Classification Metrics

Accuracy, Confusion Matrix, Precision, Recall, F1 score, AUC & ROC

Confusion matrices are the result of classification problems. There are four possible values that make up the result: True Positive, False Negative, False Positive, and True Negative.

		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

- **True Positive** — these are cases in which the classification model created correctly predicted positive.
- **False Negative** — these are cases in which the classification model predicted false, but are actually positive. These are also considered Type II errors.
- **False Positive** — these are the cases in which the classification model predicted positive, but are actually negative. These are also considered Type I errors.
- **True Negative** — these are the cases in which the classification model correctly predicted negative.

- **Sensitivity (Recall)** — also known as the True Positive Rate or Recall. Outcomes that are correctly predicted as positive.
- **Specificity** — also known as the True Negative Rate. Outcomes that are correctly predicted as negative.
- **Accuracy** — Outcomes that are correctly labeled at true. it's not compulsory that a good result. Many times data imbalanced problem. Than we use Precision and Recall.
- **Negative Predictive Value** — Outcomes that are correctly labeled as false.
- **Precision** — Outcomes that are correctly predicted positive.

When use Precision and Recall?

- If the negative samples are important, we should focus on precision. Otherwise, we should focus on recall.

- However, as to F1-score, the value higher, the model is better.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

AUC ROC Curve

Setting different thresholds for classifying positive class for data points will inadvertently change the Sensitivity and Specificity of the model. And one of these thresholds will probably give a better result than the others, depending on whether we are aiming to lower the number of False Negatives or False Positives.

ID	Actual	Prediction Probability	>0.6	>0.7	> 0.8	Metric
1	0	0.98	1	1	1	
2	1	0.67	1	0	0	
3	1	0.58	0	0	0	
4	0	0.78	1	1	0	
5	1	0.85	1	1	1	
6	0	0.86	1	1	1	
7	0	0.79	1	1	0	
8	0	0.89	1	1	1	
9	1	0.82	1	1	1	
10	0	0.86	1	1	1	
			0.75	0.5	0.5	TPR
			1	1	0.66	FPR
			0	0	0.33	TNR
			0.25	0.5	0.5	FNR

The metrics change with the changing threshold values. We can generate different confusion matrices and compare the various metrics that we discussed in the previous section. But that would not be a prudent thing to do. Instead, what we can do is generate a plot between some of these metrics so that we can easily visualize which threshold is giving us a better result.

The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the ‘signal’ from the ‘noise’. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

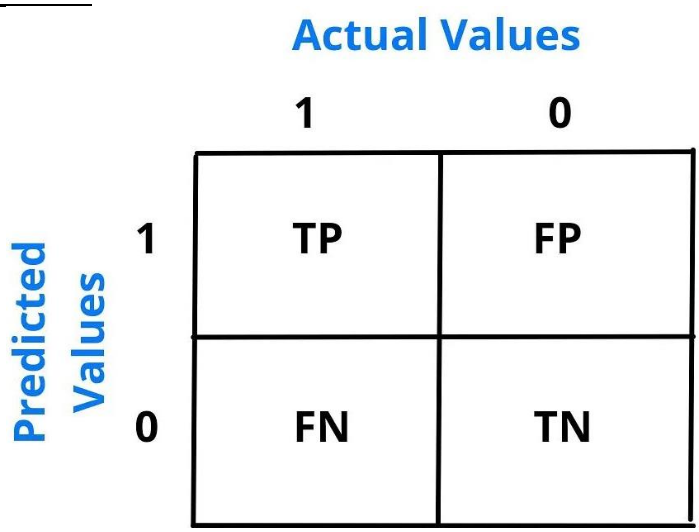
- The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.
- When AUC = 1, then the classifier is able to perfectly distinguish between all the Positive and the Negative class points correctly. If, however, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.
- When 0.5<AUC<1, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. This is so because the classifier is able to detect more numbers of True positives and True negatives than False negatives and False positives.
- When AUC=0.5, then the classifier is not able to distinguish between Positive and Negative class points. Meaning either the classifier is predicting random class or constant class for all the data points.

So, the higher the AUC value for a classifier, the better its ability to distinguish between positive and negative classes.

OR

Evaluation Metrics For Classification

Confusion Matrix:-



What is a confusion Metrix

- Confusion Matrix is the visual representation of the Actual v/s Predicted values.
- It measures the performance of our Machine Learning classification model and looks like a table-like structure.
- This is how a Confusion Matrix of a binary classification problem looks like:

Elements of Confusion Matrix

It represents the different combinations of Actual vs Predicted values. Let's define them one by one.

- **TP (True Positive):** The values which where actually positive and were predicted positive.
- **FP (False Positive):** The values which were actually negative but falsely predicted as positive. Also Known as Type II Error.
- **FN (False Negative):** The values which were actually positive but falsely predicted as negative. Also known as Type II Error.
- **TN (True Negative):** The values which were actually negative and were predicted negative.

Precision And Recall

Precision : Out of all positive predictions, how many are actually positive.

$$Precision = \frac{Predictions\ Actually\ Positive}{Total\ Predicted\ Positive}$$

$$Precision = \frac{True\ Positives}{True\ Positive + False\ Positive}$$

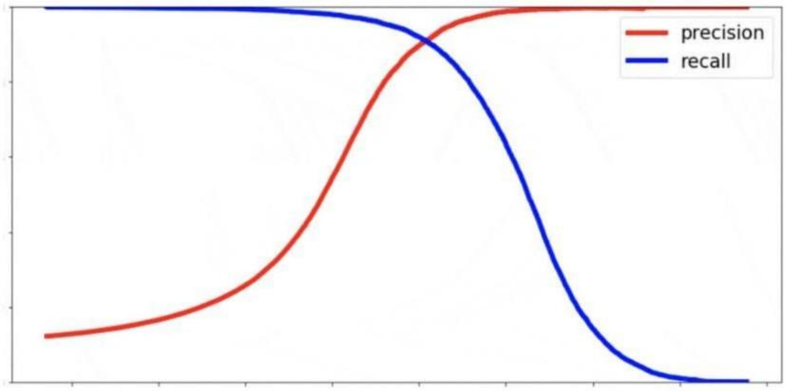
Precision is used when avoiding False Positive is more important than encountering False Negative.

Recall : Out of all actual positive, how many have been predicted as positive.

$$Recall = \frac{Predictions\ Actually\ Positive}{Total\ Actual\ Positive}$$

$$Recall = \frac{True\ Positives}{True\ Positive + False\ Negative}$$

Precision is used when avoideing False Negative is more important than encountering False Positive



- High Precision, Low Recall
- High Recall, Low Precision
- Precision and Recall are used in diverse case and prioritizing one over the other depends upon the use case.

High Precision, Low Recall

If we aim for very high Precision i.e. trying to strictly predict the True Positive and avoiding False Positive, there is a good chance that the False Negative will go higher. Therefore, the recall will drop to a lower value.

High Recall, Low Precision

Similarly, If we aim for very high Recall i.e. trying to strictly predict all the actual positive as True positive and avoiding False negative, there is a good chance that the False positive will go higher. Therefore, the precision will drop to a lower value.

F1 Score

F1-score is a **HARMONIC MEAN** of Precision and Recall. And so it gives a combined idea about these two metrics. It is maximus when Precision is equal to Recall.

When we try toincrease the precision of our model, the recall goes down, and vice-versa. The F1-score captures both the trends in a single value.

F1-score is a harmonic mean of Precision and Recall, But why are taking a harmonic mean and not an arithmetic mean?

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$$

- This is because harmonic mean punishes extreme values more.
- Let us understand this with an example. We have a binary classification model with the following results:

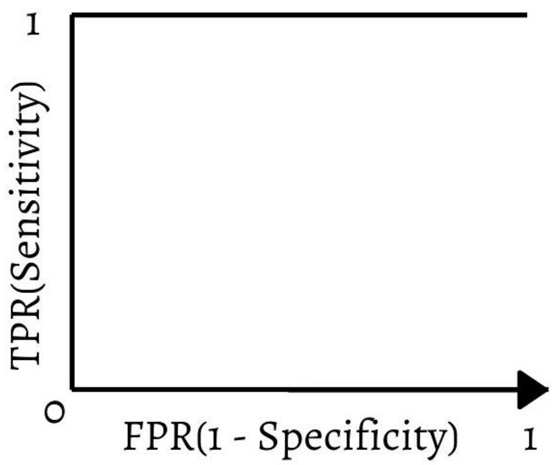
  Precision: 0, Recall: 1

  Here, if we take the arithmetic mean, we get 0.5. It is clear that the above result comes from a dumb classifier which just ignores the input and just predicts one of the classes as outputs.
- Now, if we were to take HM, we will gat 0 which is accurate as this model is useless for all purposes.
- This seems simple. There are situations however for which a data scientist would like to give a percentage more importance/weight to either precision or recall.

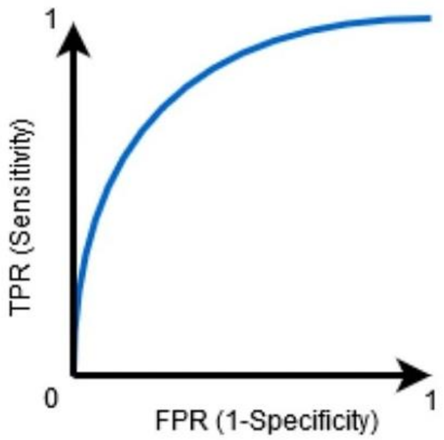
AUC ROC Curve

The Receiver Operator Characteristics (ROC) curve is an evaluation metric for **BINARY CLASSIFICATION** problems. It is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the 'signal' from the 'noise'. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

Note: The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.



When AUC = 1, then the classifer is able to perfectly distinguish between all the Positive and the Negative class points correctly. If, hower, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.



When 0.5<AUC<1, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. This is so because the classifier is able to detect more numbers of True positives and True negatives than False negatives and False positives.

Log Loss :- AUR ROC considers the predicted probabilities for determining our model's performance. However, there is an Issue with AUC – ROC, it only takes into account the order of probabilities and hence it does not take into account the model's capability to predict higher probability for samples more likely to be positive.

In [1]:
```
import pandas as pd
```

In [3]:
```
df = pd.read_csv('hotel_booking.csv')
```

In [5]:
```
df.sample()
```

Out[5]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_nun
11420	Resort Hotel	1	86	2017	May	

1 rows × 36 columns

In [6]:
```
df.shape
```

Out[6]:
```
(119390, 36)
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'stays_in_weekend_nights',
       'stays_in_week_nights', 'adults', 'children', 'babies', 'meal',
       'country', 'market_segment', 'distribution_channel',
       'is_repeated_guest', 'previous_cancellations',
       'previous_bookings_not_canceled', 'reserved_room_type',
       'assigned_room_type', 'booking_changes', 'deposit_type', 'agent',
       'company', 'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date', 'name', 'email',
       'phone-number', 'credit_card'],
      dtype='object')
```

In [8]:
```
# Deleting unuseful columns
df = df.drop(['days_in_waiting_list', 'arrival_date_year', 'assigned_room_type', 'b
              'reservation_status', 'country', 'days_in_waiting_list','name', 'ema
```

In [9]:
```
df.shape
```

Out[9]:
```
(119390, 26)
```

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
hotel                                  0
is_canceled                            0
lead_time                              0
arrival_date_month                     0
arrival_date_week_number               0
arrival_date_day_of_month              0
stays_in_weekend_nights                0
stays_in_week_nights                   0
adults                                 0
children                               4
babies                                 0
meal                                   0
market_segment                         0
distribution_channel                   0
is_repeated_guest                      0
previous_cancellations                 0
previous_bookings_not_canceled         0
reserved_room_type                     0
deposit_type                           0
agent                              16340
company                           112593
customer_type                          0
adr                                    0
required_car_parking_spaces            0
total_of_special_requests              0
reservation_status_date                0
dtype: int64
```

In [11]:
```
df = df.bfill().ffill()
```

In [12]:
```
df.dtypes
```

Out[12]:
```
hotel                              object
is_canceled                         int64
lead_time                           int64
arrival_date_month                 object
arrival_date_week_number            int64
arrival_date_day_of_month           int64
stays_in_weekend_nights             int64
stays_in_week_nights                int64
adults                              int64
children                          float64
babies                              int64
meal                               object
market_segment                     object
distribution_channel               object
is_repeated_guest                   int64
previous_cancellations              int64
previous_bookings_not_canceled      int64
reserved_room_type                 object
deposit_type                       object
agent                             float64
company                           float64
customer_type                      object
adr                               float64
required_car_parking_spaces         int64
total_of_special_requests           int64
reservation_status_date            object
dtype: object
```

In [13]:
```
l = []
```

```python
for i in df.columns:
    if df[i].dtypes =='O':
        l.append(i)
```

In [14]:
```python
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in l:
    df[i] = le.fit_transform(df[i])
```

In [15]:
```python
# df.dtypes
```

In [16]:
```python
x = df.drop('is_canceled',axis=1)
y = df['is_canceled']
```

In [17]:
```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20)
```

In [18]:
```python
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
```

In [19]:
```python
dt.fit(x_train,y_train)
```

Out[19]:
```
DecisionTreeClassifier()
```

In [18]:
```python
dt_pre = dt.predict(x_test)
```

In [19]:
```python
data = pd.DataFrame({'Actual Value':y_test,'Predicted Value':dt_pre})
```

In [20]:
```python
data
```

Out[20]:

	Actual Value	Predicted Value
247	1	1
81798	1	1
114200	0	0
110394	0	0
33450	0	0
...	...	...
60337	1	1
13293	1	1
50184	1	1
43940	1	1
94686	0	0

23878 rows × 2 columns

In [21]:
```python
from sklearn.metrics import accuracy_score
accuracy_score(y_test, dt_pre)
```

Out[21]:
```
0.9377669821593099
```

In [22]:
```python
from sklearn.metrics import classification_report, confusion_matrix
cf_matrix = confusion_matrix(y_test,dt_pre)
print(cf_matrix)
print(classification_report(y_test,dt_pre))
```

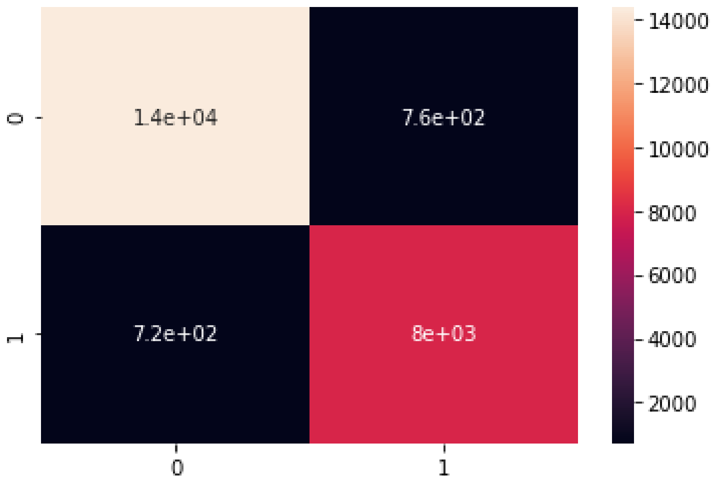
```
[[14351   764]
 [  722  8041]]
              precision    recall  f1-score   support

           0       0.95      0.95      0.95     15115
           1       0.91      0.92      0.92      8763

    accuracy                           0.94     23878
   macro avg       0.93      0.93      0.93     23878
weighted avg       0.94      0.94      0.94     23878
```

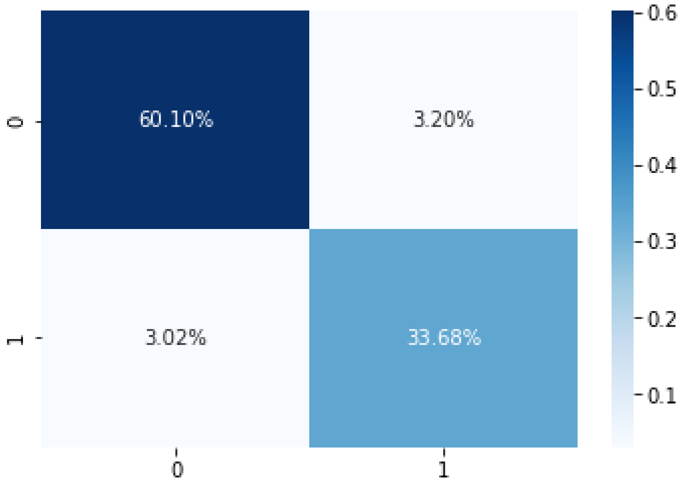
In [23]:
```python
import seaborn as sns
sns.heatmap(cf_matrix, annot=True)
```

Out[23]: <AxesSubplot:>



In [24]:
```python
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True,
            fmt='.2%', cmap='Blues')
```

Out[24]: <AxesSubplot:>



In [25]:
```python
# non-binary classifier (3x3 in this case)

# make_confusion_matrix(cf_matrix_3x3, figsize=(8,6), cbar=False)
```

In [26]:
```
confusion = confusion_matrix(y_test, dt_pre)
confusion.ravel()
```

Out[26]:
```
array([14351,   764,   722,  8041], dtype=int64)
```

In [27]:
```
from sklearn.metrics import precision_score
precision_positive = precision_score(y_test, dt_pre, pos_label=1)
precision_negative = precision_score(y_test, dt_pre, pos_label=0)

precision_positive, precision_negative
```

Out[27]:
```
(0.9132311186825667, 0.9520997810654813)
```

In [28]:
```
from sklearn.metrics import recall_score
recall_sensitivity = recall_score(y_test, dt_pre, pos_label=1)
recall_specificity = recall_score(y_test, dt_pre, pos_label=0)

recall_sensitivity, recall_specificity
```

Out[28]:
```
(0.9176081250713226, 0.9494541845848495)
```

In [29]:
```
from sklearn.metrics import f1_score
f1_positive = f1_score(y_test, dt_pre, pos_label=1)
f1_negative = f1_score(y_test, dt_pre, pos_label=0)

f1_positive, f1_negative
```

Out[29]:
```
(0.9154143897996357, 0.9507751424407049)
```

In [30]:
```
from sklearn.metrics import log_loss
log_loss(y_test,dt_pre)
```

Out[30]:
```
2.149477871483121
```

In [31]:
```
# AUC ROC Curve

from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split

# generate two class dataset
x, y = make_classification(n_samples=1000,n_classes=2,n_features=20,random_state=24
                                                                                    
# split into train-test sets
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,random_state=

# train models
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier

# logistic regression
model1 = LogisticRegression()
# knn
model2 = KNeighborsClassifier(n_neighbors=4)

# fit model
model1.fit(x_train, y_train)
model2.fit(x_train, y_train)

# predict probabilities
```

```python
pred_prob1 = model1.predict_proba(x_test)
pred_prob2 = model2.predict_proba(x_test)

from sklearn.metrics import roc_curve

# roc curve for models
fpr1, tpr1, thresh1 = roc_curve(y_test, pred_prob1[:, 1], pos_label=1)
fpr2, tpr2, thresh2 = roc_curve(y_test, pred_prob2[:, 1], pos_label=1)

# roc curve for tpr = fpr
random_probs = [0 for i in range(len(y_test))]
p_fpr, p_tpr, _ = roc_curve(y_test, random_probs, pos_label=1)

from sklearn.metrics import roc_auc_score

# auc scores
auc_score1 = roc_auc_score(y_test, pred_prob1[:, 1])
auc_score2 = roc_auc_score(y_test, pred_prob2[:, 1])

print(auc_score1, auc_score2)

print(auc_score1, auc_score2)

print(auc_score1, auc_score2)
```

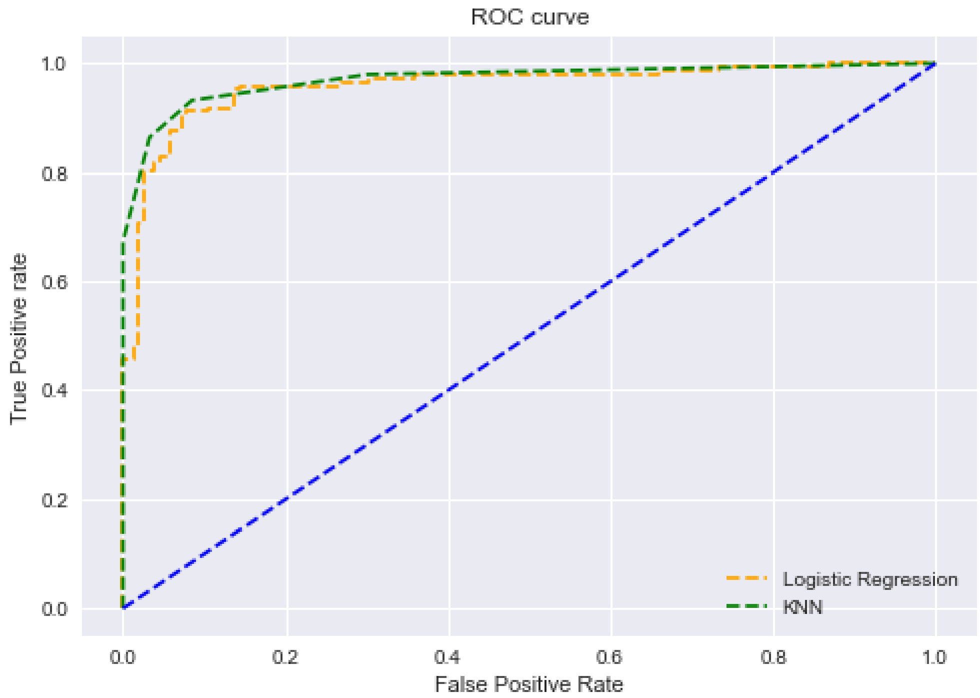
```
0.958249966653328 0.970432617491441
0.958249966653328 0.970432617491441
0.958249966653328 0.970432617491441
```

In [32]:

```python
# matplotlib
import matplotlib.pyplot as plt
plt.style.use('seaborn')

# plot roc curves
plt.plot(fpr1, tpr1, linestyle='--',color='orange', label='Logistic Regression')
plt.plot(fpr2, tpr2, linestyle='--',color='green', label='KNN')
plt.plot(p_fpr, p_tpr, linestyle='--', color='blue')
# title
plt.title('ROC curve')
# x label
plt.xlabel('False Positive Rate')
# y label
plt.ylabel('True Positive rate')

plt.legend(loc='best')
plt.savefig('ROC',dpi=300)
plt.show();
```


ROC curve
1.0
0.8
0.6
0.4
0.2
0.0
True Positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
Logistic Regression
KNN