Model_comparison

November 8, 2021

0.0.1 Comparing the performance of multiple ML models

[ ]:
```
import pandas as pd
import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn import preprocessing
from imblearn.combine import SMOTETomek

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
# from sklearn.neural_network import MLPClassifier
# from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,␣
↪recall_score, roc_curve, roc_auc_score, f1_score
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/externals/six.py:31:
FutureWarning: The module is deprecated in version 0.21 and will be removed in
version 0.23 since we've dropped support for Python 2.7. Please rely on the
official version of six (https://pypi.org/project/six/).
  "(https://pypi.org/project/six/).", FutureWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:144:
FutureWarning: The sklearn.neighbors.base module is  deprecated in version 0.22
and will be removed in version 0.24. The corresponding classes / functions
should instead be imported from sklearn.neighbors. Anything that cannot be
imported from sklearn.neighbors is now part of the private API.
  warnings.warn(message, FutureWarning)
```

[ ]:
```
#Import Drive API and authenticate
from google.colab import drive
#Mount Drive to the Colab VM
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
#Load the dataset into pandas DataFrame
df = pd.read_csv("/content/drive/MyDrive/Capstone_project/v2_credit_default.
↪csv")
```

[ ]:
```
# #  Selected features Based on Pearson and Spearman's rank correlations with␣
↪the dependent variable, and XGBoost feature importance rankings:
# df = df[['AGE', 'LIMIT_BAL', 'Pay_Apr', 'Repay_Sept', 'Pay_Sept','Default' ]]
```

[ ]:
```
df.shape
```

[ ]:
```
(29965, 24)
```

[ ]:
```
#Seperate the independent and dependent variables.
df_independent = df.drop(['Default'], axis=1)
df_default = df['Default']
```

[ ]:
```
# split the data into 75% training+validation and 25% test
X_train, X_test, y_train, y_test = train_test_split(df_independent, df_default,␣
↪test_size=0.25, random_state=1)
```

[ ]:
```
# Scale X_train and X_test
X_train_scaled = preprocessing.MinMaxScaler().fit_transform(X_train)
X_test_scaled = preprocessing.MinMaxScaler().fit_transform(X_test)
```

[ ]:
```
# Balance the training data using SMOTE Tomek
X_smt, y_smt = SMOTETomek(random_state=1).fit_sample(X_train_scaled, y_train.
↪squeeze())
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarning: Function safe_indexing is deprecated; safe_indexing is deprecated
in version 0.22 and will be removed in version 0.24.
  warnings.warn(msg, category=FutureWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarning: Function safe_indexing is deprecated; safe_indexing is deprecated
in version 0.22 and will be removed in version 0.24.
  warnings.warn(msg, category=FutureWarning)
```

[ ]:
```
#List the classifiers to be compared
names = ["Logistic_Regression", "Decision_Tree","Nearest_Neighbours", "QDA",␣
↪"Naive_Bayes", "Random_Forest", "AdaBoost", "GradientBoost"]
```

```
classifiers = [
    LogisticRegression(random_state=1, C= 50, penalty= 'l1', solver=␣
 ↪'liblinear'),
    DecisionTreeClassifier(max_depth=5),                #set max_depth
    KNeighborsClassifier(),                             #n_neighbors=5
    QuadraticDiscriminantAnalysis(),
    GaussianNB(),
    RandomForestClassifier(max_depth=5, n_estimators=100),
    AdaBoostClassifier(n_estimators=100),
    GradientBoostingClassifier(n_estimators=100, learning_rate=1.0),
    ]
```

[ ]:
```
accuracies = []
precisions = []
recalls = []
f1_scores = []

for name, clf in zip(names, classifiers):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)
    precision = precision_score(y_test, y_pred)
    precisions.append(precision)
    recall = recall_score(y_test, y_pred)
    recalls.append(recall)
```

[ ]:
```
scores_df = pd.DataFrame()
scores_df['name'] = names
scores_df['accuracy'] = accuracies
scores_df['precision'] = precisions
scores_df['recall'] = recalls
scores_df
```

[ ]:

	name	accuracy	precision	recall
0	Logistic_Regression	0.813267	0.709845	0.250305
1	Decision_Tree	0.818340	0.659478	0.353837
2	Nearest_Neighbours	0.757742	0.385733	0.177832
3	QDA	0.618793	0.334424	0.746650
4	Naive_Bayes	0.391484	0.247533	0.870889
5	Random_Forest	0.813801	0.676177	0.288672
6	AdaBoost	0.815136	0.653892	0.332521
7	GradientBoost	0.804058	0.582543	0.373934