Teamname on Kaggle : Overfitting Final AUC: 0.80181 Rank: 12

Team leader: Shengquan Ni Team members: Yangguang He, Shengquan Ni, Changchuan Shen

Model	Negative log-likelihood on training	Negative log-likelihood on validation	Training accuracy	Validation accuracy	Validation AUROC
KNN	8.568	10.119	0.752	0.707	0.712
Random forests	7.527	9.248	0.782	0.732	0.768
Neural network	10.095	10.096	0.708	0.708	0.702
Ensemble model	5.147	8.766	0.851	0.746	0.802

KNN:

For feature choosing, we used the reconstructed input data using the first 9 principal directions. Thus, the shape of the dataset was changed to (200000,9). We used KNeighborsClassifier from sklearn to build the KNN model and 5-fold CV to measure the performance of it. We used balltree on KNN to improve its speed. For hyperparameter setting, we found the AUC peak of the KNN is around k=10 to 20 after we tried with 10 different k choices from 1 to 100. Finally, we chose K=11 because it could give the best AUC score (≈0.712).

Random forests:

We used the raw input to train this model because the prediction of a decision tree would not be affected by pre-processing to normalize the magnitude of each feature, but we decreased the number of the data points to 50000 so that the training process would be faster. Thus, the shape of the dataset was changed to (50000,14). We used RandomForestClassifier from sklearn to build the random forests and 5-fold CV to measure the performance of the model. For hyperparameter setting, we hold “n_estimators” and “min_samples_split” and other parameters fixed and gradually decreased the search scope of “max_depth”. We used the same technique for selecting good “n_estimators” and “min_samples_split”. Finally, we decided to use “max_depth” = 25, “n_estimators” = 50, ”min_samples_split” = 60. After choosing optimal hyperparameters, we train the model with the whole raw input, and got the AUC score around 0.768.

Neural networks:

We used the reconstructed input data using all principal directions because it performs best. We also normalized the data because pre-processing was very important when training NN. Thus, the shape of the dataset was changed to (200000,14). We used MLPClassifier from sklearn to build the NN model and 5-fold CV to measure the performance of it. We noticed that we have 200000 data points. So, we decided to set “solver” = ’adam’ because it “works pretty well on relatively large datasets” as stated in the document. We set “activation” = ’tanh’ because it gave the best AUC among all choices. We set “early_stopping” = True to prevent overfitting and decrease the training time. We set the number of

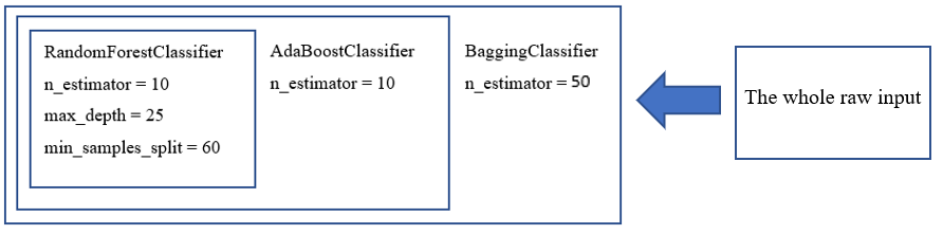
hidden layer to 4. The first layer has 128 nodes, the second layer has 64 nodes, the third layer has 32 nodes, the fourth layer has 16 nodes. However, the performance of the model was still bad, the AUC score was always around 0.8.

Other models:

We also tried SVM (AUC≈0.659), linear models (best AUC≈0.581), Gaussian Naïve Bayes (AUC≈0.628). Their performances were poor, so we discarded them.

Ensemble model:

We noticed that RandomForest classifier performs the best on the three models we have tried. Thus, we decided to build an ensemble learner based on RandomForest. The reason why we didn't use the other two models was that their predictions may mislead the ensemble model, like the bucket effect. We applied AdaBoost on my RandomForest model because we think AdaBoost works better on classification problem and GradientBoost works better on regression problem. However, after applying AdaBoost, we noticed that the accuracy on training data (≈0.99) is much higher than it on validation (≈0.70), that looked like overfitting. So, we decided to use Bagging to decrease the chance of overfitting. The final structure and hyperparameter settings of our ensemble model is shown below. The performance of the ensemble was satisfying, and we got the AUC score around 0.79.



Conclusion:

We think RandomForest, KNN and Boosted learners are all particularly appropriate for this data. RandomForests prevents overfitting, can be trained in a short time and it doesn't require pre-processing or feature selection. After investigation, we find the data seems not linearly separable in low-dimensional space, so linear models are expected to perform poorly, but KNN is still a good choice. KNN is robust regarding the search space, that means the data doesn't have to be linearly separable and it only has 2 parameters to tune (distance metric and k). Boosted learner is a boosted ensemble of some base learner and it can reduce both bias and variance. Thus, it will perform better than those base learners. We think SVM worked poorly for this data due to the enormous dataset. Training a SVM on such a big dataset takes very long time. Although we can do subsampling, it is not guaranteed that the pre-processed data can represent the whole dataset. NN is also not suitable for this data because it might be trapped in local optima and it is very hard to tune the number of hidden layers and hidden nodes in each layer when we didn't know much about the data. In addition, the data seems to have some outliers. NN might be too sensitive to ignore them.