Analyzing Financial Indicators of US stocks and Predict the Future Trends

Yongxuan Zhang 40084728
Haitun Liao 40080732
Anqi Wang 40057695
Bin Xue 40059809

Introduction

The US Stock market is a hot-debated topic with competitions between companies nowadays.

This project leverages the algorithms and techniques of big data learned in class, put into practice by analyzing real-world data: US Stocks between 2014 - 2018.

This presentation will show:

- Machine learning data analytic techniques to reveal the trend of datasets.
- The workflow of our data analysis
- The experiments setup and conclusions

Objectives

- We formulate a **binary classification** problem, that is we aim to provide stock gain prediction via machine learning methods and help to make decisions on purchase.
- Compare the prediction performance between Scikit-learn and Spark MLlib.
- Compare the running time performance between Scikit-learn and Spark MLlib.

Data source

- This data repository[1] contains five datasets: Financial Indicators of US stocks from 2014 to 2018 csv files
- Each dataset contains 200+ financial indicators, that are commonly found in the 10-K filings releases yearly by publicly-traded companies.
- There are approximately 4000 data samples in each dataset.

Technology We Used

Technology:

- Scikit-learn.
    - Scikit-learn is a popular free software machine learning library for the Python programming language
    - It contains plenty of built-in algorithms and can be easily implemented with user-friendly APIs.
- MLlib
    - MLlib is Apache Spark's scalable machine learning library
    - It is usable in different languages and platforms

Algorithms

We select the following algorithms which can be found in the aforementioned two libraries.

- **Logistic regression:** This works with binary classification problems. Through data, it can learn the coefficients of a logistic regression model
- **Naive Bayes:** This classifier belongs to probabilistic classifiers. It can differentiate independent assumptions or different objects between certain features.
- **Linear Support Vector Machine:** This is a linear model and able to solve linear and non-linear problems. It works for classification and regression practical problems.
- **Decision tree classifier:** This uses Gini impurity and “entropy” for the information gain to measure the quality of a split at each node. It contains the greedy algorithm to find locally optimal decisions.
- **Random forest classifier:** This algorithm calculates the average of a number of decision tree classifiers, which are based on various sub-dataset. It uses the average to enhance predictive accuracy and adjust over-fitting.
- **Gradient-boosted tree classifier:** This classifier works in groups that combining many weak learning models, use these to create a strong predictive model by using machine learning algorithms
- **Multilayer perceptron classifier:** This model learns a function by training on a dataset. It optimizes the log-loss function through a supervised learning algorithm. Each element corresponding to the number of neurons in its related hidden layer.

Data Analysis Procedure

Multiple steps will be carried out in our project:

1) Data Preprocessing: The raw data will be preprocessed before feeding into models. Including handling missing values, data oversample and feature normalization.
2) Model Tuning: We will conduct grid search to tune the hyperparameters of each model with 3-fold validation.
3) Results comparison: We will report and analyze the performance of different algorithms via several metrics including accuracy, recall, precision, and f1 score.

Data preprocessing

- Handle missing values
    - We use 0 value to fill the missing values and drop samples with a percentage of missing value larger than 50%.
- Normalization
    - Features will be normalize into [-1,1].
- Oversample
    - Oversample makes correctness of prediction increase(see results of 2017)

Experiment Setup

To validate the proposed objective. We set up three cases of experiments as following:

- **Case1:** Compare the performance between scikit-learn and Spark on data within the same year and next year;
- **Case2:** Compare the performance with and without oversampling on scikit-learn;
- **Case3:** Run the models on each dataset for 5 times and take the averaged running time to compare the running time between scikit-learn and Spark.

Results-Case 1

Comparison of sklearn and spark(same year)

Scikit-learn	2014	2015	2016	2017	2018	Averaged exclude 2014 and 2017
LR	49.53%	80.44%	79.63%	11.41%	81.82%	80.63%
NB	59.88%	80.19%	14.50%	44.09%	21.85%	38.85%
SVM	45.83%	80.70%	80.75%	7.36%	82.64%	81.37%
CART	31.19%	80.74%	81.26%	0.00%	80.19%	80.73%
RF	50.87%	82.89%	80.28%	26.16%	82.34%	**81.84%**
GBDT	45.61%	82.38%	81.01%	11.11%	81.28%	81.56%
MLP	47.12%	80.06%	72.03%	30.36%	79.06%	77.05%

Spark	2014	2015	2016	2017	2018	Averaged exclude 2014 and 2017
LR	0.00%	83.52%	80.12%	0.00%	82.52%	**82.05%**
NB	1.14%	83.53%	80.07%	0.00%	82.11%	81.91%
SVM	3.88%	83.33%	80.10%	0.68%	82.33%	81.92%
CART	47.45%	82.87%	80.12%	28.69%	78.79%	80.59%
RF	47.01%	84.54%	79.20%	22.79%	82.21%	81.99%
GBDT	48.01%	83.48%	76.65%	23.08%	80.12%	80.09%
MLP	33.40%	83.26%	80.10%	0.00%	81.30%	81.55%

- For case 1, we first split the data within one year to training and testing set to validate the data quality. The split rate is 4:1.
- We use 3-fold validation to do the parameters search.
- We report the F1 score considering the data imbalance. The total results shall be attached as appendix.

Results-Case 1

Comparison of sklearn and spark(different year)

Scikit-learn next year	2014-2015	**2015-2016**	2016-2017	2017-2018
LR	41.49%	**78.84%**	43.38%	9.01%
NB	80.70%	**78.79%**	10.79%	79.78%
SVM	34.47%	**78.18%**	43.30%	0.65%
CART	24.06%	**78.33%**	43.29%	0.00%
RF	24.06%	**77.96%**	42.58%	0.00%
GBDT	33.18%	**79.07%**	43.29%	3.41%
MLP	43.27%	**80.30%**	38.53%	40.61%

Spark next year	2014-2015	**2015-2016**	2016-2017	2017-2018
LR	0.00%	**80.30%**	43.29%	0.07%
NB	3.58%	**80.30%**	43.31%	39.59%
SVM	3.05%	**80.33%**	43.23%	0.07%
CART	54.38%	**74.63%**	41.00%	39.59%
RF	41.56%	**78.67%**	43.73%	14.61%
GBDT	32.44%	**77.13%**	42.58%	23.95%
MLP	19.61%	**80.25%**	43.15%	0.00%

- For this case , we use the data of one year to train and the data of next year to test.
- Because data of 2014 and 2017 is bad data, we only see the results of 2015-2016.

Results-Case 2

Comparison of row data and data after oversample(same year)

Year	Positive	Negative	Positive Percentage
2014	1634	3808	30.03%
2015	2891	4120	41.24%
2016	3218	4797	40.15%
2017	1370	4960	21.64%
2018	3046	4392	40.95%

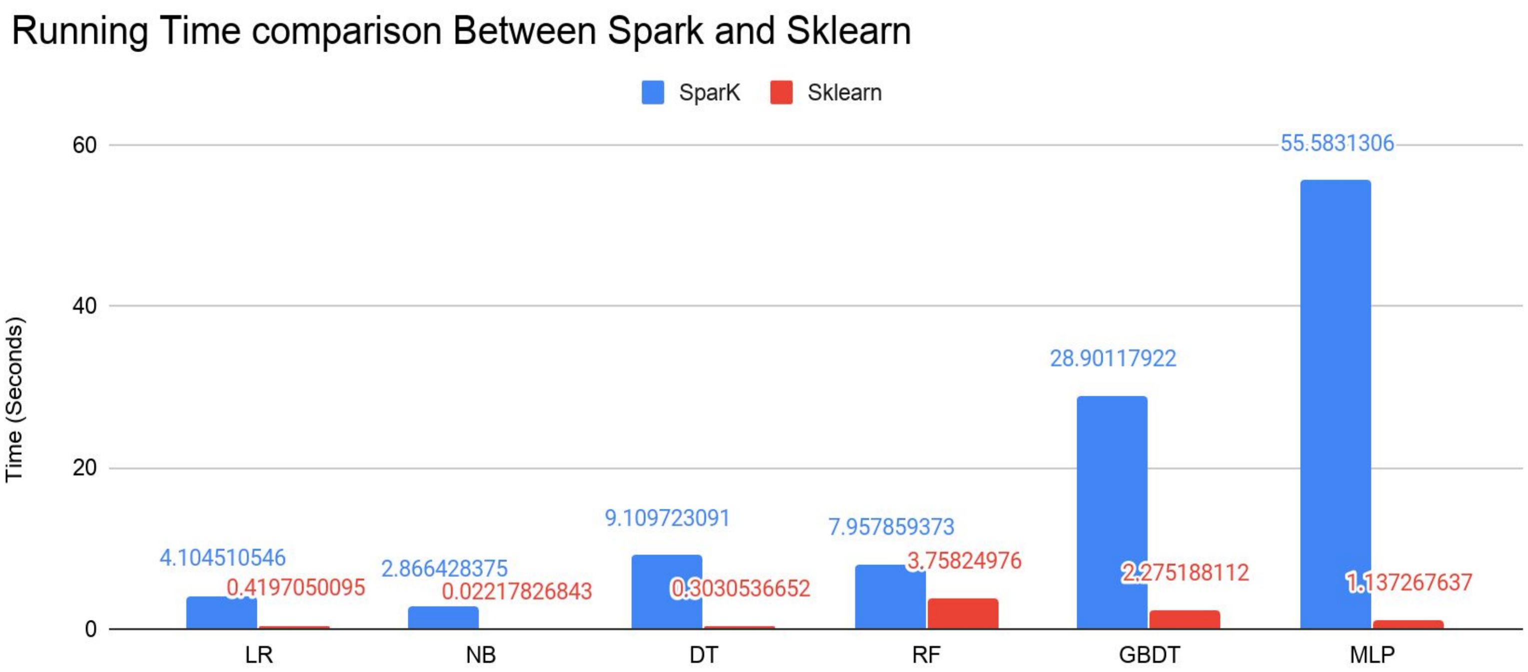
Oversampling Results:

Scikit-learn	2014	2015	2016	2017	2018
LR	55.60%	70.86%	68.71%	39.73%	71.82%
NB	59.75%	79.84%	14.73%	44.37%	22.81%
SVM	5.95%	80.09%	80.80%	11.73%	82.62%
CART	53.78%	75.66%	72.84%	35.80%	77.09%
RF	56.75%	80.85%	76.31%	35.87%	80.10%
GBDT	57.57%	81.04%	78.55%	36.73%	81.77%
MLP	53.41%	76.34%	71.96%	38.72%	74.58%

Oversampling Improvements:

	Imbalance	oversample	Improvement
2014	47.37%	56.14%	8.77%
2015	81.12%	77.43%	-3.68%
2016	68.12%	63.85%	-4.27%
2017	20.52%	38.54%	18.02%
2018	71.09%	68.03%	-3.06%

Results-Case 3



Conclusions

- From the results of Case 1 we can find out that:
  - The prediction performance between scikit-learn and Spark is small on the selected dataset.
  - When the training data is of poor quality, it is hard to obtain high accuracy to predict the future year.
- From the results of Case 2 we can see that the oversampling method has positive effect on highly imbalanced datasets(year 2014 and 2017).
- From the results of Case 3 we can conclude that scikit-learn is faster than Spark regarding the training time on our dataset.

Conclusions

- Predict a stock is worth buying or not is hard

Q&A

Thank You

Reference

[1] Project and data reference from site:
https://www.kaggle.com/cnic92/200-financial-indicators-of-us-stocks-20142018/data

[2] Apache Spark. “Machine Learning Library (MLlib).” *The Apache Software Foundation,* spark.apache.org/docs/1.1.1/mllib-guide.html.

[3] Patel, Jigar, et al. "Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques." *Expert systems with applications* 42.1 (2015): 259-268.

[4] Chatzis, Sotirios P., et al. "Forecasting stock market crisis events using deep and statistical machine learning techniques." *Expert Systems with Applications* 112 (2018): 353-371.