[0:3,1:3].values
y = data.iloc[0:3,4:6].values
```

Train and test the variables:

If the dataset used, have the more number of datas in it, it perform the train and teat variables for better accuracy.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0)
```

After train_test_split,

- *75% of data in x and y goes to x_train and y_train*
- *25% of data in x and y goes to x_test and y_test*

```
print(x.shape)

print (x_train.shape)

print (x_test.shape)
```

Output:

```
(890,12)
(667.5,12)
(222.5,12)
```

Regressor :

```
From sklearn.linear_model import LinearRegression

titanic=LinearRegression ()
```

Fit the model

It is used for mapping or plotting the inputs with output.

```
titanic.fit(x_train,y_train)
```

Output:

LinearRegression ()

Prediction of the model:

```
def score_model_class(model):
    model.fit(X_train,y_train)
    y_pred =model.predict(x_test)
```

Accuracy:

```
print("accuracy_score test: ", accuracy_score(y_test,y_pred)*100)
```

Output:

```
accuracy_score test:  83.05084745762712
```

Clusterer:

The technique to segregate Datasets into various groups, on basis of having similar features and characteristics, is being called Clustering.

Kmeans Algorithm is an Iterative algorithm that divides a group of n datasets into k subgroups /clusters based on the similarity and their mean distance from the centroid of that particular subgroup/ formed.

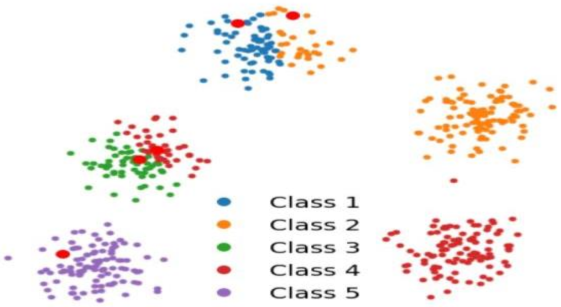
```
Identified_clusters =kmeans.fit_predict(x)
Identified_clusters
```

<u>Output:</u>

```
array([2, 3, 1, 0, 4])
```

```
data_with_clusters =data.copy()
data_with_clusters['Clusters'] = identified_clusters
Plt.scatter(data_with_clusters[‘Longitude’],data_with_clusters[‘Latitude’],c=data_
with_clusters[‘Clusters’],cmap=’rainbow’)
```

Output:



Conclusion:

EDA is the most important part of any analysis. You will get to know many things about your data. You will find answers to your most of the questions with EDA. I have tried to show most of the python functions used for exploring the data with visualizations.