Insider Trading Roles Classification Prediction on United States conventional stock or non-derivative transaction

Amartya Mukherjee, Betty Qin, Maggie Zhao, Muyuan Yang

December 2020

1 Background Material

The United States is one of the most frequently traded financial markets in the world.The Thomson Router insider trading table 1 dataset captures all insider activities as reported on SEC (U.S. Securities and Exchange Commission) forms 3, 4, 5, and 144. There had been studies shown, insider trading leads to abnormal returns after the transactions. Insiders' in-depth knowledge in their firm and its customers leads to market believe it is worthy mimicking their trades. Non-insider clients would sought after insider trading details from brokers and make trading decisions. (Shkilko et al., 2017).

2 Motivation

According to Shkilko et al.,brokers tend to mimic trades 90 minutes around the time of insider reports to the exchange. Therefore, quickly filter, identify and react to key insider trades is beneficial for investors. We believe that using variables (such as transaction date, security type, and transaction amount), we could predict the roles code for a new transaction. The reason for the chosen prediction is that the role of the insider gives investors signals of potential internal activities and private information. This is crucial for investors to detect important market signals from those insider trading activities, such that they could benefit from the market.

3 Discussion/ Analysis

3.1 Data Preprocessing

We selected over 2000 constituents in NASDAQ Composite Index and in the database as our training set. The index includes companies in various sectors such as energy, materials, industrials and different sizes. Then, we randomly selected 1000 companies for testing sets. We then filter the dataset by its cleanse indicator, which shown its completeness and confidence in correctness. There are 175,726 and 62,982 records in training and testing set respectively.

Role codes is our target variable. One insider could have multiple roles in a company. We divided the role codes into 4 levels hierarchy, which shown the important of the trades. For example, a CEO and a General Counsel are considered as level 1, and beneficial owners are considered as level 4. In our dataset, 4 variables contain the insiders' roles in the company, and we selected the highest level for each insider's roles as target variable.

In the dataset, we recognize the date different between SEC Receipt Date and transaction date is key variable, and by trials and errors, we realize most of the dates in the datesets do not add value to the prediction. Therefore we abandons most dates and columns such as tickers, trader's names, company names to limits number of variable required and the generalisation of variables required to predict role codes.

3.2 Models Building

3.2.1 Support Vector Machine

The Support Vector Machine classification algorithm was imported from the sklearn package in Python. In multi-class classification, the One-Vs-Rest scheme is used. This scheme divides the multi-class classification problem into multiple binary classification problems where every class is compared to the rest of the classes (i.e. Every data point labelled 1 is compared to every data point labelled 2, 3 or 4, then every data point labelled 2 is compared to every data point labelled 1, 3 or 4, and so on). Each of these binary classification problems is done using Support Vector Machines. The Linear Support Vector Classification algorithm is used in this report as it is more efficient with large datasets. The categorical feature "SECTITLE", as well as all the date features, "FDATE", "TRANDATE", "SECDATE", "SIGDATE" are removed from the dataset during the training of the model as that gives a better accuracy. During pre-processing, all the columns of the dataset are normalized using the StandardScaler function imported from sklearn.

The Linear Support Vector Machine model has a train accuracy of 60.50 percent and a test accuracy of 60.26 percent. The proportion of points labelled 2 in the train set is 60.54 percent and the proportion of points labelled 2 in the test set is 60.24 percent. This shows that the model does not outperform a model that classifies every data point with the value 2, which is undesirable.

Potential areas of improvement include using different kernels. Due to the dataset being large, the Linear Support Vector Classification algorithm is used as it is significantly faster than all other Support Vector Classification algorithms in the sklearn package. However, it will be interesting to see how, given more processing power, other kernels such as polynomial and RBF will perform.

3.2.2 K-Nearest Neighbors

The K-Nearest Neighbors classification algorithm was imported from the sklearn package in Python. This algorithm classifies a point based on the majority of the k points that have the smallest distance from it. The distance metric used in this report is the Euclidean distance. The categorical feature "SECTITLE", as well as all the date features, "FDATE", "TRANDATE", "SECDATE", "SIGDATE" are removed from the dataset during the training of the model as that gives a better accuracy. During pre-processing, all the columns of the dataset are normalized using the StandardScaler function imported from sklearn. Grid search shows that setting the k parameter to 11 gives the best results.

The K-Nearest Neighbors model has a train accuracy of 79.13 percent and a test accuracy of 65.12 percent, thus showing better performance compared to the Support Vector Machine model. This also shows that the K-Nearest Neighbors model outperforms a model that classifies every data point with the value 2.

Potential areas of improvement include balancing the dataset. While the model outperforms a model that classifies every data point with the value 2, the reason a lot of points get misclassified is because data points with the label 2 appear more frequently than all other classes. Thus, it will be interesting to see how the K-Nearest Neighbors model will perform if a more balanced dataset is provided.

3.2.3 Decision Tree

We implemented and trained all three models: Fine Tree, Boosted Tree and Bagged Tree using Classification Learner App from MATLAB. Fine Tree is a decision tree with maximum 100 splits while Boosted Tree and Bagged Tree, as their names suggest, are decision tree with boosting and decision tree with bagging respectively. "SECTITLE", the only categorical feature, is removed in the training of final model as excluding this feature results in better test accuracy. This is because the test data contains some categories of "SECTITLE" that are not contained in the train data. To avoid overfitting, we use holdout cross validation which takes 25 percent of the train data as validation data.

Fine Tree has a train accuracy of 67.83 percent and test accuracy of 64.05 while Boosted Tree has a train accuracy of 67.12 percent and test accuracy of 63.86. Even with a longer training time(40

seconds versus 6 seconds), Boosted Tree does not outperform than Fine Tree. On the other hand, Bagged Tree is a big improvement with train accuracy of 97.63 percent and test accuracy of 70.21 percent. There is a big difference between the train accuracy and test accuracy, but this is unlikely due to overfitting. We tried reducing the maximum number of splits and in consequence both train accuracy and test accuracy dropped although train accuracy dropped more significantly. Test data is randomly selected and most if not all the companies in test data are not in train data. The relatively low test accuracy indicates that prediction on insider roles given the information we have is a hard task.

There are high variabilities in our data, so it is reasonable that Bagged Tree performed way better than the other two models. The feature transaction price "TPRICE" has a standard deviation of 204.17. The feature number of shares "SHARES" has a standard deviation of 1.2873e+05. Bagging generates many subsets of original data. Then it trains a classifier for each of the subsets and takes the average as the final classifier. In this way, variability can be reduced if the subsets are not strongly correlated.

3.2.4 Neural Network

We used Python and Keras to construct the neural network models and set up GPU environment using CUDA to train the models. First, we built a regular feed forward neural network with two hidden layers. The two hidden layers are dense layers with 64 nodes and 32 nodes respectively and use Relu activation function. The output layer has 4 nodes since we have 4 output classes and uses Softmax activation function. We trained the model with batch size 128 and 20 epochs. The test accuracy is merely 60 percent which is worse than all Decision Tree models.

There are potential improvements to our model. One of the options is Convolutional Neural Network (CNN) which is introduced in class. However, CNN takes 2d array as input and is mainly used for image classification. The next option is Long Short-Term Memory (LSTM) which is a modified version of Recurrent Neural Network (RNN). LSTM does a good job at preventing vanishing/exploding gradient problem faced by RNN. We replaced the first dense layer with a LSTM layer which has same number of nodes as before. LSTM model has train accuracy of 66.85 percent and test accuracy of 63.86 percent which are on par with those of Fine Tree and Boosted Tree.

LSTM can capture sequential information present in input data using recurrent connection on hidden nodes, so it is good at classifying Time Series data. Our data set contains several date features like filed date "FDATE" and transaction date "TRANDATE" which are features of Times Series data. Therefore, the LSTM model outperforms regular feed forward neural network.

3.3 Cross Validation

In this section, the cross-validation performance of the models described above in classifying the role codes. The metric that was used was classification accuracy and K-Fold cross-validation was used with $K = 10, 20$.

The summaries of the results of K-Fold cross validation can be found in the appendix. From Figure 1, the highlighted cells are the most optimal scores in each category. The training and validation accuracy for neural networks is determined by the average training and validation scores from each fold. Since 20 epochs were run, the average of 20 scores were taken. It is observed that bagged tree performs the best across all categories in terms of predictive accuracy and the stability of its results in 10-Fold cross-validation.

From Figure 2, the same conclusions can be made about bagged trees as in 10-Fold cross-validation. Looking at the differences in average K-Fold accuracy between 10-Fold and 20-Fold, it is observed that most models have an increase in their scores when the cross-validation folds increase. An exception to this is the SVM model, which has decreased from 60.47% to 60.38%. However, this trend of increase is also seen in the median cross validation scores and top-5 average cross-validation scores as well. The standard deviations in the cross validation estimates are also seen to rise as folds increase as well. An explanation for what was observed could be attributed to having a larger training set when folds are increased. By having a larger training set, the models were able

to predict the validation data better. However, the validation set decreased when folds increased. This would result in a higher variation in validation accuracy across all models. The accuracy per fold for each model in both the 10-Fold and 20-Fold scheme can be found in the appendix.

From Figure 3 and Figure 7, it can be seen that the cross validation accuracy for boosted and fine trees are closer to the training score than the testing score. For bagged trees, the cross validation scores are situated between training and testing scores and are stable in both the 10-Fold and 20-Fold case. In Figure 4 and Figure 8, the cross validation scores can be seen to have more variance compared to trees, but their scores are closer to the testing score than the training score. In Figure 5 and Figure 9, most cross validation scores outperform the average training, average validation, and testing scores. This was also observed in the neural network cross validation averages in Figure 1 and Figure 2. In Figure 6 and 10, it can be seen that there is large variation in cross validation scores for SVM. However, the scores fluctuate around the training and testing score.

The performance seen in the tree algorithms is due to its ability to learn different decision rules to output a classification. Since the training data in cross validation is similar to the training data in fitting, the trees would create similar decision rules to fitting on the whole training set. The greater drop seen in bagged trees suggests that this algorithm is more sensitive to reduction in training data compared to boosted and fine trees. However, it would output more stable predictions based on the information it has due to its ensemble structure. The performance of the K-nearest neighbours algorithm appears to be sensitive to changes in the K-Fold training data. This is because changing the training data would change the K points that are nearest to the testing point, which would result in varying predictions depending on the data set. Consequently, this algorithm is seen to have the third highest standard deviation in 10-Fold and second highest in 20-Fold. In the neural network, the higher scores in cross validation is due to the implementation of dropout in the network. This would allow the network to generalize the cross validation training data better, which would result in higher validation scores since both sets come from the same overall training set. However, the variation seen is due to the random initialization of weights and the change in training data after each iteration. In addition, the large variation in support vector machines is due to the changes in the cross validation training data. In the training and testing data, it is observed to have poor performance classifying insider roles. This suggests that the data is mostly not linearly separable. With this information, it would suggest that changes in the training data from cross validation would result in greatly varying linear decision boundaries in each fold. This would also explain why the support vector machine performed poorly when the folds increase. Since this data set has a lot of variability and is most likely not linearly separable, adding more points to training would add complexity to the problem. This would make predictions not as certain as with fewer folds.

4 Conclusion

From this analysis on classifying insider trading roles from transaction data, it was seen that the current models used (decision trees, K-nearest neighbours, neural network, and support vector machines) predicted insider roles with satisfactory accuracy. This was partially due to the variability and imbalance of the training data set where role 3 is the rarest label. In addition, the variability was also seen in the testing data and there were new companies in the testing data. From how the models have performed, this may indicate further adjustments to the models need to be made (e.g. different kernels in support vector machines, using convolutional neural networks with long short-term memory), the data set needs to be balanced, or that there needs to be more variables available in the data for prediction. With this current circumstance, the bagged tree performed the best in all categories from cross validation, which is mostly attributed to its ensemble structure. There is potential with this task to classify insider roles, however the suggestions above will need to be investigated to see if they improve prediction accuracy.

5 Appendix

Model	10-Fold CV Average	10-Fold CV Median	Top 5 10-Fold CV Average	Train Accuracy	Test Accuracy	10-Fold CV Standard Deviation
Bagged Tree	0.83767	0.83805	0.83926	0.9763	0.7021	0.002008344
Boosted Tree	0.6701	0.6696	0.67266	0.6712	0.6386	0.003141125
Fine Tree	0.67505	0.6749	0.6772	0.6783	0.6405	0.00249143
SVM	0.604742967	0.601849614	0.622335191	0.605140958	0.602823029	0.021551606
NN	0.692225349	0.692047238	0.70334475	0.66853	0.6386	0.017780091
KNN	0.677292792	0.67663461	0.688723412	0.7913	0.6512	0.016643956

Figure 1: 10-Fold Cross Validation

Model	20-Fold CV Average	20-Fold CV Median	Top 5 20-Fold CV Average	Top 10 20-Fold CV Average	Train Accuracy	Test Accuracy	20-Fold CV Standard Deviation
Bagged Tree	0.839595	0.8387	0.84464	0.84246	0.9763	0.7021	0.003619316
Boosted Tree	0.67001	0.66995	0.67616	0.67361	0.6712	0.6386	0.004759135
Fine Tree	0.675705	0.67545	0.68118	0.67893	0.6783	0.6405	0.004378593
SVM	0.603803688	0.606419303	0.640709164	0.626133914	0.605140958	0.602823029	0.028868849
NN	0.718539172	0.719683707	0.740069389	0.733513802	0.66853	0.6386	0.017550747
KNN	0.686096166	0.683890093	0.71342038	0.702160945	0.7913	0.6512	0.021978476

Figure 2: 20-Fold Cross Validation

Figure 3: Accuracy per Fold 10-Fold Cross Validation for Trees

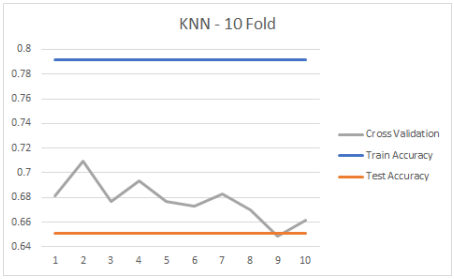

Figure 4: Accuracy per Fold 10-Fold Cross Validation for K-Nearest Neighbours

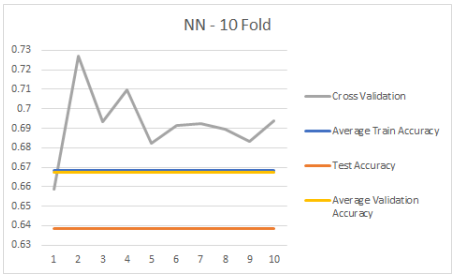

Figure 5: Accuracy per Fold 10-Fold Cross Validation for Neural Network

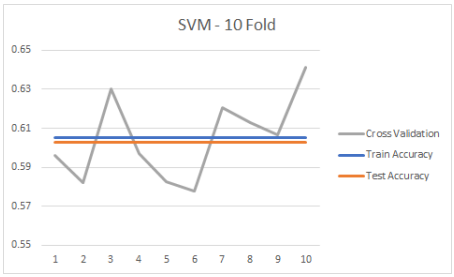

Figure 6: Accuracy per Fold 10-Fold Cross Validation for Support Vector Machine

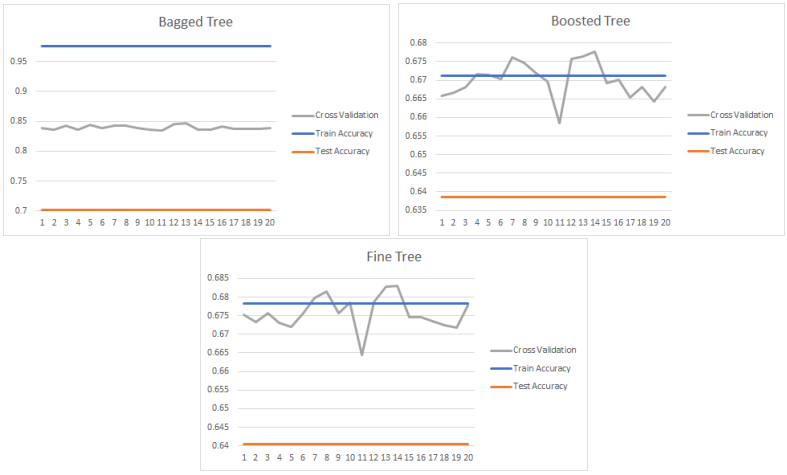

Figure 7: Accuracy per Fold 20-Fold Cross Validation for Trees

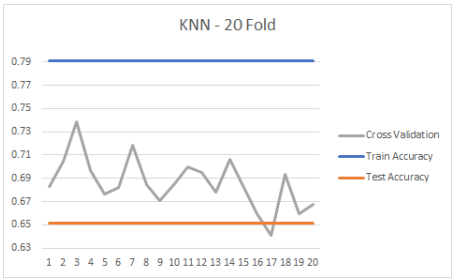

Figure 8: Accuracy per Fold 20-Fold Cross Validation for K-Nearest Neighbours

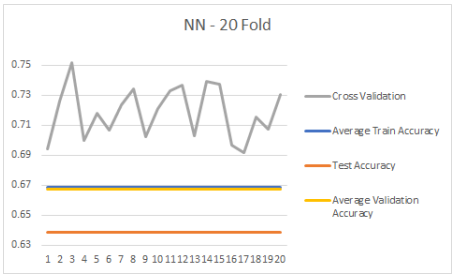

Figure 9: Accuracy per Fold 20-Fold Cross Validation for Neural Network

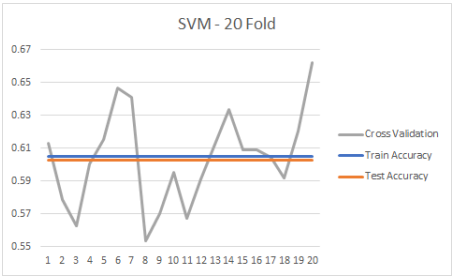

Figure 10: Accuracy per Fold 10-Fold Cross Validation for Support Vector Machine

References

[1] Brownlee, J. (2020, April 12) One-vs-Rest and One-vs-One for Multi-Class Classification. Retrieved from https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

[2] Shkilko, A., Smith, B. F., & McNally, W. J. (2017). Do Brokers of Insiders Tip Other Clients? Informs Management Science, 63(2), http://pubsonline.informs.org/journal/mnsc/

[3] Thomson Routers. (n.d.). US Insider Filing Feed Specification. Retrieved December 12, 2020, from https://wrds-www.wharton.upenn.edu