Assignment 15 sep

Perform Data preprocessing on Titanic dataset

1.Data Collection.

2.Data Preprocessing

- Import the Libraries.
- Importing the dataset.
- Checking for Null Values.
- Data Visualization.
- Outlier Detection
- Splitting Dependent and Independent variables
- Perform Encoding
- Feature Scaling.
- Splitting Data into Train and Test

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Importing the Libraries

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

Importing the dataset

```
df = pd.read_csv("/content/drive/MyDrive/SmartInternz-Notebooks/Titanic-Dataset.csv")
```

```
df
```

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

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embark
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	

```
df.tail()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
				Johnston								

```
df.shape
```

```
(891, 12)
```

```
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
```

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
df.columns
```

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

```
df['Survived'].value_counts()
```

```
0    549
1    342
Name: Survived, dtype: int64
```

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

```
male      577
female    314
Name: Sex, dtype: int64
```

```
df['Embarked'].value_counts()
```

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

```
#Dropping the unwanted columns from the dataset
df.drop(['Name','SibSp','Parch','Ticket'],axis=1,inplace=True)
df.head()
```

	PassengerId	Survived	Pclass	Sex	Age	Fare	Cabin	Embarked
0	1	0	3	male	22.0	7.2500	NaN	S
1	2	1	1	female	38.0	71.2833	C85	C
2	3	1	3	female	26.0	7.9250	NaN	S
3	4	1	1	female	35.0	53.1000	C123	S
4	5	0	3	male	35.0	8.0500	NaN	S

```
df.drop('Cabin',axis=1,inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 7 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   Sex          891 non-null    object
 4   Age          714 non-null    float64
 5   Fare         891 non-null    float64
 6   Embarked     889 non-null    object
dtypes: float64(2), int64(3), object(2)
memory usage: 48.9+ KB
```

```
df.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Sex', 'Age', 'Fare', 'Embarked'], dtype='object')
```

Checking for null values

```
df.isnull().any()
```

```
PassengerId    False
Survived       False
Pclass         False
Sex            False
Age             True
Fare           False
Embarked        True
dtype: bool
```

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

```
PassengerId      0
Survived         0
Pclass           0
Sex              0
Age            177
Fare             0
Embarked         2
dtype: int64
```

```
df['Age'].fillna(df['Age'].mean(),inplace=True)
```

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

```
PassengerId    0
Survived       0
Pclass         0
Sex            0
Age            0
Fare           0
Embarked       2
dtype: int64
```

```
df['Embarked'].fillna(df['Embarked'].mode,inplace=True)
```

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

```
PassengerId    0
Survived       0
Pclass         0
Sex            0
Age            0
Fare           0
Embarked       0
dtype: int64
```

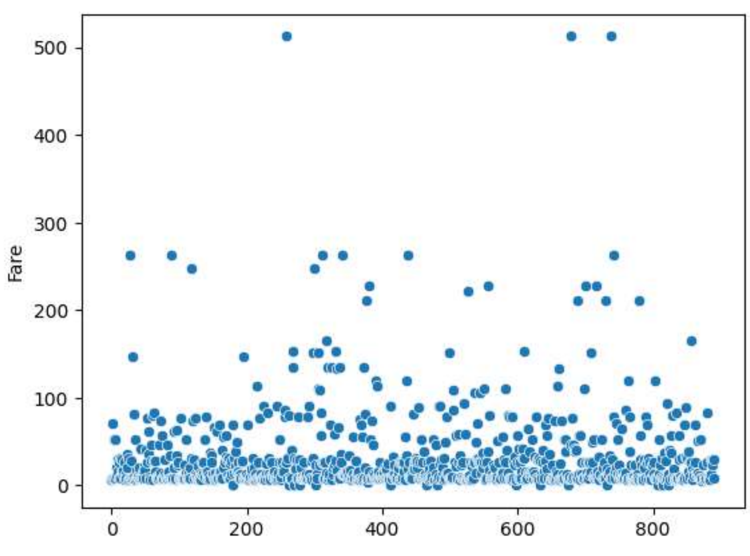
Data Visualization

```
df.head()
```

	PassengerId	Survived	Pclass	Sex	Age	Fare	Embarked
0	1	0	3	male	22.0	7.2500	S
1	2	1	1	female	38.0	71.2833	C
2	3	1	3	female	26.0	7.9250	S
3	4	1	1	female	35.0	53.1000	S
4	5	0	3	male	35.0	8.0500	S

```
sns.scatterplot(df['Fare'])
```

```
<Axes: ylabel='Fare'>
```




```
sns.distplot(df['Fare'])
```

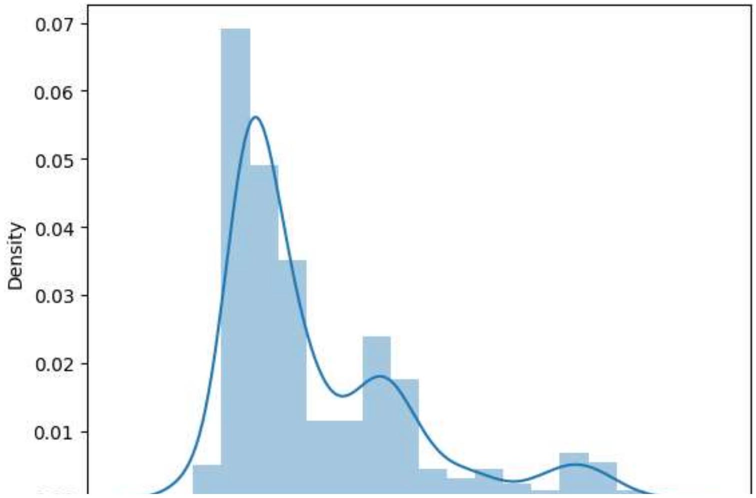
```
<ipython-input-84-70b4b4beb1b5>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Fare'])
<Axes: xlabel='Fare', ylabel='Density'>
```



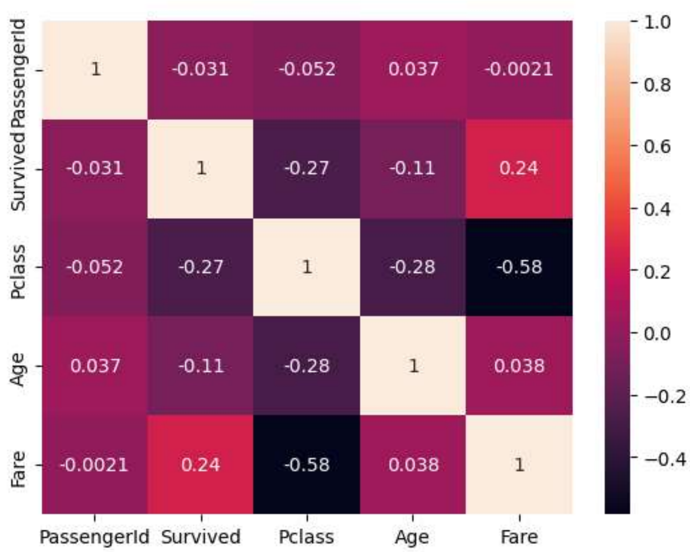

```
df.corr()
```

```
<ipython-input-78-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in D
  df.corr()
```

	PassengerId	Survived	Pclass	Age	Fare
PassengerId	1.000000	-0.031013	-0.051723	0.037345	-0.002137
Survived	-0.031013	1.000000	-0.269336	-0.107115	0.242890
Pclass	-0.051723	-0.269336	1.000000	-0.280856	-0.583530
Age	0.037345	-0.107115	-0.280856	1.000000	0.038018
Fare	-0.002137	0.242890	-0.583530	0.038018	1.000000

```
sns.heatmap(df.corr(),annot=True,)
```

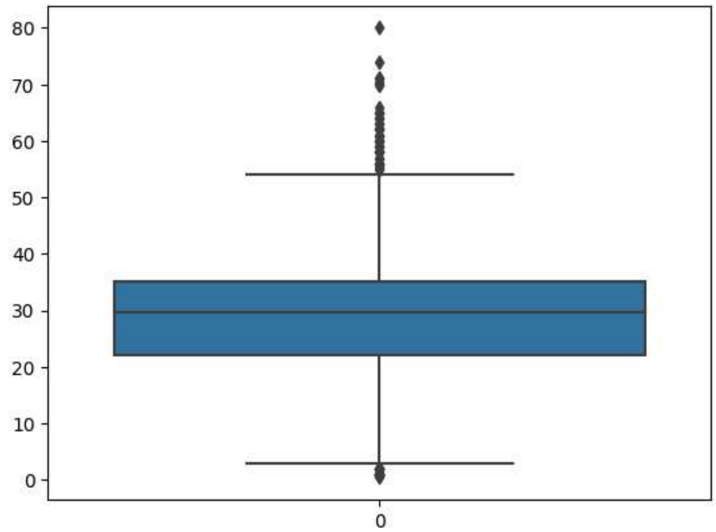
```
<ipython-input-79-8df7bcac526d>:1: FutureWarning: The default value of numeric_only in D
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```



Outlier detection

```
sns.boxplot(df['Age'])
```

```
<Axes: >
```



```
#Removing outliers using IQR method
q1=df.Age.quantile(0.25)
q3=df.Age.quantile(0.75)
```

```
q1,q3
```

```
(22.0, 35.0)
```

```
IQR=q3-q1
```

```
upper_limit=q3+1.5*IQR
```

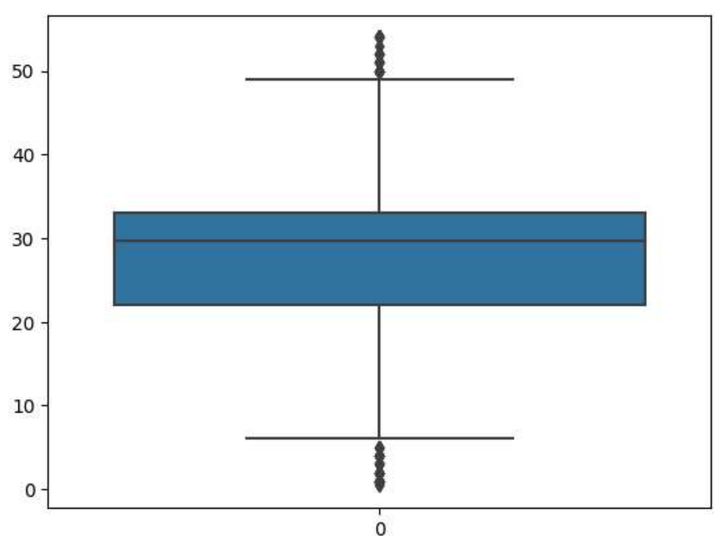
```
upper_limit
```

```
54.5
```

```
df = df[df['Age']<upper_limit]
```

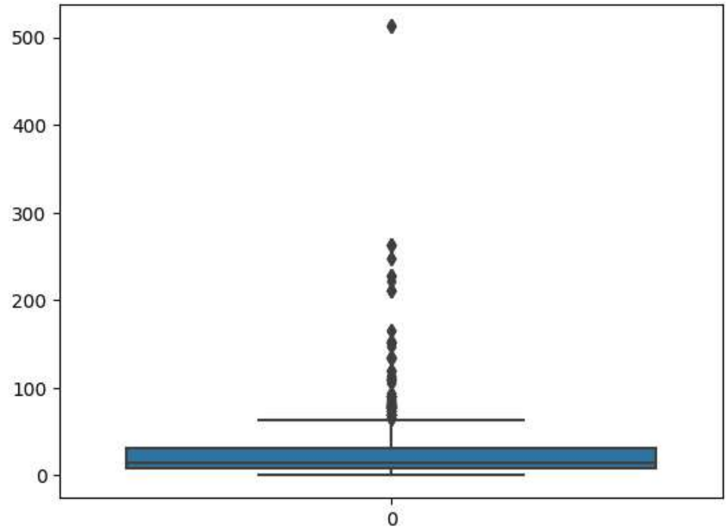
```
sns.boxplot(df['Age'])
```

```
<Axes: >
```



```
sns.boxplot(df['Fare'])
```

<Axes: >




```
Q1=df.Fare.quantile(0.25)
Q3=df.Fare.quantile(0.75)
```

```
Q1,Q3
```

(7.8958, 30.5)

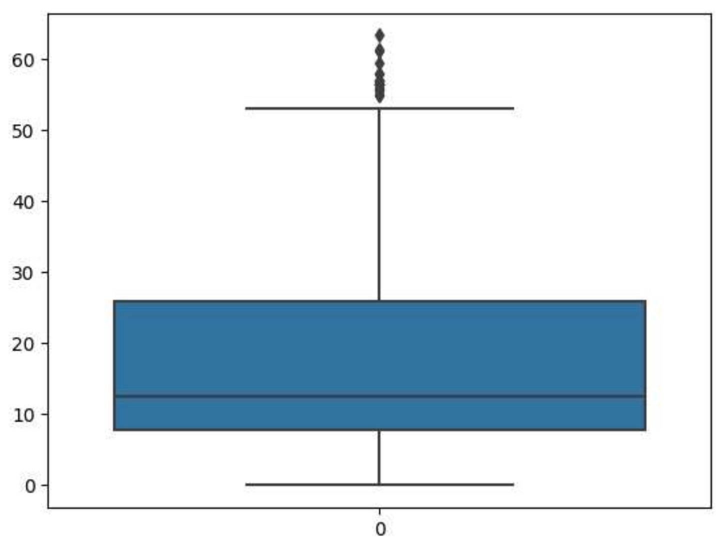
```
IQR=Q3-Q1
```

```
upper_limit_=Q3+1.5*IQR
```

```
df = df[df['Fare']<upper_limit_]
```

```
sns.boxplot(df['Fare'])
```

<Axes: >



Splitting dependant and independant variables

```
df.head()
```

	PassengerId	Survived	Pclass	Sex	Age	Fare	Embarked
0	1	0	3	male	22.000000	7.2500	S
2	3	1	3	female	26.000000	7.9250	S
3	4	1	1	female	35.000000	53.1000	S
4	5	0	3	male	35.000000	8.0500	S

```
df.shape
```

```
(741, 7)
```

```
x=df.iloc[:,2:7]
y=df.iloc[:,1:2]
```

```
x.shape
```

```
(741, 5)
```

```
y.shape
```

```
(741, 1)
```

Encoding

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
```

```
x['Sex']=le.fit_transform(x['Sex'])
```

```
x.head()
```

	Pclass	Sex	Age	Fare	Embarked
0	3	1	22.000000	7.2500	S
2	3	0	26.000000	7.9250	S
3	1	0	35.000000	53.1000	S
4	3	1	35.000000	8.0500	S
5	3	1	29.699118	8.4583	Q

```
x['Embarked']=le.fit_transform(x['Embarked'])
```

```
x.head()
```

	Pclass	Sex	Age	Fare	Embarked
0	3	1	22.000000	7.2500	2
2	3	0	26.000000	7.9250	2
3	1	0	35.000000	53.1000	2
4	3	1	35.000000	8.0500	2
5	3	1	29.699118	8.4583	1

```
print(le.classes_)
```

```
['C' 'Q' 'S']
```

```
print(dict(zip(le.classes_,range(len(le.classes_)))))
```

```
{'C': 0, 'Q': 1, 'S': 2}
```

Feature Scaling

Feature scaling using MinMaxScaler

```
from sklearn.preprocessing import MinMaxScaler
```

```
ms=MinMaxScaler()
```

```
x_scaled=pd.DataFrame(ms.fit_transform(x),columns=x.columns)
```

```
x_scaled.head()
```

	Pclass	Sex	Age	Fare	Embarked
0	1.0	1.0	0.402762	0.114429	1.0
1	1.0	0.0	0.477417	0.125082	1.0
2	0.0	0.0	0.645390	0.838091	1.0
3	1.0	1.0	0.645390	0.127055	1.0
4	1.0	1.0	0.546456	0.133499	0.5

Splitting the data into train and test

```
from sklearn.model_selection import train_test_split
```

```
x_train,y_train,x_test,y_test = train_test_split(x_scaled,y,test_size=0.2,random_state=0)
```

```
x_train.shape,y_train.shape,x_test.shape,y_test.shape
```

```
((592, 5), (149, 5), (592, 1), (149, 1))
```

```
x_train.head()
```

	Pclass	Sex	Age	Fare	Embarked
392	1.0	1.0	0.701381	0.136722	1.0
506	1.0	1.0	0.546456	0.111272	1.0
247	1.0	0.0	0.440090	0.139682	1.0
577	1.0	0.0	0.328108	0.228134	0.0
251	0.0	1.0	0.546456	0.481389	1.0

```
y_train.head()
```

	Pclass	Sex	Age	Fare	Embarked
196	0.5	1.0	0.328108	0.205182	1.0
187	0.5	1.0	0.776036	0.426148	1.0
14	0.5	1.0	0.546456	0.205182	1.0
31	1.0	0.0	0.328108	0.284099	1.0
395	1.0	0.0	0.402762	0.155268	1.0