

---

# ASSIGNMENT-3

---

Logistic Regression and Naïve Bayes



MARCH 16, 2021  
RISHABH SHARMA  
20MAI0082

Github Link :-

<https://github.com/rishabh5197/Machine-Learning-Assignments/tree/main/Assignment-3>

In [1]:

```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
# reading dataset
read = pd.read_csv("titanic_data.csv")
```

In [3]:

```
read.head(10)
```

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	
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	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
9	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	

In [4]:

```
# selecting the columns what are needed.
df=read[['Survived','Pclass','Sex','Age','Fare']]
```

In [5]:

```
# replacing male with 1 and females with 0.
df["Sex"] = df["Sex"].apply(lambda sex:1 if sex=="male" else 0)
```

<ipython-input-5-038c01d3b808>:2: SettingWithCopyWarning:  
 A value is trying to be set on a copy of a slice from a DataFrame.  
 Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)  
`df["Sex"] = df["Sex"].apply(lambda sex:1 if sex=="male" else 0)`

In [6]:

```
# Handling missing values - Data Imputation
print(df.isnull().sum())
```

```
Survived    0
Pclass      0
Sex         0
Age        177
Fare        0
dtype: int64
```

In [7]:

```
df["Age"] = df["Age"].fillna(df["Age"].median())
```

<ipython-input-7-0f86aa1fb408>:1: SettingWithCopyWarning:  
 A value is trying to be set on a copy of a slice from a DataFrame.  
 Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)  
`df["Age"] = df["Age"].fillna(df["Age"].median())`

In [8]:

```
# taking a look at a dataframe
df.head()
```

Out[8]:

	Survived	Pclass	Sex	Age	Fare
0	0	3	1	22.0	7.2500
1	1	1	0	38.0	71.2833

	Survived	Pclass	Sex	Age	Fare
2	1	3	0	26.0	7.9250
3	1	1	0	35.0	53.1000
4	0	3	1	35.0	8.0500

In [9]:

```
# Set the predictor (x) and response (y) variables
x = df.drop("Survived",axis=1)
y = df["Survived"]
```

In [10]:

```
# splitting the dataset
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.3,random_state = 0)
```

In [11]:

```
# import logistic regression
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(xtrain,ytrain)
ypredict = lr.predict(xtest)
```

In [12]:

```
ypredict
```

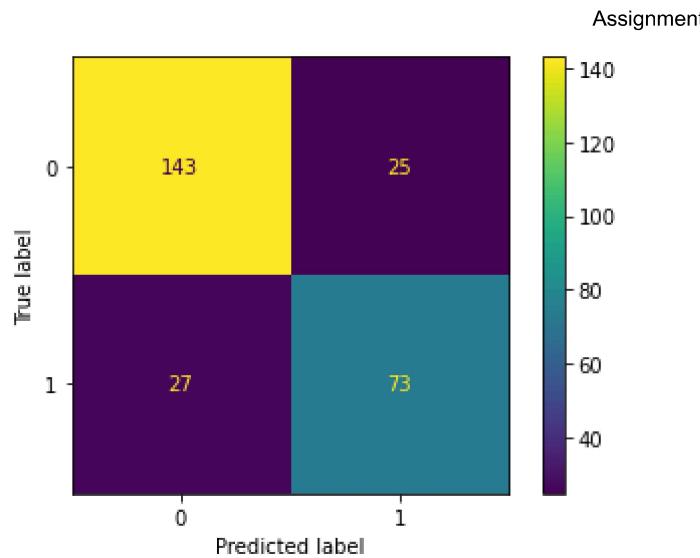
Out[12]:

```
array([0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0], dtype=int64)
```

In [13]:

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix,plot_confusion_matrix,accuracy_score,classification_report
print(confusion_matrix(ytest,ypredict))
plot_confusion_matrix(lr,xtest,ytest)
plt.show()
```

```
[[143  25]
 [ 27  73]]
```



```
In [14]: print("Accuracy Score :- ",accuracy_score(ytest,ypredict))
```

Accuracy Score :- 0.8059701492537313

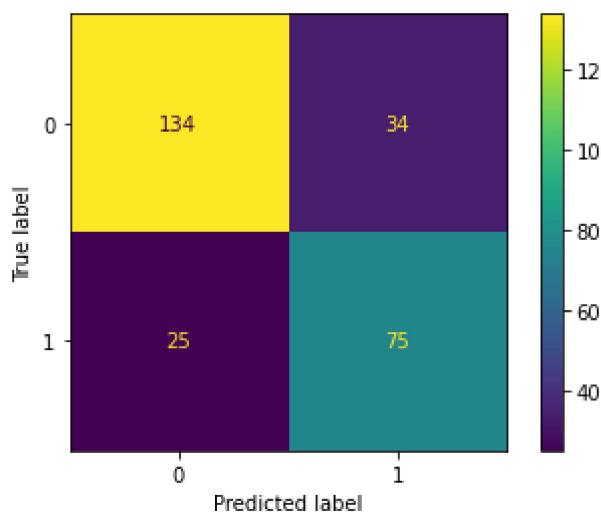
```
In [15]: print(classification_report(ytest,ypredict))
```

	precision	recall	f1-score	support
0	0.84	0.85	0.85	168
1	0.74	0.73	0.74	100
accuracy			0.81	268
macro avg	0.79	0.79	0.79	268
weighted avg	0.81	0.81	0.81	268

```
In [16]: from sklearn.naive_bayes import GaussianNB
gb = GaussianNB()
gb.fit(xtrain,ytrain)
ypredict = gb.predict(xtest)
```

```
In [17]: print(confusion_matrix(ytest,ypredict))
plot_confusion_matrix(gb,xtest,ytest)
plt.show()
```

[[134 34]  
[ 25 75]]



```
In [18]: print("Accuracy Score :- ",accuracy_score(ytest,ypredict))
```

Accuracy Score :- 0.7798507462686567

```
In [19]: print(classification_report(ytest,ypredict))
```

	precision	recall	f1-score	support
0	0.84	0.80	0.82	168
1	0.69	0.75	0.72	100
accuracy		0.78	0.78	268
macro avg	0.77	0.77	0.77	268
weighted avg	0.79	0.78	0.78	268

**Conclusion :- Logistic Regression is much better than Naive Bayes**