Ubiquant Market Prediction: A Kaggle Competition

Anfeng Peng
(anfengp2), Department of Computer Science, MCS

Yunchun Pan
(yunchun2), Department of Computer Science, MCS

Yu Bu
(yubu2), Department of Computer Science, MCS

1 INTRODUCTION

Risks and returns differ significantly based on investment types and different factors, and they can cause large fluctuations in financial statuses of investors. As investors have more complicated investment situation nowadays, there is an increasing desire to utilize computer-based algorithms and models to forecast investment returns and provide investors insights of future financial market trend. Hoping to offer investors some insights and help them make scientific investment strategies, we apply our data analysis skills by mining financial datasets and build several machine learning models to estimate investment return rates based on different metrics of investments. The problem to which we are solving is the Kaggle competition Ubiquant Market Prediction.

The original dataset has 18.55 GB. Due to limitation of computing power we only use 10% data for project. Each data sample contains time id, investment id, 300 anonymous scaled data features, and a target investment return rate. We plot the distribution of average and standard deviation of investment returns, as well as explore the correlation between features. The analysis drawn from mining the dataset helps us gain insights of data preprocessing and model development to optimize performance of return forecasting task.

We proceed to implement several baseline models to forecast investment returns. The first two baseline models are Decision Tree Regressor and Light Gradient Boosting Machine (LGBM). Grid search and cross validation are used to tune their hyper-parameters, which are maximum tree depth and minimum number of samples required to split internal node and to be a leaf node. Last baseline model is Multi-layer Perceptron (MLP). Rectified Linear Unit (ReLU) activation function and optimizer Adam is used to optimize training process. All these baselines models are trained on 70% investment data and evaluated on remaing 30% test data by metric of mean squared error (MSE). All three models produce high MSE score on testing set, indicating unsatisfying performance of our models on forecasting investment return rates.

2 PROJECT OUTLINE

Our approach to investment return prediction task demonstrates the following highlights:

(1) Comprehensive data analysis of investment returns to help model selection and training options
(2) Feature selection based on importance during data preposessing
(3) Outlier removal during model training step
(4) Both RMSE metric and correlation metric used for model training to boost overall accuracy
(5) Performance evaluation of 3 different models as well as in-depth analysis of individual models
(6) Evaluation and discussion of the impact of feature selection and the dataset overall

3 DATA MINING

The intuition behind observing the data before building our models is that we could gain a proper understanding of investment features and distributions of returns, thereby selecting better strategies to approach the problem. In addition, the observation of overall data patterns can provide guidance and validation of our model choices.

3.1 Investment Return Rates Distribution

We start with plotting the histogram to observe the general distribution of the investment return rates. The histogram demonstrates approximately symmetric normal distribution of investment return rates with a mean just below 0, indicating that investments in the dataset are generally not very profitable.

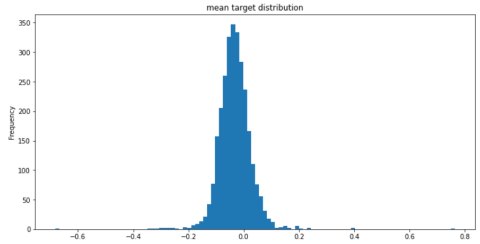

Figure 1: Investment Return Distribution V.S. Investment Count

Next, we plot the distribution of standard deviations of return rates against investment observations. The plot shows a normal distribution with slight right skewness, indicating more investments with low return rates in the dataset than those with high return rates.

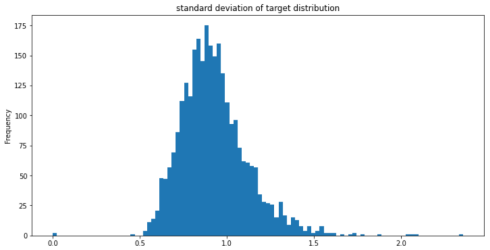

Figure 2: Investment Return std V.S. Investment Count

Then, we plot the joinplot of return rate mean versus investment observations. From the plot, we observe that the mean varies little and the scattered return rates remain mostly the same along the

x-axis, so we conclude that there is little correlation between return rates and investment ids and investment id is not an important feature for prediction task.

Similarly, We create the joinplot of return rate standard deviation

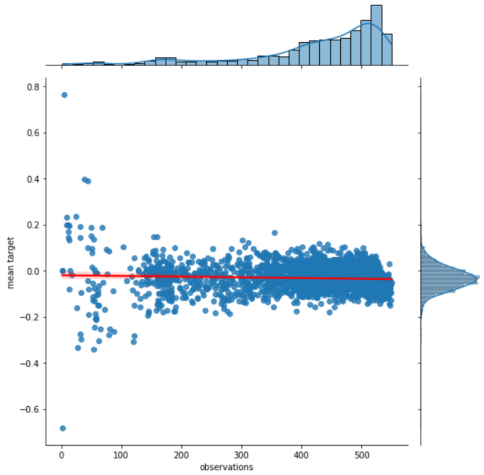

Figure 3: Investment Return Mean V.S. Investment Count

against investment observations. As suggested, standard deviation decreases as the number observation increases, which implies that introducing a new investment sample with appeared investment id can lead model to make a more confident return rate prediction than the one with new id. However, the predicted return rate will likely be closer to its mean than intended, if the introduced observation diverges from others. In addition, the model may lose accuracy if a brand new investment id is introduced.

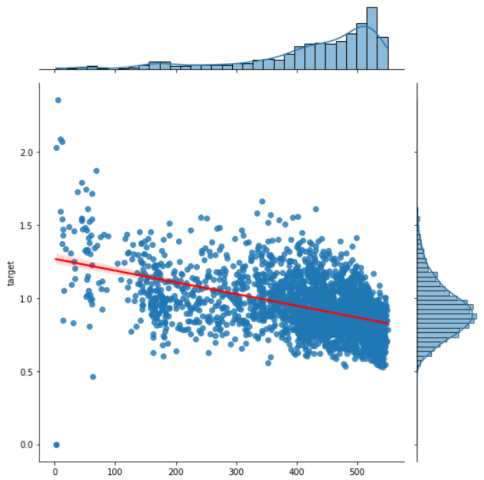

Figure 4: Investment Return std V.S. Investment Observations

On the other hand, it is observed that the distribution of returns grouped by time ids, which share similar overall normal pattern as investment ids, presents noticeable discontinuity and many outliers. Also, standard deviations and average of return rates display no particular pattern against observation counts. Therefore, we decide to exclude time id from training step. Below is a general for a brief showcase of time patterns.

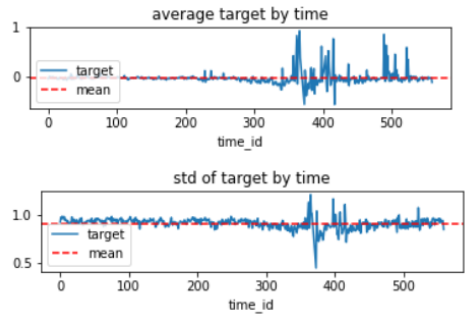

Figure 5: Investment Return std & Mean V.S. Time

Finally, we briefly analyzed the uniqueness of 300 features as indicated in 6 with the nunique() function.

```
train[features].nunique().hist()
```

Listing 1: code listing for unique

The plot shows there are large number of unique investment return values, which suggests a predictive model is more suitable than a clustering model for this task.

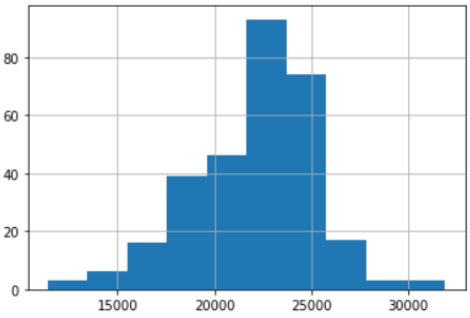

Figure 6: Feature uniqueness histogram

3.2 Feature Correlation

Subsequently, we analyzed 300 investment features. We first measure their correlation with investment return rates by importance score on a scale between 1 and -1. A score above 0 indicates that the feature helps predict return rate, while a score below 0 indicates the feature being not helpful. From the plot, we observe that the maximum importance score is only approximately 0.055, which suggests that these feature are not able to help prediction task much. Among these features, we selected 92 features with score above 0 for training decision tree model, and final performance is then compared to another model version trained by all 300 features.

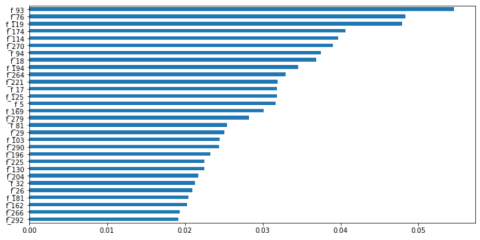

Figure 7: Importance metric - Top 30 features

Furthermore, we evaluated the importance metric with respect to investment ids. We identify 45.55% of selected features with respect to return rate appear in the features with importance scores towards investment ids higher than 0. If the predictive model is given a known investment id, all of these features, even if with negative correlation to target, should be considered during training.

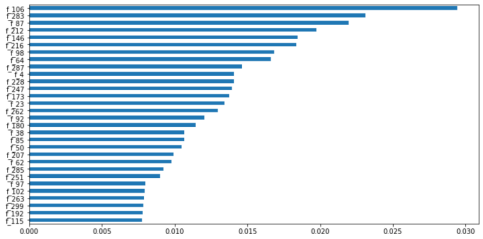

Figure 8: Importance metric - Top 30 features

3.3 Strategy

By observing investment features, we derive our modeling strategy. For LGBM, we drop investment id due to our speculation that the testing data will come with unseen investment ids. Based on our analysis above, such unseen investment ids might lead to instability of prediction. In addition, we drop time id feature because it has little to no impact on the final prediction result, as discussed earlier. Similarly, we choose features with positive importance scores with respect to return rate for training, while manually selecting some features important to the investment ids that are not present in the existing set and does not denote high negativity in the previous metric.

4 EXPERIMENT

4.1 Naive Decision Tree Regressor Approach

4.1.1 Inspiration. This section describes the decision tree[4] approach to make predictions based on provided data. The naive strategy is to implement a simple decision tree regressor. Since our task is to predict unknown investment returns based on implicit decision rules inferred by investment features, decision tree is naturally one of the best choices among prediction models.

4.1.2 Background. Since the task is to predict a continuous target value, a regression tree should be used. During the process, the tree iterates through each feature, attempting to split the data population into 2 smaller nodes based on comparison rules of current feature learned earlier to minimize mean squared error (MSE). Then, in order to provide a prediction with input data, the program also iterates through each node from the root to come up with an expected target value based on possibilities.

4.1.3 Parameters. For decision tree regressor, we consider the following maximum tree depths: 2, 3, 4, 5, 7, 10. Tuning this parameter helps tree model make good number of comparisons between samples to reach good prediction, as well as avoid overfitting happened when an overly deep tree learns many noises. The minimum number of samples required to split an internal node and required to be at a leaf node varies between 2, 3, 5, 7, 10. Tuning these two parameters helps avoid underfitting and overfitting happened when tree model makes predictions based on overly simple or complicated comparison rules.

4.1.4 Training & Model Evaluation. We randomly select 70% data samples as training set and include all other samples as testing set. Following [2], we conduct a grid search with 5-fold cross validation to find the optimal hyper-parameters for MSE metric. Then, we train decision tree regressor with optimal hyper-parameters on the entire training set with all other parameters set to default values. Then, model predicts return values of training and testing set and calculate their MSE.

Below is the pseudo code snippet for training and model evaluation process:

```
for trial in range(num_trials):
    X_train, X_test, y_train, y_test = split_train_test(
     data_X, data_y, test_size = 0.3)
    model = GridSearch(model, model_params, cv = 5,
     scoring = mse)
    model.fit(X_train, y_train)
    train_mse = MSE(model, X_train, y_train)
    test_mse = MSE(model, X_test, y_test)
```

Listing 2: code listing for Decision Tree Regressor

The grid search fits 5 folds for each of 150 combinations of maximum depth, minimum number of samples required to split an internal node and required to be a leaf node, which is in total of 750 fits. Each fits takes approximately 5 seconds to finish.

```
[CV 5/5] END max_depth=2, min_samples_leaf=2, min_samples_split=10; MSE: (test=0.864) total time=   4.6s
[CV 2/5] END max_depth=2, min_samples_leaf=3, min_samples_split=2; MSE: (test=0.867) total time=   5.6s
[CV 4/5] END max_depth=2, min_samples_leaf=3, min_samples_split=2; MSE: (test=0.862) total time=   4.5s
[CV 1/5] END max_depth=2, min_samples_leaf=3, min_samples_split=3; MSE: (test=0.884) total time=   4.7s
[CV 3/5] END max_depth=2, min_samples_leaf=3, min_samples_split=3; MSE: (test=0.861) total time=   4.6s
```

Figure 9: Training result

4.2 LGBM [3] Regressor Approach

4.2.1 LGBM Background. We chose LGBM as second baseline model because it offer faster divergence than other decision tree models while maintaining the same or even better accuracy in prediction. This ideal is achieved by the adoption of leaf-wise tree growth instead of conventional level-wise tree growth, thus reducing more loss in each iteration.

4.2.2 Training. Because the data we use follows highly similar patterns and rules, cross-validation is utilized to help with hyper-parameter tuning. The training program uses time series split to divide the loaded dataset(10% of all data) into 10 batches with training data and validation data. In each training batch, we select random but reasonable values as hyper-parameters. During training process, we utilize L2 norm and correlation metric to drive LGBM towards optimum.

Additionally, after establishing features with positive importance scores, we filtered out the features with negative impact on the target during training while manually keeping some features that maintains crucial relations with investment ids. Based on the two different set of meaningful features, the team realizes the possibility that the fetched testing data could have either historical investment ids or unseen investment id. Therefore two models can be trained here:

(1) Trained with features important to target values to cope with records with unknown investment ids
(2) Trained with investment id and its relevant features to predict target values to cope with records with existing investment ids

Below is the pseudo code snippet for training:

```
features = feature_analysis()
models = [None] * 10
for fold, (train_index, test_index) in enumerate(tscv.
    split(train_data)):
    train, valid = cv_split(train_data)
    lgbm = LGBMRegressor(
        num_leaves=2 ** np.random.randint(3, 8),
        learning_rate = 10 ** (-np.random.uniform(0.1,2))
    ,
        n_estimators = 2000,
        min_child_samples = 1000,
        subsample=np.random.uniform(0.5,1.0),
        subsample_freq=1
    )
    lgbm.fit()
    models[fold] = lgbm
    estimate_accuracy_and_loss()
```

Listing 3: code listing for LGBM Regressor

The training process uses cross-validation to generate 10 different models, each with a different set of hyper-parameters. Then we will run each model against the testing dataset to make aggregated prediction, which is then normalized to obtain average decision.

4.3 MLP Approach

From Achkar's work, we find that mlp and LSTM are good network for Stocks prediction and it should also work for investment prediction.[1]. We want to do the proof of concept first using MLP. This solution is using neural network to make a prediction and it also includes the technology of data mining. It can be considered as a MLP regression network.

4.3.1 MLP Background. The full name of MLP is muti-layer perception.[5] It is a feed-forward neural network, so input data is fed to the MLP network and the loss is the error of prediction. It can be regarded as multiple full connection layers. The circle in the structure diagram is a unit in a layer.

4.3.2 Network Structure. The structure of the MLP is [600, 300, 100, 30, 1]. So it is a 5-layer network in the mlp and we hope that it is enough to do the prediction. Before the last linear layer, we add a normalization and a dropout layer after each dense layer avoid. The activation function that we use is ReLU.

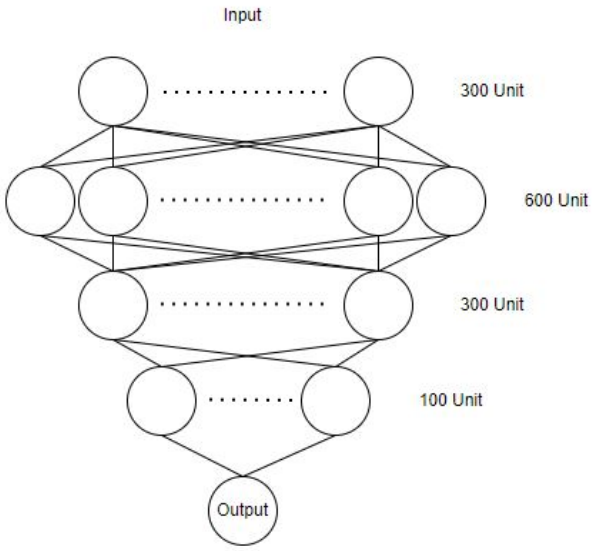

Figure 10: The mlp structure

4.3.3 Coding. We use Keras to build the network. Keras is from TensorFlow and make building network easy. After importing the data set, we first check the NaN value in the dataset and remove them if it contains. Multiple changes are added to the MLP network and all results are shown in the experiment section.

Algorithm 1 MLP Implementation

```
model <- Sequential
for Each hidden layers do
    model adds a Dense layer, set the activation function ReLU
    model adds a Normalization
    model adds a dropout layer
end for
model adds a Dense layer, set the activation function as Linear,
set the output dimension 1.
Compile the model, set the optimizer to Adam, set loss function
to MSE
Train the model
Plot results
```

5 EXPERIMENT RESULT

5.1 Performance Metric

Mean squared error (MSE) is used to evaluate performance of models for training and evaluation steps. The equation for calculating this metric is specified below:

$$MSE = \frac{1}{n}\sum(Y_i - \hat{Y}_i)$$

Where n is the number of investments, Y_i is the true investment return, $\hat{Y}_i$ is the predicted return value.

5.2 Decision Tree Regressor Result

The best Decision Tree Regressor obtained by grid search gives the following evaluation metric results.

DATASET	MODEL	TRIAL	TRAIN_MSE	TEST_MSE
market_price	DT	1	0.742014	0.917683

Figure 11: Selected model

MSE on training set is 0.742 and that on testing set is 0.917, which indicate that predicted investment rate deviates from true rates by approximately 0.9. Since the range of true investment rates is from -1 to 1, these two scores around 0.9 are significantly high and indicate bad performance of decision tree regressor trained with defined parameters.

5.3 LGBM Result

Prediction results were obtained and has shown us great potential. Even though the model is not substantially successful in predicting

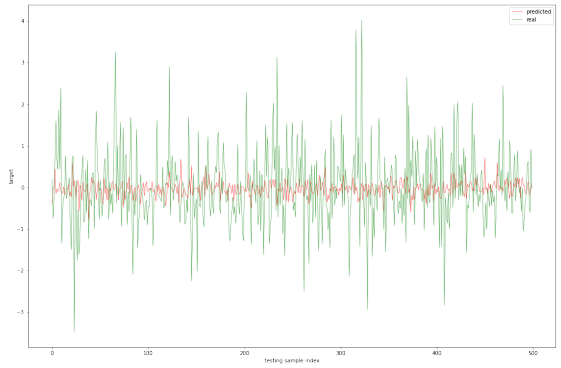

Figure 12: Predicted vs real target

target values, we see a highly similar overall pattern. Noted that our data comes from investment products, the model can be quite meaningful in determining the economical potential and business decisions. Also noted that our results indicate the model tends to be quite conservative in making predictions, thereby showing consistently low std and immunity to outliers.

Furthermore, we studied the main differences between training with selected features and training with all features. The result obtained also validate our expectation: models trained with selected feature were able to obtain, on average 0.5 lower MSE compared to models trained with all features, after 3 runs.

5.4 MLP Result

We have only 1 metric in the evaluation part which is MSE.

(1) MSE. Mean Squre Error

Table 1: MLP Result Table

Models	Basic Partial	Basic Whole	Basic No Outliers
Train MSE	0.6669	0.7744	0.6079
Test MSE	0.8570	0.8158	0.8638

The modal has been trained 50 epochs. Basic Partial model uses 10% data for training and testing, Basic Whole model uses all data and Basic No Outliers model uses 10% data without outliers. We can see that the Basic Whole model has the lowest testing MSE, so its targets are more closed to the labels but their MSE are all pretty high.

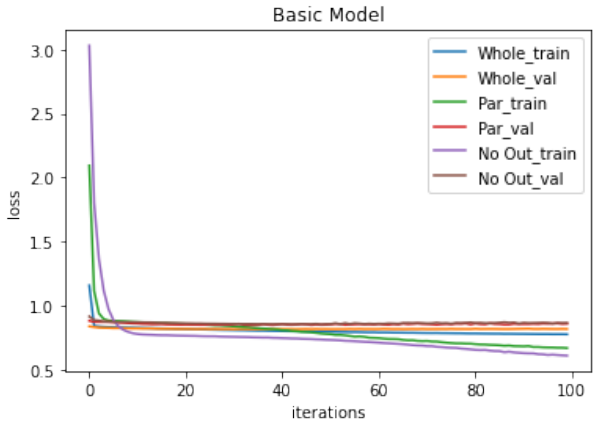

Figure 13: Basic Model Result

6 ABLATION STUDY

We also changes partial of our basic model by removing a layer, removing normalization, using other activation function, normalizing train data, and using other optimizer. The best result is given by removing all normalization layer after each dense layers. Corresponding to this result, using normalization for the train data increases the testing MSE. So for this problem, z-normalization is not a good. Their testing MSE are very closed to each other, but using SGD Optimization gives the worst result. From the ablation study, the small change of model doesn't affect the prediction a lot, so simple MLP network doesn't fit this problem.

Table 2: Ablation Study Result Table

Models	Train MSE	Test MSE
Remove Layer	0.7337	0.8544
Remove norm	0.7276	0.8540
Norm Train	0.6487	0.8600
Other Act	0.7720	0.8595
Other Opt	0.8794	0.8719

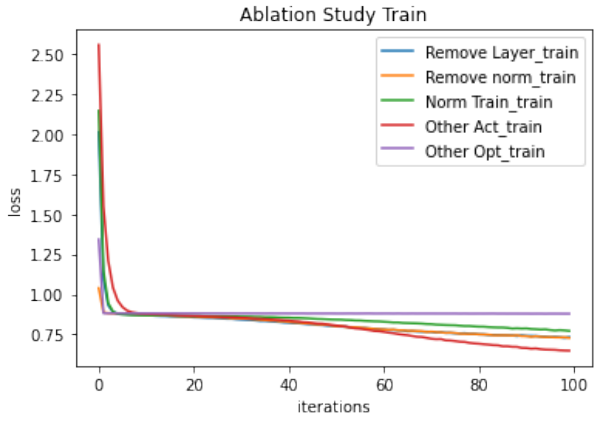

Figure 14: Ablation Result Train

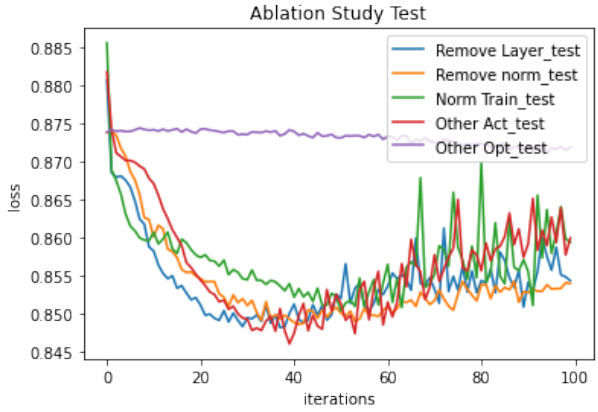

Figure 15: Ablation Result Test

7 FUTURE WORK

7.1 Decision Tree Regressor & LGBM

(1) Since training dataset contains more than 70,000 samples, we might consider using higher value for parameters to tune Decision Tree Regressor.
(2) More trials of grid search can be performed and the average of trail results can be calculated to reduce the chance of randomness.
(3) Due to the size of dataset being more than 18GB, we only utilize 10% data for project. More training samples can be included to improve model's accuracy in the future.
(4) Avoid overfitting within LGBM when the larger dataset is used.

7.2 MLP

(1) From the ablation study, changing part of the current MLP network can not improve the accuracy, or reducing the loss. New research shows that MLP-Mixer can give relevant result as CNN or transformer, so defining a complex MLP network works for this problem.
(2) WE can see that after 40 epochs, the training loss is lower than testing loss and continuing decreases but the testing loss is flat. So the rest of training makes the model over-fitting. Early-stop should be considered for this model.
(3) Using whole data set gives the best result. The ablation study uses 10% data, so using all data for training and doing ablation study can gives a more clear future direction.

8 CONTRIBUTION

Yu Bu: Performed data mining and feature correlation study of dataset. Implemented and improved LGBM model based on MSE and correlation metrics. Completed project outline, data mining, LGBM approach explanation, and LGBM result part of the report.

Yunchun Pan: Implemented and improved decision tree regressor model based on MSE. Completed introduction and decisiont tree regressor approach explanation and result part of the report. Proofread the whole report.

Anfeng Peng: Implemented and improved MLP model and tested a couple different MLP models. Did ablation study based on designed MLP network. Completed MLP model part of the report.

REFERENCES

[1] Roger Achkar, Fady Elias-Sleiman, Hasan Ezzidine, and Nourhane Haidar. 2018. Comparison of BPA-MLP and LSTM-RNN for Stocks Prediction. In *2018 6th International Symposium on Computational and Business Intelligence (ISCBI)*. 48–51. https://doi.org/10.1109/ISCBI.2018.00019
[2] Rich Caruana and Alexandru Niculescu-Mizil. 2006. An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the 23rd International Conference on Machine Learning* (Pittsburgh, Pennsylvania, USA) *(ICML '06)*. Association for Computing Machinery, New York, NY, USA, 161–168. https://doi.org/10.1145/1143844.1143865
[3] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 3149–3157.
[4] Anthony J. Myles, Robert N. Feudale, Yang Liu, Nathaniel Woody, and Steven D. Brown. 2004. An introduction to decision tree modeling. *Journal of Chemometrics* 18 (2004).
[5] Marius-Constantin Popescu, Valentina Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. 2009. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems* 8 (07 2009).