

TIME SERIES PROJECT

RANDOM FOREST PREDICTION

Shufan Zhang

Linglan Wang

Zhen Guo

ABSTRACT

1. INTRODUCTION

Nowadays, Machine Learning method plays a key roles to forecasting the future value and trends. Machine learning could precisely predict trends or result in a expansive range, as well as richer specifications of functional form. Over recent decades, machine learning algorithms have achieved great success in diverse domain. The advantage of ML algorithms is cross-learning, so the prepossessing of time series data needs to be thought over well to ensure it applied to the two canonical problems successfully: predicting returns in the cross section and time series. What's more, people always like to add decision trees in the Machine Learning model as decision tree could provide effective ways to analyze fully possible consequence of a certain decision. Therefore, in this project, we use Simple Decision Tree and Random Forest Decision Tree to predict the price trend and calculate its cumulative return.

2. METHODOLOGY

Review of Decision Tree Classifier and Random Forest Classifier

Decision tree is a tree structure where a internal node represents feature, a branch denotes the way a decision is made, and a leaf represents an output. The root or the topmost node creates partitions based on the value of features. It partitions the tree recursively, which is called recursive partitioning. The decision tree is a parameter-free method which does not depend on the probability distribution assumptions of the data. It can handle high dimensional data with high accuracy. The steps of a decision tree are stated as follows:

- Apply Attribute Selection Measures (ASM) to split the data by choosing the best feature/attribute.
- Create a decision node using the best attribute and split the data into smaller subsets.
- Repeat the above process recursively for each subset until one of the followings are satisfied:
 - All subsets belong to the same attribute value
 - All attributes are taken as decision nodes

Attribute Selection Measures (ASM) selects the best possible criterion to partition the data on each internal node. It assigns a rank to each feature by computing their scores with a particular measure. Then the feature with the highest score is selected perform the partition. Most common measures include Information Gain, Gain Ratio and Gini Index. Take Information Gain as an example.

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N} I(D_{left}) - \frac{N_{right}}{N} I(D_{right}) \quad (1)$$

where:

f = feature split on

D_p = data of the parent node

D_{left} = data of the left child node

D_{right} = data of the right child node

I = impurity criterion

N = total number of data points

N_{left} = number of data points on the left child node

N_{right} = number of data points on the right child node

A common impurity criterion used in practice is entropy, which is defined as:

$$I = - \sum_{j=1}^c p_j \log_2(p_j) \quad (2)$$

where:

p_j = proportion of data points that belongs to class j for a node

* $p_j = 0$ if all data on that node belongs to the same class

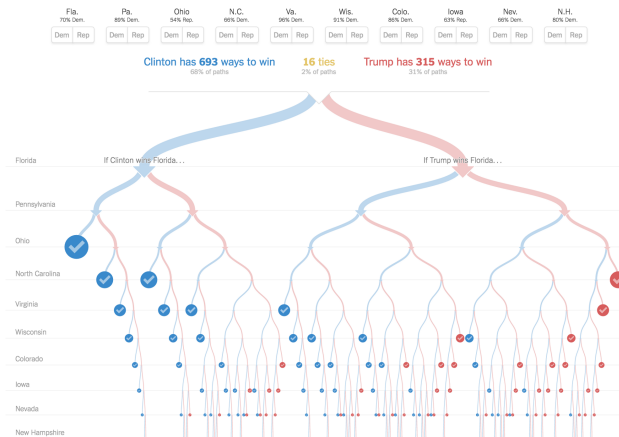


Figure 1: Apply decision tree on presidential election prediction

Random forest classification applies a supervised learning algorithm to partition. It prevents overfitting by developing decision trees on random subsets. The algorithms works in 4 steps:

- Select random samples from the dataset
- Construct a decision tree for each random sample and record the result generated by each decision tree
- Each decision tree model votes for the results
- The classification with the most votes become the final decision

Model

We construct a decision tree classifier and a random forest classifier. For both classifiers, the models focus on predicting the returns of stocks k days from now. In the study, k is set to be 5. For each day, the 5-day-return (NEXTRET) is defined as $\frac{PRC_{5d}}{PRC} - 1$, where PRC_{5d} denotes the closing price of the stock 5 days in the future, PRC denotes closing price today. Then we partition NEXTRET into three categories with an upper threshold and a lower threshold. If the NEXTRET is greater than the upper threshold, it is assigned to class I. If NEXTRET is less than the lower threshold, it is assigned to class II. Otherwise, it is put into class III. Values for class I, II and III are 1, 0, -1 respectively.

Then we fit a decision tree classifier and a random forest classifier using the training set. The decision tree classifier uses a balanced class weight, which uses the values of dependent

variable to automatically adjust weights inversely proportional to class frequencies in the input data. Gini impurity is employed as impurity criterion. Our random forest consists of 10 decision trees, each with Gini impurity. The number of features to consider when making partition is the square root of number of available features. The maximum depth of a tree is set to be 6. And it also uses a balanced class weight.

After the models are trained, they are tested on the testing set. The performances are evaluated by a variable called error, which is defined as the percentage of testing samples on which the models gives the wrong classification.

Train/Test Set

In order to do the validation of our model, we separate our data as training dataset, which used to fit the model, and test dataset, which used to provide an unbiased evaluation of a final model fit on the training dataset. We select data from previous 130 trading days as Train set and the last 40 trading days as Test set. We select the photoshots of

7	x_train
---	---------

1]:

	BIDLO	ASKHI	RET	TRN	OPENPRC	sprtrn	PRCMAXGAP	PRCMINGAP	PRC
400	147.616209	148.959920	-0.000451	0.004138	148.556807	0.003538	0.064658	0.129186	148.739168
401	148.691178	150.188455	0.008969	0.002194	149.113487	0.002824	0.055193	0.136928	150.073280
402	148.844745	150.169269	-0.007419	0.002286	150.034898	-0.000262	0.063080	0.130477	148.959920
403	147.405055	149.157455	-0.007990	0.003372	148.624002	-0.001442	0.071642	0.123474	147.769786
404	146.397272	147.453045	-0.005651	0.002077	147.251488	-0.007114	0.077732	0.118492	146.934756
...
695	158.590000	161.370000	-0.027945	0.004779	161.320010	-0.015561	0.124700	0.213025	158.620000
696	158.750000	160.670000	0.007691	0.002361	160.630000	0.009105	0.116116	0.219032	159.840000
697	159.848010	162.250000	0.007070	0.002521	159.860000	0.006416	0.108281	0.224514	160.970000
698	162.740010	165.350010	0.015904	0.003157	163.009990	0.010939	0.090931	0.236654	163.530000
699	162.509990	164.217800	-0.003363	0.001588	163.620000	-0.001387	0.094613	0.234078	162.980000

300 rows x 9 columns

Figure 2: X train dataset

```
In [35]: 1 upthreshold=0.03
2 downthreshold=0.03
3 y = 0 + (NEXTRET > upthreshold) - (NEXTRET < -downthreshold)
4 y_train = y[train_range[0]:train_range[1]]
5 y_train
```

```
Out[35]: 400    0
401    0
402    0
403    0
404    0
..
695    1
696    0
697    1
698    0
699    1
Name: PRC, Length: 300, dtype: int64
```

Figure 3: Y Train dataset

```
In [45]: 1 test_range=[700, 740]
2 x_test = x.iloc[test_range[0]:test_range[1], :]
3 x_test.head()
```

```
Out[45]:
```

	BIDLO	ASKHI	RET	TRN	OPENPRC	sprtrn	PRCMAXGAP	PRCMINGAP	PRC
700	163.12000	164.82001	0.004050	0.003916	163.52000	0.009956	0.090198	0.237167	163.64000
701	162.67999	164.34000	-0.000061	0.004813	163.61000	-0.002000	0.090265	0.237120	163.63000
702	165.59000	168.99001	0.023773	0.004995	166.30000	0.002763	0.064947	0.254835	167.52000
703	162.42999	167.47000	-0.011640	0.006146	167.14999	-0.003919	0.077490	0.246059	165.57001
704	166.57370	169.24001	0.020354	0.003753	166.97000	0.006872	0.055996	0.261099	168.94000

Figure 4: X Test Dataset

```
In [50]: 1 test_range=[700, 740]
2 y_test = y[test_range[0]:test_range[1]]
3 y_test.head()
```

```
Out[50]: 700    1
701    1
702    0
703    1
704    0
Name: PRC, dtype: int64
```


Figure 5: Y Test Dataset

Application – Trading Strategy

We perform a trading strategy based on the two models developed on the testing data. For every stock we choose, if the NEXTRET today is classified in class I, buy the stock today and sell it after 5 days. If the NEXTRET is classified in class II, sell the stock today and buy it back after 5 days. For NEXTRET in class III, no trade will be made on that day. The daily returns as well as accumulated returns of each stock during the testing period are recorded.

3. DATA

This study aims to predict stock prices and returns on a daily basis, thus the inputs of the model need to be daily data relevant to the stock in order to get the most reliable results. Most financial ratios of a company whose stock we are interest in predicting are either in monthly, quarterly or annual basis. Approximating the daily data by interpolating the monthly, quarterly and annual data will leads to significant inaccuracy, since the dynamics of those financial ratios such as Earnings per Share, Price-Earnings, Debt-Equity are unknown to us.



		PRC	next_5D_MTM
PERMNO	date		
10145	20160104	102.57	-0.040460
	20160105	103.41	-0.041002
	20160106	102.24	-0.050274
	20160107	99.23	-0.004938
	20160108	98.43	-0.011887
	20160111	98.42	-0.011075
	20160112	99.17	-0.029545
	20160113	97.10	-0.005355
	20160114	98.74	-0.009317
	20160115	97.26	-0.008020

Figure 6: Next 5D MTM

To build a model with accurate real-time data, a set of daily information of stock prices are selected to train the model. All data come from the CRSP database provided by Wharton

Research Data Services (WRDS). Here is the list of variables used by this research:

- Closing Price (PRC)
 - If Closing Price is not available, take the average of Closing Bid and Closing Ask as Closing.
- Share Volume (VOL)
 - Total number of shares of a stock sold on day I. It is expressed in units of one share.
- Price Open (OPENPRC)
- Ask or High (ASKHI)
 - The highest trading price during the day, or the closing ask price on days when the closing price is not available.
- Bid or Low (BIDLO)
 - The lowest trading price during the day, or the closing bid price on days when the closing price is not available.
- Closing Bid (BID)
- Closing Ask (ASK)
- Holding Period Return (RET)
 - The return for a sale on day I. It is based on a purchase on the most recent time previous to I when the security had a valid price. Usually, this time is I - 1.
- Number of Shares Outstanding (SHROUT)
- Return on the Standard & Poor's Composite Index (SPRTRN)
 - $(\text{SPRTRN}(t) / \text{SPRTRN}(t-1)) - 1$
- Cumulative Factor to Adjust Price (CFACPR)
 - Factors adjust prices for splits events
- Cumulative Factor to Adjust Shares Outstanding (CFACSHR)

- Factors adjust shares for split events

The period of data is from 01/04/2016 to 12/31/2019, which contains 1006 trading days. 300 of the S&P 500 component stocks are selected for this research, and 295 stocks remain after stocks with invalid data are filtered out.

The first step of data processing is adjusting all price and share data for splits events such as stock splits and stock dividends so that the shares and prices at different data are directly comparable. For prices, divide raw data by the Cumulative Factor to Adjust Shares Outstanding. For shares, multiply raw data by the Cumulative Factor to Adjust Shares Outstanding. After adjusting prices and shares data, three new variables are introduced based on the following computations:

- Share Turnover Ratio (TRN)
 - Share Volume / Number of Shares Outstanding
- Distance to Maximum Ratio (MAXGAP)
 - For stock price each day, given by $[\max(\text{previous 252 prices}) - \text{current price}] / \text{current price}$.
- Distance to Minimum Ratio (MINGAP)
 - For each data point of stock price, given by $[\min(\text{previous 252 price}) - \text{current price}] / \text{current price}$

The final step of data processing is to perform standardization. 'PRC', 'ASKHI', 'BIDLO', 'OPENPRC', 'RET', 'TRN', 'SPRTRN' are standardized according to the following method: For each data point at current time t , compute the average and standard deviation of data in previous 252 trading days. Subtract each day's value by the corresponding average, and divide the difference by the corresponding standard deviation. It is worth noting that only data starting from the 253th trading day can be standardized by this routine. Therefore, we only use the rest 754 days' data in training and testing the model.

4. EMPIRICAL RESULT

Error

The mean error for the simple decision tree is **0.418**. The median for the simple decision tree is **0.425**, the maximum for the simple decision tree is **0.875**, and the minimum is **0**. The first quantile of the error of simple decision tree is **0.325**, and the third quantile **0.525**. The standard deviation of the error for simple decision tree is **0.177**.

The mean error for the random forest is **0.401**. The median error for the random forest is **0.375**, the maximum for the random forest is **1**, and the minimum is **0**. The first quantile of the error for the random forest is **0.20**, and the third quantile is **0.55**. The standard deviation of the error for the random forest is **0.230**.



Figure 7: Error for simple decision tree

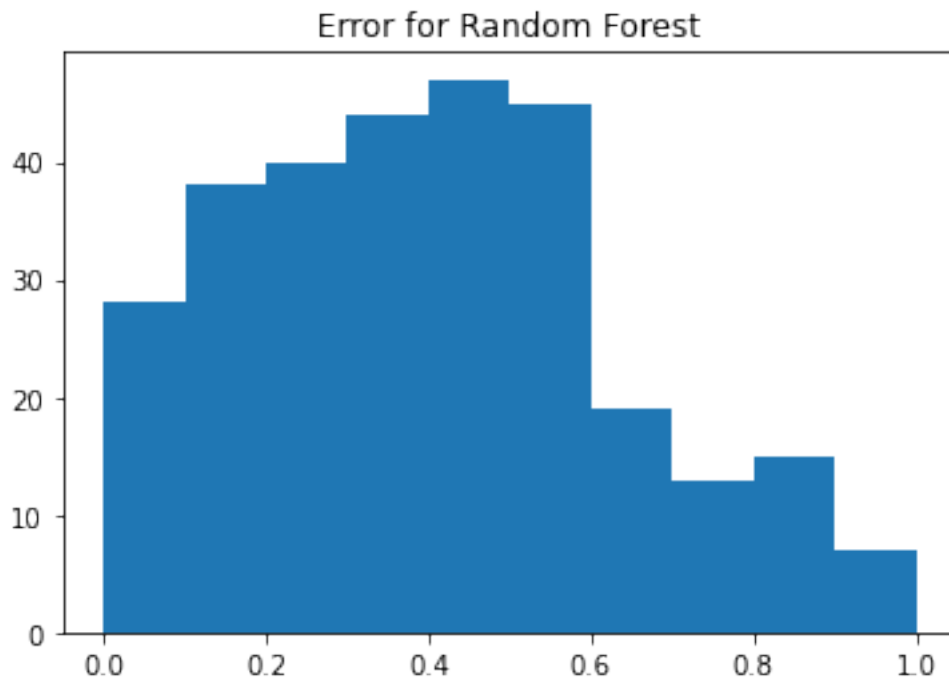


Figure 8: Error for Random Forest

Accumulative Return

The mean of accumulative return for simple decision tree is **-0.020**. The median of accumulative return for simple decision tree is **0.0**. The first quantile of accumulative return for simple decision tree is **-0.129**, and the third quantile is **0.119**. The Maximum of accumulative return is **1.033**. The Min value of accumulative return is **-1.087**. The standard deviation of accumulative return is **0.273**.

The mean of accumulative return for random forest is **-0.017**. The median of accumulative return for random forest is **-0.058**. The first quantile for accumulative return of random forest is **-0.100**, and the third quantile is **0.034**. The maximum for accumulative return of random forest is **0.750**. The min value for accumulative return of random forest is **-1.613**. The standard deviation for accumulative return of random forest is **0.276**.

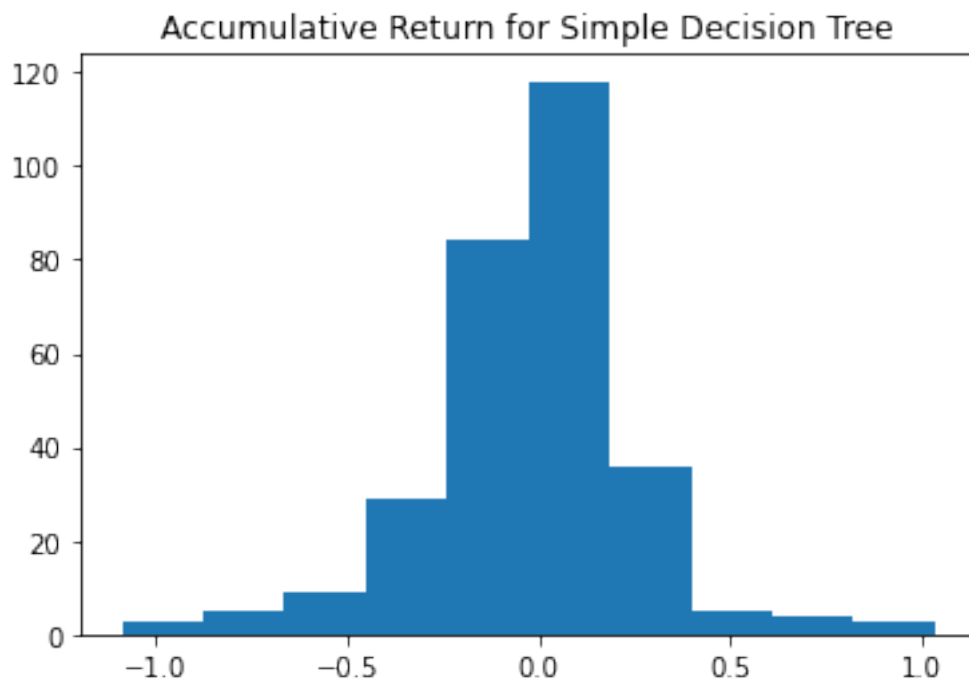


Figure 9: accumulative return for simple decision tree

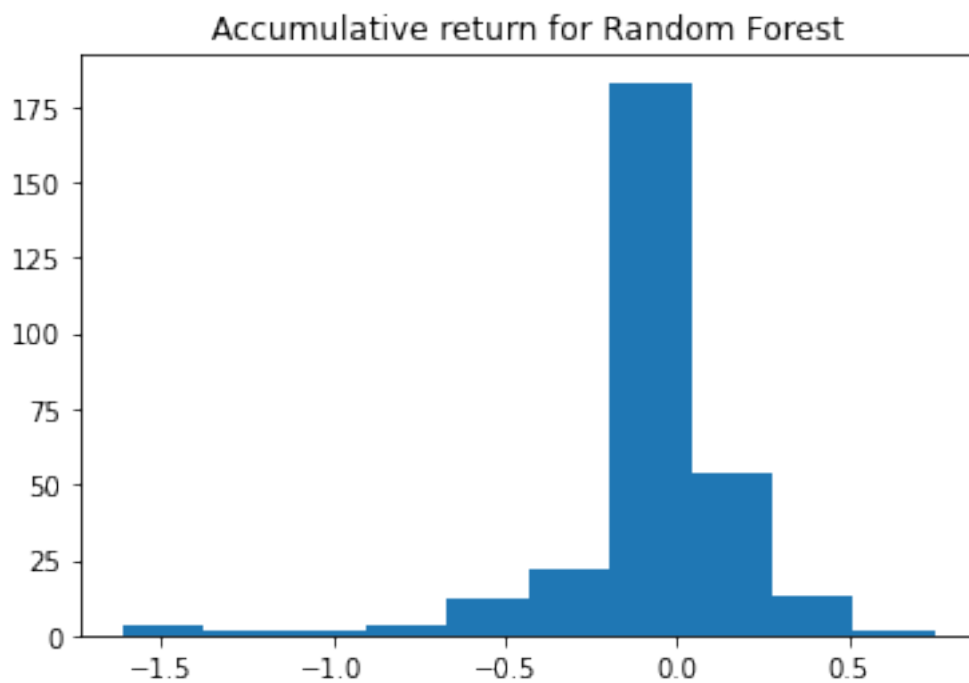


Figure 10: Accumulative Return for Random Forest

5. CONCLUSION

The trading strategy under both the decision tree model and the random forest model generate some good accumulated returns on the 296 stocks. The trading strategy applying random forest gives a maximum return of 124.52% over the testing period, while the strategy using decision tree gives a maximum return of 103.33%. On average, however, the strategy's performances are not satisfactory. The strategy under either classifier has a negative average accumulated return. There are several possible future improvement of our models and the trading strategy. A direction of improvement might be finding more predictors with significant predictive power on future stock returns. This model might also be improved by balancing out the data before constructing decision trees and random forest. This may include re-sampling and clustering the classes with too many samples. Moreover, other trading strategies may be used to take advantage of the forecasting model.

Bibliography

- [1] Decision Tree Classification in Python. www.datacamp.com/community/tutorials/decision-tree-classification-python.
- [2] Rahul. "Learn ML Algorithms by Coding: Decision Trees - Lethal Brains." *Medium*, 10 Aug. 2018, lethalbrains.com/learn-ml-algorithms-by-coding-decision-trees-439ac503c9a4.