Importing Libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sbs
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
```

Importing Datasets:

```
test_file=pd.read_csv("test.csv")
train_file=pd.read_csv("train.csv")
test_file.info()
print("-------------------------------------------------------")
train_file.info()
```

```
<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: 36.0+ KB
```

```
-------------------------------------------------------
<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
```

TOP 10 ROWS OF Test & Train:

```
test_file.head(10)
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embar
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	

```
train_file.head(10)
```

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

Checking the shape of Datasets:

```
print("Test.csv Shape=",test_file.shape)
print("Train.csv Shape=",train_file.shape)
```

```
Test.csv Shape= (418, 11)
Train.csv Shape= (891, 12)
```

Checking the Columns in our Datasets:

```
print("Columns Of Test.csv\n",test_file.columns)
```

```
Columns Of Test.csv
 Index(['PassengerId', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',
       'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```
print("Columns Of Train.csv\n",train_file.columns)
```

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

Checking the total Nan values of all Columns:

```
print("Test.csv Columns Having Nan Values:\n",test_file.isnull().sum())
```

```
Test.csv Columns Having Nan Values:
 PassengerId      0
Pclass           0
```

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

```
print("Train.csv Columns Having Nan Values:\n",train_file.isnull().sum())
```

```
Train.csv Columns Having Nan Values:
 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
```

Dropping Duplicates:

```
test_file=test_file.drop_duplicates()
test_file=test_file.dropna()
train_file=train_file.drop_duplicates()
train_file=train_file.dropna()
```

Handling Missing Values In Train.csv:

```
mean_of_train_age=train_file.Age.mean()
mode_of_train_embarked=train_file.Embarked.mode()#I am taking mode bcz it is a categoricial  column
mode_of_train_cabin=train_file.Cabin.mode()#I am taking mode bcz it is a categoricial  column
```

```
train_file.Age.fillna(value=mean_of_train_age, inplace=True)
train_file['Embarked'].fillna(train_file['Embarked'].mode()[0], inplace=True)
train_file['Cabin'].fillna(train_file['Cabin'].mode()[0], inplace=True)
#-----------------------------------------------------------------------
test_file.Age.fillna(value=mean_of_train_age, inplace=True)
test_file['Cabin'].fillna(test_file['Cabin'].mode()[0], inplace=True)
mean_of_train_fare=train_file.Fare.mean()
test_file.Fare.fillna(value=mean_of_train_fare, inplace=True)
```

```
print("Train.csv Columns Having Nan Values:\n",train_file.isnull().sum())
```

```
Train.csv Columns Having Nan Values:
 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
```

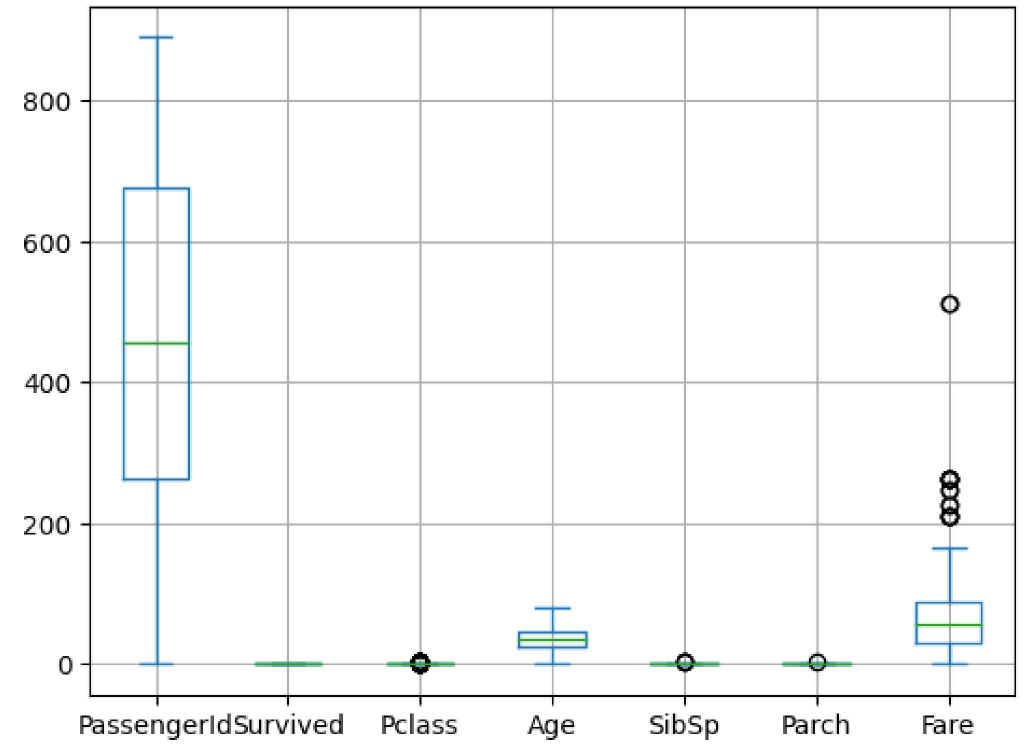
```
print("Test.csv Columns Having Nan Values:\n",test_file.isnull().sum())
```

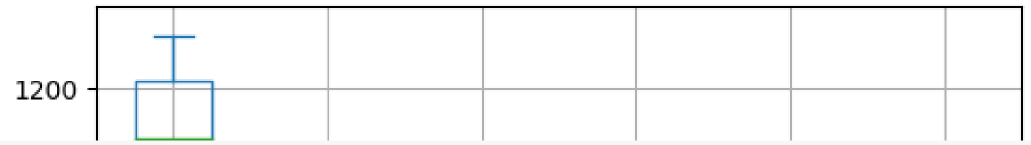
```
Test.csv Columns Having Nan Values:
 PassengerId    0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

Checking Outliers:

```
train_file.plot.box(grid=True)
test_file.plot.box(grid=True)
```

```
<Axes: >
```





```
def out_zscore(data):
    global outliers,zscore
    outliers = []
    zscore = []
    threshold = 3
    mean = np.mean(data)
    std = np.std(data)
    for i in data:
        z_score= (i - mean)/std
        zscore.append(z_score)
        if np.abs(z_score) > threshold:
            outliers.append(i)
    return print("Total number of outliers are",len(outliers))
```

Checking the total number of outliers of these columns:

```
print("Number of Outliers in Train.csv:\n")
print("OUTLIERS IN Pclass COLUMNS=")
out_zscore(train_file.Pclass)
print("OUTLIERS IN SibSp COLUMNS=")
out_zscore(train_file.SibSp)
print("OUTLIERS IN Parch COLUMNS=")
out_zscore(train_file.Parch)
print("OUTLIERS IN Fare COLUMNS=")
out_zscore(train_file.Fare)
print("-----------------------------------------------------------------------------------")
print("Number of Outliers in Test.csv:\n")
print("OUTLIERS IN Pclass COLUMNS=")
out_zscore(test_file.Pclass)
print("OUTLIERS IN SibSp COLUMNS=")
out_zscore(test_file.SibSp)
print("OUTLIERS IN Parch COLUMNS=")
out_zscore(test_file.Parch)
print("OUTLIERS IN Fare COLUMNS=")
out_zscore(test_file.Fare)
```

```
Number of Outliers in Train.csv:

OUTLIERS IN Pclass COLUMNS=
Total number of outliers are 10
OUTLIERS IN SibSp COLUMNS=
Total number of outliers are 3
OUTLIERS IN Parch COLUMNS=
Total number of outliers are 1
OUTLIERS IN Fare COLUMNS=
Total number of outliers are 2
-----------------------------------------------------------------------------------
Number of Outliers in Test.csv:
```

```
OUTLIERS IN Pclass COLUMNS=
Total number of outliers are 3
OUTLIERS IN SibSp COLUMNS=
Total number of outliers are 1
OUTLIERS IN Parch COLUMNS=
Total number of outliers are 1
OUTLIERS IN Fare COLUMNS=
Total number of outliers are 1
```

Removing Outliers:

```
def remove_outliers(data):
    lower_threshold = np.quantile(data, 0.25)
    upper_threshold = np.quantile(data, 0.75)
    filtered_data = [x for x in data if lower_threshold <= x <= upper_threshold]
    return filtered_data
```

```
print("Values After Removing Outliers From Fare:\n",remove_outliers(train_file.Fare))
print("Values After Removing Outliers From Parch:\n",remove_outliers(train_file.Parch))
print("Values After Removing Outliers From SibSp:\n",remove_outliers(train_file.SibSp))
print("Values After Removing Outliers From Pclass:\n",remove_outliers(train_file.Pclass))
#-------------------------------------------------------------------------------------------
print("Values After Removing Outliers From Fare:\n",remove_outliers(test_file.Fare))
print("Values After Removing Outliers From Parch:\n",remove_outliers(test_file.Parch))
print("Values After Removing Outliers From SibSp:\n",remove_outliers(test_file.SibSp))
print("Values After Removing Outliers From Pclass:\n",remove_outliers(test_file.Pclass))
```

```
Values After Removing Outliers From Fare:
 [71.2833, 53.1, 51.8625, 35.5, 76.7292, 61.9792, 83.475, 61.175, 34.6542, 63.3583, 77.2875, 52.0, 77.2875, 53.1, 79.2, 66.6, 33.5, 30.69
Values After Removing Outliers From Parch:
 [0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0,
Values After Removing Outliers From SibSp:
 [1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0,
Values After Removing Outliers From Pclass:
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
Values After Removing Outliers From Fare:
 [82.2667, 61.175, 61.9792, 57.75, 52.5542, 76.2917, 60.0, 52.0, 78.85, 55.4417, 75.2417, 57.75, 83.1583, 50.4958, 55.4417, 39.0, 83.1583
Values After Removing Outliers From Parch:
 [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
Values After Removing Outliers From SibSp:
 [1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0,
Values After Removing Outliers From Pclass:
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

standardize_features():

```
from sklearn.preprocessing import MinMaxScaler
def normalize_features(data):
# Create a MinMaxScaler object
  scaler = MinMaxScaler()

# Reshape the data to a 2D array (required by the scaler)
  data_2d = [[value] for value in data]

# Apply Min-Max scaling
  normalized_data = scaler.fit_transform(data_2d)

# Flatten the normalized data back to a 1D array
  normalized_data = normalized_data.flatten()

  print(normalized_data)
```

```
arr=['PassengerId', 'Survived', 'Pclass' , 'Age' , 'SibSp' , 'Parch' , 'Fare']
for i in arr:
  print("Feature ",str(i),"After Normilizing\n",normalize_features(train_file[i]))
  print("---------------------------------------------------------")
```

```
[0.         0.00225225 0.00563063 0.01013514 0.01126126 0.02252252
 0.02477477 0.02927928 0.05743243 0.05968468 0.06869369 0.0731982
 0.08333333 0.09797297 0.10247748 0.10698198 0.10810811 0.11373874
 0.12274775 0.13175676 0.13738739 0.13851351 0.15202703 0.15315315
 0.15540541 0.16554054 0.16891892 0.19031532 0.19481982 0.1981982
 0.20495495 0.21621622 0.21734234 0.21846847 0.22972973 0.23423423
 0.24099099 0.24436937 0.25112613 0.25788288 0.27477477 0.27815315
 0.28153153 0.28265766 0.28828829 0.29391892 0.29504505 0.30067568
 0.3018018  0.30630631 0.30855856 0.32657658 0.3277027  0.33333333
 0.33558559 0.34234234 0.34459459 0.34684685 0.34797297 0.3490991
 0.35698198 0.35810811 0.36486486 0.36711712 0.36936937 0.37162162
 0.37274775 0.37725225 0.37837838 0.38063063 0.38175676 0.38288288
 0.38738739 0.39977477 0.41103604 0.41441441 0.41554054 0.42342342
 0.43806306 0.44144144 0.44256757 0.46283784 0.48198198 0.48310811
 0.48761261 0.48873874 0.49211712 0.5        0.5045045  0.50788288
 0.50900901 0.51238739 0.51689189 0.51914414 0.53153153 0.54391892
 0.54617117 0.5472973  0.55292793 0.55743243 0.55968468 0.56644144
 0.56756757 0.57545045 0.57882883 0.57995495 0.58445946 0.58783784
 0.60247748 0.60585586 0.60698198 0.61148649 0.61824324 0.625
 0.62725225 0.64189189 0.64301802 0.64864865 0.65315315 0.65540541
 0.65765766 0.65990991 0.66441441 0.67342342 0.68468468 0.69481982
 0.6981982  0.7027027  0.70495495 0.70833333 0.71058559 0.72072072
 0.72522523 0.72747748 0.74099099 0.74436937 0.7545045  0.76351351
 0.76576577 0.77477477 0.7759009  0.78490991 0.78603604 0.78716216
 0.78828829 0.79504505 0.79842342 0.80067568 0.80405405 0.80518018
 0.80630631 0.81418919 0.82094595 0.82882883 0.83333333 0.83445946
 0.83783784 0.84121622 0.84459459 0.8536036  0.85810811 0.86036036
 0.86824324 0.87612613 0.87837838 0.8795045  0.88738739 0.89527027
 0.90202703 0.90653153 0.90990991 0.9222973  0.92567568 0.93918919
 0.95945946 0.96396396 0.96959459 0.97522523 0.97972973 0.98085586
 0.98873874 0.99774775 1.        ]
Feature  PassengerId After Normilizing
 None
---------------------------------------------------------
[1. 1. 0. 1. 1. 1. 1. 0. 1. 0. 0. 1. 0. 1. 0. 0. 1. 0. 0. 0. 1. 0. 1. 0.
 0. 0. 1. 0. 0. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 0. 1. 0. 0. 1.
 1. 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 1. 0. 1. 1.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 0. 1. 0. 1. 0. 1. 1.
 1. 0. 0. 1. 0. 1. 0. 1. 0. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 0. 0. 1. 1. 1. 1. 0. 0. 1.
 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 0. 1. 1. 1. 1. 0. 1. 1. 0. 0. 1.
 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1.]
Feature  Survived After Normilizing
 None
---------------------------------------------------------
[0.  0.  0.  1.  0.  0.5 0.  0.  0.  0.  0.  0.5 1.  0.  0.  0.  0.  0.
 0.  0.  0.5 0.  0.  0.  0.  0.5 0.  0.  0.  0.  0.5 0.5 0.  0.  1.  0.
 0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.5 0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.5 0.  0.  0.  0.  0.  0.  0.5 0.
 0.5 0.  0.  0.  0.  0.  0.  0.  1.  0.  1.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.5 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.5 0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.5
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.
 0.  0.  0.  0.  1.  0.  0.5 0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.
 0.5 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0. ]
Feature  Pclass After Normilizing
```

```
arr=['PassengerId', 'Pclass' , 'Age' , 'SibSp' , 'Parch' , 'Fare']
for i in arr:
  print("Feature ",str(i),"After Normilizing\n",normalize_features(test_file[i]))
  print("---------------------------------------------------------")
```

```
[0.         0.00497512 0.02985075 0.03482587 0.039801   0.05472637
 0.07960199 0.08457711 0.08955224 0.09452736 0.10199005 0.1119403
 0.11691542 0.12935323 0.13930348 0.14179104 0.15174129 0.15422886
 0.15671642 0.16169154 0.17164179 0.19900498 0.20895522 0.21890547
 0.24129353 0.24875622 0.25373134 0.26119403 0.26368159 0.27363184
 0.2960199  0.32338308 0.34328358 0.35820896 0.36318408 0.38308458
 0.41044776 0.41293532 0.41542289 0.42039801 0.42288557 0.4278607
 0.45771144 0.47263682 0.48756219 0.50497512 0.51243781 0.52238806
 0.55223881 0.55721393 0.56467662 0.5721393  0.57960199 0.59701493
 0.64179104 0.64676617 0.6840796  0.69900498 0.72885572 0.73134328
 0.73631841 0.75124378 0.75621891 0.76865672 0.77114428 0.78109453
 0.79353234 0.80348259 0.82338308 0.84079602 0.85323383 0.85572139
 0.87562189 0.89303483 0.89552239 0.90049751 0.91044776 0.94029851
 0.94278607 0.95273632 0.95771144 0.96517413 0.97512438 0.97761194
 0.98258706 0.99253731 1.        ]
Feature  PassengerId After Normilizing
 None
---------------------------------------------------------
[0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.5 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.5 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.5 0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.5 0.5 0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.5 0.  0.  0. ]
Feature  Pclass After Normilizing
 None
---------------------------------------------------------
[0.29333333 0.61333333 0.62666667 0.28       0.53333333 0.38666667
 0.58666667 0.58666667 0.78666667 0.30666667 0.36       0.32
 0.46666667 0.16       0.4        0.78666667 0.36666667 0.45333333
 0.42       0.72       0.88       0.34666667 1.         0.56
 0.23333333 0.46666667 0.82666667 0.         0.46666667 0.45333333
 0.69333333 0.8        0.29333333 0.37333333 0.54666667 0.62666667
 0.70666667 0.46666667 0.84       0.48       0.22666667 0.34666667
 0.06666667 0.61333333 0.42666667 0.54666667 0.65333333 0.28
 0.50666667 0.84       0.62666667 0.58666667 0.53333333 0.34666667
 0.6        0.33333333 0.30666667 0.69333333 0.84       0.38666667
 0.72       0.72       0.74666667 0.32       0.33333333 0.14666667
 0.50666667 0.38666667 0.76       0.58666667 0.65333333 0.77333333
 0.32       0.4        0.64       0.70666667 0.72       0.29333333
 0.66666667 0.22666667 0.62666667 0.38666667 0.56       0.25333333
 0.65333333 0.48       0.50666667]
Feature  Age After Normilizing
 None
---------------------------------------------------------
[0.33333333 0.33333333 0.33333333 0.         0.         0.33333333
 0.33333333 0.         0.         0.33333333 1.         0.
 0.         0.66666667 0.         0.33333333 0.         0.
 0.         0.66666667 0.33333333 0.33333333 0.33333333 0.33333333
 0.         0.         0.33333333 0.33333333 0.         0.33333333
 0.         0.33333333 0.         0.         0.         0.
 0.33333333 0.         0.         0.33333333 0.33333333 0.33333333
 0.         0.33333333 0.         0.         0.33333333 0.
 0.33333333 0.33333333 0.33333333 0.33333333 0.33333333 0.33333333
 0.         0.33333333 0.33333333 0.33333333 0.33333333 0.33333333
 0.33333333 0.         0.33333333 0.         0.         0.66666667
 0.         0.         0.         0.         0.         0.66666667
 0.33333333 0.         0.         0.33333333 0.         0.
```

```python
from sklearn.preprocessing import OneHotEncoder
# Create an instance of OneHotEncoder
"""
encoder = OneHotEncoder(categories=[['female' , 'male']])
train_file['Sex'] = encoder.fit_transform(train_file[['Sex']])"""
train_file.Sex
#for females it is 0 and for males it is 1
```

```
1      female
3      female
6        male
10     female
11     female
        ...
```

```
871    female
872      male
879    female
887    female
889      male
Name: Sex, Length: 183, dtype: object
```

```
df = pd.DataFrame(train_file)
encoder = OneHotEncoder()
encoded_data = encoder.fit_transform(df[['Sex']])
encoded_df = pd.DataFrame(encoded_data.toarray(), columns=encoder.get_feature_names_out(['Sex']))
df_encoded = pd.concat([df.drop('Sex', axis=1), encoded_df], axis=1)
df_encoded.head()
```

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
1	2.0	1.0	1.0	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1.0	0.0	PC 17599	71.2833	C85	
3	4.0	1.0	1.0	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	1.0	0.0	113803	53.1000	C123	

```
df = pd.DataFrame(test_file)
encoder = OneHotEncoder()
encoded_data = encoder.fit_transform(df[['Sex']])
encoded_df = pd.DataFrame(encoded_data.toarray(), columns=encoder.get_feature_names_out(['Sex']))
df_encoded = pd.concat([df.drop('Sex', axis=1), encoded_df], axis=1)
df_encoded.head()
```

	PassengerId	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
12	904.0	1.0	Snyder, Mrs. John Pillsbury (Nelle Stevenson)	23.0	1.0	0.0	21228	82.2667	B45	S
14	906.0	1.0	Chaffee, Mrs. Herbert Fuller (Carrie Constance...	47.0	1.0	0.0	W.E.P. 5734	61.1750	E31	S
			Ryerson							