Ubiquant Market Prediction Project Report

Zizheng Lin
CSE
HKUST
zlinai@connect.ust.hk

Duanyi Yao
IOT
HKUST(GZ)
dyao@connect.ust.hk

Abstract

Ubiquant market prediction is a popular data competition in kaggle. Our goal is to predict the market return by using 3,141,410 samples with 304 features and target. We adopt TabNet which is proposed by Google AI to tackle the data. The TabNet can achieve end-to-end learning when input raw tabular data without any processing, which facilitates the prediction with multiple financial features.

We first introduce the background of market prediction and common models. Then we explain the TabNet structure and its functionality. We conduct the experiments by comparing the TabNet with some widely used baselines for Ubiquant task, where the TabNet achieves the best performance. Moreover, we analyze some important hyper-parameters of TabNet and the importance of some features.

1 Introduction

1.1 Background

The prediction of financial market is one of the most important and challenging problems. To attempt to predict returns, there are many computer-based algorithms and models for financial market trading. Big data analytic techniques developed with machine learning algorithms are gaining more attention in various financial application fields, including stock market investment. This is mainly because machine learning algorithms do not require any assumptions about the data and often achieve higher accuracy than econometric and statistical models; for example, artificial neural networks (ANNs), fuzzy systems, and genetic algorithms are driven by multivariate data with no required assumptions. With nonlinear, data-driven, and easy-to-generalize characteristics, multivariate analysis with ANNs has become a dominant and popular analysis tool in finance and economics. (1) and (2) review the use of using ANNs as a forecasting method in different areas of finance and investing, including financial engineering.

ANNs using different deep learning algorithms are categorized as deep neural networks (DNNs), which have been applied to many important fields. Moreover, it is critical for neural networks with different topologies to achieve accurate results with a deliberate selection of input variables.

The most influential and representative inputs can be chosen using mature dimensionality reduction technologies, such as principal component analysis (PCA), and its variants fuzzy robust principal component analysis (FRPCA) and kernel-based principal component analysis (KPCA), among others. PCA is a classical and well-known statistical linear method for extracting the most influential features from a high-dimensional data space.

Besides PCA methods, variants of ensemble decision trees (DTs) are also effective methods for handling multi-dimension dataset in financial prediction. It has some benefits when applied in tabular datasets: (i) they are representionally efficient for decision manifolds with approximately hyperplane boundaries which are common in tabular data; and (ii) they are highly interpretable in their basic

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

form (e.g. by tracking decision nodes) and there are popular post-hoc explainability methods for their ensemble form, e.g. ((3)) (iii) they are fast to train.

TabNet(4) is a new canonical DNN architecture for tabular data, which combines DNNs with DTs. It enable gradient descent based end-to-end learning for tabular data unlike traditional tree-learning. It outperforms traditional DNNs and tree-based learning, which has multiple benefits: (i) efficiently encoding multiple data types like images along with tabular data; (ii) alleviating the need for feature engineering, which is currently a key aspect in tree-based tabular data learning methods; (iii) learning from streaming data and perhaps most importantly (iv) end- to-end models allow representation learning which enables many valuable application scenarios including data-efficient domain adaptation(5), generative modeling(6) and semi-supervised learning(7).

1.2 Data overview

This dataset contains features derived from real historic data from thousands of investments. Our challenge is to predict the value of an obfuscated metric relevant for making trading decisions. This is a forecasting competition, where we need to use the train dataset to predict the return.

The train data set contains following features and target:

1. row id: A unique identifier for the row.
2. time id: The ID code for the time the data was gathered. The time IDs are in order, but the real time between the time IDs is not constant and will likely be shorter for the final private test set than in the training set.
3. investment id: The ID code for an investment. Not all investment have data in all time IDs.
4. $[f_0 : f_{299}]$: Anonymized features generated from market data.
5. target: The target.

1.3 Data analysis

From above information, we counted the number of samples 3141410 and number of Features 304. There are only 1211 unique time_id's. A relatively small number compared to the number of samples. Now let's understand how many entries (samples) we have for each time_id 1.

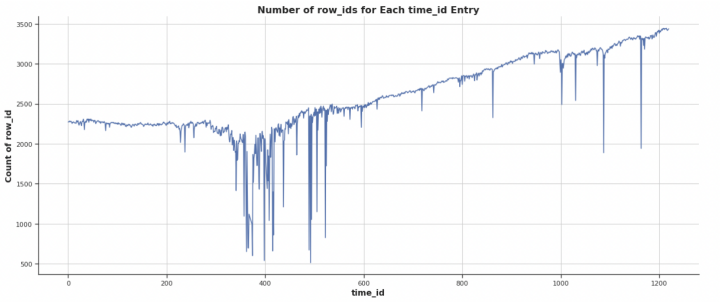

Figure 1: Time-id vs Samples

There are 3579 unique investment_id, each has different entries. 2 shows the relationship between investment_id and the number of entries.

The target is the labels for training and its numerical mean is -0.02109, standard deviation is 0.91971, the minimum is -9.41965 and the maximum is 12.03861.

1.4 Our model

In this competition, we use TabNet(4) which is proposed by GoogleAI for processing the financial tabular data and predicting the return. The input of TabNet is over 300 features and the model can achieve end-to-end learning with sequential attention to choose the most valuable features.

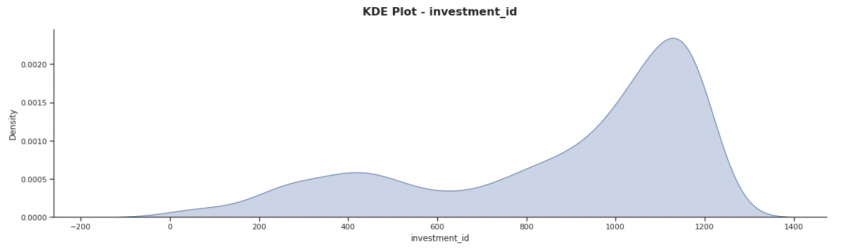

Figure 2: Investment-id vs Samples

2 Model

TabNet was proposed by the researchers at Google Cloud in the year 2019. The idea behind TabNet is to effectively apply deep neural networks on tabular data which still consists of a large portion of users and processed data across various applications such as healthcare, banking, retail, finance, marketing, etc.

2.1 Model structure

The TabNet encoder is composed of a feature transformer, an attentive transformer and feature masking. A split block divides the processed representation to be used by the attentive transformer of the subsequent step as well as for the overall output. For each step, the feature selection mask provides interpretable information about the model's functionality, and the masks can be aggregated to obtain global feature important attribution. The TabNet decoder is composed of a feature transformer block at each step.

3 shows the TabNet architecture for encoding tabular data and 4 shows the decoder of the TabNet.

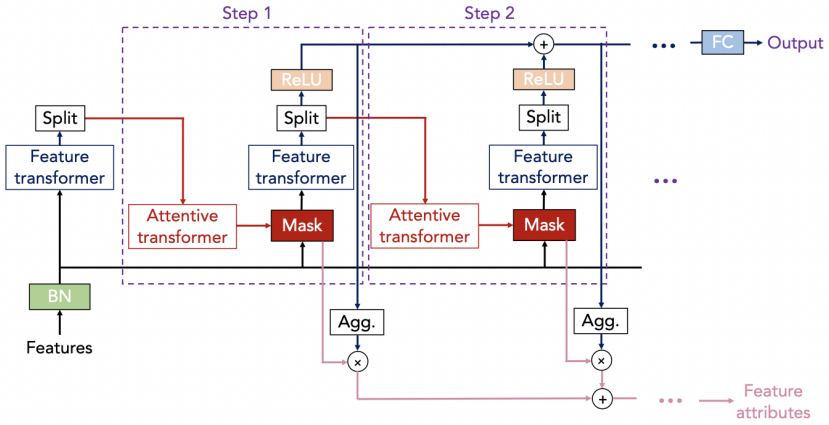

Figure 3: Encoder

2.1.1 Feature transformer

Feature transformer contains 4 layer network, where 2 are shared across all decision steps and 2 are decision step-dependent. Each layer is composed of a fully-connected (FC) layer, BN and GLU nonlinearity.

2.1.2 Attentive transformer

The attentive transformer, the Attentive part of the Encoder, was inspired by the idea of focusing on only some of the data in the input when processing text and images. In the tabular data, it can be seen

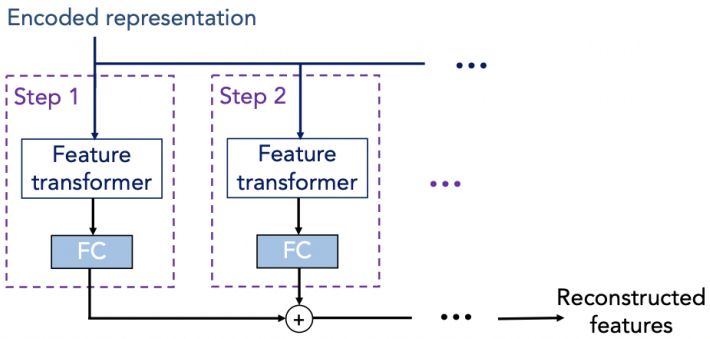

Figure 4: Decoder

as selecting only a few of the dimensions of the data to process at each step, i.e. feature selection, which can simplify the model parameters and learning efficiency.

An attentive transformer consists of a fully connected (FC) layer, a BatchNorm layer, and Prior scales layer, and a Sparsemax layer. It receives input and after passing through the fully connected layer and Batch normalization layer, then it passes through the prior scales layer.

2.1.3 Attentive mask

The output from the attentive transformer step, are then fed into the a attention mask, which it helps in identify the selected features. It quantifies aggregate feature importance in addition to analysis of each step. Combining the masks at different steps requires a coefficient that can weigh the relative importance of each step in the decision.

3 Experiments

In this section, we first describe the settings like data split and evaluation splits in Section 3.1. Then we introduce the baselines used in the experiments to compare with TabNet in Section 3.2. Afterwards, we present and analyze the main results of TabNet and baselines on our data split in Section 3.3. Then we investigate the effect of varying the values of some crucial hyper-parameters of TabNet in Section 3.4. Finally, we carry out feature importance analysis in Section 3.5.

3.1 Settings

Since we are tackling a time series prediction task, it is not reasonable to adopt cross-validation, because the timestamp of data used to train the model should be always earlier than that of data used in evaluation. Thus, we split the dataset based on the timestamp, where the instances whose "time_id" attribute is smaller than 1126 are in the train set while the others are in the test set. In this way, we obtain an approximately $9:1$ train/test ratio. The model is trained on the train set for a certain number of epochs and then tested on the test set to evaluate its performance.

Regarding evaluation metric, we follow the official evaluation scheme which adopts *Pearson correlation coefficient*. Given a sample with n pairs $(x_i, y_i)_{i=1,2,\ldots,n}$, the Pearson correlation coefficient between x and y, denoted as r_{xy}, is computed as follows:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}},$$

where $\overline{x}$ and $\overline{y}$ are the sample mean of the x and y respectively. This metric essentially measures the linear correlation between two samples.

We adopt the following hyper-parameter configuration of TabNet to obtain the main results:

Table 1: Performance comparison between all baselines and TabNet on train/test sets.

Models \ Splits	Train	Test
LR	0.1251 ± 0.0000	0.1093 ± 0.0000
LGBM	0.1401 ± 0.0002	0.1126 ± 0.0002
MLP+CNN Ensemble	0.1436 ± 0.0003	0.1170 ± 0.0001
TabNet	$\mathbf{0.1577} \pm 0.0002$	$\mathbf{0.1281} \pm 0.0001$

1. batch size: 1024
2. width of the decision prediction layer: 8
3. width of the attention embedding for each mask: 8
4. number of decision steps: 3
5. gamma (coefficients for feature reusage in the masks): 1.3
6. number of independent Gated Linear Units layers at each step: 2
7. number of shared Gated Linear Units at each step: 2
8. momentum for batch normalization: 0.02
9. sparsity loss coefficient: 0.001

In addition, we adopt entmax (8) function as the masking function to use for selecting features since it can generalize softmax and sparsemax functions. As for optimization algorithm, we use Adam (9) with 0.0249 learning rate which is adjusted by Cosine Annealing Warm Restarts (10) scheme. All results are obtained by three repetitive experiments under three different random seeds.

3.2 Baselines

The following models are used as baselines for comparison:

Linear Regression (LR) is a canonical algorithm for handling regression tasks in machine learning/statistics. It assumes a linear relationships between the observed values (i.e., features) and the values to be predicted (e.g., target). The coefficients of observed values and the bias are directly estimated via minimizing certain "distance", such as least squares error, between the estimated linear model and the target values in the training set. Even though the linear assumption might be too restrictive for many real-world problems, this model is still widely used by many practitioner due to its simplicity and high interpretability.

Light Gradient Boosting Machine (LGBM) (11) is a efficient and light-scale gradient boosting framework based on decision trees algorithm. It adopts two novel techniques including Gradient-based One Side Sampling and Exclusive Feature Bundling. These two techniques address the limitations of histogram-based algorithm that is primarily used in all previous version of Gradient Boosting Decision Tree models. Briefly speaking, the model first fit a decision tree model on the training data, and then adjust the weight of each instance in the training data based on the prediction results of the previous tree. Afterwards, it fits another new tree on the adjusted data, and makes the prediction using the ensemble of the new tree and the previous tree. The above procedure is done iteratively for a pre-specified number of steps, and the final model is the ensemble of all trees obtained along the training process.

Multi-Layer Perceptron (MLP) + Convolutional Neural Network (CNN) Ensemble is an ensemble model customized for Ubiquant competition. Essentially, it is constructed by aggregating the predictions from four MLP models and a one-dimensional CNN model. Such an ensemble enables the model to effectively capture the hidden relationships between features and the target values. The aggregation scheme is taking the average over the normalized prediction values of all models, where the normalization follows the Gaussian normalization.

Figure 5: Scores when taking different number of decision steps

3.3 Main Results

The performance of all baselines and TabNet on train/test sets is shown in Table 1, where we have the following observations and analyses:

1. TabNet outperforms all baselines on both train and test set. The absolute performance gains on train set range from 0.0141 to 0.0326. On test set, the absolute performance gains range from 0.0111 to 0.01888. Such performance gains demonstrate that TabNet is indeed more powerful than the baselines in tackling tabular datasets.
2. The standard deviation of the TabNet is small compared to the LGBM and MLP+CNN baselines, which shows that TabNet is very stable with respect to different random seeds, making it a suitable solution to various real-life challenges involving tabular datasets.
3. Without using any ensemble technique, TabNet still outperforms the ensemble baseline (MLP+CNN Ensemble). This shows that TabNet is a desirable choice for solving various learning problems about tabular datasets since it is already powerful when being used as a single model, saving tremendous efforts of building complicated ensemble models. However, to further improve the performance of TabNet, ensemble with other models such as LGBM or MLP might be a good choice, which we don't investigate due to the time limitation.

3.4 Hyper-Parameter Analysis

In this section, we study the effects of varying the values of the following two crucial hyper-parameters of TabNet: number of decision steps and sparsity regularization coefficients.

3.4.1 Number of Decision Steps

This hyper-parameter specifies how many times the TabNet would process the features in the encoder and decoder. We investigate the effect of this parameter by checking its performance when taking the values in $\{2, 3, 4, 5, 6\}$. The mean scores of all configurations on the test set are shown in Figure 5. From the results, we have the following observations:

1. The scores range from 0.1097 to 0.1281. When taking 3 decision steps in encoder and decoder, TabNet achieves the optimal performance. The performance degrades when taking less or more decision steps.
2. The performance drops monotonically when increasing the number of decision steps starting from 3, and the performance in 6 steps is even worse than in 2 steps. This indicates that Taking too many decision steps might cause the model to overfit the dataset, which prevents the model from effectively capturing the characteristics of the dataset.

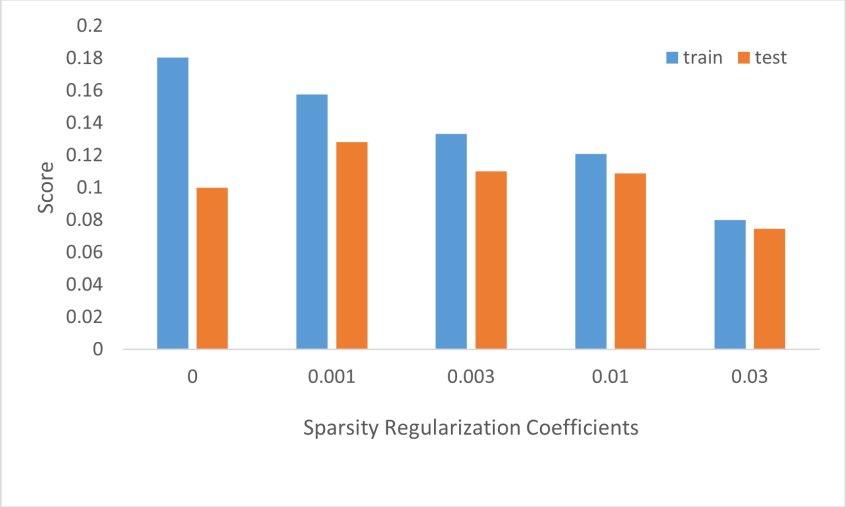

Figure 6: Scores when taking different sparsity regularization coefficients.

Table 2: Performance of instances removing different features on test sets.

Features	Test Score
Full features	**0.1281**
- feature no.17	0.1217
- feature no.102	0.1270
- feature no.284	0.1192

3.4.2 Sparsity Regularization Coefficient

Sparsity Regularization Coefficient controls the strength of the sparsity loss which influences the sparsity of the feature selection procedure. In order to investigate the regularization effects, we record the results of TabNet on both train and test sets, where the coefficient takes values in $\{0, 0.001, 0.003, 0.01, 0.03\}$. The results are shown in Figure 6. From the figure, we have the following observations:

1. The instance with the coefficient taking the value of 0.001 achieves the optimal generalization capability. The performance on the train set degrades monotonically as the the value of the coefficient increases, which is reasonable since the higher the coefficient, the sparser the model is, which means it becomes more difficult to the model to fit the distribution of the train set.
2. The gaps between the train scores and the corresponding test scores range from 0.0055 to 0.0802, where the value decreases monotonically as the coefficient increases. This phenomenon is reasonable since the higher the coefficient, the strength of regularization becomes higher, which means the overfitting issue is less severe. However, when the value of the coefficient becomes too high, the model become too sparse to capture the characteristics of the dataset, which also harms the performance of the model during generalizing to test set.

3.5 Feature Importance Analysis

We randomly select three features to drop from the train set, and then test the model which has been trained on the new train set on the test set to inspect the importance of each feature. The mean scores of all configurations are shown in Table 2. From the table, we have the following observations:

1. All three features contribute positively to the performance of the model, where the feature no.284 has the greatest significance among the three features, whereas the feature no.102 has the smallest significance.

2. For features without significant influence on the model performance, it might be helpful to remove those features or aggregate them into one single feature to reduce both time and memory consumption.

4 Conclusion

In this project report, we first introduce the background of the Ubiquant task and some common models. Then we present a brief overview and thorough analysis of the dataset. Afterwards, we explain the model (i.e., TabNet) we adopt to tackle the task, starting from its overall structure to each of its crucial components. Then we conduct the experiments by comparing the TabNet with some widely used baselines for Ubiquant task, where the TabNet achieves the best performance. Moreover, we analyze some important hyper-parameters of TabNet and the importance of some features.

References

[1] G. Zhang, B. Eddy Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[2] A.-P. Refenes, A. N. Burgess, and Y. Bentz, "Neural networks in financial engineering: A study in methodology," *IEEE transactions on Neural networks*, vol. 8, no. 6, pp. 1222–1267, 1997.

[3] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," *arXiv preprint arXiv:1802.03888*, 2018.

[4] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," *CoRR*, vol. abs/1908.07442, 2019.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[6] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[7] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. Salakhutdinov, "Good semi-supervised learning that requires a bad GAN," *CoRR*, vol. abs/1705.09783, 2017.

[8] B. Peters, V. Niculae, and A. F. Martins, "Sparse sequence-to-sequence models," in *Proc. ACL*, 2019.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[10] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with warm restarts," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[11] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

5 Group Work Contribution

1. Zizheng Lin was responsible for 50% of the total workload of the project, including (1) conducting the experiments of comparing TabNet with baselines, hyper-parameter analysis and feature analysis; (2) writing the Experiment section and Conclusion Section of the report; (3) Delivering the presentation and making the presentation video;
2. Duanyi Yao was responsible for 50% of the total workload of the project, including (1) conducting the experiments of exploratory data analysis; (2) Writing the Introduction and Model section of the report; (3) Making the presentation slides.