Voting_classifier_teste

November 4, 2019

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from datetime import datetime
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import StratifiedKFold, GridSearchCV
from sklearn.ensemble import RandomForestClassifier,␣
 ↪GradientBoostingClassifier, AdaBoostClassifier, VotingClassifier,␣
 ↪BaggingClassifier
from sklearn.metrics import roc_auc_score, roc_curve, auc,␣
 ↪precision_recall_curve
from sklearn.metrics import classification_report, confusion_matrix
from xgboost import XGBClassifier
from mlxtend.plotting import plot_learning_curves
from yellowbrick.model_selection import LearningCurve
import matplotlib.gridspec as gridspec
import itertools
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn import tree
from sklearn.naive_bayes import MultinomialNB
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder
```

[2]:
```
def timer(start_time=None):
    if not start_time:
        start_time = datetime.now()
        return start_time
    elif start_time:
        tmin, tsec = divmod((datetime.now() - start_time).total_seconds(), 60)
        print('\n Tempo Necessário: %i minutos and %s segundos.' % (tmin,␣
 ↪round(tsec, 2)))
```

[3]:
```
train = pd.read_csv('trainRF.csv', dtype={'id': np.int32, 'target': np.int8})
X_train = train.iloc[:,1:68]
```

```
Y_train = train.loc[:, train.columns == 'Y']
test = pd.read_csv('testRF.csv', dtype={'id': np.int32})
X_test = test.iloc[:,1:68]
Y_test = test.loc[:, test.columns == 'Y']
#all_features = [x for x in train.drop(['ID', 'Y'], axis=1).columns]
```

[4]:
```
print(X_train.shape)
```

```
(109992, 67)
```

[5]:
```
print(X_test.shape)
```

```
(65995, 67)
```

[6]:
```
X_train.head()
```

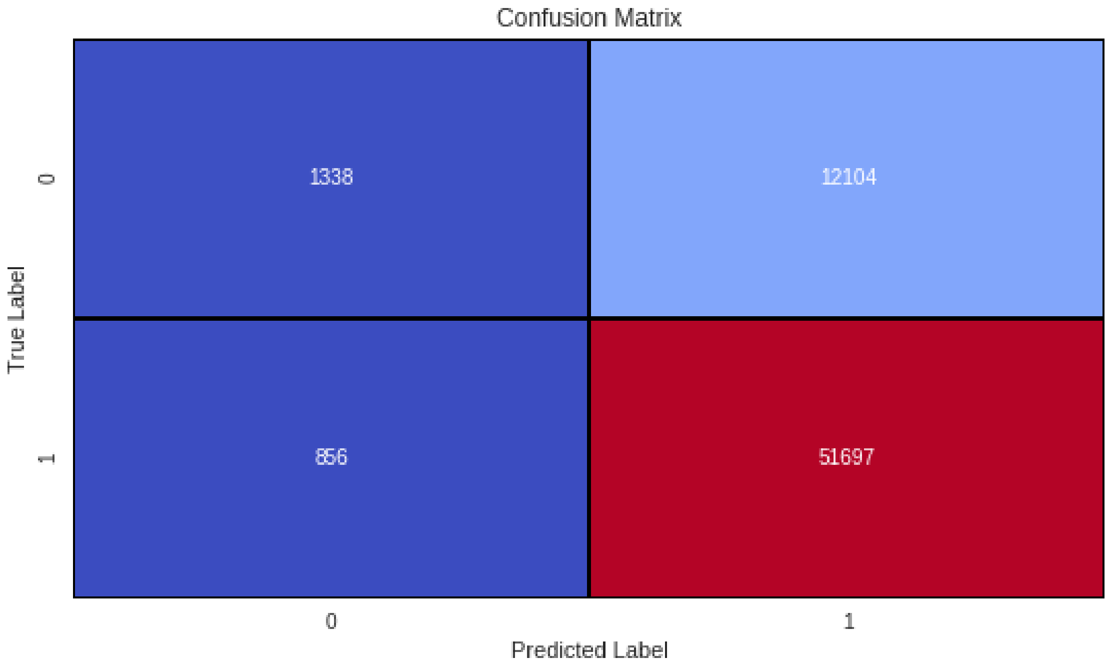
[6]:
```
   v1  v3  v6  v7  v8  v9  v10  v16  v17  v18  …    v687   v689  v690  v691  \
0   0   0   0   0   0   0    0    0    0    0  …   -1.00  -1.00   0.0   0.0
1   0   0   0   0   0   0    0    0    0    0  …  100.00  -1.00   0.0   0.0
2   0   0   0   0   0   0    0    0    0    0  …   99.90  -1.00   0.0   0.0
3   0   0   0   0   0   0    0    0    0    0  …   -1.00  -1.00   1.0   1.0
4   0   0   0   0   0   0    0    0    0    0  …   54.83  21.66   0.0   0.0

   v694  v56  v64     v72     v74     v78
0     0  0.0  0.0     0.0     0.0     0.0
1    15  0.0  0.0     0.0     0.0     0.0
2    15  0.0  0.0     0.0     0.0     0.0
3     0  0.0  0.0  4330.0  2260.0  1800.0
4     1  0.0  0.0     0.0     0.0     0.0

[5 rows x 67 columns]
```

Voting Ensemble

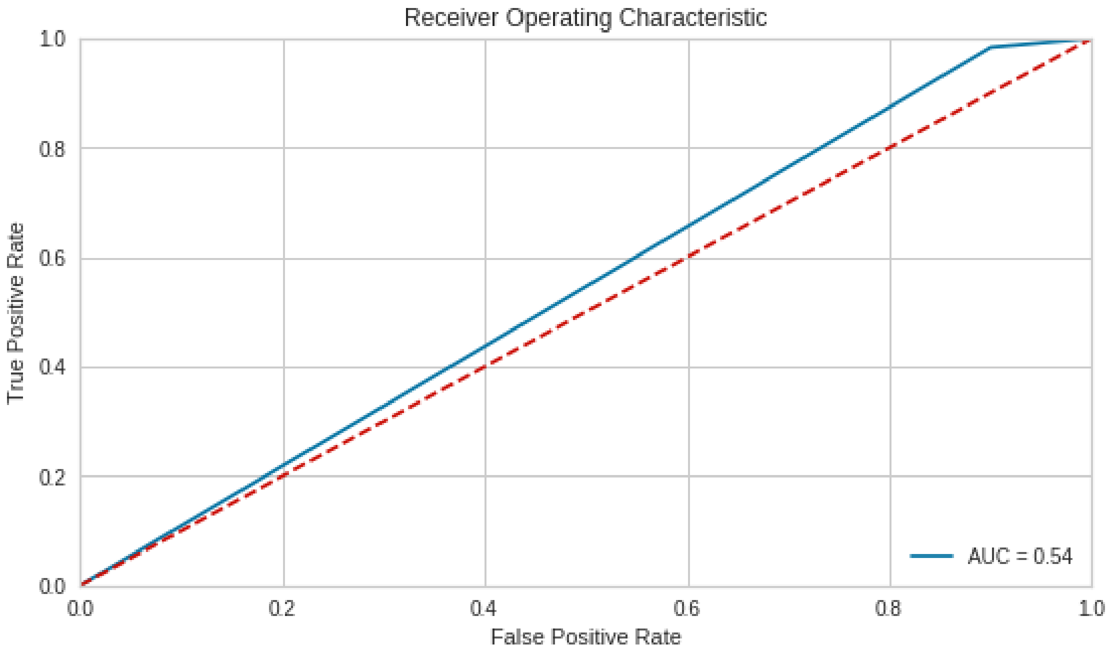
[7]:
```
from sklearn.ensemble import VotingClassifier
from sklearn.svm import SVC
starttime = timer(None)
start_time = timer(None)
model1 = RandomForestClassifier(n_estimators=200, max_depth=2,␣
 ↪criterion='gini', n_jobs=4, random_state=1)
model2 = DecisionTreeClassifier(class_weight=None, criterion='entropy',␣
 ↪max_depth=None, max_features=None, max_leaf_nodes=None,)
model3 = AdaBoostClassifier(n_estimators=200, random_state=1, learning_rate=1)
model4 = XGBClassifier(base_score=0.2, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=0.1, colsample_bytree=1, gamma=1,
              learning_rate=0.1, max_delta_step=1, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=300, n_jobs=4,
              nthread=None, objective='reg:squarederror', random_state=1,
              reg_alpha=0, reg_lambda=10, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
model = VotingClassifier(estimators=[('bg', model1), ('dtc', model2), ('ada',␣
 ↪model3), ('xgb', model4)], voting='hard')
model.fit(X_train,Y_train.values.ravel())
model.score(X_test,Y_test.values.ravel())
predictions_model = model.predict(X_test)
scores = cross_val_score(model, X_train, Y_train.values.ravel(), cv=3)
timer(start_time)

false_positive_rate, true_positive_rate, thresholds = roc_curve(Y_test,␣
 ↪predictions_model)
roc_auc = auc(false_positive_rate, true_positive_rate)
matrix_model = confusion_matrix(Y_test, predictions_model)



plt.figure(1)
plt.figure(figsize=(9,5))
model_heatmap = sns.heatmap(matrix_model,annot=True, cbar=False, fmt="d", cmap␣
 ↪='coolwarm', linecolor ='black', linewidths = 1)
bottom, top = model_heatmap.get_ylim()
model_heatmap.set_ylim(bottom + 0.5, top - 0.5)
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
plt.show()

plt.figure(2)
plt.figure(figsize=(9,5))
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',
label='AUC = %0.2f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.0,1.0])
plt.ylim([-0.0,1.0])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()

plt.figure(3)
precision, recall, thresholds = precision_recall_curve(Y_test,␣
 ↪predictions_model)
plt.figure(figsize = (9,5))
plt.plot(recall, precision)
plt.plot([0, 1], [0.5, 0.5], linestyle = '--')
plt.xlabel('Recall', fontsize = 16)
plt.ylabel('Precision', fontsize = 16)
plt.xticks(size = 18)
plt.yticks(size = 18)
plt.title('Precision-Recall', fontsize = 28)
plt.show();


print("Classification Report")
print(classification_report(Y_test, predictions_model))

print("Acurácia do Modelo Cross Validation")
print(scores.mean())
```

```
Tempo Necessário: 2 minutos and 26.33 segundos.
```

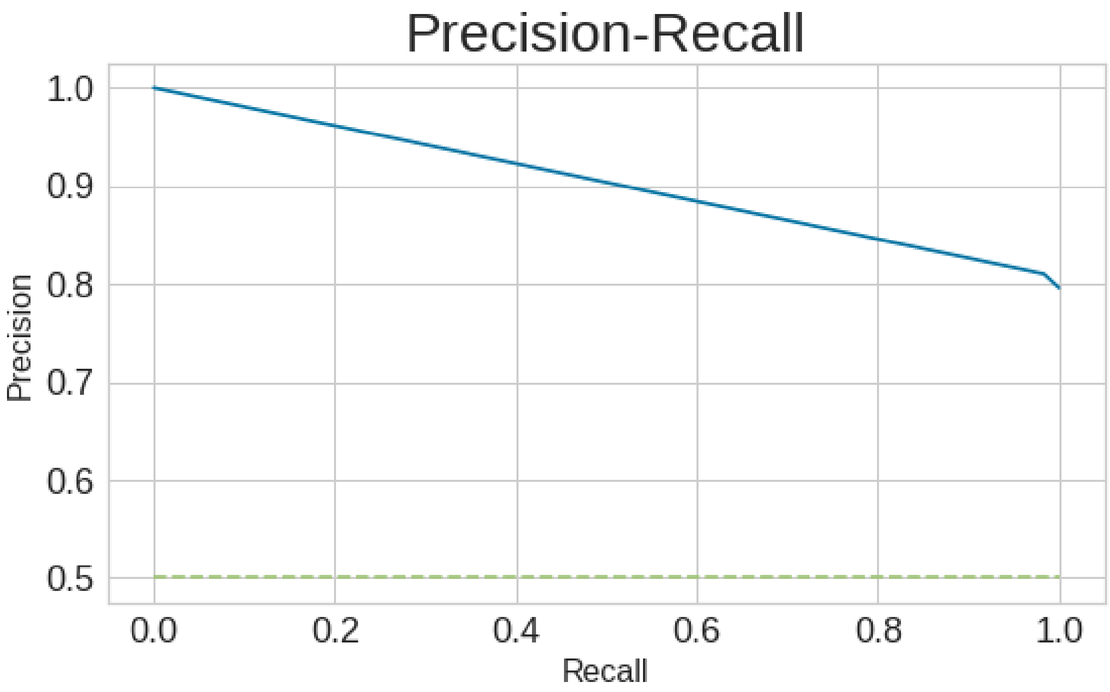
```
<Figure size 432x288 with 0 Axes>
```



```
<Figure size 432x288 with 0 Axes>
```



```
<Figure size 432x288 with 0 Axes>
```



```
Classification Report
```

```
              precision    recall  f1-score   support

           0       0.61      0.10      0.17     13442
           1       0.81      0.98      0.89     52553

    accuracy                           0.80     65995
   macro avg       0.71      0.54      0.53     65995
weighted avg       0.77      0.80      0.74     65995

Acurácia do Modelo Cross Validation
0.8005309322324199
```

[ ]: