In [31]:

```

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

Type *Markdown* and LaTeX: α^2

In [32]:

```
data1=pd.read_csv("train.csv")
data2=pd.read_csv("test.csv")
data = pd.concat([data1,data2],axis=0)
```

In [33]:

```
data.head(5)
```

Out[33]:

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/ time conv
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	
1	1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	
2	2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	
3	3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	
4	4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	

5 rows × 25 columns

In [34]:
```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 129880 entries, 0 to 25975
Data columns (total 25 columns):
 #   Column                             Non-Null Count   Dtype  
---  ------                             --------------   -----  
 0   Unnamed: 0                         129880 non-null  int64  
 1   id                                 129880 non-null  int64  
 2   Gender                             129880 non-null  object 
 3   Customer Type                      129880 non-null  object 
 4   Age                                129880 non-null  int64  
 5   Type of Travel                     129880 non-null  object 
 6   Class                              129880 non-null  object 
 7   Flight Distance                    129880 non-null  int64  
 8   Inflight wifi service              129880 non-null  int64  
 9   Departure/Arrival time convenient  129880 non-null  int64  
 10  Ease of Online booking             129880 non-null  int64  
 11  Gate location                      129880 non-null  int64  
 12  Food and drink                     129880 non-null  int64  
 13  Online boarding                    129880 non-null  int64  
 14  Seat comfort                       129880 non-null  int64  
 15  Inflight entertainment             129880 non-null  int64  
 16  On-board service                   129880 non-null  int64  
 17  Leg room service                   129880 non-null  int64  
 18  Baggage handling                   129880 non-null  int64  
 19  Checkin service                    129880 non-null  int64  
 20  Inflight service                   129880 non-null  int64  
 21  Cleanliness                        129880 non-null  int64  
 22  Departure Delay in Minutes         129880 non-null  int64  
 23  Arrival Delay in Minutes           129487 non-null  float64
 24  satisfaction                       129880 non-null  object 
dtypes: float64(1), int64(19), object(5)
memory usage: 25.8+ MB
None
```

In [35]:
```
del data["Unnamed: 0"]
del data["id"]
```

In [36]:

```
print(data.isnull().sum())
```

```
Gender                                 0
Customer Type                          0
Age                                    0
Type of Travel                         0
Class                                  0
Flight Distance                        0
Inflight wifi service                  0
Departure/Arrival time convenient      0
Ease of Online booking                 0
Gate location                          0
Food and drink                         0
Online boarding                        0
Seat comfort                           0
Inflight entertainment                 0
On-board service                       0
Leg room service                       0
Baggage handling                       0
Checkin service                        0
Inflight service                       0
Cleanliness                            0
Departure Delay in Minutes             0
Arrival Delay in Minutes             393
satisfaction                           0
dtype: int64
```

In [37]:

```
from sklearn.impute import SimpleImputer
imputer_int=SimpleImputer(missing_values=np.nan)
data['Arrival Delay in Minutes'] = imputer_int.fit_transform(data[['Arrival De
```

In [38]:

```
data.isnull().sum()
```

Out[38]:

```
Gender                               0
Customer Type                        0
Age                                  0
Type of Travel                       0
Class                                0
Flight Distance                      0
Inflight wifi service                0
Departure/Arrival time convenient    0
Ease of Online booking               0
Gate location                        0
Food and drink                       0
Online boarding                      0
Seat comfort                         0
Inflight entertainment               0
On-board service                     0
Leg room service                     0
Baggage handling                     0
Checkin service                      0
Inflight service                     0
Cleanliness                          0
Departure Delay in Minutes           0
Arrival Delay in Minutes             0
satisfaction                         0
dtype: int64
```

In [39]:

```
from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
data["Gender"]=LE.fit_transform(data["Gender"])
data["Customer Type"]=LE.fit_transform(data["Customer Type"])
data["Type of Travel"]=LE.fit_transform(data["Type of Travel"])
data["Class"]=LE.fit_transform(data["Class"])
data["satisfaction"]=LE.fit_transform(data["satisfaction"])
```

In [40]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 129880 entries, 0 to 25975
Data columns (total 23 columns):
 #   Column                             Non-Null Count   Dtype  
---  ------                             --------------   -----  
 0   Gender                             129880 non-null  int32  
 1   Customer Type                      129880 non-null  int32  
 2   Age                                129880 non-null  int64  
 3   Type of Travel                     129880 non-null  int32  
 4   Class                              129880 non-null  int32  
 5   Flight Distance                    129880 non-null  int64  
 6   Inflight wifi service              129880 non-null  int64  
 7   Departure/Arrival time convenient  129880 non-null  int64  
 8   Ease of Online booking             129880 non-null  int64  
 9   Gate location                      129880 non-null  int64  
 10  Food and drink                     129880 non-null  int64  
 11  Online boarding                    129880 non-null  int64  
 12  Seat comfort                       129880 non-null  int64  
 13  Inflight entertainment             129880 non-null  int64  
 14  On-board service                   129880 non-null  int64  
 15  Leg room service                   129880 non-null  int64  
 16  Baggage handling                   129880 non-null  int64  
 17  Checkin service                    129880 non-null  int64  
 18  Inflight service                   129880 non-null  int64  
 19  Cleanliness                        129880 non-null  int64  
 20  Departure Delay in Minutes         129880 non-null  int64  
 21  Arrival Delay in Minutes           129880 non-null  float64
 22  satisfaction                       129880 non-null  int32  
dtypes: float64(1), int32(5), int64(17)
memory usage: 21.3 MB
```

In [41]:
```
IndepVar = []
for col in data.columns:
    if col != 'satisfaction':
        IndepVar.append(col)

TargetVar = 'satisfaction'

x = data[IndepVar]
y = data[TargetVar]
```

In [42]:
```
x.columns
```

Out[42]:
```
Index(['Gender', 'Customer Type', 'Age', 'Type of Travel', 'Class',
       'Flight Distance', 'Inflight wifi service',
       'Departure/Arrival time convenient', 'Ease of Online booking',
       'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort',
       'Inflight entertainment', 'On-board service', 'Leg room service',
       'Baggage handling', 'Checkin service', 'Inflight service',
       'Cleanliness', 'Departure Delay in Minutes',
       'Arrival Delay in Minutes'],
      dtype='object')
```

In [44]:

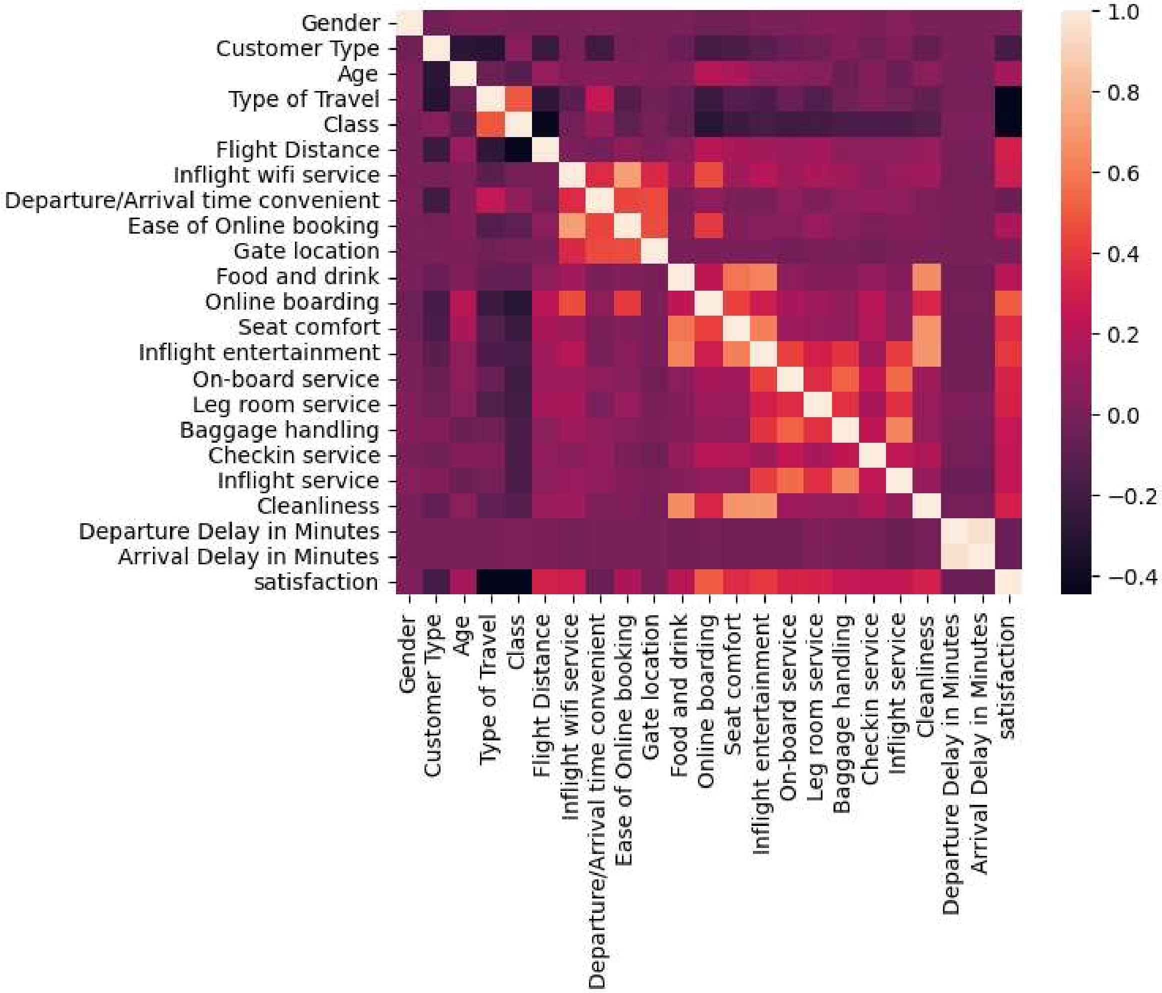
```
corr = data.corr()
```

In [45]:

```
sns.heatmap(corr)
```

Out[45]:

```
<AxesSubplot:>
```



In [46]:

```
from sklearn.model_selection import train_test_split
x_train,x_test, y_train, y_test=train_test_split(x,y,test_size=0.33,random_sta
```

In [47]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform (x_test)
```

In [48]:

```
print(x_train.shape,x_test.shape)
```

```
(87019, 22) (42861, 22)
```

Selecting Meaningful Features

In [19]:

```
from sklearn.feature_selection import SelectKBest,chi2
best = SelectKBest(score_func=chi2,k=10)
fit = best.fit(x,y)
scores = pd.DataFrame(fit.scores_)
colNames = pd.DataFrame(x.columns)
features = pd.concat([colNames,scores],axis=1)
features.columns = ['attribute','score']
imp_features=features.nlargest(10,'score')
print(imp_features)
```

```
                     attribute         score
5              Flight Distance  9.645859e+06
21    Arrival Delay in Minutes  4.298388e+04
20  Departure Delay in Minutes  3.293872e+04
11             Online boarding  1.834034e+04
3               Type of Travel  1.815163e+04
4                        Class  1.696271e+04
2                          Age  1.353949e+04
13      Inflight entertainment  1.091611e+04
12                Seat comfort  7.993086e+03
6         Inflight wifi service  6.758345e+03
```

In [20]:

```
c = ['Flight Distance','Arrival Delay in Minutes','Departure Delay in Minutes'
'Inflight entertainment','Seat comfort','Inflight wifi service']
x1=data.loc[:,c]
print(x1.head(1))
```

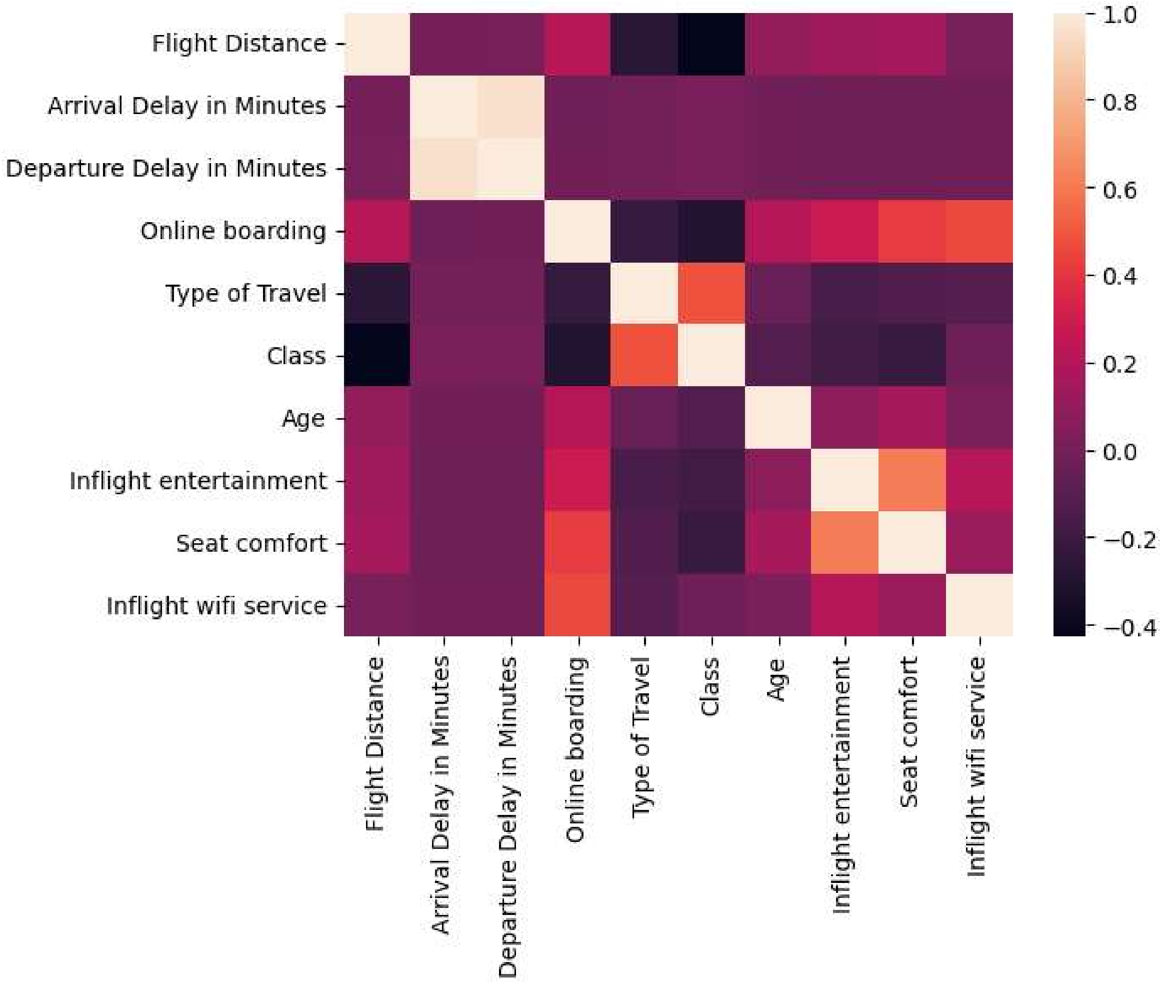
```
   Flight Distance  Arrival Delay in Minutes  Departure Delay in Minutes  \
0              460                      18.0                           25   

   Online boarding  Type of Travel  Class  Age  Inflight entertainment  \
0                3               1      2   13                       5   

   Seat comfort  Inflight wifi service  
0             5                      3  
```

In [21]:
```
x1corr = x1.corr()
sns.heatmap(x1corr)
```

Out[21]:
```
<AxesSubplot:>
```



In [22]:
```
from sklearn.model_selection import train_test_split
x_train,x_test, y_train, y_test=train_test_split(x1,y,test_size=0.33,random_st
```

In [23]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform (x_test)
```

In [24]:
```
print(x_train.shape,x_test.shape)
```

```
(87019, 10) (42861, 10)
```

In [ ]:

PCA

In [25]:

```
from sklearn.decomposition import PCA
pca=PCA(n_components = 2)
x_r=pca.fit(x).transform (x)
```

In [26]:

```
print(x_r)
```

```
[[-730.35390612    9.38687236]
 [-955.33902476  -16.04965605]
 [ -48.33502762  -21.0399969 ]
 ...
 [-362.3515567   -20.99952443]
 [ -63.35407309  -21.00030257]
 [-926.31430049  -21.0677217 ]]
```

In [ ]:

```
```

Common Techniques to identify overfitting and underfitting:

1.learning curves 2.cross-validation 3.Regularization Techniques

In [27]:

```
#using Regularization Technique
from sklearn.linear_model import LogisticRegression
# Initialize logistic regression with L2 regularization
clf = LogisticRegression(penalty='l2', C=1.0)
# Train the model and evaluate
clf.fit(x_train, y_train)
train_accuracy = clf.score(x_train, y_train)
test_accuracy = clf.score(x_test, y_test)
print("Training accuracy:", train_accuracy)
print("Testing accuracy:", test_accuracy)
```

```
Training accuracy: 0.8411611257311622
Testing accuracy: 0.8430274608618558
```

In [28]:

```
from sklearn.model_selection import cross_val_score
# Perform cross-validation
scores = cross_val_score(clf, x, y, cv=5)  # clf is your trained model, X is y
print("Cross-validation scores:", scores)
print("Mean accuracy:", np.mean(scores))
```

```
C:\Users\srira\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\srira\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\srira\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\srira\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
Cross-validation scores: [0.82884201 0.81817832 0.82025716 0.81729289 0.80031
568]
Mean accuracy: 0.8169772097320603

C:\Users\srira\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Applying different algorithms

LogisticRegression

In [49]:

```
from sklearn.linear_model import LogisticRegression
ModelLR = LogisticRegression()


ModelLR.fit(x_train, y_train)

y_pred = ModelLR.predict(x_test)

y_pred_prob = ModelLR.predict_proba(x_test)

print('Model Name: ',ModelLR )


from sklearn.metrics import confusion_matrix
actual = y_test
predicted = y_pred
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, r
print('Confusion matrix : \n', matrix)
```

```
Model Name:  LogisticRegression()
Confusion matrix : 
 [[15534  3094]
 [ 2296 21937]]
```

In [52]:

```
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : tp:{} fn:{} fp:{} tn:{}\n'.format(tp, fn, fp, tn) )


sensitivity = round(tp/(tp+fn), 3)
specificity = round(tn/(tn+fp), 3)
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3)
precision = round(tp/(tp+fp), 3)
f1Score = round((2*tp/(2*tp + fp + fn)), 3)

from math import sqrt

print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
```

```
Outcome values : tp:15534 fn:3094 fp:2296 tn:21937

Accuracy : 87.4 %
Precision : 87.1 %
Recall : 83.4 %
F1 Score : 0.852
Specificity or True Negative Rate : 90.5 %
```

Random Forest

In [53]:

```
from sklearn.ensemble import RandomForestClassifier
ModelRF = RandomForestClassifier()
ModelLR.fit(x_train, y_train)

y_pred = ModelLR.predict(x_test)

y_pred_prob = ModelLR.predict_proba(x_test)

print('Model Name: ',ModelLR )


from sklearn.metrics import confusion_matrix
actual = y_test
predicted = y_pred
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, r
print('Confusion matrix : \n', matrix)
```

```
Model Name:  LogisticRegression()
Confusion matrix :
 [[15534  3094]
 [ 2296 21937]]
```

In [54]:

```
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : tp:{} fn:{} fp:{} tn:{}\n'.format(tp, fn, fp, tn) )


sensitivity = round(tp/(tp+fn), 3)
specificity = round(tn/(tn+fp), 3)
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3)
precision = round(tp/(tp+fp), 3)
f1Score = round((2*tp/(2*tp + fp + fn)), 3)

from math import sqrt

print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
```

```
Outcome values : tp:15534 fn:3094 fp:2296 tn:21937

Accuracy : 87.4 %
Precision : 87.1 %
Recall : 83.4 %
F1 Score : 0.852
Specificity or True Negative Rate : 90.5 %
```

KNeighbourClassifier

In [55]:

```
from sklearn.neighbors import KNeighborsClassifier
ModelKNN = KNeighborsClassifier(n_neighbors=5)
ModelLR.fit(x_train, y_train)

y_pred = ModelLR.predict(x_test)

y_pred_prob = ModelLR.predict_proba(x_test)

print('Model Name: ',ModelLR )


from sklearn.metrics import confusion_matrix
actual = y_test
predicted = y_pred
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, r
print('Confusion matrix : \n', matrix)
```

```
Model Name:  LogisticRegression()
Confusion matrix :
 [[15534  3094]
 [ 2296 21937]]
```

In [56]:

```
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : tp:{} fn:{} fp:{} tn:{}\n'.format(tp, fn, fp, tn) )


sensitivity = round(tp/(tp+fn), 3)
specificity = round(tn/(tn+fp), 3)
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3)
precision = round(tp/(tp+fp), 3)
f1Score = round((2*tp/(2*tp + fp + fn)), 3)

from math import sqrt

print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
```

```
Outcome values : tp:15534 fn:3094 fp:2296 tn:21937

Accuracy : 87.4 %
Precision : 87.1 %
Recall : 83.4 %
F1 Score : 0.852
Specificity or True Negative Rate : 90.5 %
```

RandomForestClassifier

In [57]:

```
from sklearn.ensemble import RandomForestClassifier
ModelRF = RandomForestClassifier()
ModelRF.fit(x_train, y_train)

y_pred = ModelRF.predict(x_test)

y_pred_prob = ModelRF.predict_proba(x_test)

print('Model Name: ',ModelRF )


from sklearn.metrics import confusion_matrix
actual = y_test
predicted = y_pred
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, r
print('Confusion matrix : \n', matrix)
```

```
Model Name:  RandomForestClassifier()
Confusion matrix :
 [[17545  1083]
 [  550 23683]]
```

In [58]:

```
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : tp:{} fn:{} fp:{} tn:{}\n'.format(tp, fn, fp, tn) )


sensitivity = round(tp/(tp+fn), 3)
specificity = round(tn/(tn+fp), 3)
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3)
precision = round(tp/(tp+fp), 3)
f1Score = round((2*tp/(2*tp + fp + fn)), 3)

from math import sqrt

print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
```

```
Outcome values : tp:17545 fn:1083 fp:550 tn:23683

Accuracy : 96.2 %
Precision : 97.0 %
Recall : 94.2 %
F1 Score : 0.956
Specificity or True Negative Rate : 97.7 %
```

In [ ]:

In [ ]: