

**NAME :**

- **ADITYA RAJ-20BEE0209**

**TEAM NUMBER – 180**

**APPLIED DATA SCIENCE**

# ASSIGNMENT -2

```
In [30]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import os

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

```
Out[30]:
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows x 15 columns

```
In [18]: #Handle the missing values(Mean mode median imputation)
df.isnull().sum()
```

```
Out[18]: survived      0
pclass      0
sex          0
age        177
sibsp       0
parch       0
fare        0
embarked     2
class       0
who         0
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel) C

```
fare      0
embarked  2
class     0
who       0
adult_male 0
deck     688
embark_town 2
alive     0
alone     0
dtype: int64
```

```
In [20]: def mean_median(df,variable):
         df[variable+'_mean'] = df[variable].fillna(df[variable].mean())
         df[variable+'_median'] = df[variable].fillna(df[variable].median())
```

```
In [23]: mean_median(df, 'age')
         df.head()
```

```
Out[23]:
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone	age_mean	age_median
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False	22.0	22.0
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False	38.0	38.0
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True	26.0	26.0
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False	35.0	35.0
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True	35.0	35.0

```
In [32]: #Finding the outliers(using INTER QUARTILE RANGE)
         Q1=df['fare'].quantile(0.25)
         Q3=df['fare'].quantile(0.75)
         IQR=Q3-Q1
```

```
In [33]: Q1 = df['fare'].quantile(0.25)
         Q3 = df['fare'].quantile(0.75)
         IQR = Q3 - Q1
         whisker_width = 1.5
         Fare_outliers = df[(df['fare'] < Q1 - whisker_width*IQR) | (df['fare'] > Q3 + whisker_width*IQR)]
         Fare_outliers.head()
```

```
Out[33]:
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
27	0	1	male	19.0	3	2	263.0000	S	First	man	True	C	Southampton	no	False

	id	age	sex	height_cm	weight_kg	body_fat_percent	team	position	height_m	is_goalkeeper	nationality	has_scholarship	is_injured		
31	1	1	female	NaN	1	0	146.5208	C	First	woman	False	B	Cherbourg	yes	False
34	0	1	male	28.0	1	0	82.1708	C	First	man	True	NaN	Cherbourg	no	False
52	1	1	female	49.0	1	0	76.7292	C	First	woman	False	D	Cherbourg	yes	False

```
In [45]: import warnings
warnings.filterwarnings("ignore")
plt.tight_layout()

print("Previous Shape With Outlier: ",df.shape)
sns.boxplot(df['age'],orient='v',ax=axes[0])
axes[0].title.set_text("Before")
#HANDLING OUTLIER #
Q1 = df.age.quantile(0.25)
Q3 = df.age.quantile(0.75)
print(Q1,Q3)
IQR = Q3-Q1
print(IQR)
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
print(lower_limit,upper_limit)
df2 = df
# REPLACING ALL THE Large values with MAX QUANTILE VALUE #

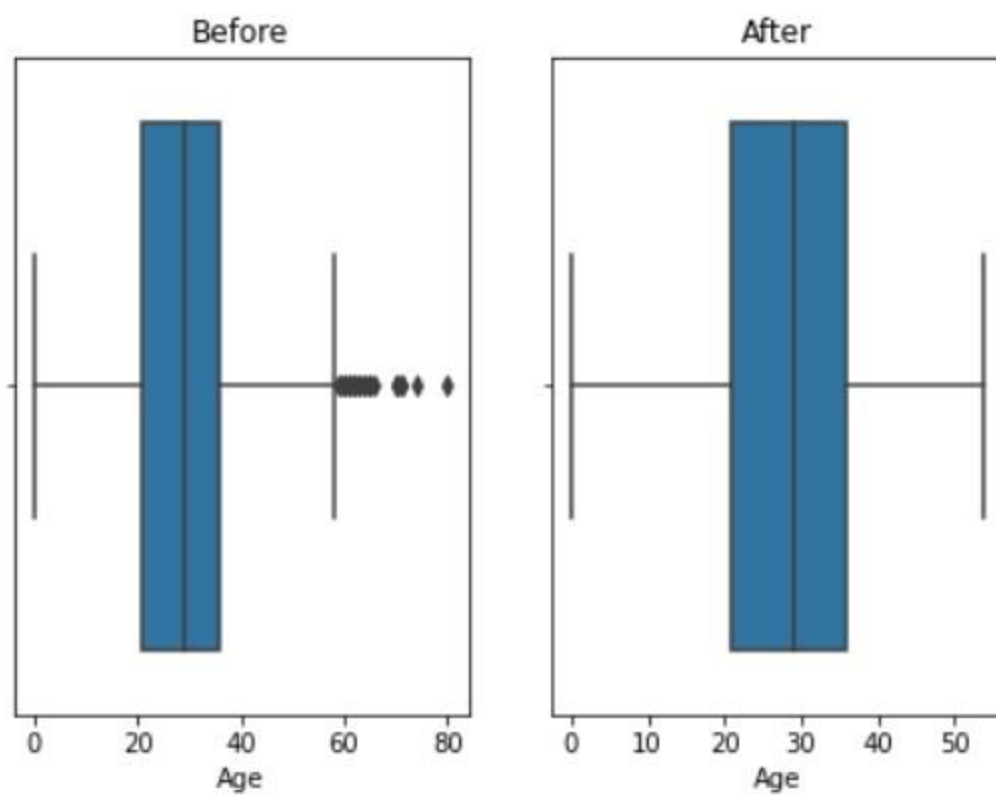
df2['age'] = np.where(df2['age']>upper_limit,upper_limit,df2['age'])
df2['age'] = np.where(df2['age']<lower_limit,lower_limit,df2['age'])
print("Shape After Removing Outliers:", df2.shape)
sns.boxplot(df2['age'],orient='v',ax=axes[1])
axes[1].title.set_text("After")
plt.show()
```

```
Previous Shape With Outlier: (891, 15)
20.125 38.0
17.875
-6.6875 64.8125
Shape After Removing Outliers: (891, 15)

<Figure size 640x480 with 0 Axes>
```

In [ ]:

Previous Shape With Outlier: (891, 11)  
Shape After Removing Outliers: (891, 11)



Jupyter Dh sb assm 2 Last Checkpoint: 2 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Notebook saved Trusted Python 3 (ipykernel)

```
In [58]: #split the data into training and testing
X_train, X_test, y_train, y_test = train_test_split(
    df['embarked'],
    df['survived'],

    test_size=0.3,
    random_state=0,
)
```

```
In [52]: #categorical columns encoding
X_train_enc = pd.get_dummies(X_train, drop_first=True)
X_test_enc = pd.get_dummies(X_test, drop_first=True)

X_train_enc.head()
```

```
Out[52]:
```

	Q	S
857	0	1
52	0	0
386	0	1
124	0	1
578	0	0

```
In [59]: # Splitting Dataset into the Independent variables:
X = df.iloc[:, :-1].values
print(X)

[[0 3 'male' ... nan 'Southampton' 'no']
 [1 1 'female' ... 'C' 'Cherbourg' 'yes']
 [1 3 'female' ... nan 'Southampton' 'yes']
 ...
 [0 3 'female' ... nan 'Southampton' 'no']
 [1 1 'male' ... 'C' 'Cherbourg' 'yes']
 [0 3 'male' ... nan 'Queenstown' 'no']]
```

```
In [60]: ## Splitting Dataset into the dependent variables:
Y = df.iloc[:, -1].values
print(Y)

[False False  True False  True  True  True False False False  True
  True False  True  True False  True False  True  True  True  True
  False False  True False  True  True  True False  True  True False False
  True  True False False False  True False  True  True False  True]
```

## # SCALING OF INDEPENDENT VARIABLE

```
In [51]: #Using the concept of feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:,3:] = sc.fit_transform(X_train[:,3:])
X_test[:,3:] = sc.transform(X_test[:,3:])
```

```
In [52]: print(X_train)

[[ 2.03000000e+02  3.40000000e+01  0.00000000e+00 ... -4.88677777e-01
 -1.93649167e-01  5.44862368e-01]
 [ 4.40000000e+02  3.10000000e+01  0.00000000e+00 ... -4.88677777e-01
 -1.93649167e-01  5.44862368e-01]
 [ 1.03000000e+02  2.10000000e+01  0.00000000e+00 ... -4.88677777e-01
 -1.93649167e-01  5.44862368e-01]
 ...
 [ 7.92000000e+02  1.60000000e+01  0.00000000e+00 ... -4.88677777e-01
 -1.93649167e-01  5.44862368e-01]
 [ 7.06000000e+02  3.90000000e+01  0.00000000e+00 ... -4.88677777e-01
 -1.93649167e-01  5.44862368e-01]
 [ 8.59000000e+02  2.40000000e+01  0.00000000e+00 ...  2.04633819e+00
 -1.93649167e-01 -1.83532587e+00]]
```

```
In [67]: #DESCRIPTIVE STATS
df.describe
```

```
Out[67]: <bound method NDFrame.describe of
0      0      3      male  22.0      1      0      7.2500      S      Third
1      1      1      female  38.0      1      0      71.2833      C      First
2      1      3      female  26.0      0      0      7.9250      S      Third
3      1      1      female  35.0      1      0      53.1000      S      First
4      0      3      male  35.0      0      0      8.0500      S      Third
..      ...      ...      ...      ...      ...      ...      ...      ...
886     0      2      male  27.0      0      0      13.0000      S      Second
887     1      1      female  19.0      0      0      30.0000      S      First
888     0      3      female  NaN      1      2      23.4500      S      Third
889     1      1      male  26.0      0      0      30.0000      C      First
890     0      3      male  32.0      0      0      7.7500      Q      Third

who      adult_male  deck  embark_town  alive  alone
0      man      True  NaN  Southampton  no  False
1      woman      False  C  Cherbourg  yes  False
2      woman      False  NaN  Southampton  yes  True
3      woman      False  C  Southampton  yes  False
4      man      True  NaN  Southampton  no  True
..      ...      ...      ...      ...      ...
886     man      True  NaN  Southampton  no  True
887     woman      False  B  Southampton  yes  True
888     woman      False  NaN  Southampton  no  False
889     man      True  C  Cherbourg  yes  True
890     man      True  NaN  Queenstown  no  True

[891 rows x 15 columns]>
```

```
In [69]: df2 = df["age"].mean()
df2
```

```
Out[69]: 29.69911764705882
```

```
In [70]: df3=df["age"].median
df3
```

```
Out[70]: <bound method NDFrame._add_numeric_operations.<locals>.median of 0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
...
886     27.0
887     19.0
888     NaN
889     26.0
890     32.0
Name: age, Length: 891, dtype: float64>
```

```
In [72]: df4=df["age"].mode
df4
```

```
Out[72]: <bound method Series.mode of 0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
...
886     27.0
887     19.0
888     NaN
889     26.0
890     32.0
Name: age, Length: 891, dtype: float64>
```

7-5-11

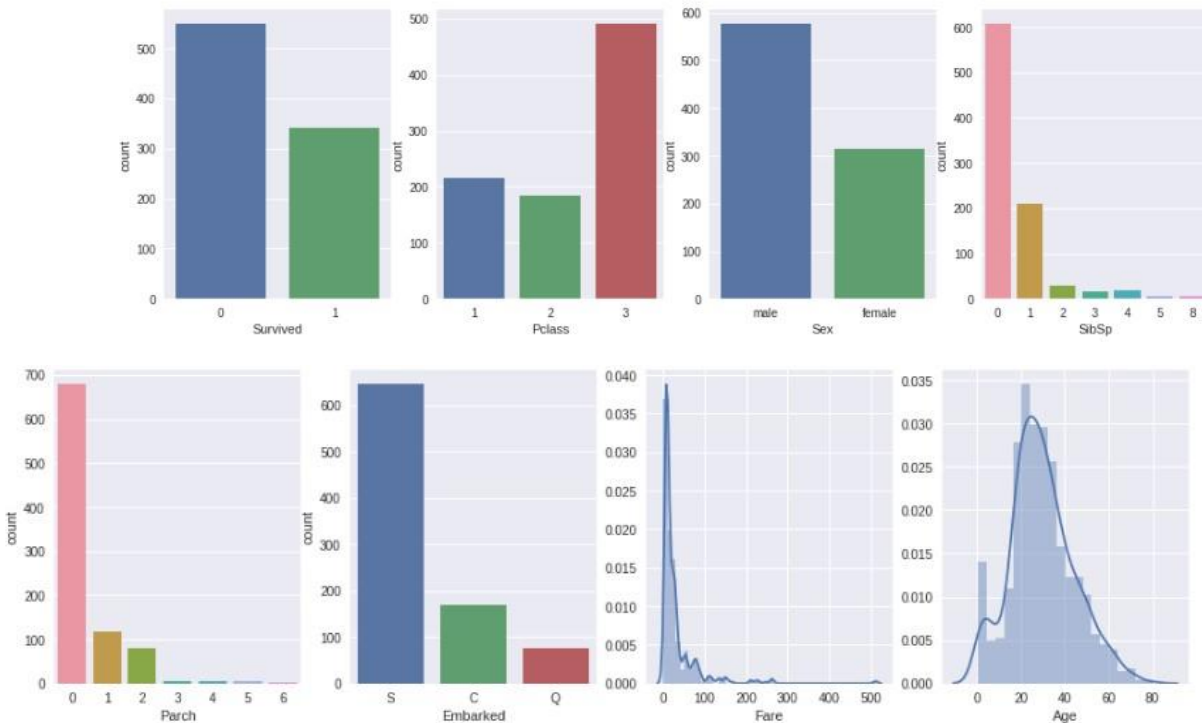
## #UNIVARIATE ANALYSIS

In [17]:

```
fig, axes = plt.subplots(2, 4, figsize=(16, 10))
sns.countplot('Survived', data=train, ax=axes[0,0])
sns.countplot('Pclass', data=train, ax=axes[0,1])
sns.countplot('Sex', data=train, ax=axes[0,2])
sns.countplot('SibSp', data=train, ax=axes[0,3])
sns.countplot('Parch', data=train, ax=axes[1,0])
sns.countplot('Embarked', data=train, ax=axes[1,1])
sns.distplot(train['Fare'], kde=True, ax=axes[1,2])
sns.distplot(train['Age'].dropna(), kde=True, ax=axes[1,3])
```

Out[17]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7ffa6366bdd8>

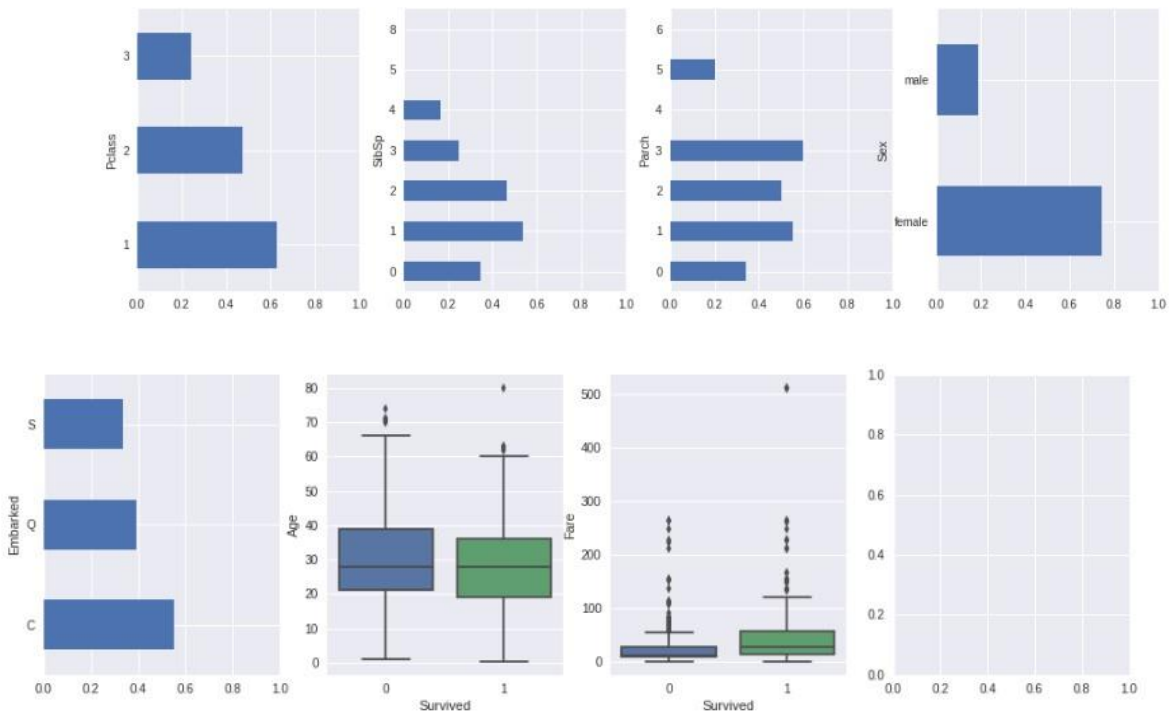


## # BIVRIATE ANALYSIS



```
In [18]: figbi, axesbi = plt.subplots(2, 4, figsize=(16, 10))
train.groupby('Pclass')['Survived'].mean().plot(kind='barh',ax=axesbi[0,0],xlim=[0,1])
train.groupby('SibSp')['Survived'].mean().plot(kind='barh',ax=axesbi[0,1],xlim=[0,1])
train.groupby('Parch')['Survived'].mean().plot(kind='barh',ax=axesbi[0,2],xlim=[0,1])
train.groupby('Sex')['Survived'].mean().plot(kind='barh',ax=axesbi[0,3],xlim=[0,1])
train.groupby('Embarked')['Survived'].mean().plot(kind='barh',ax=axesbi[1,0],xlim=[0,1])
sns.boxplot(x="Survived", y="Age", data=train,ax=axesbi[1,1])
sns.boxplot(x="Survived", y="Fare", data=train,ax=axesbi[1,2])
```

```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffa631ff9e8>
```



## # MULTIVARIATE ANALYSIS

```
In [20]: import seaborn as sns

f, ax = plt.subplots(figsize=(10, 8))
corr = train.corr()
sns.heatmap(corr,
            mask=np.zeros_like(corr, dtype=np.bool),
            cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, ax=ax)
```

```
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffa590f7160>
```

