XGBoost Model

```
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
sklearn.__version__
from xgboost import XGBClassifier

# Import necessary modules
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.metrics import r2_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_report,confusion_matrix
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
from sklearn.metrics import classification_report
from sklearn.ensemble.forest import RandomForestClassifier
from sklearn.metrics import make_scorer
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import cross_validate
from sklearn.model_selection import train_test_split
#from sklearn.metrics import ross_validate
from sklearn.svm import SVR
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: The
  warnings.warn(message, FutureWarning)
```

```
df = pd.read_csv('breast_cancer_data.csv')
df.head()
```

```
print(df.shape)
df.describe().transpose()
```

```
(569, 32)
```

	count	mean	std	min	25%	
id	569.0	3.037183e+07	1.250206e+08	8670.000000	869218.000000	90602

```
target_column = ['diagnosis']
predictors = list(set(list(df.columns))-set(target_column))
df[predictors] = df[predictors]/df[predictors].max()
df.describe().transpose()
```

	count	mean	std	min	25%	50%	
id	569.0	0.033327	0.137186	0.000010	0.000954	0.000994	0.00
diagnosis	569.0	0.372583	0.483918	0.000000	0.000000	0.000000	1.00
radius_1ean	569.0	0.502572	0.125366	0.248346	0.416222	0.475631	0.56

```
X = df[predictors].values
y = df[target_column].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=40)
print(X_train.shape);
print(X_test.shape)
```

```
     (398, 31)
     (171, 31)
```

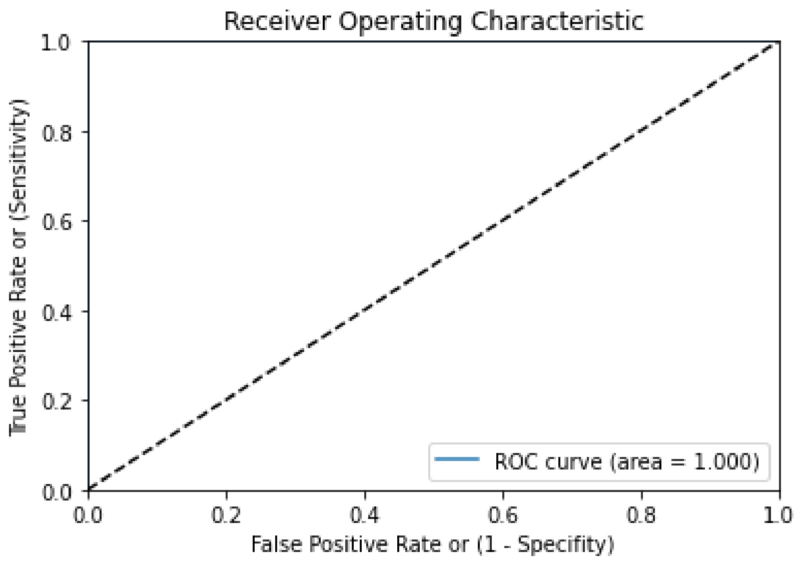
concave points_1ean	569.0	0.243137	0.192857	0.000000	0.100944	0.166501	0.36

```
model = RandomForestClassifier()

#scoring = {'accuracy','recall', 'precision','f1','roc_auc','specificity': make_scorer(recall_sco
scoring = {
    'accuracy': make_scorer(accuracy_score),
    'sensitivity/recall': make_scorer(recall_score),
    'specificity': make_scorer(recall_score,pos_label=0),
    'precision': make_scorer(precision_score),
    'f1':make_scorer(f1_score),
    'roc_auc':make_scorer(roc_auc_score)
}

cv_results = cross_validate(model, X, y.ravel(), cv=5, scoring=scoring)
# Getting the test set true positive scores
#print(cv_results.keys())
#print(cv_results.values())
cr = pd.DataFrame(cv_results)
cr
```

	fit_time	score_time	test_accuracy	test_sensitivity/recall	test_specificity
0	0.177191	0.012270	0.938596	0.906977	0.957746
1	0.199161	0.012567	0.938596	0.860465	0.985915
2	0.198979	0.012591	0.982456	0.976190	0.986111
3	0.191579	0.011537	0.973684	0.952381	0.986111
4	0.175838	0.012699	0.973451	0.976190	0.971831

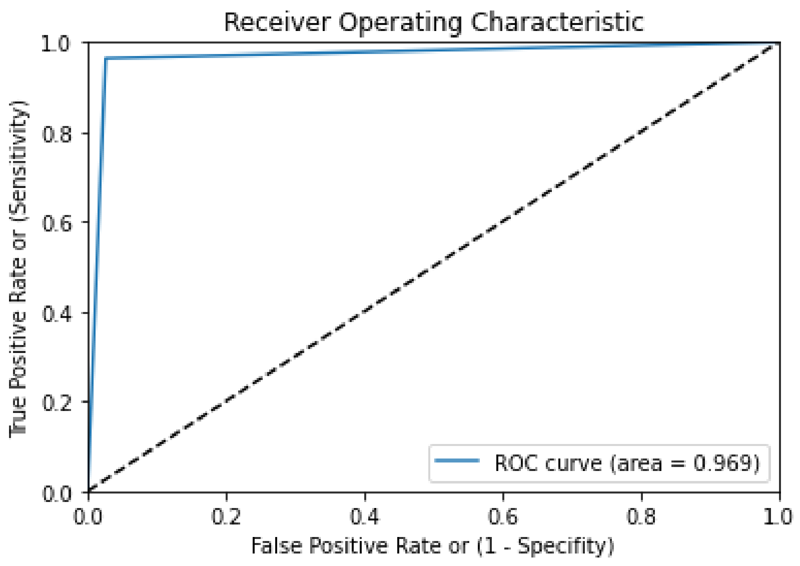
```
model = XGBClassifier()
model.fit(X_train,y_train.ravel())

predict_train = model.predict(X_train)
predict_test = model.predict(X_test)


print("Confustion Matrix For Training Data")
print("------------------------------------")
print(confusion_matrix(y_train,predict_train))
print("------------------------------------")
print("Accuracy:", accuracy_score(y_train,predict_train))
print("Sensitivity/Recall:",metrics.recall_score(y_train,predict_train))
tn, fp, fn, tp = confusion_matrix(y_train,predict_train).ravel()
specificity = tn / (tn+fp)
print("Specificity:", specificity)
print("Precision:", metrics.precision_score(y_train,predict_train))
print("F-Score:", metrics.f1_score(y_train,predict_train))
print("Mens Squre Error:", mean_squared_error(y_test,predict_test))
print("Root Mens Squre Error:", np.sqrt(mean_squared_error(y_test,predict_test)))
print("ROC_AUC scores:",metrics.roc_auc_score(y_train,predict_train, average="macro"))

# Compute fpr, tpr, thresholds and roc auc
fpr, tpr, thresholds = roc_curve(y_train,predict_train)
roc_auc = auc(fpr,tpr)

# Plot ROC curve
plt.plot(fpr, tpr, label='ROC curve (area = %0.3f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')  # random predictions curve
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate or (1 - Specifity)')
plt.ylabel('True Positive Rate or (Sensitivity)')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
```

```
    Confustion Matrix For Training Data
    ------------------------------------
    [[242   0]
     [  0 156]]
    ------------------------------------
    Accuracy: 1.0
    Sensitivity/Recall: 1.0
    Specificity: 1.0
    Precision: 1.0
    F-Score: 1.0
    Mens Squre Error: 0.029239766081871343
    Root Mens Squre Error: 0.17099639201419234
    ROC_AUC scores: 1.0
    <matplotlib.legend.Legend at 0x7f6425dc2490>
```



```
print("Confustion Matrix For Testing Data")
print(confusion_matrix(y_test,predict_test))
print("-----------------------------------")
print("Accuracy:", accuracy_score(y_test,predict_test))
print("Sensitivity/Recall:",metrics.recall_score(y_test,predict_test))
tn, fp, fn, tp = confusion_matrix(y_test,predict_test).ravel()
specificity = tn / (tn+fp)
print("Specificity:", specificity)
print("Precision:", metrics.precision_score(y_test,predict_test))
print("F-Score:", metrics.f1_score(y_test,predict_test))
print("Mens Squre Error:", mean_squared_error(y_test,predict_test))
print("Root Mens Squre Error:", np.sqrt(mean_squared_error(y_test,predict_test)))
print("ROC_AUC scores:",metrics.roc_auc_score(y_test,predict_test, average="macro"))

# Compute fpr, tpr, thresholds and roc auc
fpr, tpr, thresholds = roc_curve(y_test,predict_test)
roc_auc = auc(fpr,tpr)

# Plot ROC curve
plt.plot(fpr, tpr, label='ROC curve (area = %0.3f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')  # random predictions curve
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate or (1 - Specifity)')
plt.ylabel('True Positive Rate or (Sensitivity)')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
```

```
     Confustion Matrix For Testing Data
     -------------------------------------
     [[112   3]
      [  2  54]]
     -------------------------------------
     Accuracy: 0.9707602339181286
     Sensitivity/Recall: 0.9642857142857143
     Specificity: 0.9739130434782609
     Precision: 0.9473684210526315
     F-Score: 0.9557522123893805
     Mens Squre Error: 0.029239766081871343
     Root Mens Squre Error: 0.17099639201419234
     ROC_AUC scores: 0.9690993788819877
     <matplotlib.legend.Legend at 0x7f6425e69710>
```



✓ 0s completed at 20:49