CMPT459 Data Mining

Milestone2

By

Decision Tree - Ziyi An 301371687

Logistic Regression - Chen Zhao 301308092

SVM - Duo Lu 301368672

Github: https://github.com/Duo-Lu/CMPT459

March 6, 2020

1 Introduction

This is the report for the data mining project, milestone two, which is majorly about training classifiers, tuning parameters, and examine performance in various metrics. The report is question-wise constructed, while results from each required classifier are provided by individual group members within most of the questions, which also means questions that have similar conclusions of all members (ie. question 3) would only be answered once. Codes of all the work are uploaded to the Git repository and were run in Jupyter Notebook Python environment.

2 Feature Selection

Regarding the bonus question, external data of locations of NYC subway stations were used to compute a feature of the distance from each rent to its nearest station.

For Decision Tree:

It is rather dynamic of choosing features due to certain impacts it has upon the performance, and a few more features were computed from the original data, such as the ratio of price over the number of baths and bedrooms, as well as from external data mentioned above. Initially, a total of thirteen features were used to train the decision tree model, and scalars such as MinMaxScaler were included in the training pipeline. Later on, as I intended to filter out some meaningless features, SelectKBest function was used to choose features according to ANOVA F-values. In addition, RandomForest classifier is also capable of providing important scores as shown.

For Logistic Regression:

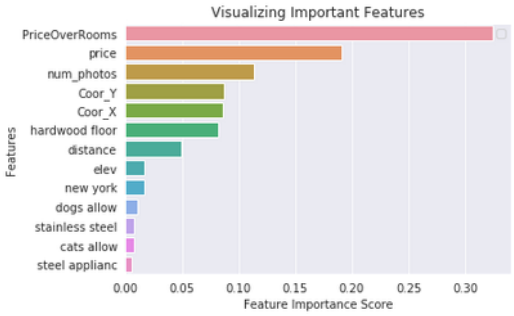


Figure 1: Feature importance score for Decision Tree

Thirty features chosen and used to train the logistic regression model include the provided features such as 'bedrooms', 'bathrooms' and computed features such as 'distance' to the nearest subway location and average price over rooms. Also, by calculating the term frequency and b-gram for 'features' and 'description' respectively after stemmer and stop words elimination, we got 22 categorical features such as 'stainless steel' as part of my training dataset. After training and testing the score, some features are not relative to boost the score, which are removed by using SelectKBest K features sklearn package according to ANOVA F-values.

For SVM:

Price over rooms and all the basic features we got from milestone one, plus distance to its nearest station are being used for training the SVM model. Moreover, we use DBSCAN to find roughly 24 clusters(most dense areas) and calculate the minimum distance to the nearest cluster. However, it tends not to improve the model. Apparently, SelectKBest function is not suitable for SVM to choose features because it needs several iterations to fit the model.

3 Python Libraries

For Decision Tree:

The main libraries responsible for this classifier are sklearn.tree and sklearn.ensemble (RandomForestClassifier). Additional classes that helped to build the training pipeline are SelectKBest from sklearn.feature_selection and scalers from sklearn.preprocessing. Some involved performance metrics are from sklearn.metrics (classification_report, roc_auc_score).

For Logistic Regression:

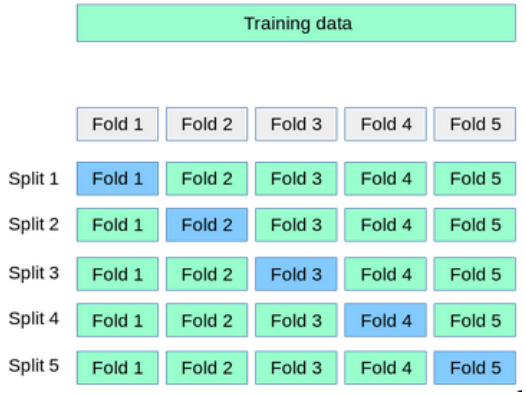
The main libraries responsible for this classifier are sklearn.linear_model.LogisticRegression. Additional classes that helped to build the training pipeline are SelectKBest from sklearn.feature_selection, scalers from sklearn.preprocessing and sklearn.pipeline. Some involved performance metrics are from sklearn.metrics (classification_report, roc_auc_score). To split the training data and achieve the cross validation, sklearn.model_selection.train_test_split, sklearn.model_selection.cross_val_score and sklearn.model_selection.KFold are imported.

For SVM:

Main library for classifiers is sklearn.svm. Majority libraries for build the model and performance validation are same as Decision Tree and Logistic Regression.

4 Cross-Validation

There are ways of performing this approach in Python, such as using StratifiedKFold then train test split, or using cross validate directly. The procedures are the same, which is to separate training data into k (ie. 5) folds and each time, use k-1 of them to train the model. The resulting model is then validated on the remaining fold, and the performance measure (ie. accuracy score) could be examined separately or together as the average.



5 Initial Performance

For Decision Tree:

With all of the features being used and mostly default settings of the classifier, the average accuracy score is 0.6560659464607859 in cross-validation. The classification report is generated as shown:

Then, the test data is processed to have the same features and is predicted using this (very naive) model, obtained a log-loss score of 11.37993.

The performance is, unsurprisingly, poor. One major cause is related to question 6, overfitting, as, by default, the decision tree

classification report:

	precision	recall	f1-score	support
high	0.25	0.27	0.26	820
low	0.79	0.80	0.79	8064
medium	0.37	0.34	0.36	2720
accuracy			0.66	11604
macro avg	0.47	0.47	0.47	11604
weighted avg	0.65	0.66	0.65	11604

Figure 3:

tends to gain a 100% training accuracy. Therefore, without any pruning, the training score overwhelmed the validation score which will be illustrated later. Another issue might be the imbalanced data distribution as high-interest records may be overlooked. This is observed from the precision and recall rate (or F1 score) of low-interest records, which are much higher than others.

For Logistic Regression:

With all of the features being used and mostly default settings of the classifier, the average accuracy score is 0.6897064070544069 in cross-validation. The classification report is generated as shown below. Then, the test data is processed to have the same features and is predicted using this (very naive) model, obtained a log-loss score of 2.11326. The performance is not good as expected. One major cause is the different ranges among features and relevance of features.

	precision	recall	f1-score	support
high	0.30	0.01	0.02	650
low	0.73	0.96	0.83	6518
medium	0.42	0.13	0.20	2115
accuracy			0.71	9283
macro avg	0.48	0.37	0.35	9283
weighted avg	0.63	0.71	0.63	9283

Figure 4: First cross validation of total of 5

	precision	recall	f1-score	support
high	0.23	0.00	0.01	651
low	0.72	0.97	0.83	6517
medium	0.41	0.11	0.18	2115
accuracy			0.70	9283
macro avg	0.46	0.36	0.34	9283
weighted avg	0.62	0.70	0.62	9283

Figure 5: Second cross validation of total of 5

	precision	recall	f1-score	support
high	0.67	0.01	0.02	651
low	0.72	0.97	0.83	6517
medium	0.40	0.10	0.16	2115
accuracy			0.70	9283
macro avg	0.60	0.36	0.34	9283
weighted avg	0.65	0.70	0.62	9283

Figure 6: Third cross validation of total of 5

	precision	recall	f1-score	support
high	0.31	0.01	0.02	650
low	0.72	0.96	0.82	6517
medium	0.41	0.12	0.18	2115
accuracy			0.70	9282
macro avg	0.48	0.36	0.34	9282
weighted avg	0.62	0.70	0.62	9282

Figure 7: Fourth cross validation of total of 5

	precision	recall	f1-score	support
high	0.32	0.02	0.03	650
low	0.73	0.95	0.82	6517
medium	0.38	0.12	0.19	2115
accuracy			0.70	9282
macro avg	0.48	0.36	0.35	9282
weighted avg	0.62	0.70	0.62	9282

Figure 8: Fifth cross validation of total of 5

For SVM:

SVM is complex computationally(training all the dataset needs to take approximately 20 minutes). Therefore, we cannot use the whole training data to select features or tune parameters. Sampling 10% of the training data(with replacement) might be appropriate, but that will affect the accuracy. With linear kernel SVM and all the features, the average accuracy score is

0.6946291041751115 in cross-validation. Then, the test data is processed to have the same features and is predicted using this model, obtained a log-loss score of 1.60754. The main reason why the performance is not quite good is that we have a high dimensional dataset which the linear kernel cannot separate very well.

```
classification report:
                precision    recall  f1-score   support

           0       0.73      0.97      0.83      8655
           1       0.38      0.09      0.14      2725
           2       0.56      0.11      0.18       933

    accuracy                           0.71     12313
   macro avg       0.56      0.39      0.38     12313
weighted avg       0.64      0.71      0.63     12313
```

Figure 9:

6 Improvements

For Decision Tree:

Several modifications were applied. Firstly, features are selected manually (intuitively) and/or by build-in functions as also mentioned in question 1. Features such as price, price-over-rooms and (exist of) hardwood floor seem to be more contributing.

```
Best 3 features:
  price,   hardwood floor,   PriceOverRooms,
score is: 0.7027341093299742
```

Figure 10:

Secondly, scalers like min-max scaler were used to normalize the values of some numeric attributes. Other modifications are, for example, setting parameters to prune the tree, reducing overfitting; taking only partial data from low-interest records to balance the distribution; ensembling multiple decision trees (ie. 1000 trees) to see if variance and correlation could be reduced. The outcomes will be discussed in question 7.

For Logistic Regression:

- Change the range chosen for term frequency to get rid of too large or too small numbers of terms and b-gram
- Set the logistic regression solver to newton-cg
- Set the logistic regression multi-class to multinomial
- Use MinMaxscaler to standardize the numerical data
- Use 'SelectBest' to choose the most related k features

For SVM:

- Turning the parameters of the classifier, such as using Gaussian Radial Basis Kernel, regularization the parameter
- Summarize and combine some useful features like "prices", "bathrooms" and "bedrooms"
- Sample the training dataset to get representative dataset for training SVM model
- Use StandardScaler and MinMaxscaler to map numerical data into particular range

7 Overfitting

For Decision Tree:

The overfitting is obvious in early trials of decision tree classifier training. It could be observed through the difference between the training score and the validation score. For example, before modification, the scores are as such:

```
trainning vs. validation score:  0.9690597259329484 0.6550327473285074
```

Figure 11:

The most apparent way to avoid overfitting is to limit the depth of the decision tree and on the minimum number of records in leaf nodes, which results in a difference as such:

```
trainning vs. validation score:  0.7178315952770835 0.703464322647363
```

Figure 12:

For Logistic Regression:

In the logistic regression, overfitting can be witnessed when the score of train data is obviously larger than the score of test data. The largest gap of train data and test data is still smaller than 0.1 approximately, which should be not considered as overfitting. To prevent overfitting in logistic regression, we can change the bias, regularization or reduce features.

For SVM:

In general, the support vector machine is the high-bias and low-variance algorithm, even with Gaussian Radial Basis Kernel which increases the risk of overfitting(complex decision surfaces). However, the SVM we trained is quite good for avoiding overfitting. For example, the training and validation score are as such:

```
trainning vs. validation score:  0.6964290548772234 0.7077885162023877
```

Figure 13:

8 Improved Performance

For Decision Tree:

After applying most of the modifications mentioned in question 5, the decision tree classifier gained a cross-validation accuracy score of about 0.703 (varying in small range depending on slight adjustments of parameters), and the log-loss score on test data is 0.79592. In addition, the random forest classifier with 1000 estimators gains an accuracy score of 0.7126236296522273 and a log-loss score of 0.79588 on test data.

The most effective modification seems to be the pruning since overfitting is generally the major issue of decision tree classifier. The second-best approach is the selection of features, as the log-loss score varies from 0,9 to 0.7 by using less number of features, while the accuracy score does not change too much. Other approaches contribute a little to the result such as the ensemble classifier. The scaler helps to standardize numeric values to be better selected yet influence little to the final score. The partial sampling on low-interest records did very poorly in improving the result since the accuracy score dropped to about 0.59 despite the precision and recall were slightly higher than before.

For Logistic Regression:

After applying modifications mentioned in previous questions, the accuracy score increased to 0.71 and the log loss score decreased to 0.70332.

By changing the 'chi2' to 'f_classif' in SelectBest function and shrink the range of term frequency, I witness the most obvious change which increase the the accuracy score by around 0.2 and decrease the log loss score to 0.73 from 2.11, since the combination of these two methods increase the relevance of features which influence the most of the final score. Also, the rest methods such as logistic regression solver

and MinMax increase the accuracy score from the perspectives of changing the approach of weight calculation and scales respectively.

For SVM:

Unfortunately, only changing the kernel function to Gaussian Radial Basis can improve the accuracy score to 0.714232896 and the log loss score decreased to 0.78314. The more complex decision surface will be more useful in high dimensional dataset.

The larger values of parameter C, the smaller-margin hyperplane will train(avoid misclassification to improve accuracy score). But due to complex computationally for SVM, large C is not suitable for training the model. MinMaxscaler will decrease the running time indeed. However, it didn't help decrease the log-loss score. Also, features like "Price over rooms" and "distances" only slightly decrease the log-loss score. Moveover, sampling training data to build the model only increases the accuracy score.

9 Additional Metrics

For Decision Tree:

Other than the precision, recall, and F1-score, AUC of ROC were used to examine the classifier performance on the training data. The plot (after modification) is as such: The conclusion agrees with the classification report, which is that the classifier is more capable of separating low-interest records than the other two

Compared with the multiclass logarithmic loss metric, it is less sensitive to the modification I did, but it might imply that the interest level distribution in the test data set is similar (or even more skewed) compared with the training.

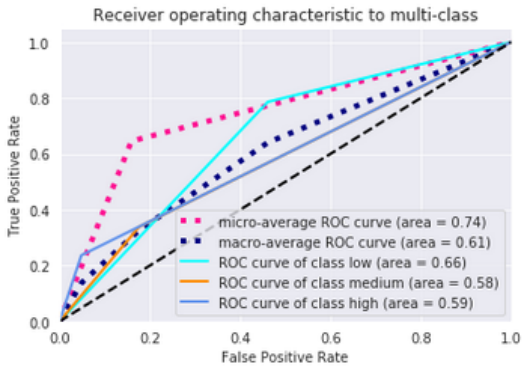


Figure 14: ROC curve for Decision Tree

For Logistic Regression:

I use a f1_score to evaluate the final score. The result is shown in the graph below. Compared with the multiclass logarithmic loss metric, it is quite similar to the actual score of log loss score. Thus, I believe my score is trustable.

```
f1_score(y_true, y_scores, average='micro')
0.7051016890727335
```

Figure 15:

For SVM:

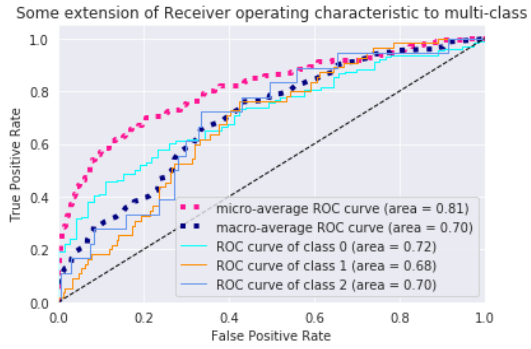


Figure 16:

Conclusions are the same as the Decision Tree above.

10 Additional Feature

For Decision Tree:

The additional feature ('distance') improves little of the performance. It is the low ranked according to either ANOVA F-values or the RandomForest classifier.

For Logistic Regression:

I define a new feature 'distance' to improve the accuracy.

0.702626366964721 for distance added

0.7023462874157692 for distance removed

Thus, I got a minor improvement after adding the 'distance' feature. If you do not train two different versions of your classifier, intuitively speaking, more convenient transportation will increase customers' interest.

For SVM:

Use DBSCAN to find roughly 24 clustering which are the most dense areas in NYC. Compute the distance to the most nearest cluster.