Implementing Various Classification and Ensemble Methods to Predict Rainfall

TIMOTHY KIM

CS 178: Machine Learning & Data Mining | 19 March 2018

MODEL PERFORMANCE

	Training	Validation	Test (Public)	Test (Private)
KNN	0.8587	0.7363	*	*
Decision Tree	0.6189	0.5977	*	*
AdaBoost w/Decision Tree	0.5990	0.5993	*	*
Gradient Boost w/ Decision Tree	0.7101	0.6054	*	*
Logistic Regression	0.5733	0.5729	*	*
Logistic Regression Ensemble (KNN, AdaBoost w/DT, and Gradient w/DT)	0.8043	0.7445	0.7572	0.7557

* Kaggle did not let me test the performance of these classifiers due to some unknown error.

For some reason the leaderboard did not update for me after 11:57 pm and therefore it shows my previous score of 0.574. However, my result for

MODELS USED

K-Nearest Neighbors Classifier

I used the K-Nearest Neighbors Classifier from the sklearn library. The stratified k-fold cross-validation method is implemented with the KNN Classifier to fit the training data. After testing extensively on which value of k worked the best (from 1 to 20), k=7 was the greatest performer on our validation set with 0.73699. However, I did notice that the AUC score did not increase as significantly after 5. The changes were either minimally increasing or even minimally decreasing, so I decided that a greater number of neighbors does not impact the accuracy enough to be viable.

Decision Tree

The decision tree was used only to stack it with the AdaBoost and Gradient Boost Classifiers in the sklearn library. By iterating over the set of all possible max_depth options from 1 to 20, I found that the accuracy increases as max_depth increases. Therefore, I decided to test the Decision Tree with the AdaBoost and Gradient Boost Classifier to determine what value max_depth can be set to so that the learner is fairly accurate and time efficient. Using AdaBoost with the default 50 estimators, I attempted to increase the max depth of the tree and record how much time each test took and how much the AUC score increased.

Determining Optimal Max Depth of Decision Tree through AdaBoost Classification

Max_depth =	Time (sec)	Score Increase (%)	Score Increase / Time
1	11	1.97	0.179
2	23	2.44	0.106
3	34	2.44	0.071
4	42	3.08	0.073
5	54	1.90	0.035
6	62	2.93	0.047
7	75	4.15	0.053
8	83	3.89	0.046
9	96	4.85	0.050

I decided to use 7 as max depth because although I realize there may be errors in computing, max depth as 7 seemed like a reasonable compromise between efficiency and performance.

AdaBoost Classifier

The AdaBoost Classifier was used as one of the classification models because it is a good boosting classifier for minimizing loss in other classifiers. This is because it changes the weights of each instance in the sample distribution so that it puts more emphasis on the wrongly predicted instances. The AdaBoost Classifier used from the sklearn library included weak learners. I decided to use 100 weak learners as an arbitrary value but after testing at a different time, I realized that the AUC score does not increase after a 100 compared to before 100.

Gradient Boost Classifier

The Gradient Boost Classifier was used to boost the performance of these other classifiers by calculating the residual of the errors and minimizing these said errors.

ENSEMBLE PREDICTION

I combined the boost models (AdaBoost and Gradient Boost) with the Decision Tree because I can easily put weights on the tree compared to any other learning models. Additionally, I found that AdaBoost and Gradient Boost work well with weak learners such as Decision Trees. The Logistic Regression classifier was used because it could output a confidence probability instead of a discrete output. Also it seemed like a efficient wrapper class that could combine the KNeighbors, AdaBoost and GradientBoost Classifiers seamlessly, and output more accurate predictions rather than using the vote or average stacking methods.

CONCLUSION

The most influential methods I believe I implemented was the Logistic Regression ensemble to stack the AdaBoost Classifier, Gradient Boost Classifier, and the KNeighbors Classifier. The difference between the individual and stacked classifier method is clearly displayed in the first table and the positive effect on the performance is clear. A method that I tried implementing before these was the perceptron classifier and the linear regression model. I didn't expect them to work well, but I did not expect them to give training and validation AUC performances under 0.5 (the theoretical success of a random classifier). I know that these models did not work because the data seemed to fit a nonlinear model than a linear one. Also, I attempted to