Final Project: Stock Performance Prediction by Machine Learning Methods

Application Track **4 Team Members**

Abstract

This project predicts stock performance, either stock price or trend, based on three machine learning algorithms: Support Vector Regression (SVR), Long short-term memory (LSTM) and Random Forest (RF). We construct our own dataset by combining calculated 12 technical factors and fundamental factors of 7 stocks, and pre-process the data before applying them to specific models by normalizing and dimension reduction techniques. We test the performance of SVR and LSTM by root mean squared error (RMSE) while RF by accuracy. After the experiments, SVR model performs better than LSTM in predicting stock price. RF model shows relatively high accuracy in predicting the price trend and the performance closely relates to choice of class and decision path.

1 Motivations

Quantitative Finance, widely recognized as "rocket science of the Wall Street", has transformed the modern financial industry. This is an area of finance in which sophisticated mathematical models are used to predict the markets and hedge risks. There are two branches in the Quantitative Finance, which are P-quant and Q-quant. Q-quant is where people utilize stochastic calculus to approximate the theoretical risk-neutral fair prices. P-quant, on the other hand, tries to harness the power of statistical tools and machine learning algorithms to estimate the prices and trends in the real probability world.

This project focuses on the P-quant branch and includes three advanced and popular algorithms in this field, namely, Support Vector Regression (SVR), Long short-term memory recurrent neural network (LSTM) and random forest (RF). Our project follows the application track and aims to predict the future stock prices and returns of seven S&P 500 listed companies from different industries, including Apple, Coke Cola, Hewlett-Packard, Kroger, McDonald's, Microsoft and Royal Caribbean. Since these industries have significantly different trading momentum and financial factors, we want to investigate the performances of predictions among them.

Machine learning has been harnessed in financial industry for a half century since James Simons founded Renaissance Technologies. These above mentioned algorithms showed great adaptability to high-volatile, noise-saturated time-series data since they were born. Scholars, quant researchers and financial engineers still use them till today but with different feature spaces. However, majority only focuses on technical perspective and their researches can only explain market-related information, i.e., trading prices and volumes. Our vital innovation is blending data from underlying corporations' financial statements into our feature space; thus, the intrinsic value of the stock also reflects in our prediction.

2 Problem Statement

Our project works on implementing basic machine learning models into stock predictions. We select 7 stocks covering different industries. The tricky and interesting parts are calculating the technical factors, combining technical factors and fundamental factors as feature space, and determining the

simulating models. The input matrix is in the form of $[V, t_1, ..., t_i, f_1, ..., f_j]$, where V is the trading Volume, t_i is the columns of daily technical factors and f_j is the according fundamental factors. The output is a column of labels for RF method or close prices for SVR and LSTM methods. Finally, we manage to train the different models with proper hyper-parameters, which can help us to formulate good fitting curves and prediction results.

The technical factors we choose are included in Table 1. The adjustment to the whole 106 fundamental factors will also be explained later.

Table 1: Technical Factors

Technical factors	Equation
Moving Average(MA)	$\sum_{i=0}^{t-1} \frac{C_{t-i}}{n}$
Exponential Moving Average(EMA)	$C_t \times \lambda + EMA_{t-1} \times (1-\lambda)$
Relative Strength Index(RSI)	$100 - \frac{100}{1+(\sum_{i=0}^{n-1} UP_{t-i}/n)/(\sum_{i=0}^{n-1} DW_{t-i}/n)}$
Accumulation Distribution Oscillator (ADO)	$\frac{H_t - C_{t-1}}{H_t - Lt}$
Moving Average Convergence Divergence (MACD)	$MACD(n)_{t-1} + \frac{2}{n+1} * (DIFF_t - MACD(n)_{t-1})$
Average True Range(ATR)	$\frac{1}{n}\sum_{i=1}^{n} Max_i((HL), Abs(HC_p), Abs(LC_p)]$
Momentum	$C_t - C_{t-i}$
Williams R% (W%R)	$\frac{H_n - C_t}{H_n - Ln} \times 100$
Stochastic Oscillator	$\frac{100 \times (C_t - LL_{t-(n-1))}}{HH_{t-(n-1)} - LL_{t-(n-1)}}$
Commodity Channel Index (CCI)	$\frac{M_t - SM_t}{0.015 * D_t}$

C_t is the closing price, λ is the weight, UP_t means upward price change while DW_t is the downward price change at time t, L_t is the low price and H_t the high price at time t, $DIFF_t = EMA(12)_t - EMA(26)_t$, LL_t and HH_t implies lowest low and highest high in the last t days, respectively. $M = \frac{H_t + L_t + C_t}{3}$, $SM_t = \frac{\sum_{i=1}^{n} M_{t-i+1}}{n}$, $D_t = \frac{\sum_{i=1}^{n} (M_{t-i+1} - SM_t)}{n}$,

3 Related Work

People have tried to predict the performance of stock and trend of market for years and applied various machine learning methods on it. A fundamental problem we concern about is which machine learning method should be chosen.

For regression method, Bruno (2018) points out that SVR method could have significant prediction error on daily stock price. One of Bruno (2018)'s target is to minimize these errors by choosing optimized hyper-parameters and kernel functions. To select the optimal parameters, Ling-Jing Kao (2013) proposes the trial-and-error (also called cross-validation) method. The method performs well but it's very time-consuming and data-intensive. Cherkassky and Ma (2004) come up with an analytic parameter selection method. The analytic method is based on sketching the structure of training data to determine the best value of parameters, which works well in their experiment. The grid search proposed by Hsu et al. (2009) also performs well in their study. The grid search is a straightforward method using exponentially growing sequences to identify good parameters. In this project, we will combine the advantages of both methods.

The choice of kernel function determines the performance of SVR model. Sheng-Hsun Hsu (2009) proposes the polynomial kernel in their study because it tends to achieve better performance in the experiments. Ye and Huang (2011) uses the radial basis function (RBF) as the kernel function because of its capabilities and simple implementation. In our project, we will test on linear kernel, RBF and polynomial kernel in SVR.

For LSTM, lots of applications has been implemented in modeling financial time-series data. Fischer(2018) predicted returns of portfolios constructed by S&P 500 stocks and achieved approximate patterns like the actual world. Cao (2019) combined CEEMDAN signal decomposition algorithm with LSTM model, which achieved satisfying performance. People always use ARIMA to compare

with LSTM. Recently, Althelaya (2018) forecasted economic and financial time series by ARIMA and LSTM and found LSTM achieved 80% higher accuracy than ARIMA on average. Thus, we decided to use LSTM as one of our prediction methods.

For classifier method, Patel(2014) use random forest to forecast stock trend with weekly data, and get prediction with enough accuracy. There are many other papers using classifiers to forecast stock performance and trend. Nevertheless, few of them has the same background and goal with us. Our goal is to use classifier to classify daily return rate to more than two intervals. We finally choose Random Forest method because it is highly interpretable.

4 Methodologies applied

4.1 Data preprocessing

We first manually calculate the results of technical factors for each day. And then, we match the seasonal fundamental factors (106 of them in total) to each daily technical factor and combine the daily prices and volume data to construct our own dataset. Moreover, we carefully test some parameters like lag periods for some technical factors, and finally set it differently in order to maximize the impacts of different factors. As for fundamental seasonal data, there are some collinearities exist. For example, all of the components of the DuPont Equation (An accounting equation) are included and they are perfectly linear correlated. Therefore, we choose several methods to standardized the data, and test the performances of each methods in every model, trying to figure out the best method for financial data standardization.

The first method is standardizing all the technical and fundamental factors. The transform equation is as below:

$$\widetilde{x_{ni}} = \frac{x_{ni} - \overline{x}_i}{\sigma_i} \tag{1}$$

The purpose of this step is to rescale the variables and make them have zero mean and unit variance.However, We cannot eliminate the relationships between variables, which can be shown by the covariance matrix with contents $\rho_{ij} = \frac{1}{N}\sum_{n=1}^{N} \widetilde{x_{ni}}\widetilde{x_{nj}}$. However, by using PCA we can make a more substantial normalization to offer zero mean and unit variance. What's more, different variables become decorrelated in this way. Suppose the dimention of the feature space is D. From the class, we know that the general solution to the minimization of the objective function is choosing the eigenvectors of the covariance matrix, which is given by:

$$\mathbf{S}u_i = \lambda_i u_i \tag{2}$$

where $\mathbf{S}$ is the covariance matrix, and u_i is the eigenvector according to the eigenvalue λ_i. We can rewrite above equation in the form:

$$\mathbf{SU} = \mathbf{UL} \tag{3}$$

where $\mathbf{L}$ is a $D \times D$ diagonal matrix with elements λ_i, and $\mathbf{U}$ is a $D \times D$ orthogonal matrix with columns given by u_i. Then we can transform the original data points x_n into y_n:

$$y_n = \mathbf{L}^{-\frac{1}{2}}\mathbf{U}^T(x_n - \overline{x}) \tag{4}$$

We can prove that the attributes within the transformed data points are decorrelated by calculating the covariance matrix:

$$\frac{1}{N}\sum_{n=1}^{N} y_n y_n^T = \frac{1}{N}\sum_{n=1}^{N} \mathbf{L}^{-\frac{1}{2}}\mathbf{U}^T(x_n - \overline{x})(x_n - \overline{x})^T\mathbf{U}\mathbf{L}^{-\frac{1}{2}} \tag{5}$$

$$= \mathbf{L}^{-\frac{1}{2}}\mathbf{U}^T\mathbf{S}\mathbf{U}\mathbf{L}^{-\frac{1}{2}} \tag{6}$$

$$= \mathbf{L}^{-\frac{1}{2}}\mathbf{L}\mathbf{L}^{-\frac{1}{2}} = \mathbf{I} \tag{7}$$

We can find that in this way the covariance matrix is an identity matrix, which means that now the attributes are not correlated to each other and has been normalized. We apply PCA to standardize the fundamental factors and combine them with the original technical factors in order to remain the volatility. Moreover, the 106 fundamental factors seem too much compared to technical factors, so we also use PCA to lower the dimension of the fundamental factors to the same level of technical factors.

In summary, we transform the data into three forms:

1. Both technical and fundamental factors are standardized.

2. Just normalized fundamental factors by PCA and combine with the original technical factors.

3. Use PCA to lower the dimension of fundamental factors (to 12 which is the dimension of the technical factors) and then combine with the original technical factors.

We use these different versions of combined data to train our models, and see which way of processing data fits the price curve or reflect the change mode of the stock best. After doing that, we test the model and compare the effects between different models.

4.2 Support Vector Regression

SVM uses $(\mathbf{x}_k, y_k)_{k=1}^N$ training observations to build a linear model, mapping variables on a greater dimensions. Separation between classifications is achieved by an optimal hyperplane, which is calculated based on N observations, where $\mathbf{x}$ is the independent variable vector, and y is classification $y_k \in (0, 1)$ for each sample. Thus, the classification hyper-plane is given by:

$$f(\mathbf{x}) = \mathbf{w}\mathbf{x}^T + b \tag{8}$$

where $\mathbf{x}, \mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n) \in R^n$ and $b \in R$ respectively means the input vector, the weight vector and the intercept. The optimization problem for training the linear SVR is given by:

$$min \frac{1}{2} \|\mathbf{w}\|^2 + c \sum_{k=1}^{N} (\xi_k + \xi_k^*) \tag{9}$$

Subject to,

$$y_k - \mathbf{w}\mathbf{x}_k^T - b \leq \varepsilon + \xi_k \; and \; \mathbf{w}\mathbf{x}_k^T + b - y_k \leq \varepsilon + \xi_k^* \tag{10}$$

where c is the penalty for incorrect estimation, $\varepsilon > 0$ is the regularization factor that weights the trade-off between the estimation value and the observation value, and ξ and ξ^* are slack variables, and $k = 1, ..., N$. After solving this optimization problem, the SVR seeks to solve the following nonlinear regression problems:

$$f(\mathbf{x}) = \mathbf{w}\varphi(\mathbf{x})^T + b \; = \; \sum_{k=1}^{N} (\alpha_k - \alpha_k^*) \varphi(\mathbf{x}_k) \varphi(\mathbf{x}_k)^T + b \tag{11}$$

where $\varphi(\mathbf{x})$ denotes a mapping function that maps the input vector $\mathbf{x}$ into a higher dimensional feature space, and where α and α^* are the Lagrange multipliers. The inner product of functions $\varphi(\mathbf{x})$ and $\varphi(\mathbf{x})^T$ can be replaced by a kernel function $K(\cdot)$. Thus, the general form of the SVR is given by:

$$f(\mathbf{x}) = \sum_{k=1}^{N} (\alpha_k - \alpha_k^*) K(\mathbf{x}, \mathbf{x}_k) + b \tag{12}$$

The common forms of kernel functions used by SVR in stock prediction is linear, radial and polynomial functions, respectively given by:

$$K(x_i, x_j) = x_i^T x_j \tag{13}$$

$$K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2} \tag{14}$$

$$K(x_i, x_j) = (x_i^T x_j + 1)^d \tag{15}$$

The shape of kernel function will directly influences the values generated by the SVR. Also, the constant c in Eq.(2) and d in Eqs.(7) and (8) should be optimised.

4.3 Long Short-Term Memory Recurrent Neural Network

A single Cell in the recurrent neural network can be imagined as several duplicates of traditional ones but they connect to each other and each one pass information to a successor. Alongside the time,

past information would not be useful in explaining and predicting the future, even worse, function as noise.

Long Short-Term Memory is a special kind of recurrent neural network designed to avoid long-term dependencies problem. It is majorly used to model time-series data. In our project, trends of financial data are nearly perfect modeled by this technique. The vital part to LSTM is Cell State, shown in figure one, denoted by C_t at time t. h_t is our output prediction for each Cell at time t.

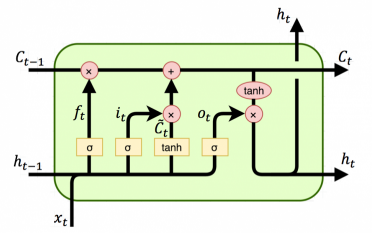

Figure 1: LSTM Cell

There are 4 major steps in LSTM. The first step is called "forget gate layer", which determines what past information should be kept or eliminated. It is denoted by f_t, and defined in equation 17. W_f is the weight and b_f is bias term.

$$f_t = \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right) \tag{16}$$

The second step is determined by i_t, defined in equation 18 and $\tilde{C}$ defined in defined in equation 19. This step is decided which information can pass on to the next Cell State.

$$i_t = \sigma\left(W_i\left[h_{t-1}, x_t\right] + b_i\right) \tag{17}$$

$$\tilde{C}_t = \tanh\left(W_C\left[h_{t-1}, x_t\right] + b_C\right) \tag{18}$$

The third step is to generate new Cell State, and the process is described as equation 20.

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t \tag{19}$$

The fourth step, which is the last, is to generate new hidden layers to filter the output of previous hidden layer h_{t-1} and new input x_t. This step is described in equation 21 and 22.

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{20}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{21}$$

Our network's structure contains 4 LSTM layers and each one contains 100 above Cells. We also add 20% dropout regularization layer following each LSTM layer, which means dropout layer will randomly sets input units to 0 with a frequency of 20% to avoid over-fitting problems. The feature space is past-60-day close prices and corresponding financial and technical indicators the output space is close price of the next day. The network is trained by ADAM algorithm.

4.4 Random Forest Classifier for Return Rate Interval Prediction

Random forest classifier (RF) is an ensemble method based on decision tree classifier. When training the model, we feed data randomly to train 100 decision trees with greedy growth method and optimize the weights among trees. The prediction is made by weighted majority vote of different trees. By doing so, the risk of overfitting could be significantly reduced. Another reason for using random forest classifier is that RF classifier is a highly interpretable classifier.

We find the version 1 and 2 data with normalization or scaling don't fit in RF method, because such normalization dos not help the RF model perform better but make the result harder to interpret. We compute the daily return rate by close price of stock, cut the interval into n+1 pieces with even length, and label them from 0 to n.

Tuning parameters is the most challenging and time-consuming part when apply RF classifier in this problem. When tuning the parameters, we export the decision trees with highest five weights and check if their decision path is reasonable. By tuning the split strategy, depth, and minimum samples in single nodes, we cut down branches with problem and significantly improved the accuracy of our model. We find that by setting maximum depth to 15, minimum samples in single nodes to 15, most models get satisfied outcome.

5 Evaluation

Our data is from 7 stocks distributed in different industries. We retrieve the 20 years'(Dec.2000 - Sep.2020) stock prices from Yahoo! Finance and quarterly fundamental data from FactSet. The raw data contains 106 fundamental factors and we calculate 12 technical factors from the price data. After applying different versions of pre-processed data into different models, we decide to use version 3 data, whose fundamental factors' dimension is lowered by PCA. The reason is that this method reduce the weight of fundamental factors, even though it loses some interpretability.

Root Mean Squared Error(RMSE) are used to measure the performance of SVR and LTSM. The RMSE will be calculated by:

$$RMSE = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(d_i - \hat{d}_i)^2} \tag{22}$$

, where T means total test sample, d denotes real sample value and $\hat{d}$ is the predicted value.
Accuracy will measure the performance of RF model. Computation of accuracy requires True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN). These parameters are defined as:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{23}$$

The dataset will be divided into training data and testing data. Training data will take 90-percent of the dataset. We will predict 1-day based on testing data. SVR and LTSM will predict the stock price by regression techniques. RF will predict the trend of next day using classification techniques.

5.1 Support Vector Regression

According to the Section 3, we will optimize the kernel function, penalty constant c, the degree d using the analytic parameter selection method and the grid search. According to the grid search, c will take the value of 2^{-1}, 2, 2^3, ... ,2^9 and d will take 2 and 3. The kernel function with the parameter set of C and d which generate the minimum forecasting RMSE will be considered the best setting for SVR. After putting the training dataset in the model, we can predict the 7 stocks' price from 2018-10 to 2019-10. Applying RMSE model in predicted price and actual price, we will get RMSEs for each kernel function related with each set of parameters. The RMSE results with linear kernel, rbf kernel and polynomial kernel are respectively in table 1, table 2, table3 (Appendix). The bold RMSE means the stock's lowest RMSE in the table.

As we can see, in radial kernel function, the lowest RMSEs scatter with respect to c and they do not have a consensus to the best c. In linear kernel, there are 5 stocks have the lowest RMSE when c equals to 2^-1, while AAPL and MSFT have the lowest RMSE when c is 2^9. In the table of polynomial kernel function, the lowest RMSEs concentrate when c is 2^{-1}. After comparing the RMSE in each kernel function, we compare the RSMEs among kernel functions. It can be seen that linear kernel with c equals to 2^{-1} has the lowest RMSEs in all stocks including AAPL and MSFT whose RMSEs are not the optimal when c is 2^{-1}. This is because the RMSEs of AAPL and MSFT have trivial difference between c is 2^{-1} and their optimal c.

For now, we can conclude that the linear kernel with c being 2^{-1} is the best setting. After getting the optimal pair, we will use them to predict the price using SVR and finally we will compare SVR's RMSEs with LSTM's. The RMSE can be seen in table 2.

Table 2: SVR Prediction Performance

Company	AAPL	COKE	HP	KR	MCD	MSFT	RCL
RMSE	2.42188	11.2643	2.06266	0.8107	4.72045	4.12	4.72209

5.2 Long Short-Term Memory Recurrent Neural Network

According to Methodologies section, we used the past-60-day close prices and corresponding financial and technical indicators to predict the next-day close price for abovementioned 7 companies. We

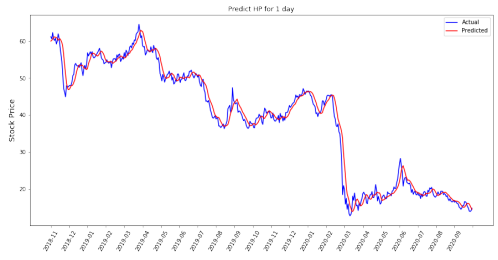

Figure 2: Predict HP Close Price by SVR

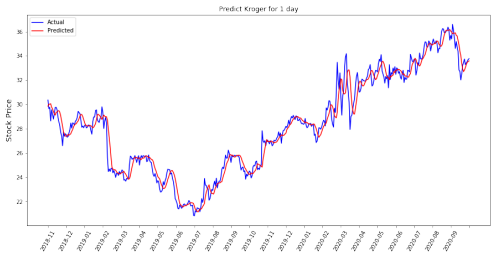

Figure 3: Predict Kroger Close Price by SVR

utilized stock prices and underlying corporations' financial information ranging from December 2000 to October 2018 as our training set, which including 4500 data points, to avoid over-fitting. Our test set is the corresponding data ranging from November 2018 to September 2020, including approximate 500 data points. The table 3 shows the performance of our LSTM on prediction of stock's close prices by RMSE.

Table 3: LSTM Prediction Performance

Company	AAPL	COKE	HP	KR	MCD	MSFT	RCL
RMSE	10.2794	28.4109	5.4385	1.3689	18.7583	18.0003	26.0261

According to the table, RMSE difference between stocks and industries are not statistically significant considering the approximate 500 data points in our test size. Following figures show LSTM 1-day close price prediction curves of Hewlett-Packard and Kroger. Other prediction graphs are in the Appendix.

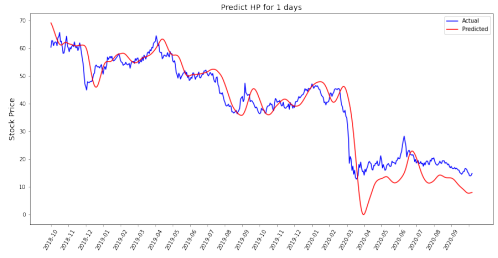

Figure 4: Predict HP Close Price by LSTM

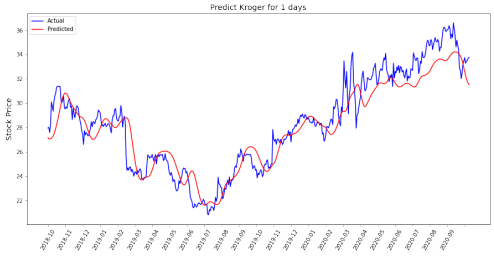

Figure 5: Predict Kroger Close Price by LSTM

5.3 Random Forest Classifier

The performances of some models are listed below.

Table 4: RF interval prediction

Model Num.	End of interval	Length of interval	Amount of classes	Accuracy
1	0.96,1	0.01	6	0.5687
2	0.96,1	0.02	4	0.584
3	0.96,1	0.04	3	0.5879
4	0.96,1.04	0.02	6	0.4008
5	0.96,1.04	0.04	4	0.5763
6	1,1.04	0.01	6	0.5573
7	1,1.04	0.02	4	0.5458
8	1,1.04	0.04	3	0.5763

Generally, decrease the length of interval or increase the number of classes would reduce model's accuracy. We found that RF method hardly make precise prediction when the number of classed exceeds 6 or the length of each interval is less than 0.05.

RF classifier has tendency to follow the prior probability of each classes in the train set. It is not overfitting, but we still need to check models' decision path to make sure the model does not label all observations to majority class. Evenly separate classes may help, but it is not always useful to manipulate cut points of interval for two reasons. First, we may set threshold for certain purpose, such as to identify the 'oversell' and 'overbuy' interval. At that time, we do not want to change the threshold. Second, the cut points may affect model's performance. Select an improper threshold may dramatically reduce model's accuracy.

The selection of cut points could affect model's accuracy. If the threshold separate observations with similar features, the accuracy of model would get extremely bad. Intuitively, that means these similar observations randomly distribute in an interval. Trying to separate that interval and classify observations into different class is overfitting. One way to identify these intervals is to find if there is large amount of data concentrate in a short interval compared to its neighbor.

Figure 6: Model 5 Decision Path Overview (detailed version in Appendix (Zoom In required))

6 Conclusion

Our project proves that the combination of the technical indicator and fundamental factor can successfully predict the stock price using SVR, LSTM and RF. As we can see in table 2 & 3 and figure 2, 3, 4&5, the RMSE of SVR is much lower than LSTM's in all stocks and the predicted curve of SVR fits more than LSTM. It is because the stock price is strongly time-varying representing complex non-linearity and SVR can be used in non-linear prediction problems. Although SVR performs the best in this project, the errors in some stocks are also non-trivial such as COKE and MCD. Furthermore, we only use three common kernel functions in building SVR model but they are not enough to the stock market. We should tried more kernel functions or build a more market-fitting kernel function by ourselves. LSTM is still superior than most traditional algorithms, such as ARIMA and GARCH. Furthermore, it is relatively easier to integrate high dimensional dataset. However, the accuracy of LSTM is still affected by the past information considering large time scale, i.e., 20 years stock prices. Forget gate layer could not obliterate all historical information deteriorating the prediction. When time scale shrinks to 3 years or less, LSTM's accuracy increases dramatically. For RF classifier, prediction with class less than 6 and length of interval longer than 0.05 could be accurate enough. However, the accuracy of RF classifier also depends on how we set the begin and end point of intervals.

Our project innovatively combines the technical indicator and fundamental factor and all model predict accurately in next-day's stock price. Both SVR and LSTM have relatively low RMSE while the accuracy in RF is in large magnitude. It seems like we make a money-making machine. However, in our project, the prediction period is too short. If we consider the noise, commission fee in the stock market and errors in the model, the cost and risk increases and the machine will break down. To get rid of these risks and make the models more usable, an idea is increasing the prediction period, say 30-day, but none of our models can predict well in such period. To make the period longer, we can use more market-fitting feature space such as financial indicator and related macro economic factors. Also, we can include the majority investors attitude and finance news based on natural language process (NLP) techniques.

References

[1] Althelaya, Khaled A., El-Sayed M. El-Alfy & Salahadin Mohammed. (2018) Evaluation of bidirectional lstm for short-and long-term stock market prediction. *2018 9th international conference on information and communication systems (ICICS). IEEE*

[2] Bruno Miranda Henrique, Vinicius Amorim Sobreiro & Herbert Kimura. (2018) Stock price prediction using support vector regression on daily and up to the minute prices. *The Journal of Finance and Data Science* **4**: 183-201

[3] Cao, Jian, Zhi Li, & Jian Li. (2019) Financial time series forecasting model based on CEEMDAN and LSTM. *Physica A: Statistical Mechanics and its Applications* **519**:127-139.

[4] Cherkassky, V. & Ma, Y. (2004) Practical selection of svm parameters and noise estimation for svm regression. *Neural Networks.*

[5] C.W. Hsu ,C.C. & C.J. Lin. A Practical Guide to Support Vector Classification. *Online: /http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdfS.*

[6] Fischer, Thomas, & Christopher Krauss. (2018) Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research* **270.2**: 654-669.

[7] Hsu S H , Hsieh P A , Chih T C. & et al. (2009) A two-stage architecture for stock price forecasting by integrating self-organizing map and support vector regression.*Expert Systems with Applications* **36(4)**:7947-7951.

[8] Hsu, C.W., Chang, D. & Lin, C.J. (2003) *A Practical Guide to Support Vector Classification.*

[9] Jigar Patel, Sahil Shah, Priyank Thakkar & K. Kotecha (2014) Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. *Expert Systems with Applications* **42**:259-268

[10] Kao L J ,Chiu C C. ,Lu C J. & et al. (2013) Integration of nonlinear independent component analysis and support vector regression for stock price forecasting.*Neurocomputing* **99**:534-542.

[11] Siami-Namini, Sima, & Akbar Siami Namin. (2018) Forecasting economics and financial time series: ARIMA vs. LSTM.*arXiv preprint* **arXiv:1803.06386**

[12] V. Cherkassky & Y. Ma (2004) Practical selection of SVM parameters and noise estimation for SVM regression.*Neural Networks* **17**:113–126.

[13] Yeh C Y., Huang C W. & Lee S J. (2011) A multiple-kernel support vector regression approach for stock market price forecasting*Expert Systems with Applications* **38(3)**:2177-2186.

Appendix

Support Vector Regression

Table 5: Radial kernel function

	$C = 2^{-1}$	$C = 2$	$C = 2^3$	$C = 2^5$	$C = 2^7$	$C = 2^9$
	MSE	MSE	MSE	MSE	MSE	MSE
AAPL	40.2711	39.0897	**38.5796**	38.8121	39.2012	40.2746
COKE	120.5115	107.0297	97.6289	88.7686	**83.9924**	84.8465
HP	2.0815	2.0766	**2.0761**	2.0773	2.0776	2.0773
KR	0.8164	**0.8112**	0.8115	0.8128	0.8142	0.8151
MCD	65.5083	48.7475	35.7457	28.1692	**27.7022**	30.9544
MSFT	90.6707	89.3219	87.4140	86.7571	84.2413	**82.2440**
RCL	5.6591	4.9714	4.7765	4.7562	4.7492	**4.7468**

Table 6: Linear kernel function

	$C = 2^{-1}$	$C = 2$	$C = 2^3$	$C = 2^5$	$C = 2^7$	$C = 2^9$
	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE
AAPL	2.42188	2.41602	2.41540	2.41527	2.41524	**2.41521**
COKE	**11.26430**	11.27064	11.27442	11.27488	11.27486	11.27497
HP	**2.06266**	2.06376	2.06392	2.06394	2.06391	2.06397
KR	**0.81070**	0.81077	0.81082	0.81083	0.81084	0.81085
MCD	**4.72045**	4.72638	4.72834	4.72918	4.72918	4.72922
MSFT	4.12000	4.10401	4.10278	4.10259	4.102582	**4.102581**
RCL	**4.72209**	4.72756	4.72960	4.72997	4.73038	4.73024

Table 7: Polynomial kernel function

		$C = 2^{-1}$	$C = 2$	$C = 2^3$	$C = 2^5$	$C = 2^7$	$C = 2^9$
		RMSE	RMSE	RMSE	RMSE	RMSE	RMSE
AAPL	degree = 2	**211.1119**	211.3755	211.4515	211.4497	211.4510	211.4369
COKE		**166.5565**	170.2635	170.7345	171.7514	171.7522	171.7558
HP		**19.4123**	19.4157	19.4157	19.4157	19.4157	19.4155
KR		10.0203	10.0210	10.0193	10.0192	10.0192	**10.0186**
MCD		**311.8398**	313.0168	313.2379	313.2828	313.3630	313.3718
MSFT		**141.9337**	158.2061	158.8954	158.8915	158.8758	158.8131
RCL		16.1378	16.1151	16.1165	16.1167	16.1158	**16.1123**
AAPL	degree = 3	**628.6810**	628.9399	628.9431	628.9476	628.9482	629.1170
COKE		**638.3118**	641.1083	641.1114	641.6841	642.5240	642.9482
HP		10.7069	10.7041	10.7012	10.7013	10.7011	**10.7007**
KR		9.6431	**9.6356**	9.6357	9.6357	9.6360	9.6370
MCD		**225.6394**	225.7646	225.7647	225.7652	226.0282	226.0589
MSFT		**986.0207**	990.3433	990.3647	990.4501	990.7947	991.8879
RCL		**27.8016**	27.8919	27.8918	27.8913	27.8892	27.8811

Long Short-Term Memory Recurrent Neural Network

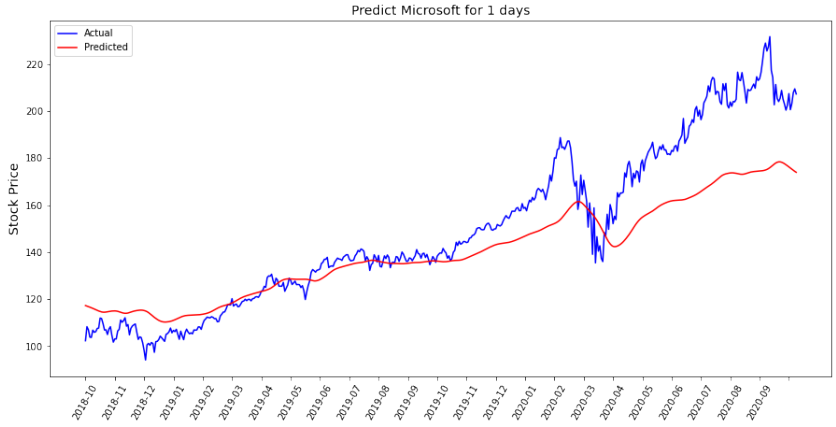

Figure 7: Predict Microsoft Close Price by LSTM

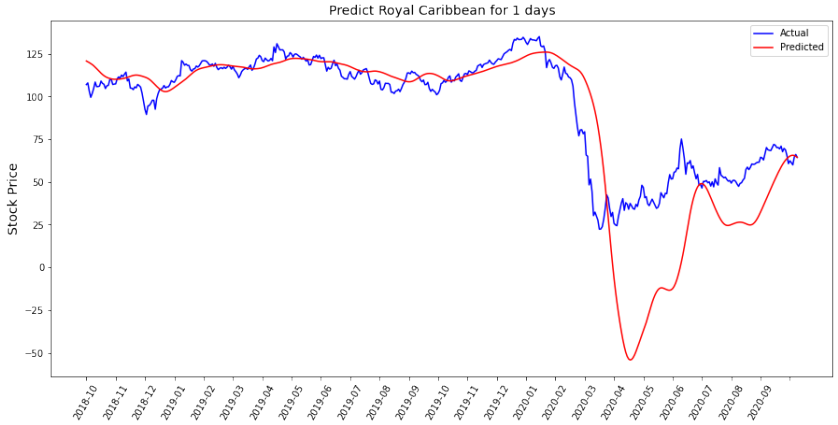

Figure 8: Predict Royal Caribbean Close Price by LSTM

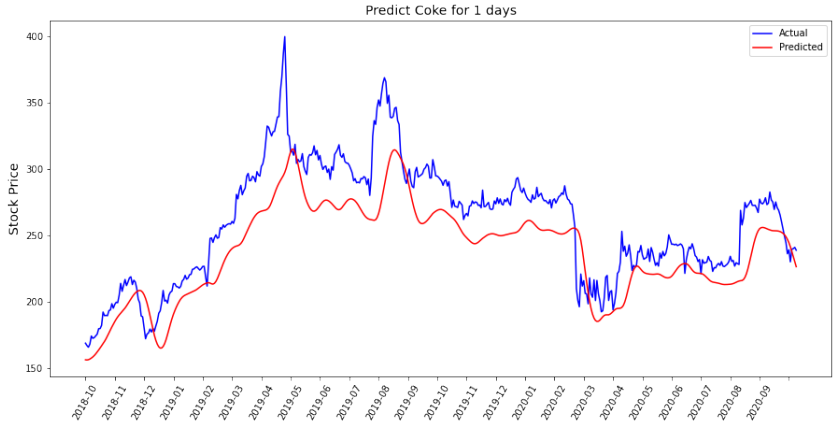

Figure 9: Predict Coke Cola Close Price by LSTM

Random Forest Classifier