

CS178 Final Project Report

Kaggle Team Name: 3BigGUAGUAs

Team Members: (Name - NetID -Kaggle Name)

Tianle Liu - tianlel1 - Tianle Liu

Yiting Wang - yitinw9 - echoyitinnn


Yuling Yan - yulingy3 - yyuling

Performance Table:

Classifier	Training Error	Validation Error	ROC_AUC	Kaggle Score
Logistic Regression	0.3080666666666667	0.30778000000000005	0.6642382384644245	0.65825
Random Forest	0.2979625	0.29800000000000004	0.6896228716219419	0.68719
Neural Network	0.2281375	0.27435	0.7470463827450654	0.74047
Ensemble	/	/	/	0.75068

Private Leaderboard Result:

Our private leaderboard result is shown below. But we forget to choose the final performance for the competition, it used our highest score defaultly. We also included a screenshot of our final ensemble file score.

53	—	3BigGUAGUAs		0.78360	10	5h
ensemble.txt			0.74991	0.75068	<input type="checkbox"/>	
5 hours ago by echoyitinnn						
add submission details						

Description for each model:

Logistic Regression	<p>For Logistic Regression, we firstly split the imported data, which is X and Y, for 75% as training data part and 25% as validation data part. Then, we set some possible polynomial degrees (1,2,3) to test the training/validation error for each respectively. During the test, we used the Logistic Regression algorithm from the source sklearn with changing the possible polynomial degrees. After the testing, we found that both error rates decreased from polynomial degree 1 to 2, and increased from polynomial degree 2 to 3. Based on what we've learned, we believed that with the polynomial degree increasing, the training data error rate should be decreasing and the validation data error rate should be decreasing to increasing (this is based on the understanding of the process from underfitting to overfitting). Although the validation data error rate changing trend is like what we've expected, the training data's trend should not have the same trend (which means from decreasing to increasing as the validation data error rate changing trend showed).</p> <p>Thus, we just picked the most suitable degree -- degree of 2 for the result of this part (this decision is based on our finding of the AUC value for the polynomial degree). We know that we need to combine it with other ways to test or optimize.</p>
----------------------------	--

Random Forest	<p>Based on what we have got and concluded from the homework, we kept some data to our testing phase (the data we kept is maxdepth's best output, which is 7). We still used the same split ratio for the training data and the validation data. For our method, we used the RandomForestClassifier (we will call RFC next) from the sklearn. Our testing focused on finding the improvements from the maxdepth = 7. Firstly, we split data train-validation with the 0.75 ratio. As we see, we have two test targets in RFC function. First is testing n_estimators. Our testing pool is [100,200,300,,,,,1500].</p> <p>After the testing n_estimator, we found out the lowest validation error rate, and its n_estimator is 800. Then we kept this n_estimator unchanged, and tested the min_samples_leaf. Our testing pool is [1,2,3,4,5]. After this testing, we found that when min_samples_leaf = 5, it reached the lowest validation error rate and achieved the highest AUC score. Thus, we put it into the submission test, then found it has optimized the score with 0.03.</p>
Neural Network	<p>For this model, we used the MLPClassifier method from the sklearn source. Firstly, we split the data with the same 0.75 train-validation ratio. Our thoughts of modifying the test is trying to find the best hidden layer size structure for the classifier. As we tried, it is hard to test the best number of nodes in layers (since it costs a lot of time and is unable to get some efficient results). Thus, we tried to find the best structure for hidden layers. We test for [1,2,3,4,5] hidden layer structure respectively. For nodes in the hidden layer, we kept it as high as possible (but also, as reasonable as possible) and unchanged for the initial input for the structure. Then we found the 1-layer structure works best. In this case, we put it into the Kaggle and run it for a test.</p>

Ensemble of each models:

For now, we have three models' Kaggle score. So our way to ensemble them with weighted separation. Based on the table above, we found that the Neural Network has the lowest validation error rate and highest Kaggle score, then the second place is Random Forest, and the last is Linear/Polynomial Regression. So our way to pick the weighted percent is based on low validation error rate and high kaggle score.

Conclusion:

After testing three models: Logistic Regression, Random Forest and Neural Network. We found that the Neural Network works best among three of them. We concluded that the reason for Neural Network has the best performance is its complexity. For a large amount of data, if we use the Logistic Regression, it is hard to deal with the data accurately just using the linear algorithm. So it is reasonable that the Logistic Regression has the worst performance. Then, for Random Forest and Neural Network, they run in a non-linear algorithm. Compared with the Neural Network, Random Forest is still simpler. Random Forest is based on the algorithm of the Decision Tree, but it actually combines many possibilities together to increase the accuracy. However, for the Neural Network, it's based on our "thinking way". It is kind of machine learning, but we believe it is far beyond machine learning, since its way to learn and predict data needs is more complex. So, according to the Neural Network's advantages on comprehension, complexity and fitting, we think it is not surprising that the Neural Network has the best performance among them. Although the Neural Network has the best performance, it doesn't mean we should ignore the high time cost for the Neural Network compared to the other two, or even other models. Maybe in the future, we have to consider the time cost for the choice way too. But for this project, based on the accuracy and fitting result, we think what we got is enough.