```
import pandas as pd
data = pd.read_excel("/content/PCOS_data_without_infertility.xlsx")
data.head(10)
# data.info()
```

	Sl. No	Patient File No.	PCOS (Y/N)	Age (yrs)	Weight (Kg)	Height(Cm)	BMI	Blood Group	Pulse rate(bpm)	RR (breaths/min)	...	Fast food (Y/N)	Re
0	1	1	0	28	44.6	152.0	19.3	15	78	22	...	1.0	
1	2	2	0	36	65.0	161.5	NaN	15	74	20	...	0.0	
2	3	3	1	33	68.8	165.0	NaN	11	72	18	...	1.0	
3	4	4	0	37	65.0	148.0	NaN	13	72	20	...	0.0	
4	5	5	0	25	52.0	161.0	NaN	11	72	18	...	0.0	
5	6	6	0	36	74.1	165.0	NaN	15	78	28	...	0.0	
6	7	7	0	34	64.0	156.0	NaN	11	72	18	...	0.0	
7	8	8	0	33	58.5	159.0	NaN	13	72	20	...	0.0	
8	9	9	0	32	40.0	158.0	NaN	11	72	18	...	0.0	
9	10	10	0	36	52.0	150.0	NaN	15	80	20	...	0.0	

10 rows × 45 columns

```
del data['Unnamed: 44']
del data['Sl. No']
del data['Patient File No.']
data['Marraige Status (Yrs)'].fillna(0,inplace = True)
data['Fast food (Y/N)'].fillna(0,inplace = True)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541 entries, 0 to 540
Data columns (total 42 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   PCOS (Y/N)              541 non-null    int64  
 1    Age (yrs)              541 non-null    int64  
 2   Weight (Kg)             541 non-null    float64
 3   Height(Cm)              541 non-null    float64
 4   BMI                     242 non-null    float64
 5   Blood Group             541 non-null    int64  
 6   Pulse rate(bpm)         541 non-null    int64  
 7   RR (breaths/min)        541 non-null    int64  
 8   Hb(g/dl)                541 non-null    float64
 9   Cycle(R/I)              541 non-null    int64  
 10  Cycle length(days)      541 non-null    int64  
 11  Marraige Status (Yrs)   541 non-null    float64
 12  Pregnant(Y/N)           541 non-null    int64  
 13  No. of aborptions       541 non-null    int64  
 14    I   beta-HCG(mIU/mL)  541 non-null    float64
 15  II    beta-HCG(mIU/mL)  541 non-null    object 
 16  FSH(mIU/mL)             541 non-null    float64
 17  LH(mIU/mL)              541 non-null    float64
 18  FSH/LH                  9 non-null      float64
 19  Hip(inch)               541 non-null    int64  
 20  Waist(inch)             541 non-null    int64  
 21  Waist:Hip Ratio         9 non-null      float64
 22  TSH (mIU/L)             541 non-null    float64
 23  AMH(ng/mL)              541 non-null    object 
 24  PRL(ng/mL)              541 non-null    float64
 25  Vit D3 (ng/mL)          541 non-null    float64
 26  PRG(ng/mL)              541 non-null    float64
 27  RBS(mg/dl)              541 non-null    float64
 28  Weight gain(Y/N)        541 non-null    int64  
 29  hair growth(Y/N)        541 non-null    int64  
 30  Skin darkening (Y/N)    541 non-null    int64  
 31  Hair loss(Y/N)          541 non-null    int64  
 32  Pimples(Y/N)            541 non-null    int64  
```

```
 33  Fast food (Y/N)         541 non-null    float64
 34  Reg.Exercise(Y/N)       541 non-null    int64
 35  BP _Systolic (mmHg)     541 non-null    int64
 36  BP _Diastolic (mmHg)    541 non-null    int64
 37  Follicle No. (L)        541 non-null    int64
 38  Follicle No. (R)        541 non-null    int64
 39  Avg. F size (L) (mm)    541 non-null    float64
 40  Avg. F size (R) (mm)    541 non-null    float64
 41  Endometrium (mm)        541 non-null    float64
dtypes: float64(19), int64(21), object(2)
memory usage: 177.6+ KB
```

```
data["AMH(ng/mL)"] = pd.to_numeric(data["AMH(ng/mL)"], errors='coerce')
data["II    beta-HCG(mIU/mL)"] = pd.to_numeric(data["II    beta-HCG(mIU/mL)"], errors='coerce')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541 entries, 0 to 540
Data columns (total 42 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   PCOS (Y/N)              541 non-null    int64
 1    Age (yrs)              541 non-null    int64
 2   Weight (Kg)             541 non-null    float64
 3   Height(Cm)              541 non-null    float64
 4   BMI                     242 non-null    float64
 5   Blood Group             541 non-null    int64
 6   Pulse rate(bpm)         541 non-null    int64
 7   RR (breaths/min)        541 non-null    int64
 8   Hb(g/dl)                541 non-null    float64
 9   Cycle(R/I)              541 non-null    int64
 10  Cycle length(days)      541 non-null    int64
 11  Marraige Status (Yrs)   541 non-null    float64
 12  Pregnant(Y/N)           541 non-null    int64
 13  No. of aborptions       541 non-null    int64
 14    I   beta-HCG(mIU/mL)  541 non-null    float64
 15  II    beta-HCG(mIU/mL)  540 non-null    float64
 16  FSH(mIU/mL)             541 non-null    float64
 17  LH(mIU/mL)              541 non-null    float64
 18  FSH/LH                  9 non-null      float64
 19  Hip(inch)               541 non-null    int64
 20  Waist(inch)             541 non-null    int64
 21  Waist:Hip Ratio         9 non-null      float64
 22  TSH (mIU/L)             541 non-null    float64
 23  AMH(ng/mL)              540 non-null    float64
 24  PRL(ng/mL)              541 non-null    float64
 25  Vit D3 (ng/mL)          541 non-null    float64
 26  PRG(ng/mL)              541 non-null    float64
 27  RBS(mg/dl)              541 non-null    float64
 28  Weight gain(Y/N)        541 non-null    int64
 29  hair growth(Y/N)        541 non-null    int64
 30  Skin darkening (Y/N)    541 non-null    int64
 31  Hair loss(Y/N)          541 non-null    int64
 32  Pimples(Y/N)            541 non-null    int64
 33  Fast food (Y/N)         541 non-null    float64
 34  Reg.Exercise(Y/N)       541 non-null    int64
 35  BP _Systolic (mmHg)     541 non-null    int64
 36  BP _Diastolic (mmHg)    541 non-null    int64
 37  Follicle No. (L)        541 non-null    int64
 38  Follicle No. (R)        541 non-null    int64
 39  Avg. F size (L) (mm)    541 non-null    float64
 40  Avg. F size (R) (mm)    541 non-null    float64
 41  Endometrium (mm)        541 non-null    float64
dtypes: float64(21), int64(21)
memory usage: 177.6 KB
```

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

```
PCOS (Y/N)                 0
 Age (yrs)                 0
Weight (Kg)                0
Height(Cm)                 0
BMI                      299
Blood Group                0
Pulse rate(bpm)            0
RR (breaths/min)           0
Hb(g/dl)                   0
Cycle(R/I)                 0
Cycle length(days)         0
Marraige Status (Yrs)      0
Pregnant(Y/N)              0
No. of aborptions          0
```

```
  I   beta-HCG(mIU/mL)      0
II    beta-HCG(mIU/mL)      1
FSH(mIU/mL)                 0
LH(mIU/mL)                  0
FSH/LH                    532
Hip(inch)                   0
Waist(inch)                 0
Waist:Hip Ratio           532
TSH (mIU/L)                 0
AMH(ng/mL)                  1
PRL(ng/mL)                  0
Vit D3 (ng/mL)              0
PRG(ng/mL)                  0
RBS(mg/dl)                  0
Weight gain(Y/N)            0
hair growth(Y/N)            0
Skin darkening (Y/N)        0
Hair loss(Y/N)              0
Pimples(Y/N)                0
Fast food (Y/N)             0
Reg.Exercise(Y/N)           0
BP _Systolic (mmHg)         0
BP _Diastolic (mmHg)        0
Follicle No. (L)            0
Follicle No. (R)            0
Avg. F size (L) (mm)        0
Avg. F size (R) (mm)        0
Endometrium (mm)            0
dtype: int64
```

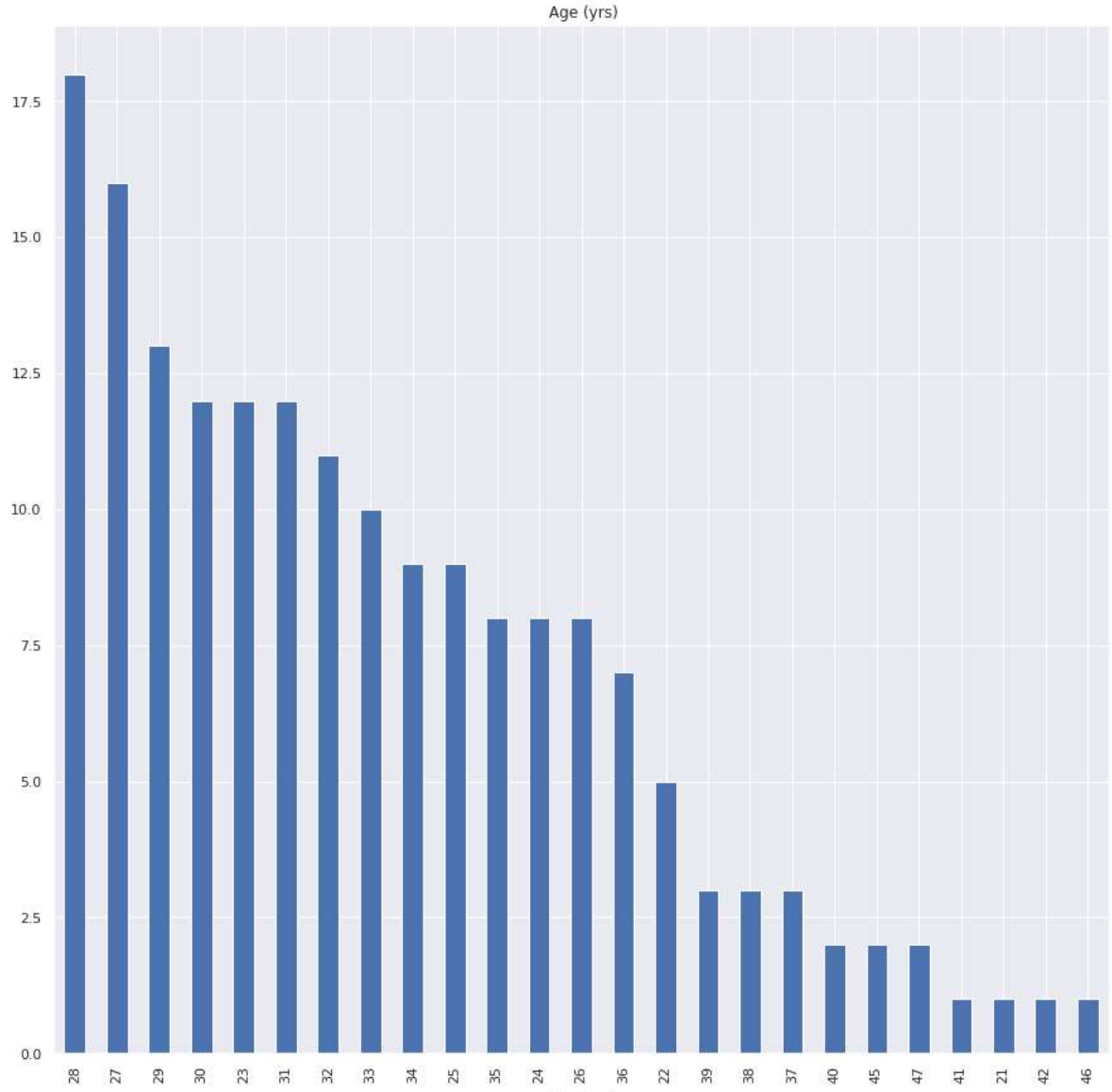
```
data["Waist:Hip Ratio"].fillna(data["Waist:Hip Ratio"].median(),inplace=True)
data["BMI"].fillna(data["BMI"].median(),inplace=True)
data["FSH/LH"].fillna(data["FSH/LH"].median(),inplace=True)
data["II    beta-HCG(mIU/mL)"].fillna(data["II    beta-HCG(mIU/mL)"].median(),inplace=True)
data["AMH(ng/mL)"].fillna(data["AMH(ng/mL)"].median(),inplace=True)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541 entries, 0 to 540
Data columns (total 42 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   PCOS (Y/N)              541 non-null    int64  
 1    Age (yrs)              541 non-null    int64  
 2   Weight (Kg)             541 non-null    float64
 3   Height(Cm)              541 non-null    float64
 4   BMI                     541 non-null    float64
 5   Blood Group             541 non-null    int64  
 6   Pulse rate(bpm)         541 non-null    int64  
 7   RR (breaths/min)        541 non-null    int64  
 8   Hb(g/dl)                541 non-null    float64
 9   Cycle(R/I)              541 non-null    int64  
 10  Cycle length(days)      541 non-null    int64  
 11  Marraige Status (Yrs)   541 non-null    float64
 12  Pregnant(Y/N)           541 non-null    int64  
 13  No. of aborptions       541 non-null    int64  
 14    I   beta-HCG(mIU/mL)  541 non-null    float64
 15  II    beta-HCG(mIU/mL)  541 non-null    float64
 16  FSH(mIU/mL)             541 non-null    float64
 17  LH(mIU/mL)              541 non-null    float64
 18  FSH/LH                  541 non-null    float64
 19  Hip(inch)               541 non-null    int64  
 20  Waist(inch)             541 non-null    int64  
 21  Waist:Hip Ratio         541 non-null    float64
 22  TSH (mIU/L)             541 non-null    float64
 23  AMH(ng/mL)              541 non-null    float64
 24  PRL(ng/mL)              541 non-null    float64
 25  Vit D3 (ng/mL)          541 non-null    float64
 26  PRG(ng/mL)              541 non-null    float64
 27  RBS(mg/dl)              541 non-null    float64
 28  Weight gain(Y/N)        541 non-null    int64  
 29  hair growth(Y/N)        541 non-null    int64  
 30  Skin darkening (Y/N)    541 non-null    int64  
 31  Hair loss(Y/N)          541 non-null    int64  
 32  Pimples(Y/N)            541 non-null    int64  
 33  Fast food (Y/N)         541 non-null    float64
 34  Reg.Exercise(Y/N)       541 non-null    int64  
 35  BP _Systolic (mmHg)     541 non-null    int64  
 36  BP _Diastolic (mmHg)    541 non-null    int64  
 37  Follicle No. (L)        541 non-null    int64  
 38  Follicle No. (R)        541 non-null    int64
```

```
 39  Avg. F size (L) (mm)    541 non-null    float64
 40  Avg. F size (R) (mm)    541 non-null    float64
 41  Endometrium (mm)        541 non-null    float64
dtypes: float64(21), int64(21)
memory usage: 177.6 KB
```

```
import matplotlib.pyplot as plt #for plotting simple graphs
import seaborn as sns #another plotting library
for i in [' Age (yrs)', 'Weight (Kg)',
       'Height(Cm) ', 'Hb(g/dl)', 'Cycle(R/I)', 'Cycle length(days)', 'No. of aborptions',
        'Hip(inch)', 'Waist(inch)',
       'PRG(ng/mL)', 'RBS(mg/dl)', 'BP _Systolic (mmHg)', 'Follicle No. (L)', 'Follicle No. (R)',
       'Avg. F size (L) (mm)', 'Avg. F size (R) (mm)', 'Endometrium (mm)']:
  sns.set(rc = {'figure.figsize':(15,15)})
  data[data['PCOS (Y/N)'] == 1][i].value_counts().plot.bar()
  plt.title(i)
  plt.show()
```


Age (yrs)
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
28
27
29
30
23
31
32
33
34
25
35
24
26
36
22
39
38
37
40
45
47
41
21
42
46


Weight (Kg)
14
12
10
8
6

```
data['PCOS (Y/N)'].value_counts()
```

```
0    364
1    177
Name: PCOS (Y/N), dtype: int64
```

```
X=data.drop(["PCOS (Y/N)"],axis = 1)
y=data["PCOS (Y/N)"]
```

```
from imblearn.over_sampling import RandomOverSampler
oversample = RandomOverSampler(sampling_strategy=0.7)
X, y = oversample.fit_resample(X, y)
y.value_counts()
```

```
0    364
1    254
Name: PCOS (Y/N), dtype: int64
```

```
import sklearn
from sklearn.preprocessing import PowerTransformer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.tree import DecisionTreeRegressor
import math
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
# from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
```

```
sscaler = MinMaxScaler() #helps us scale the dataset. This makes it easy for the model to train
cols = X.columns
x_scaled = sscaler.fit_transform(X)
X_scaled = pd.DataFrame(x_scaled, columns = cols)
X_scaled
```

```
Age  Weight  Height(Cm)  BMI  Blood  Pulse  RR  Hb(g/dl)  Cycle(R/I)  Cycle  Pimples(Y/N)
```

```
X_train,X_test, y_train, y_test = train_test_split(X , y, test_size=0.2)
```

```
rfc = RandomForestClassifier(n_jobs=-1,n_estimators=150,max_features='sqrt',min_samples_leaf=10) #creates a Random forest model
rfc.fit(X_train, y_train) #trains model on data
pred_rfc = rfc.predict(X_test) #prediction
accuracy = accuracy_score(y_test, pred_rfc)
print(accuracy)
```

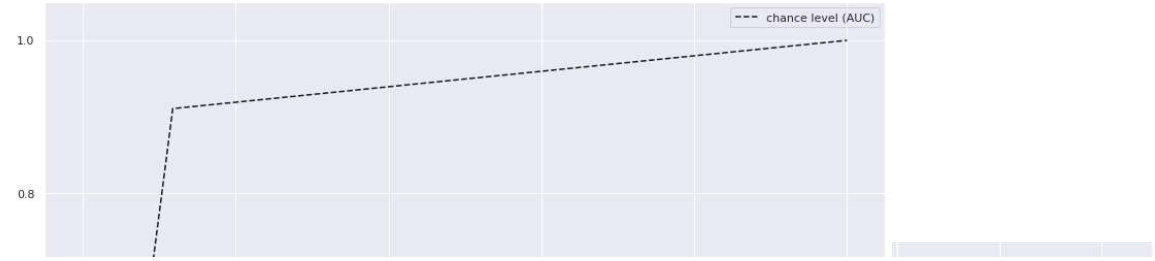
```
0.8951612903225806
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn import metrics


print('Confusion matrix :',confusion_matrix(y_test,pred_rfc))
print('Accuracy score :', accuracy_score(y_test,pred_rfc))
print('Precision Score :', precision_score(y_test,pred_rfc,pos_label=1,average='macro'))
print('Recall Score :', recall_score(y_test,pred_rfc,pos_label=1,average='macro'))
fpr, tpr, thresholds = metrics.roc_curve(y_test,pred_rfc, pos_label=1)
print('fpr :', fpr)
print('tpr :', tpr)
print('thresholds :', thresholds)
auc = metrics.auc(fpr, tpr)
print('auc :', auc)

plt.plot(fpr,tpr, "k--", label="chance level (AUC)")
plt.axis("square")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
```

```
Confusion matrix : [[60  8]
 [ 5 51]]
Accuracy score : 0.8951612903225806
Precision Score : 0.8937418513689701
Recall Score : 0.8965336134453781
fpr : [0.         0.11764706 1.        ]
tpr : [0.         0.91071429 1.        ]
thresholds : [2 1 0]
auc : 0.8965336134453781
```




```
classi_report = classification_report(y_test, pred_rfc)
print(classi_report)
```

```
                  precision    recall  f1-score   support

               0       0.92      0.88      0.90        68
               1       0.86      0.91      0.89        56

        accuracy                           0.90       124
       macro avg       0.89      0.90      0.89       124
    weighted avg       0.90      0.90      0.90       124
```

```
import xgboost as xgb

xgb_cl = xgb.XGBClassifier(learning_rate = 0.001, gamma = 0.03, max_depth = 20, subsample = 0.5 )
xgb_cl.fit(X_train, y_train)

# Predict
preds = xgb_cl.predict(X_test)

# Score
accuracy_score(y_test, preds)
```

```
0.8629032258064516
```

```
rfc = RandomForestClassifier(n_jobs=-1,n_estimators=150,max_features='sqrt',min_samples_leaf=10)
xgb = xgb.XGBClassifier(learning_rate = 0.001, gamma = 0.03, max_depth = 20, subsample = 0.5)
l = [('rf',rfc), ('xgb', xgb)]
from sklearn.ensemble import StackingClassifier
stack_model = StackingClassifier( estimators = l)
score = cross_val_score(stack_model,X_scaled,y,cv = 5,scoring = 'accuracy')
```

```
print(score)
```

```
[0.91129032 0.88709677 0.88709677 0.86178862 0.87804878]
```

```
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
warnings.filterwarnings('ignore')

#fitting logistic regression to training set
# model1 = Perceptron(eta0=1.0,max_iter=1000,tol=1e-3,random_state=42)
model1 = MLPClassifier(random_state=1, max_iter=300).fit(X_train, y_train)
model1.fit(X_train,y_train)
#prediction
prediction1 = model1.predict(X_test)
```

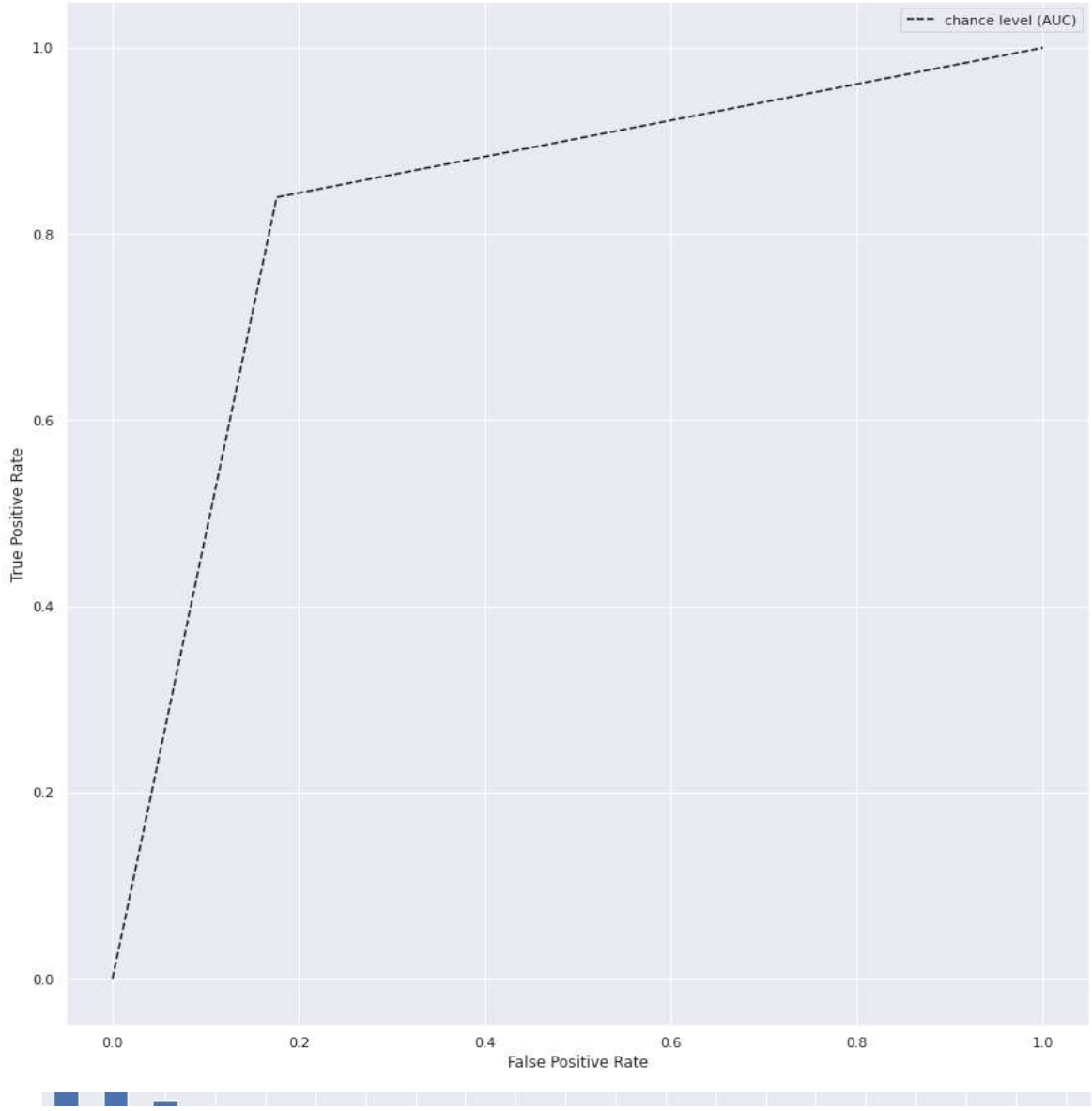
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

```
from sklearn.metrics import f1_score
from sklearn import metrics


print('Confusion matrix :',confusion_matrix(y_test,prediction1))
print('Accuracy score :', accuracy_score(y_test,prediction1))
print('Precision Score :', precision_score(y_test,prediction1,pos_label=1,average='macro'))
print('Recall Score :', recall_score(y_test,prediction1,pos_label=1,average='macro'))
fpr, tpr, thresholds = metrics.roc_curve(y_test, prediction1, pos_label=1)
print('fpr :', fpr)
print('tpr :', tpr)
print('thresholds :', thresholds)
auc = metrics.auc(fpr, tpr)
print('auc :', auc)

plt.plot(fpr,tpr, "k--", label="chance level (AUC)")
plt.axis("square")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
```

```
Confusion matrix : [[56 12]
 [ 9 47]]
Accuracy score : 0.8306451612903226
Precision Score : 0.8290743155149936
Recall Score : 0.83140756302521
fpr : [0.         0.17647059 1.        ]
tpr : [0.         0.83928571 1.        ]
thresholds : [2 1 0]
auc : 0.83140756302521
```



```
from sklearn.ensemble import AdaBoostClassifier
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train,X_test, y_train, y_test = train_test_split(X , y, test_size=0.2)
abc = AdaBoostClassifier(n_estimators=50,learning_rate=1)
model = abc.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

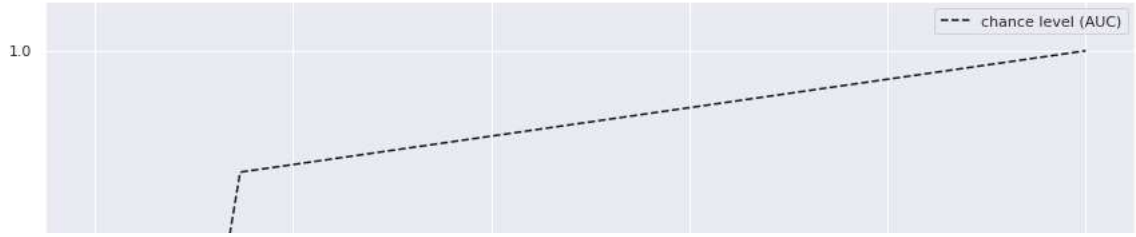
```
Accuracy: 0.8629032258064516
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn import metrics


print('Confusion matrix :',confusion_matrix(y_test,y_pred))
print('Accuracy score :', accuracy_score(y_test,y_pred))
print('Precision Score :', precision_score(y_test,y_pred,pos_label=1,average='macro'))
print('Recall Score :', recall_score(y_test,y_pred,pos_label=1,average='macro'))
fpr, tpr, thresholds = metrics.roc_curve(y_test,y_pred, pos_label=1)
print('fpr :', fpr)
print('tpr :', tpr)
print('thresholds :', thresholds)
auc = metrics.auc(fpr, tpr)
print('auc :', auc)

plt.plot(fpr,tpr, "k--", label="chance level (AUC)")
plt.axis("square")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
```

```
Confusion matrix : [[64 11]
 [ 6 43]]
Accuracy score : 0.8629032258064516
Precision Score : 0.8552910052910052
Recall Score : 0.8654421768707483
fpr : [0.         0.14666667 1.        ]
tpr : [0.         0.87755102 1.        ]
thresholds : [2 1 0]
auc : 0.8654421768707482
```




```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn import metrics


print('Confusion matrix :',confusion_matrix(y_test,y_pred))
print('Accuracy score :', accuracy_score(y_test,y_pred))
print('Precision Score :', precision_score(y_test,y_pred,pos_label=1,average='macro'))
print('Recall Score :', recall_score(y_test,y_pred,pos_label=1,average='macro'))
fpr, tpr, thresholds = metrics.roc_curve(y_test,y_pred, pos_label=1)
print('fpr :', fpr)
print('tpr :', tpr)
print('thresholds :', thresholds)
auc = metrics.auc(fpr, tpr)
print('auc :', auc)

plt.plot(fpr,tpr, "k--", label="chance level (AUC)")
plt.axis("square")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
```