Winton Wong

Midterm

ISE 364

Introduction

The objective is to create a model with data from project_data.csv and be used to predict target value with new_obs.csv.

Summary

The data set are trained for a Supporting Vector Machine, without applying balancing techniques and with overbalancing and under-balancing techniques. The scores were then compared, and the data set as is produced the most accurate results among the set of data.

Experimental Details

- Load the project_data.csv as df
- Conduct Exploratory Data Analysis for project_data.csv
- Convert the V0 to binary data
- Apply Train Test Split to df
- Train and evaluate the model with Linear SVC
- Train and evaluate the model with K Neighbor Classifier
- Train a Support Vector Machine Classifier Model (SVM), in grid, and test it with test data
- Evaluate the model with confusion matrix and classification report
- Resample data for underbalanced case with imblearn.under_sampling
- Train a Support Vector Machine Classifier Model (SVM), in grid, and test it with test data
- Evaluate the model with confusion matrix and classification report
- Resample data for overbalanced case with from imblearn.over_sampling
- Train a Support Vector Machine Classifier Model (SVM), in grid, and test it with test data
- Evaluate the model with confusion matrix and classification report
- Train and evaluate the model with Random Forest
- Used the SVM model without balancing to predict the targets from new_obs.csv

Results and discussions

The linear SVC and the K Neighbor models was first completed, they have a weighted average of 0.75, and 0.76, it felt they could be improved. The weighted average for Supporting Vector Machine without balancing, with under-balancing, and with overbalancing, are 0.86, 0.79, and 0.78, and it is 0.86 for Random Forest Model with 500 estimators. They are produced a moderately good result. Therefore, Supporting Vector Machine without balancing seems to be the best candidate to be used to predict data from new_obs.csv.

Supporting Vector Machine without balancing

```
{'C': 10, 'gamma': 1, 'kernel': 'rbf'}
[[  78  255]
 [   6 1311]]
              precision    recall  f1-score   support

           0       0.93      0.23      0.37       333
           1       0.84      1.00      0.91      1317

    accuracy                           0.84      1650
   macro avg       0.88      0.61      0.64      1650
weighted avg       0.86      0.84      0.80      1650
```

Supporting Vector Machine with under-balancing

```
{'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
[[218 115]
 [373 944]]
              precision    recall  f1-score   support

           0       0.37      0.65      0.47       333
           1       0.89      0.72      0.79      1317

    accuracy                           0.70      1650
   macro avg       0.63      0.69      0.63      1650
weighted avg       0.79      0.70      0.73      1650
```

Supporting Vector Machine with overbalancing

```
{'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}
[[ 154  179]
 [ 199 1118]]
              precision    recall  f1-score   support

           0       0.44      0.46      0.45       333
           1       0.86      0.85      0.86      1317

    accuracy                           0.77      1650
   macro avg       0.65      0.66      0.65      1650
weighted avg       0.78      0.77      0.77      1650
```

K Neighbor Classifier

```
[[  83  250]
 [ 100 1217]]
              precision    recall  f1-score   support

           0       0.45      0.25      0.32       333
           1       0.83      0.92      0.87      1317

    accuracy                           0.79      1650
   macro avg       0.64      0.59      0.60      1650
weighted avg       0.75      0.79      0.76      1650
```

Linear SVC

```
[[ 296   37]
 [1011  306]]
              precision    recall  f1-score   support

           0       0.23      0.89      0.36       333
           1       0.89      0.23      0.37      1317

    accuracy                           0.36      1650
   macro avg       0.56      0.56      0.36      1650
weighted avg       0.76      0.36      0.37      1650
```

Random Forest with 500 estimators

```
[[ 139  194]
 [  37 1280]]
              precision    recall  f1-score   support

           0       0.79      0.42      0.55       333
           1       0.87      0.97      0.92      1317

    accuracy                           0.86      1650
   macro avg       0.83      0.69      0.73      1650
weighted avg       0.85      0.86      0.84      1650
```

Body

After the data was loaded into data frames, Exploratory Data Analysis was conducted to attempt to discover trends, to check for nulls, data types, etc. The data has 12 different type of characteristics, and the target was binary of 1s and 0s. However, one of the features contains unique values of As and Bs, and it had to be converted to was binary of 1s and 0s using dummies variables.

The data set was then split with 30% for testing after training the model.

Model of choice is Supporting Vector Machine, after consulting the scikit-learn cheat sheet (1). This set of data contains more than 50 samples, we are predicting a category of 1 or 0, the data are labeled. The information gives a path that first led to Linear SVC, the K Neighbor Classifier, and then SVC. As shown in the results, both Linear SVC and K Neighbor Classifier did not perform as well as SVC. (Both models can be retained using linearSVC.py and k_neighbors.py)

Following the data clean up and setting up for training. Then the grid was paired with the Supporting Vector Machine to find the best combinations the parameter C and gamma that would produce the best result. C and gamma that produced the best results are 10, and 1.

It was discovered that the ratio of target 1s are 4 times more than 0s in target. Although the model was retrained with under balancing and overbalancing in mind afterwards, it appears such data engineering methods did not produce a better result pre outlined in the results sections.

In an exploratory attempt, the data was then again trained using random forest models with the number estimators of 100, 250, 500, 100. Results are not shown but test can be easily ran using the random_forest.py script in the zip package. They all produced comparable level of results to the Supporting Vector Machine. At the concern of overfitting it was not chosen, nor was it on the path in the cheat sheet.

Conclusions

Following the advices of the cheat sheet and the comparing of results of all the models conducted, it is determined that a Supporting Vector Machine with parameter C of 10 and gamma of 1 is the best bet in getting a higher score on this midterm or provide the best likelihood of more accurate results on the new_obs data.

Recommendations

It would be nice if there is a description of the different features, then each feature can be deeply analyzed and to be used to the advantage.

Files

random_forest.py

- A python script for a random forest model

Midterm_ISE364_WintonWong.ipynb

- An ipython notebook where the SVM models are contained, as well as my thoughts and logic

linearSVC.py

- A python script for the linear SVC model

k_neighbors.py

- A python script for the k neighbors classifier model

finalized_model.sav

- This is the saved model of SVM without any balancing

Reference

1. https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html