Popularity of Online News Article

I have devised this problem to classify the popularity of online news articles into 'Unpopular', 'Neutral' or 'Popular'. There are a lot of articles published every day as news, articles or blogs. Some of these articles get very popular and gather a lot of audience as it is shared through social media. There are other articles which are not as popular and some that are good enough that some people might read it and share but, do not create as much buzz in the social sphere.

The ability to predict which article become popular and which ones never make it to the top seem challenging and appealing at the same time depending on whether the person is the author, reader, distributor or ad hosts. Sometimes, it so happens that smaller sites get a lot of traffic because of a single article that is read by millions of people which could bring the site host to its knees. Even the author would like to know if their article is going to be read by a lot of people or just barely noticed by others. Ad distributors could make a lot of money by hosting their ads on popular articles as they are read by millions of people which could help their bottom line.

Hence, being able to predict the popularity of news articles beforehand seems challenging but, has myriad of applications. A lot of variables effect which articles are popular and which are not. It could be based on certain geographical region, certain cultural, economic, political, factor and the content and style of the article itself. However, in the Mashable dataset that has been downloaded the focus is on the content of the article itself rather than other extraneous factors.

This dataset is one of the datasets collected from UCI Machine learning repository. It was collected and donated by K. Fernandes et al who have used it for their own research to predict the popularity of online news article based on the number of shares in the social media like Facebook and Twitter. The dataset originally was used for regression modelling but, has been broken down into three bins and converted to a classification problem for this assignment.

The dataset was collected over two years and has been scrubbed to be used for various machine learning algorithms. The dataset contains a set of 61 attributes out of which 58 of the attributes are predictive, 2 are non-predictive and 1 is a field goal which is the number of shares in the social media. Even though, researchers used this dataset to do a two way prediction between popular and unpopular dataset I have found that the data is not linearly separable. Instead, there are a lot of dataset that are neither popular nor unpopular hence, I have devised a third class which are articles that are neutral in terms of numbers of shares on the social media.

The following are some of the attributes of the dataset:

> *url, timedelta, n_tokens_title, n_tokens_content, n_unique_tokens, n_non_stop_words, n_non_stop_unique_tokens, num_hrefs, num_self_hrefs, num_imgs, num_videos, average_token_length, num_keywords, data_channel_is_lifestyle*

Each one these is taken from various articles published on the Mashable website. I have included two examples below:

> *http://mashable.com/2013/01/07/astronaut-notre-dame-bcs/, 731, 9, 531, 0.503787878, 0.999999997, 0.665634673, 9, 0, 1, 0, 4.404896422*

http://mashable.com/2013/01/07/beewi-smart-toys/, 731, 10, 370, 0.559888578, 0.999999995, 0.698198195, 2, 2, 0, 0, 4.359459459

The target value is the number of shares in the social media which ranges from 1 to 800K so, instead of using the raw values I used standard normalization to fit all of the shares into a smaller range which goes from -0.29194 to 72.23. Hence, a demarcation has been created where articles that have standard shares below -0.225 are unpopular and above 0.1 are popular and those in between are neutral.

Initially, the data had a huge number of neutral dataset which caused the classifiers to over-fit for the neutral class. Hence, I did a random removal of some of the neutral data for training so that all the classes were equally represented in the training set. This caused the accuracy of the classifiers to increase by more than 15%.

Out, of the 15K data 60% data has been used for training, 20% has been set out for testing and the other 20% has been set as a holdout set which is used for 1- fold validation. The dataset has been dummy coded with values of 0 for Unpopular, 1 for Neutral and 2 for popular.

	Unpopular	Neutral	Popular	Total	No. features
Training	3303	2756	3438	9497	59
Testing	1166	873	1127	3166	59
Holdout Set	1148	870	1148	3166	59

Decision Tree Classifier

DecisionTreeClassifier from Scikit-learn was used to train and do a three way classification. Initially, the tree was expanded to fit all the data which created a tree of height of 31 and a baseline accuracy of 63.9%. After that the tree was pruned iteratively from 33 to 2 to figure out which gave the best possible accuracy which took about 10 sec. The tree of height 6 gave the best accuracy score on the holdout set of 72%.

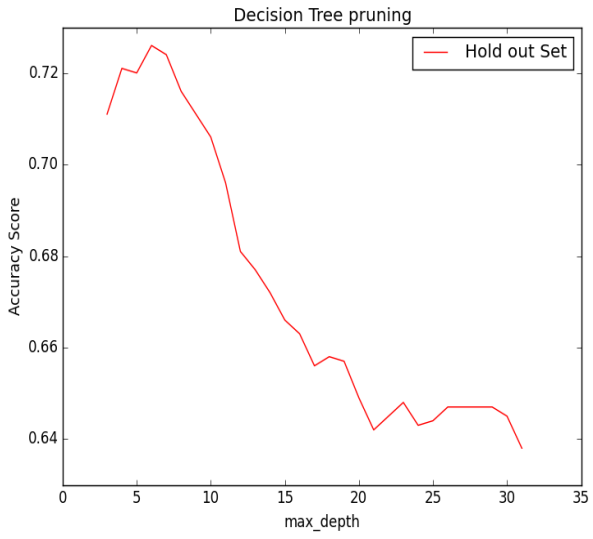


Fig. 1 Pruning tree for various depth

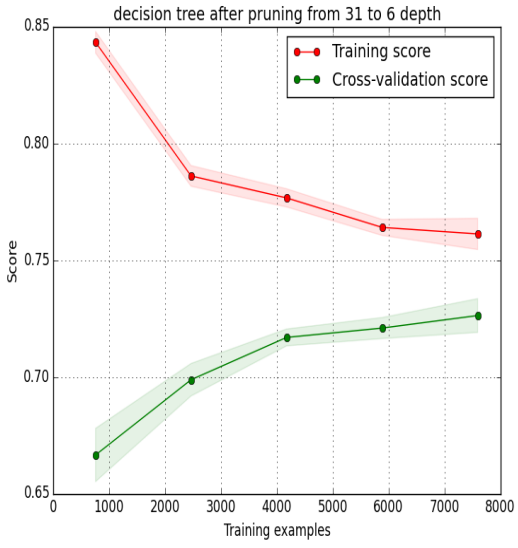


Fig. 2 Learning curve tree of depth 6

Max height of 6 was used to create the final classifier and used for testing and cross validation. From the cross validation graph we can see that as the number of data is increased accuracy on training data decreases whereas accuracy on testing set increases. It suggests that as more data is included the accuracy of the classifier should increase.

The same process was also used to run a feature selection on the dataset and used only 25 most important attributes by doing a K-best feature selection and got the following metrics. Accuracy went down by 3% which suggests that it is best to use all the attributes.

	Training	Testing	Training 25 features	Testing 25 features
Accuracy	0.756	0.726	0.768	0.733
F1 Score	0.748	0.718	0.762	0.727
Avg. Recall	0.756	0.726	0.768	0.733

Boosting

Adaboost classifier from Scikit-learn was used for ensemble learning to boost the DecisionTreeClassifier. There are two major hyper-parameters to modify depth of the decision tree and number of trees used. I have plotted the depth vs. time and n_estimators vs time complexity graph. It seems that they are directly proportional hence, increasing the depth and number of estimators takes a lot longer to fit and predict but, also gives better prediction.

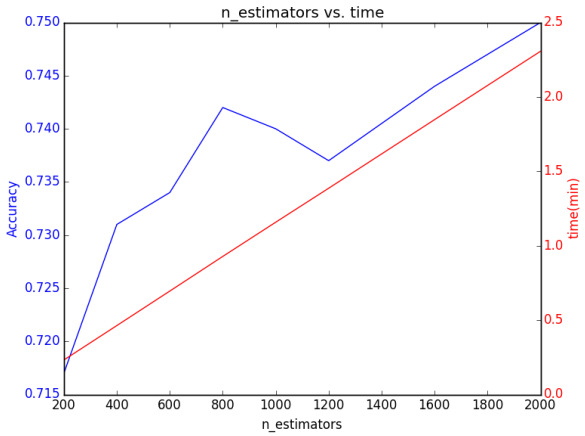


Fig. 3 Complexity curve for number of estimators

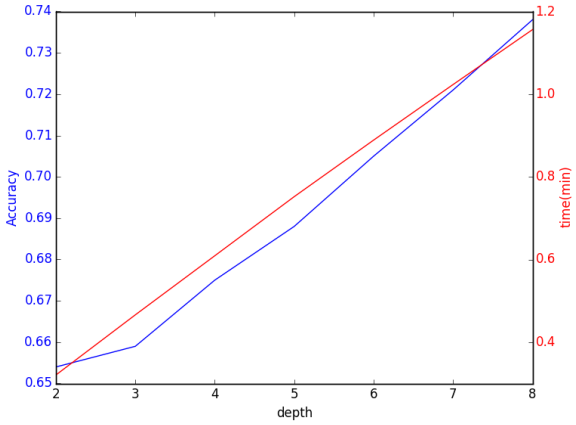


Fig. 4 Complexity curve for depth

Hence, a tree with depth of 7 and 10,000 estimators were used to train the AdaBoost classifier. It took about 30 mins to run and predict. It gave an accuracy of 75% on the testing set which is better than all the other algorithms. The following graph shows the learning curve for the boosted classifier and the accuracy metrics:

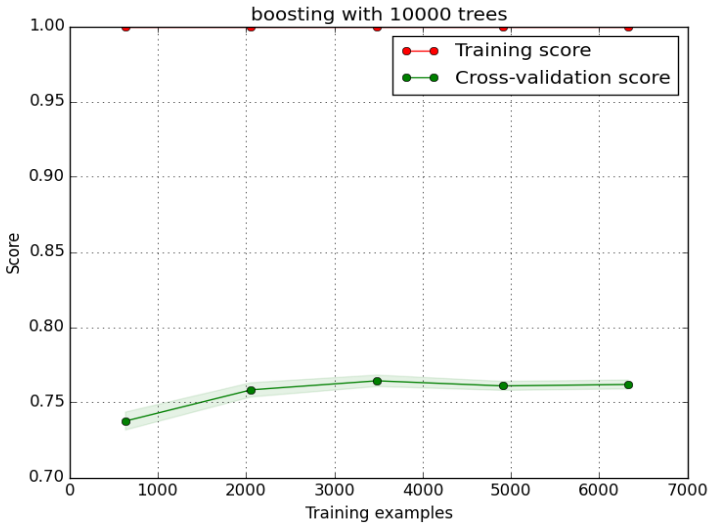


Fig. 5 boosting with 10K estimators

As we can see on the graph above, the classifier has been over-fit with the training data hence very high accuracy on the training set. But, the testing set accuracy has also increased by a little which suggests that feeding more data to train would increase the accuracy.

	Training 7 X 1000	Testing 7 X 1000	Train 7 X 5000	Test 7 X 5000
Accuracy	0.998	0.750	0.998	0.743
F1 Score	0.998	0.741	0.998	0.735
Avg. Recall	0.998	0.750	0.998	0.743
Time (min)	27.757		18	

Above, we can see that accuracy of the classifier only increases by 1% while increasing the number of estimators by two fold.

Neural Network:

MultilevelPerceptron from Weka was used to run the neural network with number of hidden nodes as 30. The number of hidden layer was chosen based on the number of input attributes which is 57. This classifier was run for various epoch which gave different accuracy rate which has been plotted below:

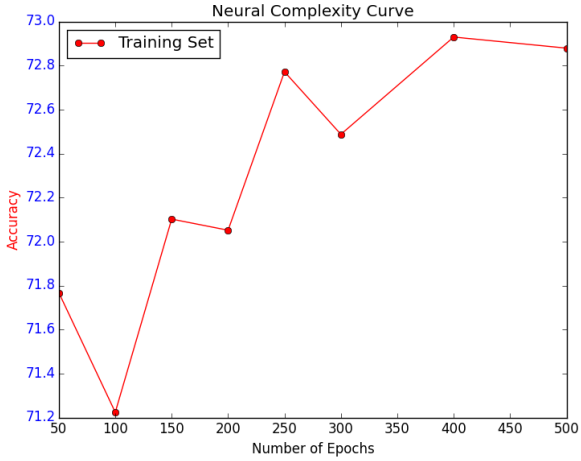


Fig. 6 Number of epochs

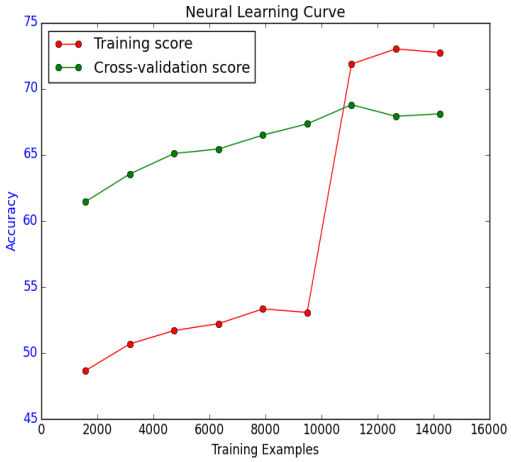


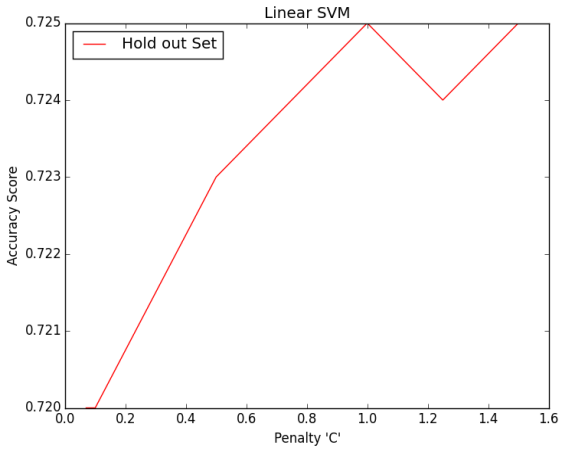
Fig. 7 Learning curve

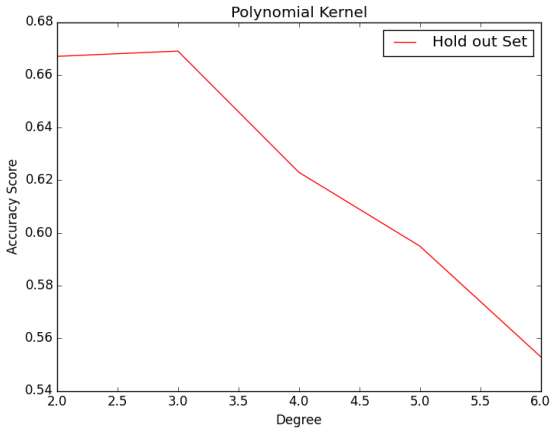
From the above complexity curve we can see that increasing the number of epochs gives better accuracy but after 400 epochs accuracy decreases. Hence, 400 epoch were used to generate the cross validation learning curve for the neural network. The model was used to find a cross-validation error which gave an accuracy of 68% using all the training data.

We can see from the learning curve that as number of training samples is increased the training accuracy increases which suggests the model is being over fitted. However, the overall answer is similar to other models. It might have

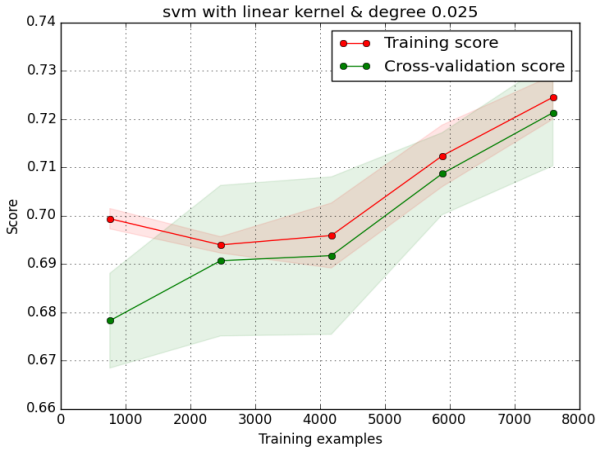
SVM:

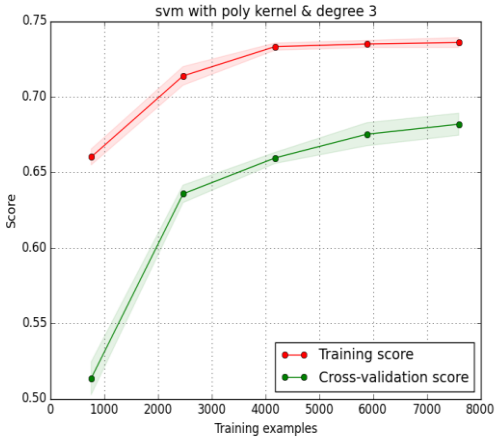
SVC module from Scikit-learn was used to predict the three way classification of popularity for online articles. I have compared linear and polynomial kernel by switching them out. I have compared penalty or ‘C’ factor for linear kernels and different degree polynomials for the polynomial kernel.





We can see that for linear kernel Penalty ‘C’ of 1.5 gives the best accuracy of 72.5 % on the holdout set. Similarly, for the polynomial kernel the best accuracy of 73.6% is given by the polynomial of degree 3. This information is used to create the best classifier and do a cross validation and testing:





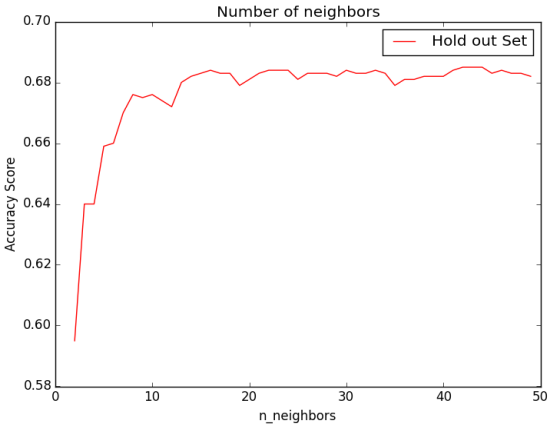
Testing Metrics:

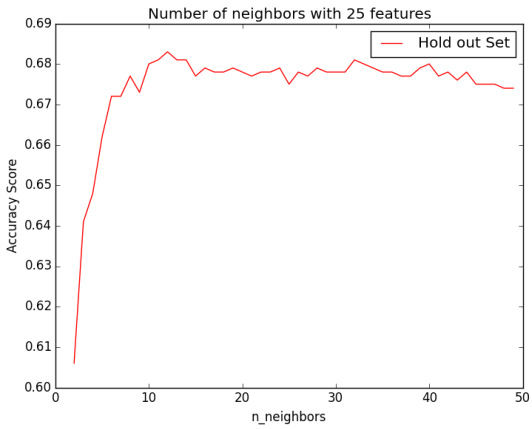
	Training Linear	Testing Linear	Training Poly	Testing Poly
Accuracy	0.733	0.736	0.712	0.682
F1 Score	0.725	0.728	0.711	0.681
Avg. Recall	0.733	0.736	0.712	0.782

Looking at the graph above and the metrics we can safely conclude that linear kernel is better by few percentage points and could be improved by adding more data. The learning and testing score both increase when adding more data

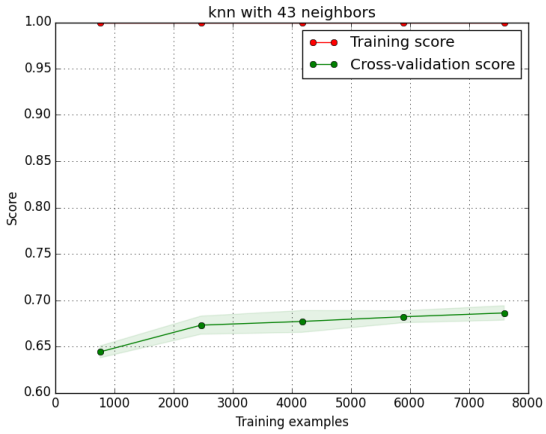
K-Nearest Neighbors:

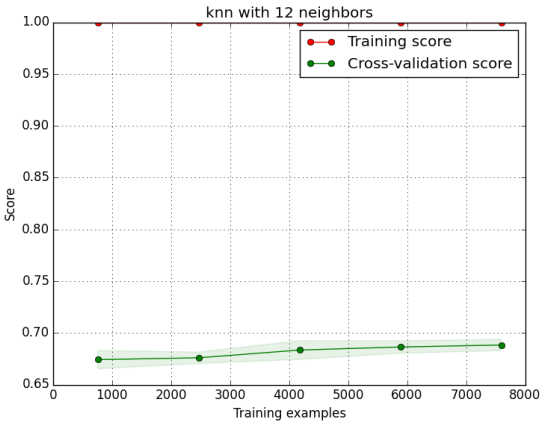
By running the training set, on a default KNeighborsClassifier with the default nearest neighbors of 5 it gave an accuracy of 66% and the dataset with 25 best features also gave an accuracy of 66% on the holdout set. To figure out the optimum number of neighbors the number of neighbors were iteratively increased and the accuracy of the hold-out set was calculated. 43 nearest neighbors gave the highest accuracy for the holdout set with all 59 attributes included whereas 12 nearest neighbor gave the best result for dataset with only 25 attributes included which were 68.5% and 68.3% respectively.





The following chart show the accuracy against the number of neighbors for both the sets with all the attributes and the cross validation learning curves.





Testing metrics for both the dataset:

	Training	Testing	Training 25 features	Testing 25 features
Accuracy	1.0	0.691	0.712	0.682
F1 Score	1.0	0.689	0.711	0.681
Avg. Recall	1.0	0.691	0.712	0.691

We can see that holding out attributes does not make a difference in the accuracy of the prediction and the training time is also only different by 10 sec. Hence, using all the features is better.

Conclusion

Popularity prediction of online news article can be done with 75% accuracy based on the ensemble boosting method. However, it takes a lot of time to train and test. Increasing the number of estimators increases the accuracy of the classifier but, by a very small margin as can be seen by going from 5000 estimators to 10,000 increased the accuracy by 1% whereas the training time increased by a whole lot.

It can also be seen that Decision Trees give similar results but, with very little training time. Decision trees correctly classified the tweets 73% of the time with very little overhead. Using only the best 25 features also increased the testing time because decision tree also uses information gain for feature selection.

Neural network did not perform on par with other algorithms. It actually performed worse than other classifiers. It could be because there are some noise in the data which is causing it to misrepresent some of the weights. There are some articles on this dataset that are hyper popular which could be causing the weights to be non-optimal.

Linear SVM also performed on par with the Decision Tree algorithm and it was very fast. It gave an accuracy of 73% on the training set which could mean that the data is linearly separable. KNN gave an average score of 69% with all the features and with only 25 features however, it performed the same meaning the data points that are in similar distance did not change based on the number of features.

References

Caruana, Rich, and Alexandru Niculescu-Mizil. "Data mining in metric space: an empirical analysis of supervised learning performance criteria." Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2004.

Hearn, Alison. "Structuring feeling: Web 2.0, online ranking and rating, and the digital 'reputation'economy." ephemera 10.3/4 (2010): 421-438.

Tatar, Alexandru, et al. "From popularity prediction to ranking online news."Social Network Analysis and Mining 4.1 (2014): 1-12.

Online News Popularity. (n.d.). Retrieved September 21, 2015. http://archive.ics.uci.edu/ml/datasets/Online+News+Popularity