EE 559 Project

Online News Popularity Prediction

Name: Yao Zhu

USC ID: 9292642940

Email: yaozhu@usc.edu

May 1st 2018

Abstract

The project is expected to predict online news popularity based on its 60 features. Among the 60 features, there are both predictive features and non-predictive features, binary features and numeric features. In this report, several approaches are used. Statistical approaches such as linear regression, logistic regression, K Nearest Neighbors. Prediction models such as Perceptron, SVM, Naïve Bayes, Random Forest and Decision Tree are discussed. Accuracy and F1 score are used as model evaluation criterion. Results shows that random forest has the best result for this classification problem with accuracy of 68% and F1 score of 0.67.

I. Description of dataset

Original news dataset is a collection of 39644 news article information scraped from Mashable over a span of 2 years. In this project, I use the reduced dataset, which has only 4594 articles, which largely reduce the processing time.

There are totally 61 features in this dataset, 58 of them are predictive, 2 nonpredictive, 1 target. Here is the description of each feature:

0. url: URL of the article (nonpredictive)
1. timedelta: Days between the article publication and the dataset acquisition (nonpredictive)
2. n tokens title: Number of words in the title
3. n tokens content: Number of words in the content
4. n unique tokens: Rate of unique words in the content
5. n non stop words: Rate of nonstop words in the content
6. n non stop unique tokens: Rate of unique nonstop words in the content
7. num hrefs: Number of links
8. num self hrefs: Number of links to other articles published by Mashable
9. num imgs: Number of images
10. num videos: Number of videos
11. average token length: Average length of the words in the content
12. num keywords: Number of keywords in the metadata
13. data channel is lifestyle: Is data channel `Lifestyle'?
14. data channel is entertainment: Is data channel `Entertainment'?

15. data channel is bus: Is data channel `Business'?

16. data channel is socmed: Is data channel `Social Media'?

17. data channel is tech: Is data channel `Tech'?

18. data channel is world: Is data channel `World'?

…

1st column url and 2nd column timedelta has been omitted when loading data into programming environment since they cannot be considered as features.

The 60th column – shares is our target, which is used for labeling each data point.

II. Preprocessing

1. Missing data

In this dataset, attribute ‘n tokens content’ stands for number of words in the content, if this is 0, which means there is no content in the article. We consider these articles as missing data. Therefore, before importing dataset into python environment, I delete all the data points whose attribute 3 equals 0. There are totally 488 data points in dataset. After dealing with them, number of data points becomes 4466.

2. Label data

Attribute ‘Shares’ is used for labeling data. Since I have deleted some data points which are missing data. I choose 1400 as threshold for popular and unpopular. Because this threshold can give us balanced labels. Those data which has shares value >=1400 are labeled 1, otherwise 0. After labeling, popular has 2273 data points, while unpopular has 2193 data points.

3. Normalization

Since there are binary and numeric features in the dataset, normalization is not simply remove mean. In my project, I keep those binary features, such as ‘Is data channel ‘Lifestyle’?’ unchanged. Other numeric features are removed by its feature mean.

4. Split train set and test set

I set aside 10% of the data samples as test set. The rest 90% is train set for feature selection and train classifier.

sklearn. cross_validation. train_test_split

III. Feature selection

1. Try PCA for feature selection – not desirable

sklearn.decomposition.PCA

mpl_toolkits.mplot3d.Axes3D

At first, I used PCA for feature reduction. After projecting data points to the first three principle directions, data points are not separable as shown in fig.1.

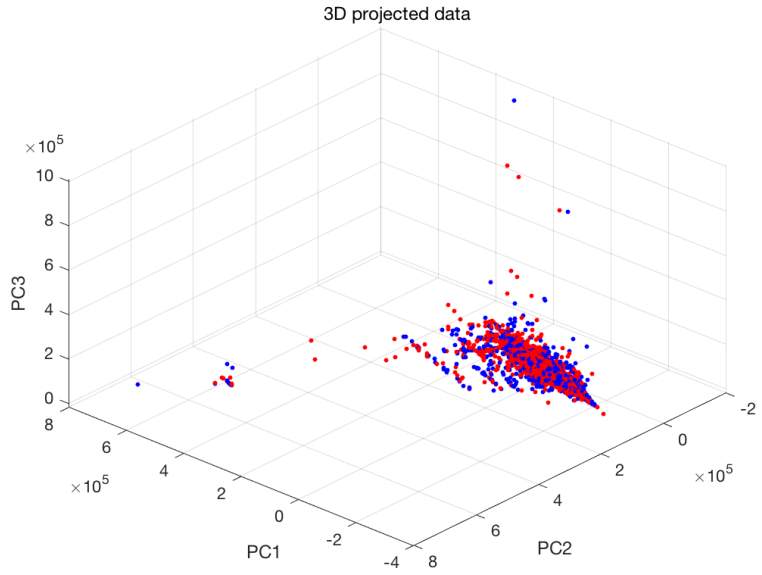

fig 1 3D PCA projected data

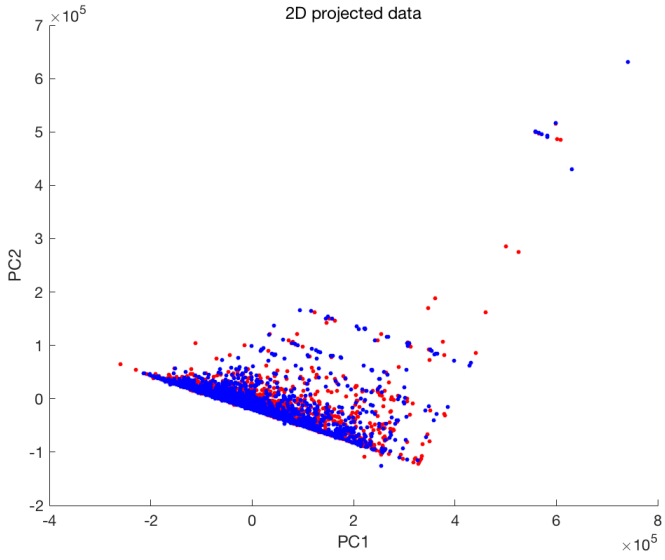

fig 2 2D PCA projected data

As seen in fig 1 and fig 2, PCA doesn't perform well on feature reduction on this dataset. This also implies that the classification result may not be very strong because features are not highly discriminate. This failure leads me to second try.

2. Try fisher score for feature selection – not desirable

Fisher score is an effective way for feature ranking. The larger fisher score is, the more likely this feature is more discriminative. The Fish score for jth feature is given by:

$$F(j) = \frac{(\bar{x}_j^1 - \bar{x}_j^2)^2}{(s_j^1)^2 + (s_j^2)^2}$$

where

$$(s_j^k)^2 = \sum_{x \in X^k} (x_j - \bar{x}_j^k)^2$$

In the equation above, the numerator represents the discrimination between popular and unpopular news, the denominator indicates the scatter within each class. However, this criterion alone is over simplified for feature selection, so I combine it with p values for feature ranking.

3. Use F test and p value together for feature selection

sklearn.feature_selection.SelectKBest, f_classif

P value is another commonly used criterion for feature ranking. A small p value indicates a strong evidence against the null hypothesis. In feature selection area, the smaller p value is, the less interaction between this feature and target. In a word, we rank features in ascending p-value and descending F-score.

In implementation, I choose Naïve Bayes classifier as the baseline because of its operating principle that the features are conditionally independent. I choose features according to its feature rank and feed the features into Naïve Bayes Classifier recursively. Train accuracy, test accuracy and F1 score are three metrics to evaluate how discriminate and how well performed the chosen features are.

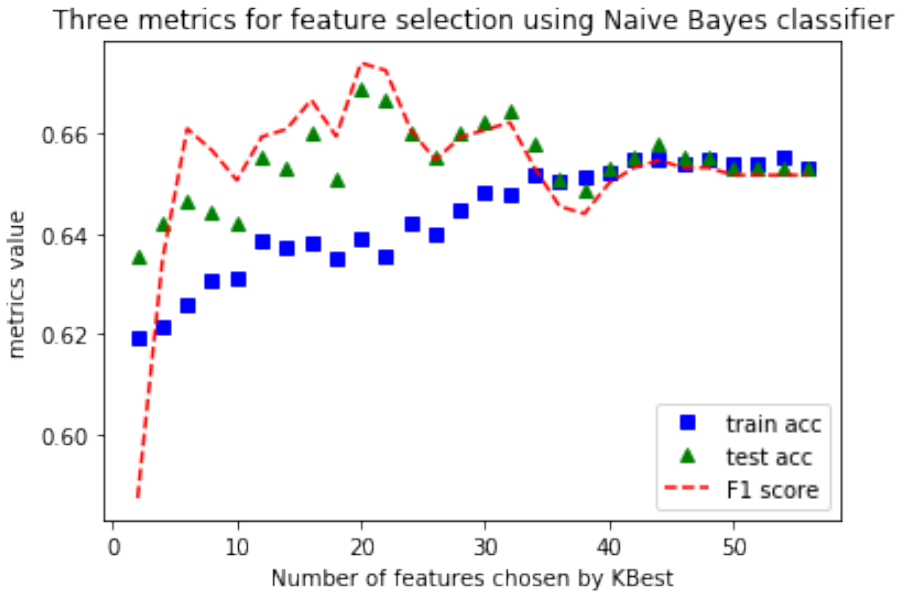

fig 3 Three metrics values with increasing number of feature selected

As seen in fig 3 above, when feature number equals 20, test accuracy reaches its maximum, so as F1 score. Clearly, 20 features are the most powerful discriminative features among 58 features. Some of the chosen feature is shown below:

Feature number	**Feature name**
39	Closeness to LDA topic 1
25	Avg. keyword (max. shares)
16	Is data channel `Tech'?
10	Average length of the words in the content
43	Text subjectivity
36	Was the article published on a Sunday?
46	Rate of negative words in the content

IV. Evaluation approach

I am using three metrics to evaluate models – accuracy, AUC and F1 score. Accuracy directly indicates the proportion of correctly classified data points. AUC measures the trade-off between true positive and false positive for each model. F1 score is the harmonic mean of precision and recall. What's more important is that, F1 score is independent of class distribution. High AUC and high F1 score are

characteristic for good models.

1. Accuracy

Accuracy calculates the proportion of correctly classified data points:

$$ACC = \sum \frac{number\ of\ articles\ which\ Prediction\ equals\ actual}{number\ of\ test\ articles}$$

2. F1 score

sklearn.metrics.f1_score

F1 score works better in terms of evaluation the model. Because F1 score is robust and shows weighted accuracy through taking the harmonic mean of precision and recall. When it comes with cross validation, I compute F1 score for each fold and report the mean of them.

3. AUC (Area under ROC curve)

sklearn.metrics. roc_curve, auc

For SVM classifier with RBF kernel, I use cross validation with 5 folds, and plot the ROC curve for each fold. ROC stands for receiver operating characteristics, which relates the True positive rate (TPR) to the False positive rate (FPR). AUC is the area under this curve.

4. Confusion matrix

sklearn.metrics.confusion_matrix

A confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i but predicted to be in group j.

In binary classification, the count of true negatives is $C_{0,0}$, false negatives is $C_{1,0}$, true positives is $C_{1,1}$ and false positives is $C_{0,1}$.

V. Classification approach and result

1. Linear regression

sklearn.linear_model.LinearRegression

Linear regression represents the least-square fit of the response to the data. It chooses the hypothesis

$$h(\theta) = \sum_{i=0}^{n} \theta_i x_i$$

by minimizing the cost function

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

due to the high variance of target, direct apply linear regression is not acceptable, since the predicted label is not integer, but float. I replace the label value from -1 and 1. Therefore, the labels for data points can be distinguished from their sign. Consider a prediction is correct if its value and the actual label has the same sign (both + or -). The result is shown below:

Train accuracy	0.6614
Test accuracy	0.6734
F1 score	0.6605

Confusion matrix:

$$\begin{bmatrix} 159 & 60 \\ 86 & 142 \end{bmatrix}$$

2. Logistic regression

sklearn.linear_model.LogisticRegression

For logistic regression, the hypothesis is

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

parameters chosen in order to maximize the likelihood

$$\prod_{i=1}^{m} p(y^{(i)}|x^{(i)}; \theta)$$

results is shown below:

Train accuracy	0.6591
Test accuracy	0.6801
F1 score	0.6742

Confusion matrix:

$$\begin{bmatrix} 157 & 62 \\ 87 & 141 \end{bmatrix}$$

One advantage for this model is that I can use a non-linear activation function (sigmoid) to show the contrast to the assumption of linear separability.

3. K nearest neighbors

sklearn.neighbors.KNeighborsClassifier

This method predicts one test sample by taking vote from its k nearest training samples. The class for

test sample will be the majority vote class number. One of the advantage for this method is that it is robust to the fluctuations in the feature values since it calculates the L2 norm across features. However, this method largely depends on the experience of people.

Results for KNN is shown below:

	K=3	**K=5**	**K=7**	**K=9**	**K=11**
Train accuracy	0.786	0.7256	0.7064	0.6892	0.6723
Test accuracy	0.606	0.6152	0.6242	0.6309	0.6242
F1 score	0.612	0.6161	0.6299	0.6325	0.6216

Confusion matrix for k=9 is shown below:

$$\begin{bmatrix} 140 & 79 \\ 86 & 142 \end{bmatrix}$$

4. Naïve Bayes

sklearn.naive_bayes.BernoulliNB

Naïve Bayes was used as baseline for feature selection since it assumes all the features are conditionally independent. I perform Naïve Bayes classifier again for selected 20 features. Results are shown below:

Train accuracy	0.6392
Test accuracy	0.6689
F1 score	0.6740

Confusion matrix:

$$\begin{bmatrix} 146 & 73 \\ 75 & 153 \end{bmatrix}$$

5. Perceptron

sklearn.linear_model.Perceptron

Results for perceptron:

Train accuracy	0.4693
Test accuracy	0.4586
F1 score	0.4319

Confusion matrix:

$$\begin{bmatrix} 113 & 106 \\ 136 & 92 \end{bmatrix}$$

6. Support Vector Machine

sklearn.svm.SVC

(1) Parameter selection for RBF kernel

I use GridSearchCV method for parameter selection. First use StratifiedKFold to partition train data into 5 folds. Next, generate C and gamma value grid. I use logspace to generate 7 C values between 10e-3 and 10e+3. Gamma value range is from 10e-5 to 10, totally 7 values as well. Last, use GridSearchCV to select the best parameter setting pair according its score, which is cross-validated train accuracy.

The best parameters are:

The best parameters are {'C': 0.1, 'gamma': 1e-05} with a score of 0.62

Also, I plot a heat map for the score of each parameter pair:

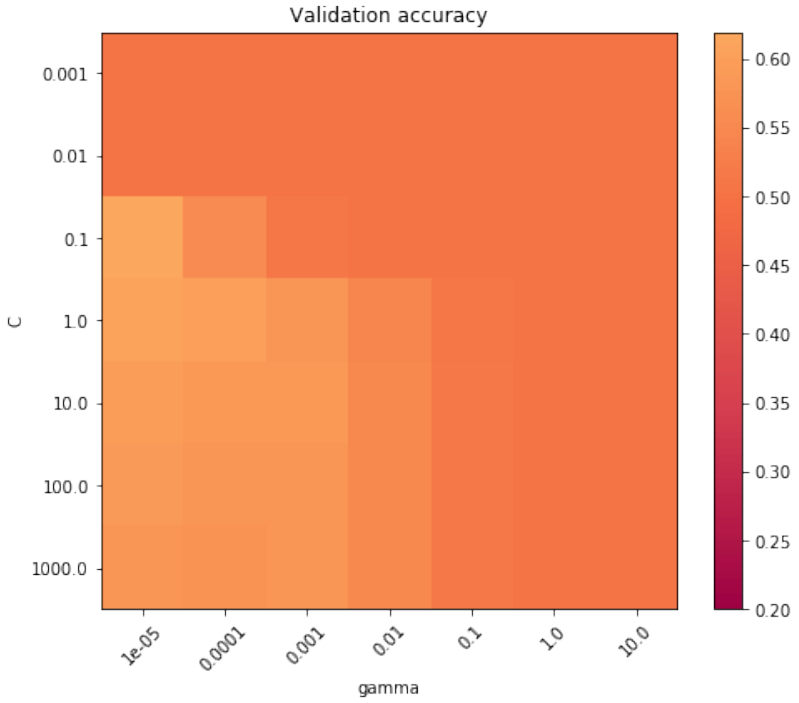

fig 4 heat map for parameter setting

As seen in the heat map, we can clearly notice that when C=0.1 and gamma=1e-5, the patch color is the lightest, which means that it has the highest validation accuracy.

(2) Apply best parameter to SVM classifier

Results:

Train accuracy	0.6193
Test accuracy	0.6353
F1 score	0.6639

Confusion matrix:

$$\begin{bmatrix} 126 & 96 \\ 67 & 161 \end{bmatrix}$$

(3) SVM classifier with other kernels

I tried linear kernel and poly kernel, sadly the project runs into an infinite loop. So, I have to give up these two kernels.

(4) ROC curve for different number of folds

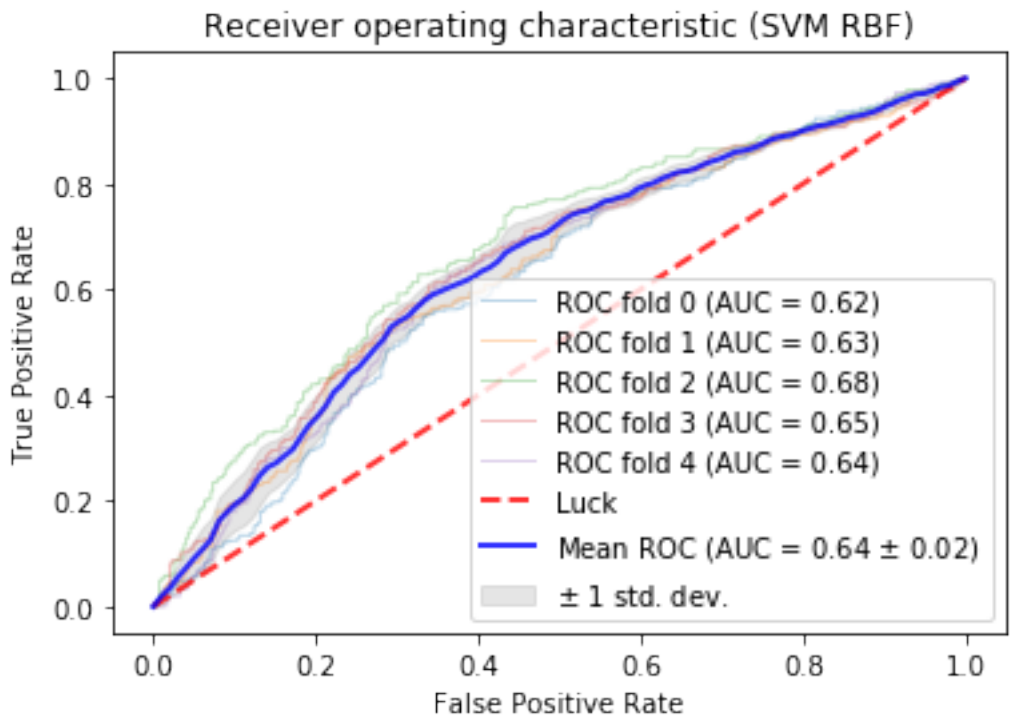

fig 5 ROC graph for SVM classifier with RBF kernel

7. Decision tree

sklearn.tree.DecisionTreeClassifier

sklearn.model_selection.cross_val_score

Decision tree is commonly used in classification method. It is a treelike graph or model to make

decisions and get consequent results. Decision tree implicitly perform variable screening or feature selection and the best feature of using it is easy to interpret and explain.

Results:

Train accuracy	0.5733
Test accuracy	0.5727
F1 score	0.5609

Confusion matrix:

$$\begin{bmatrix} 134 & 85 \\ 106 & 122 \end{bmatrix}$$

8. Random Forest

sklearn.ensemble.RandomForestClassifier

In random forest, I used 100 trees to 500 trees and have different results. Random forest is an ensemble classifier which gives a good generalization performance and avoid the overfitting problem.

	Trees=100	**Trees=200**	**Trees=300**	**Trees=400**	**Trees=500**
Train accuracy	0.6486	0.6482	0.6474	0.6517	0.6534
Test accuracy	0.6823	0.6532	0.6756	0.6599	0.6823
F1 score	0.6787	0.6437	0.6697	0.6514	0.6743

Confusion matrix when trees=500

$$\begin{bmatrix} 158 & 61 \\ 81 & 147 \end{bmatrix}$$

VI. Conclusion

Best results for each method:

Classifier	**Test accuracy**	**F1 score**
Linear regression	0.67	0.66
Logistic regression	0.68	0.67
Naïve Bayes	0.67	0.67
Perceptron	0.46	0.43
KNN (k=9)	0.63	0.63

SVM (RBF kernel)	0.64	0.66
Decision tree	0.57	0.56
Random forest (trees=500)	**0.68**	**0.67**

As we can see in the flowchart, the best model is random forest with tree number 500. Perceptron performs the least. Random Forests performed well because the ensemble model is inherently robust to over-fitting and lends itself well to our high-dimensional data. With only a little bit of hyper-parameter tuning, it very quickly outperformed the other models.