EE 559 Project

Yang Liu

liu578@usc.edu

May 1st

Classifying and Predicting Online News Popularity

Abstract

For advertisers and media companies, whether a news report has the potential to be popular or not is a big problem because putting advertisements on an online news page in advance is a good chance for promoting products. OnlineNewsPopularityReduced.csv from UCI Machine Learning Website is a data set of online news which we can study to predict popularity of news popularity based on features. The problem can be regarded as a binary classification problem if we set an appropriate threshold value of shares to label news as popular or not. I separate the original data set into training set and test set, do some preprocessing to them, and remove redundant features through PCA and Fisher scores methods. Then I use 3 different methods: k Nearest Neighbor, Artificial Neural Network/Multi-layer Perception and Support Vector Machine together with cross validation technology to train the classifiers and finally test on the test set. By using cross validation for parameter adjusting, I get the optimal model for the three classifiers. The accuracy of KNN can reach up to 63%, the accuracy of MLP can reach up to 65%, the accuracy of SVM with rbf kernel can reach up to 66% but it takes more time for computation than MLP. All of the above optimal models are based on Fisher Score parameter selection.

Reading data and observation

1. Reading Data

To read data from the csv file, I use numpy's ***genfromtxt*** function to read all numerical data and use python's inbuilt ***open*** function to read the names of features into a list of string.

2. Observing the Dataset

The ***url*** feature and ***timedelta*** feature are non- predictive, which means we can remove these two features. I strip the feature ***url*** and ***timedelta*** by slicing the data. The ***kw_min_min*** is useless as well because all values of this column are -1 and we cannot get any information in this feature. The data set has no missing value at all and all values of useful features are numerical.

Pre-processing

1. Labeling Dataset

Popularity relies on the last feature ***shares***. To make two classes ***popular*** and ***non-popular*** even balanced, I set the threshold to be 1400: 2357 data points with ***share*** value greater than 1400 are classified as ***class 1: popular***, while 2597 data points with ***share*** value less than or equal to 1400 are classified as ***class 2: non-popular***, 4954 data points totally.

2. Separating Training and Test Set

The original dataset should be randomly divided it into training set and test set. Training set is used for training classifier, adjusting parameters and cross validation while test set is used for final test only. I use ***train_test_split*** function from sklearn.model_selection to do the separation. 954 test points,19.257% from original set, and 4000 training samples.

In the later process, whatever normalization/ feature selection operation done to the training set, test set should do the same operation with same scaler. And test set should not be touched again until the final classifier is determined and need to evaluate the error rate on test set.

3. Traverse the Training Set and calculate Fisher Score

For future use, first traverse training set and calculate the number of class 1 points ***nClass1_of_train***, number of class 2 points ***nClass2_of_train***, class 1's mean vector ***u1***, variance vector ***s1***, and class 2 's mean vector ***u2,*** variance vector ***s2.*** For each feature, I calculate the Fisher score for it and store all Fisher score in list ***F_score*** for future feature selection.

4. Normalization

I normalize the training set by using the ***StandardScaler*** from sklearn.preprocessing, according to the mean and standard deviation of the training set. Then I normalize the test set by using the scaler based on training set.

Feature Selection

I use both PCA and Fisher Score method to reduce feature dimensions and compare the difference of their performance in different classifiers later.

1. ***PCA***

   For PCA feature selection, I import the package ***PCA*** from sklearn.decomposition and find the top 20 principal components. I use ***pca.fit_transform*** to fit the training data and project to ***train_data_pca***, and ***pca.transform*** to accordingly project the test data to ***test_data_pca***.

2. ***Fisher Score***

   Since ***F_score*** is already calculated during preprocessing, I bind ***F_score*** and corresponding feature's index together by using ***zip*** function, then sort the ***key*** (score) and ***value*** (index of feature) by score in descending order. The name of the feature is also addressable by address the names of features read at first step.

   I print out all the features with their indexes in the descending order of Fisher score as below:

   60 shares,27 kw_avg_avg,41 LDA_02,18 data_channel_is_world,12 num_keywords,38 is_weekend,42 LDA_03,
   9 num_imgs,25 kw_min_avg,49 rate_negative_words,26 kw_max_avg,37 weekday_is_sunday,24 kw_avg_max,
   36 weekday_is_saturday,17 data_channel_is_tech,45 global_sentiment_polarity,6 n_non_stop_unique_tokens,
   14 data_channel_is_entertainment,11 average_token_length,7 num_hrefs,47 global_rate_negative_words,
   5 n_non_stop_words,43 LDA_04,33 weekday_is_wednesday,16 data_channel_is_socmed,
   29 self_reference_max_shares,4 n_unique_tokens,30 self_reference_avg_sharess,57 title_sentiment_polarity,
   39 LDA_00,56 title_subjectivity,32 weekday_is_tuesday,59 abs_title_sentiment_polarity,40 LDA_01,
   21 kw_avg_min,2 n_tokens_title,46 global_rate_positive_words,54 min_negative_polarity,13 data_channel_is_lifestyle,
   20 kw_max_min,10 num_videos,28 self_reference_min_shares,8 num_self_hrefs,53 avg_negative_polarity,
   22 kw_min_max,3 n_tokens_content,15 data_channel_is_bus,58 abs_title_subjectivity,55 max_negative_polarity,
   23 kw_max_max,44 global_subjectivity,34 weekday_is_thursday,52 max_positive_polarity,35 weekday_is_friday,
   51 min_positive_polarity,48 rate_positive_words,50 avg_positive_polarity,31 weekday_is_monday,19 kw_min_min

   because shares is the features I use to determine the class labels at the beginning, I shouldn't use it any more during training period. After removing it out, I get the top 20 features and their indexes:

   27 kw_avg_avg,41 LDA_02,18 data_channel_is_world,12 num_keywords,38 is_weekend,42 LDA_03,
   9 num_imgs,25 kw_min_avg,49 rate_negative_words,26 kw_max_avg,,37 weekday_is_Sunday,24 kw_avg_max,
   36 weekday_is_Saturday,17 data_channel_is_tech,45 global_sentiment_polarity,6 n_non_stop_unique_tokens,
   14 data_channel_is_entertainment,11 average_token_length,7 num_hrefs,47 global_rate_negative_words

   indexes of features chosen by Fisher Score: ***[27, 41, 18, 12, 38, 42, 9, 25, 49, 26, 37, 24, 36, 17, 45, 6, 14, 11, 7, 47]***

   The final data set with reduced feature dimensions is ***train_data[:][:, indexes]*** while test set is ***test_data[:][:, indexes]***

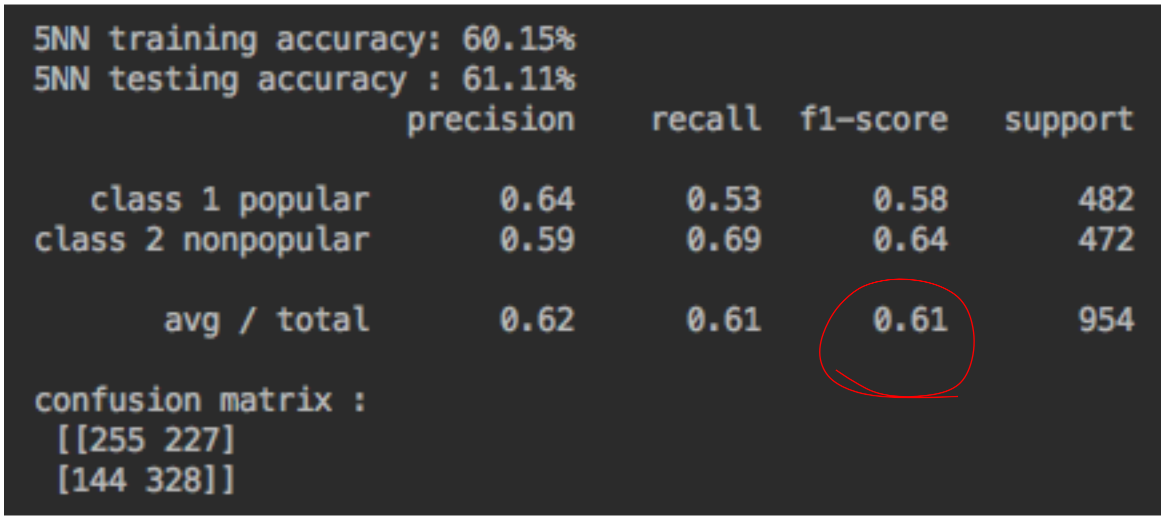
Classifiers

I use three different kinds of classifiers totally: k Nearest Neighbor (KNN), Artificial Neural Network/Multi-layer Perception (MLP) and Support Vector Machine (SVM) with cross validation to adjust the best parameters for performance.

(a) KNN

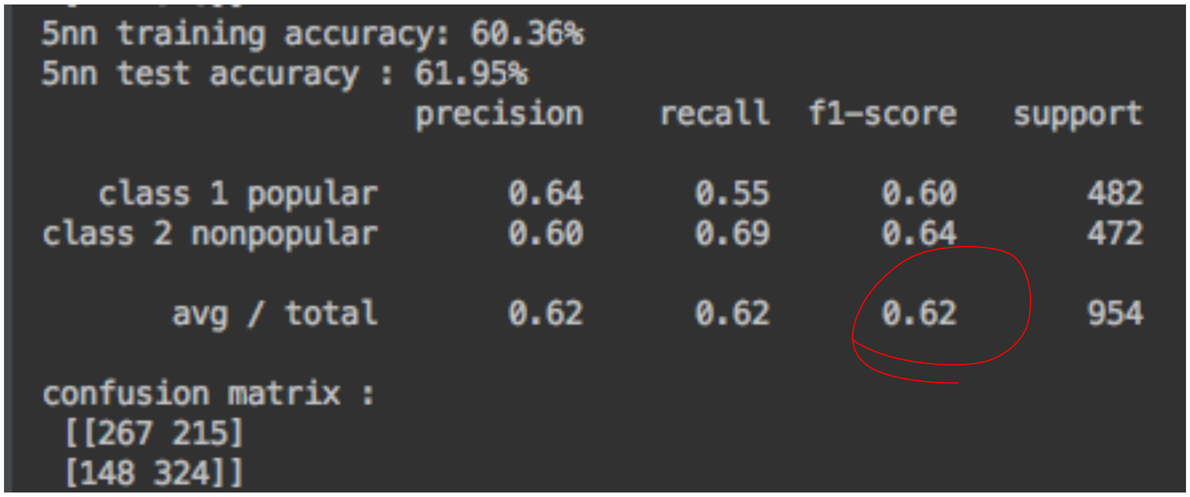
In order to have a basic knowledge of the performance and have a basic classifier to compare with while debugging, I use a 5NN classifier with no cross validation or parameter adjusting at first. But I use data set with 3 different kinds of feature dimensions. Normalization is accomplished before training.

- ***Full Feature*** I use the totally 58 useful features come with the training set and test set to do classification fitting and prediction. The classification function is ***knn_5()*** The result is as bellow:



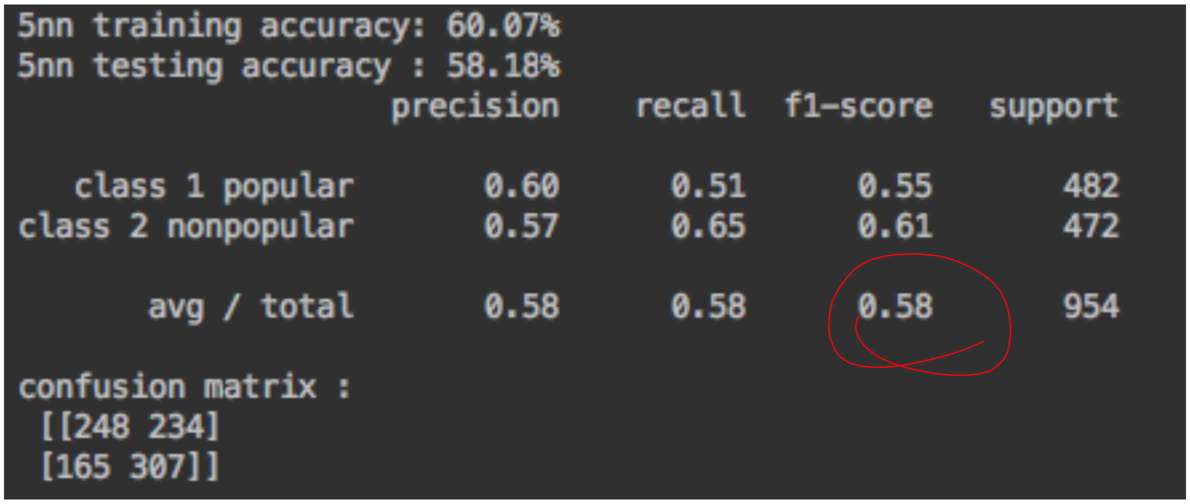
```
5NN training accuracy: 60.15%
5NN testing accuracy : 61.11%
                   precision    recall  f1-score   support

   class 1 popular       0.64      0.53      0.58       482
class 2 nonpopular       0.59      0.69      0.64       472

       avg / total       0.62      0.61      0.61       954

confusion matrix :
 [[255 227]
 [144 328]]
```

- ***PCA*** I only use the 20 features given by PCA for training and testing, the classification function is ***knn_5_pca().*** The result is as bellow:



```
5nn training accuracy: 60.36%
5nn test accuracy : 61.95%
                   precision    recall  f1-score   support

   class 1 popular       0.64      0.55      0.60       482
class 2 nonpopular       0.60      0.69      0.64       472

       avg / total       0.62      0.62      0.62       954

confusion matrix :
 [[267 215]
 [148 324]]
```

- ***Fisher score*** I only use the 20 features given by Fisher score for training and testing, the classification function is ***knn_5_fisher().*** The result is as bellow:



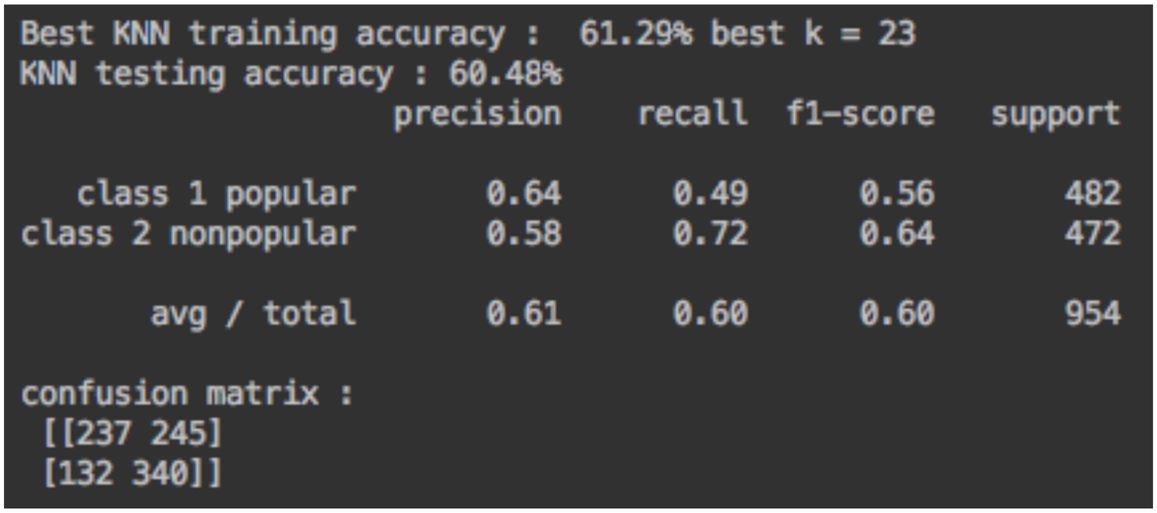
```
5nn training accuracy: 60.07%
5nn testing accuracy : 58.18%
                   precision    recall  f1-score   support

   class 1 popular       0.60      0.51      0.55       482
class 2 nonpopular       0.57      0.65      0.61       472

       avg / total       0.58      0.58      0.58       954

confusion matrix :
 [[248 234]
 [165 307]]
```

In general, performance is similar, the accuracy for 5NN classifiers is 60%. I will compare PCA and Fisher Score later.

Then I make the 5nn classifier more complex: make KNN classifiers with cross validation and parameter adjusting. In KNN classification, k is the only parameter need adjust and the choice of k mainly depends on experience. Since I don't have much experience on parameter adjusting, I use cross validation to find the best choice of k. KNN classifier relies on majority voting based on Euclidean distance so normalization is always required before training.

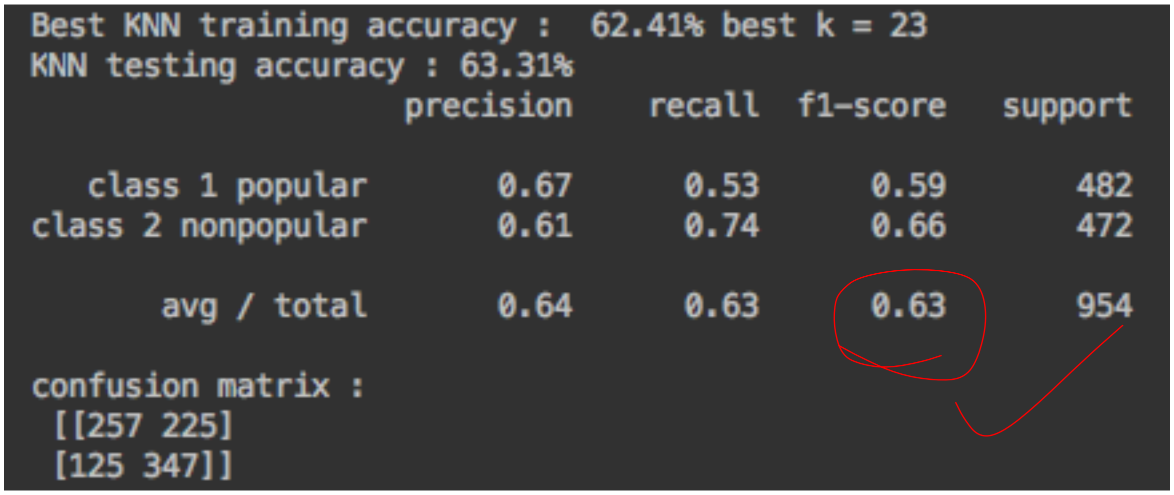
- ***Cross validation*** I introduce the Cross validation and use k as a parameter to optimize instead of assigning it 5 as before. By separating the training set into 5 subsets as 1 validation set and 4 training sets by using ***StratifiedKFold*** from sklearn.model_selection. Each time I do a model fitting on the 4 training subsets and test accuracy on the validation set. Since I use 5-fold cross validation, I run it 5 times and get the mean of the accuracy as the evaluation of training set. For each k, I calculate a mean accuracy and the final k should be the k with the greatest accuracy on training set.

- ***PCA*** I only use the 20 features given by PCA for training and testing, the classification function is ***knn_pca().*** The result is as bellow:



```
Best KNN training accuracy :  61.29% best k = 23
KNN testing accuracy : 60.48%
                  precision    recall  f1-score   support

  class 1 popular       0.64      0.49      0.56       482
class 2 nonpopular       0.58      0.72      0.64       472

       avg / total       0.61      0.60      0.60       954

confusion matrix :
 [[237 245]
 [132 340]]
```

- ***Fisher score*** I only use the 20 features given by Fisher score for training and testing, the classification function ***is knn_fisher()***. The result is as bellow:



```
Best KNN training accuracy :  62.41% best k = 23
KNN testing accuracy : 63.31%
                  precision    recall  f1-score   support

  class 1 popular       0.67      0.53      0.59       482
class 2 nonpopular       0.61      0.74      0.66       472

       avg / total       0.64      0.63      0.63       954

confusion matrix :
 [[257 225]
 [125 347]]
```

By comparing the performance of KNN classifier both on PCA and Fisher Score, I conclude that the performance on KNN classifier based on Fisher Score is a little bit better than PCA. The reason may be that the feature is well-designed and not much correlated. Fisher Score may perform better on other classifiers as well.
Interestingly, the best choice of k is always the unbound of interval I set for k, but the improvement on performance is negligible if k is big enough. So, my experience on this particular problem is that k = 11 is big enough considering the balance between performance and computation.
With KNN and Fisher Score, the training and testing accuracy could reach up 63% approximately.

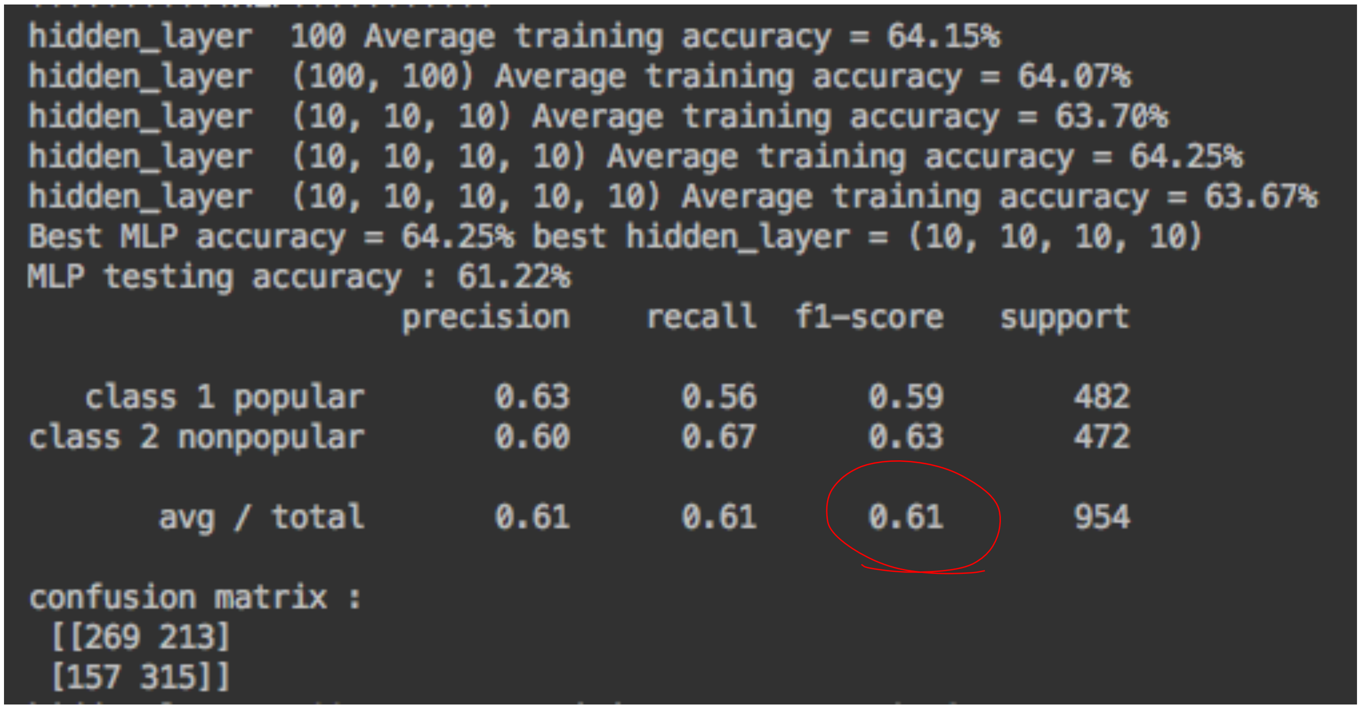
(b) Artificial Neural Networks/ Multi-layer Perception (MLP)

Artificial neural networks can be seen as multi-layer perception as well and it is realized in sklearn's packet neural_network.MLPClassifier. There are some parameters that need to adjust: solver and activation and hidden layers. Singe the dataset is not a big data set, according to scikit's manual, we should use 'lbfgs' for parameter solver.

For parameter activation, there are for choices: 'identity', 'logistic', 'tang', 'relu'. By trying all of them with 'lbfgs', 'identity' gives the best performance considering the accuracy and computation time. Then parameter hidden_layer_sizes is the main parameter that I adjust by cross validation.

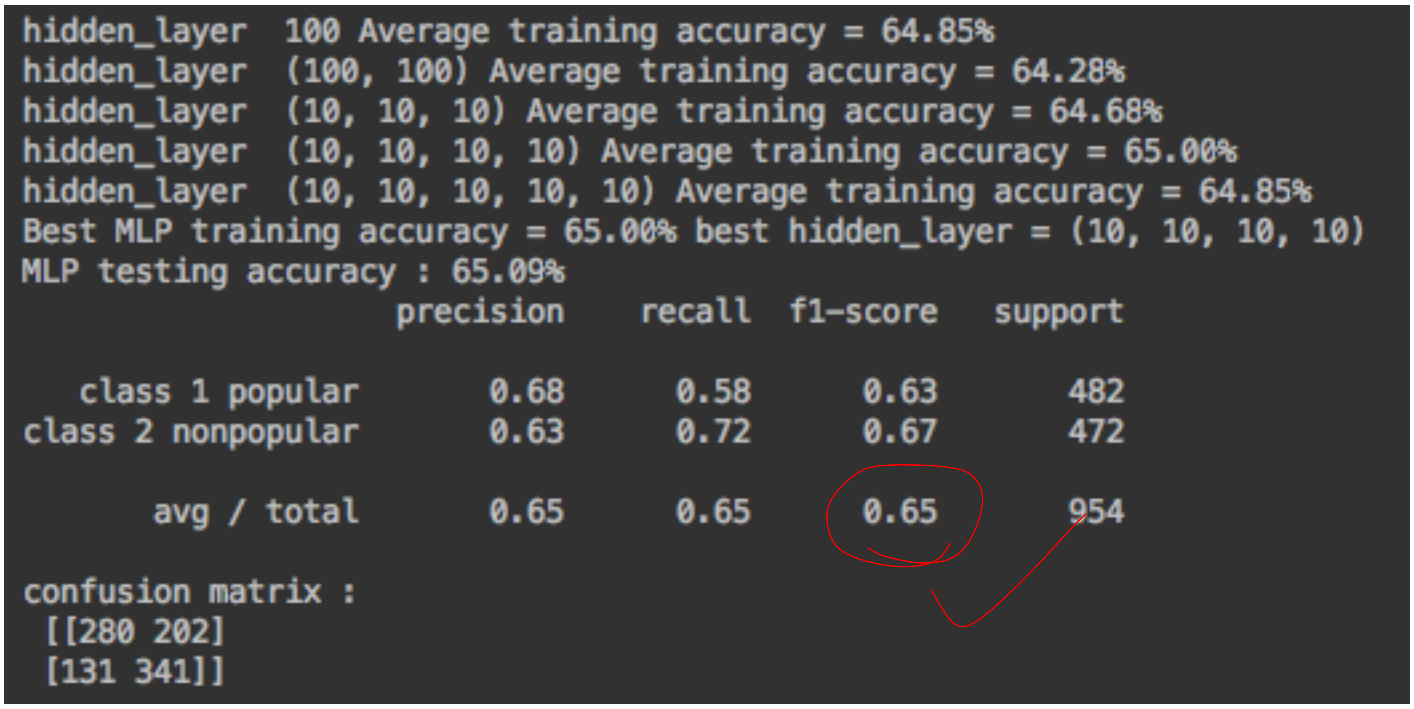
I set up a list of 5 different kinds of hidden layers: (100), (100,100), (10, 10, 10), (10, 10, 10, 10), (10, 10, 10, 10, 10) from which cross validation can choose from.

- ***PCA*** I only use the 20 features given by PCA for training and testing, the classification function is ***knn_pca().*** The result is as bellow:



```
hidden_layer  100 Average training accuracy = 64.15%
hidden_layer  (100, 100) Average training accuracy = 64.07%
hidden_layer  (10, 10, 10) Average training accuracy = 63.70%
hidden_layer  (10, 10, 10, 10) Average training accuracy = 64.25%
hidden_layer  (10, 10, 10, 10, 10) Average training accuracy = 63.67%
Best MLP accuracy = 64.25% best hidden_layer = (10, 10, 10, 10)
MLP testing accuracy : 61.22%
                    precision    recall  f1-score   support

   class 1 popular       0.63      0.56      0.59       482
class 2 nonpopular       0.60      0.67      0.63       472

       avg / total       0.61      0.61      0.61       954

confusion matrix :
 [[269 213]
 [157 315]]
```

- ***Fisher score*** I only use the 20 features given by Fisher score for training and testing, the classification function ***is knn_fisher()***. The result is as bellow:



```
hidden_layer  100 Average training accuracy = 64.85%
hidden_layer  (100, 100) Average training accuracy = 64.28%
hidden_layer  (10, 10, 10) Average training accuracy = 64.68%
hidden_layer  (10, 10, 10, 10) Average training accuracy = 65.00%
hidden_layer  (10, 10, 10, 10, 10) Average training accuracy = 64.85%
Best MLP training accuracy = 65.00% best hidden_layer = (10, 10, 10, 10)
MLP testing accuracy : 65.09%
                    precision    recall  f1-score   support

   class 1 popular       0.68      0.58      0.63       482
class 2 nonpopular       0.63      0.72      0.67       472

       avg / total       0.65      0.65      0.65       954

confusion matrix :
 [[280 202]
 [131 341]]
```

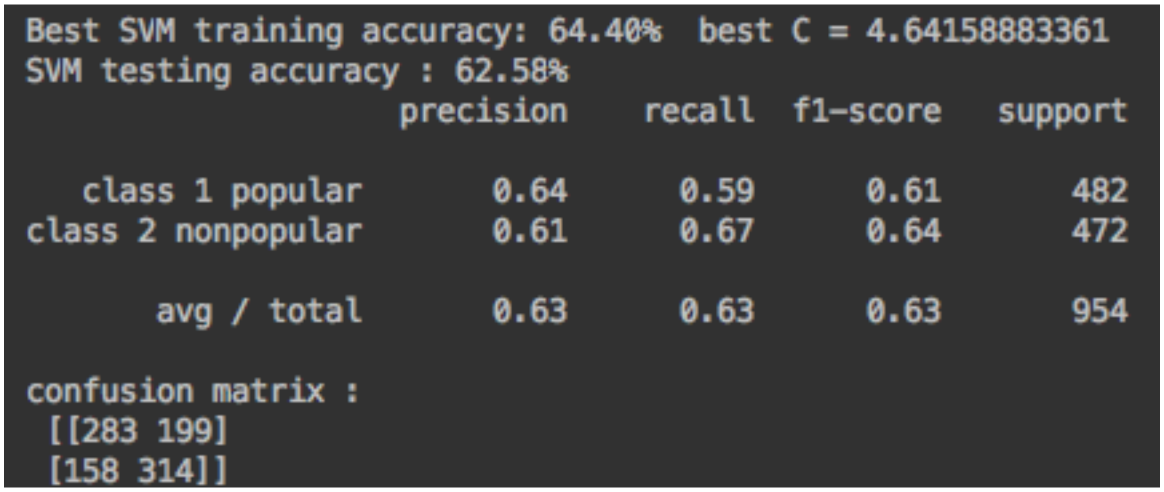
According to the result above, the performance of MLP with Fisher Score feature selection is better than PCA, which is the same form what I observe in KNN. The overall accuracy of MLP with Fisher Score reaches *up to 65% approximately, which is slightly better than KNN.*

(c) SVM

SVM is a non-statistical classifier and it is realized in sklearn's package SVM. I use two different SVMs and do parameter adjusting by cross validation. Since SVM can use lineal kernel function and rbf kernel function, I do both of them.For linear kernel function SVM, a single parameter C need to be optimized, for rbf kernel function , two parameter C and Gamma need to optimized.

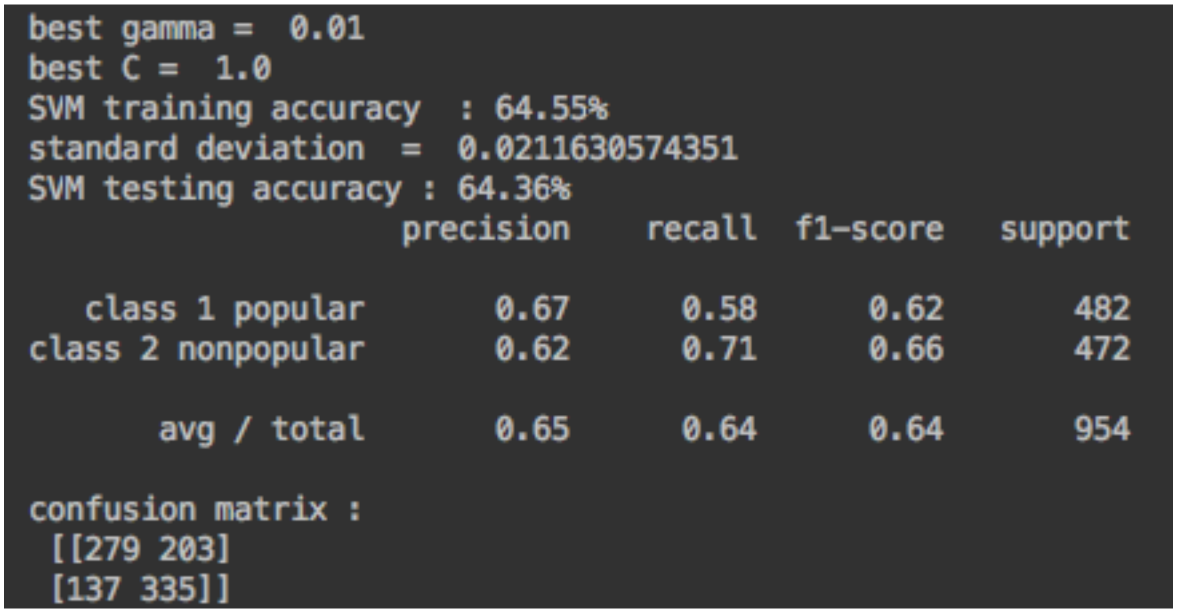
- ***PCA*** I only use the 20 features given by PCA for training and testing, the classification function is ***knn_pca().*** The result is as bellow:

  for linear kernal function SVM



```
Best SVM training accuracy: 64.40%  best C = 4.64158883361
SVM testing accuracy : 62.58%
                    precision    recall  f1-score   support

   class 1 popular       0.64      0.59      0.61       482
class 2 nonpopular       0.61      0.67      0.64       472

       avg / total       0.63      0.63      0.63       954

confusion matrix :
 [[283 199]
 [158 314]]
```

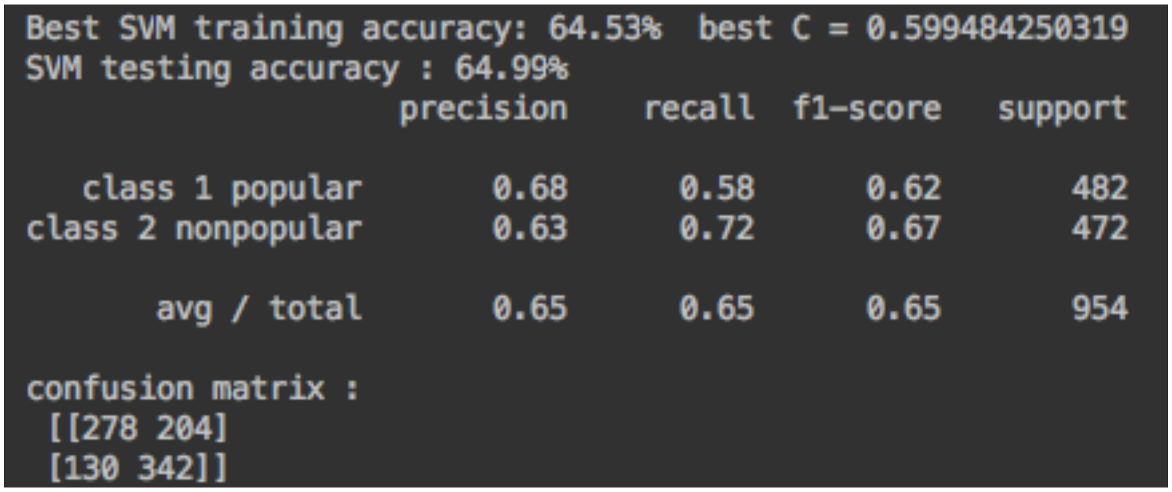
  for rbf kernal function SVM



```
best gamma =  0.01
best C =  1.0
SVM training accuracy  : 64.55%
standard deviation  =  0.0211630574351
SVM testing accuracy : 64.36%
                    precision    recall  f1-score   support

   class 1 popular       0.67      0.58      0.62       482
class 2 nonpopular       0.62      0.71      0.66       472

       avg / total       0.65      0.64      0.64       954

confusion matrix :
 [[279 203]
 [137 335]]
```

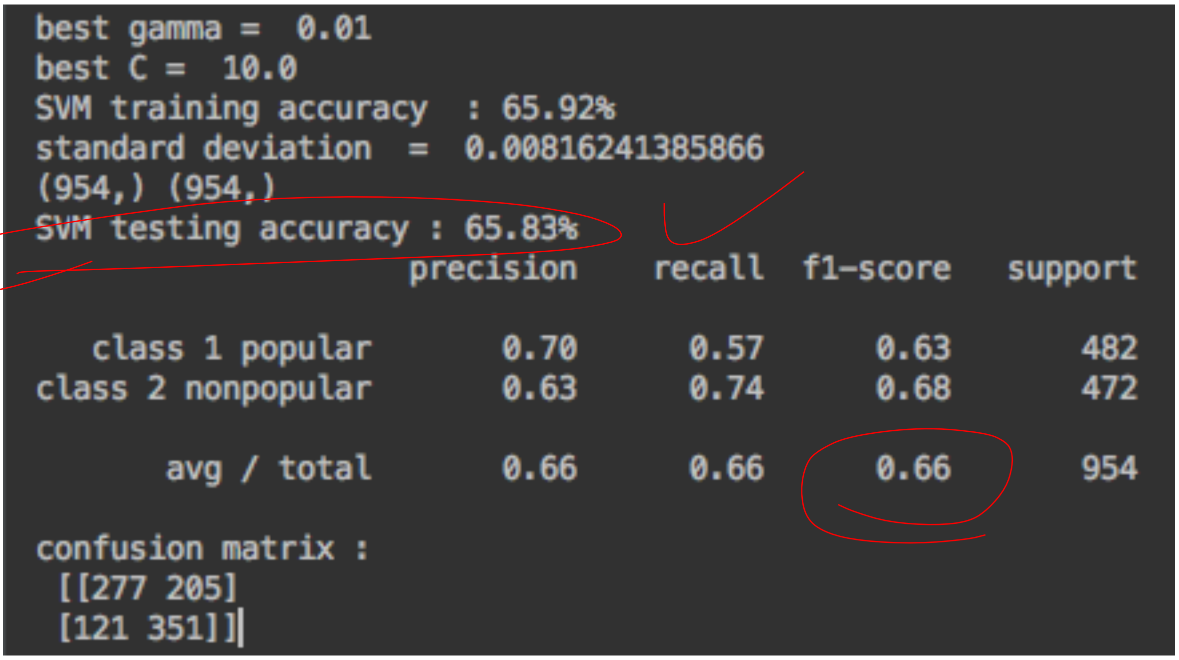
- ***Fisher score*** I only use the 20 features given by Fisher score for training and testing, the classification function ***is knn_fisher()***. The result is as bellow:

  for linear kernal function SVM



```
Best SVM training accuracy: 64.53%  best C = 0.599484250319
SVM testing accuracy : 64.99%
                    precision    recall  f1-score   support

    class 1 popular       0.68      0.58      0.62       482
class 2 nonpopular       0.63      0.72      0.67       472

       avg / total       0.65      0.65      0.65       954

confusion matrix :
 [[278 204]
 [130 342]]
```

  for rbf kernal function SVM



```
best gamma =  0.01
best C =  10.0
SVM training accuracy  : 65.92%
standard deviation  =  0.00816241385866
(954,) (954,)
SVM testing accuracy : 65.83%
                    precision    recall  f1-score   support

    class 1 popular       0.70      0.57      0.63       482
class 2 nonpopular       0.63      0.74      0.68       472

       avg / total       0.66      0.66      0.66       954

confusion matrix :
 [[277 205]
 [121 351]]
```

As we can see from the previous result, the rbf_kernel SVM using Fisher gives the best performance. The accuracy can reach up to 66% with the parameter gamma = 0.01 and C = 10. But the computation time is much longer than MLP.

Interpretation/Conclusion

- Feature selection plays an important role in the pattern recognition, better feature choice can give higher accuracy rate and shorter computation time. In this particular problem, Fisher Score is a better choice for feature selection than PCA and it can produce a accuracy 1-2% increase based on the same classifier. The reason may be that the feature is well-designed and not much correlated.
- By taking a like at the rank of score, we can have a general understanding of the importance of features, in this particular problem features like ***kw_avg_avg, LDA_02, data_channel_is_world, kw_min_avg, rate_negative_words*** are the 5 main features that is related to popularity.
- Normalization is also an important process that we need to take into account. Usually whether the data need to be normalized depend on the learning algorithm. For this particular project, classifiers like KNN, MLP and SVM all require normalization.
- Cross validation can help us use the data set more efficiently, adnother advantage of it is that we can use cross validation to do find the optimal parameter for cliassfiers. In this project, I use cross validation everywhere to get optimal parameters like k in KNN, C and gamma in SVM, hidden layers' structure in MLP.
- Train data and test data separation should be done at the beginning so that test set will not affect the training process at all.
- The classifier with best accuracy for this project is the rbf kernel SVM with gamma = 0.01 and C = 10, the accuracy is 66%. While MLP model with hidden layers(10)(10)(10)(10) also gives a good result at 65%.

very good!
A.J.

References

- http://scikit-learn.org/dev/modules/generated/sklearn.neural_network.MLPClassifier.html
- https://www.researchgate.net/post/Is_it_necessary_to_normalize_data_before_performing_principle_component_analy sis
- http://scikit-learn.org/dev/modules/generated/sklearn.neural_network.MLPClassifier.html
- https://www.researchgate.net/post/How_can_we_find_the_optimum_K_in_K-Nearest_Neighbor
- http://scikit-learn.org/dev/modules/generated/sklearn.neural_network.MLPClassifier.html
- https://www.cnblogs.com/pinard/p/6243025.html
- https://archive.ics.uci.edu/ml/datasets/online+news+popularity#
- ***https://blog.csdn.net/u011311291/article/details/78743393***
- ***https://stats.stackexchange.com/questions/69157/why-do-we-need-to-normalize-data-before-principal-component-analysis-pca***
- ***https://blog.csdn.net/haiyu94/article/details/53001726***
- ***https://www.cnblogs.com/bonelee/p/7880867.html***