Algorithmic Trading & Stock Market Price Prediction

Saumya Sheth
saumyabsheth@gmail.com
Mentored By:
Agamjot Singh , Aastha Sancheti, Samarjeet Singh

Dec 2022 - Jan 2023

Contents

1 Introduction

The Stock Market is a crucial platform for humans to trade and get equity in firms they have confidence in. For some, it serves as an additional source of income while for others it's the only source of income. The change in the trend of the stock market price has always been identified as a very important problem in the economic field. The traditional analysis method is based on economics and finance and involves *fundamental and technical analysis*. This is a tedious method and for a living, it can involve a lot of risks.

In recent years Machine Learning has blown up and one of its subsets- *Deep Learning* has a notable use in this field. In this project, we are first introduced to basics of python after which we were introduced to various data science libraries and then we started with machine learning, its basic terminologies and maths for it, then we learned various learning techniques and models and application of machine learning in finance. After that we learned about momentum effect in stock trading and also the paired switching technique for trading. After that we moved on to deep learning. We first understood the basics of neural networks, how do they work and some maths behind it. Then we also learned about various optimizers. Then we learned to implement neural networks in pytorch and then in keras. Then finaaly we went on to CNN and LSTM models, read a research paper about it and implemented the model on tata motors stock. we use *Convolutional Neural Network(CNN) - Long short-term memory(LSTM)* Model to predict stock prices.The accuracy of traditional methods is difficult to be convincing. On the contrary, this model is a *high accuracy-high performance* model.

2 Dataset

2.1 Data Collection

The quantity and quality of the data dictate how accurate the model is. Data can be collected via pre collected datasets from kaggle, UCI, etc. Another way of collecting data is web scraping using packages like BeautifulSoup.

2.2 Data Preprocessing

Data preprocessing involves wrangling the data and prepare it for training. We perform data cleaning which is required by removing the duplicates,

correcting the errors, deal with the missing values, normalize the data, data type conversion if needed, etc. We randomize the data which erases the effects of the particular order in which we collected or prepared our data. We split the data into training and testing or evaluation sets.

Data preparation also involves exploratory data analysis. Exploratory data analysis (EDA) is an approach to analyzing and understanding a dataset through summary statistics and visualizations. The goal of EDA is to identify patterns, anomalies, and relationships in the data that can be used to guide further analysis and modeling. This typically includes techniques such as plotting data, calculating summary statistics, and identifying outliers. EDA is an iterative process that can help to refine research questions and hypotheses, and is often the first step in the data analysis workflow.
We did an assignment for exploratory data analysis where we had to analyze data of an e commerce platform. We had 3 months data as 3 seperate files. We combined the three files and carried out EDA to draw some inferences working with the data which can help the firm make decisions or create new strategies. In EDA, we compared the monthwise sales, extracted the top brands according to sales/revenue, looked at how many people actually buy something out of people who have interacted with the firm, we analyzed what is the purchase rate that is what percentage of goods are buyed after being added to cart, how many average number of days a customer keeps an item in his cart before making a purchase, day wise sales like which week day has the highest sales and which has the lowest, what time of the day people purchase the most. These are the inferences which we received after doing the EDA assignment. This helped us understand the importance of EDA.

3 Machine learning

We read articles related to machine learning and then learnt about its application in Finance.

3.1 What is machine learning ?

Machine Learning is a concept which allows the machine to learn from examples and experience, and that too without being explicitly programmed. So instead of writing the code, what we do is we feed data to the generic

algorithm, and the algorithm/ machine builds the logic based on the given data. Machine Learning is a subset of artificial intelligence which focuses mainly on machine learning from their experience and making predictions based on its experience. It enables the computers or the machines to make data-driven decisions rather than being explicitly programmed for carrying out a certain task. These programs or algorithms are designed in a way that they learn and improve over time when are exposed to new data.

3.2 Supervised Learning

Supervised Learning is the process of making an algorithm to learn to map an input to a particular output. This is achieved using the labelled datasets that you have collected. If the mapping is correct, the algorithm has successfully learned. Else, you make the necessary changes to the algorithm so that it can learn correctly. Supervised Learning algorithms can help make predictions for new unseen data that we obtain later in the future. Learning gives the algorithm experience which can be used to output the predictions for new unseen data Experience also helps in optimizing the performance of the algorithm. Real-world computations can also be taken care of by the Supervised Learning algorithms.
Linear Regression and classification are types of supervised learning.

3.3 Linear regression

Linear regression is a statistical method used to establish a relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data. The goal is to find the best-fitting line through the data points. The line is represented by an equation y = mx + b, where y is the dependent variable, m is the slope of the line, x is the independent variable, and b is the y-intercept. Linear regression is useful for predicting outcomes, and it can be used with one or multiple variables. This method assumes a linear relationship between the input variables and the output variable. We also learnt how to implement linear regression. There are other types of regression techniques like logistic regression, polynomial regression, stepwise regresssion.

3.4 Classification

Classification in machine learning is a process of training a model to predict the class or category of a given input data. The model learns from a labeled dataset, where each input data is associated with a specific class or category. Once the model is trained, it can be used to predict the class of new, unseen data. The goal of classification is to accurately predict the class of new data based on its features. There are various algorithms available for classification such as Logistic Regression, Decision Trees, Random Forest, SVM, Naive Bayes and Neural Networks. The choice of algorithm depends on the characteristics of the data and the desired level of accuracy.

Then we also learnt about the overfitting problem in machine learning and how it can be resolved. A statistical model is said to be overfitted when we feed it a lot more data than necessary. When a model fits more data than it actually needs, it starts catching the noisy data and inaccurate values in the data. As a result, the efficiency and accuracy of the model decrease. It occurs when a model is too complex and has too many parameters relative to the amount of data it is trained on. The model then memorizes the training data rather than learning the underlying patterns in the data. One of the most effective ways to avoid overfitting is to use more training data. Another way is to use regularization techniques, which add a penalty term to the model's objective function to discourage excessive complexity. Reducing the number of features by selecting the most relevant ones or using dimensionality reduction techniques. Ensemble methods like bagging and boosting can also help to reduce overfitting. Using simpler models with fewer parameters. Early stopping for deep learning model which means to stop the training process before the model over-memorizes the training data. It's important to keep in mind that overfitting can be avoided or reduced but not completely eliminated. It's a trade-off between complexity and performance, and a balance needs to be struck between the two.

3.5 Unsupervised learning

Unsupervised learning is a type of machine learning where the model is trained on unlabeled data, without any prior knowledge of the output. The goal of unsupervised learning is to find patterns or structure in the data. Common techniques used in unsupervised learning include clustering, which

groups similar data points together, and dimensionality reduction, which reduces the number of features in the data. Unsupervised learning can be used for tasks such as anomaly detection, data compression, and density estimation. Some popular unsupervised learning algorithms are K-means, PCA, Autoencoder, and hierarchical clustering. In contrast to supervised learning, unsupervised learning does not require labeled data, and it can discover hidden patterns in the data to generate insights.

3.6 K-means clustering

K-means clustering is a technique used in unsupervised machine learning to group similar data points together. The algorithm starts with k initial centroids, which are randomly chosen data points. Each data point is then assigned to the closest centroid. The centroids are then recalculated based on the mean of all the data points assigned to that centroid. The process repeats until the centroids stop moving or a maximum number of iterations is reached. The final result is k clusters, where each cluster contains similar data points. The number of clusters, k, must be specified in advance.

3.7 K-Nearest neighbours

K-nearest neighbors (KNN) is a simple, non-parametric method used for classification and regression in machine learning. Given a new data point, KNN finds the k-number of closest training examples in the feature space and assigns the most common label among those k-neighbors. The main idea behind the KNN algorithm is that similar data points tend to have similar labels. The parameter k is the number of closest neighbors to examine, and it is typically chosen through cross-validation. The algorithm does not make any assumptions about the underlying distribution of the data and is sensitive to the scale and distribution of the features.
These are the things we learned in machine learning.

4 Applications in Finance

After learning the basics of machine learning we moved on to its application in finance. We learned about the application of python in automated trading in stock market. We first learnt about very basics of stock market and trading

and understood how prices fluctuate in such Markets and how that can be used to generate profits. This Process of buying at low prices and selling them at higher ones could be automated to yield astronomical profits. This is what's known as Algorithmic Trading.

4.1 Mean reversion

Mean reversion in stock price is the assumption that the price will tend to move back to the average price over time. Mean reversion trading often refers to counter-trend or reversal trading. If any stock price is currently below its average, then we should buy it, as there is every chance that it is having some down-time and will rise in coming times. Similarly if some Stock Price is above its Average, then that indicates, its time to sell. Then we learnt about the implementation of mean reversion in country equity indices.

4.2 Momentum effect in stock trading

There's something which goes by the name of Momentum Anamoly in Trading, which says that what was strongly going up in the near past will probably continue to go up shortly. Stocks which outperform peers on 3-12 month period tend to perform well also in the future. Nowadays, momentum strategies are well-known and generally accepted in both the public and academic worlds. Yet, the momentum strategy is based on a simple idea, the theory about momentum states that stocks which have performed well in the past would continue to perform well. On the other hand, stocks which have performed poorly in the past would continue to perform badly. This results in a profitable but straightforward strategy of buying past winners and selling past losers. Moreover, the strategy selects stocks based on returns over the past J months. It holds them for K months, the selection of J and K is purely dependent on the choice of the investor, but we are presenting results of 12-1 month momentum strategy. To sum it up, the stocks which have outperformed peers during the K months period tends to perform well in the upcoming period and vice versa. Moreover, the momentum effect works in a small-cap universe as well as in a large-cap universe, and it is safe to say that momentum is one of the most academically investigated effects with strong persistence. Pure momentum portfolios are created in a way that investor longs stocks with the strongest momentum and shorts stocks with the lowest momentum. However, this pure momentum portfolio recorded

the disastrous year 2009 with more than 80% drawdown. Despite the crash, the momentum factor is still a strong performance contributor in long-only portfolios (long stocks with the strongest momentum without shorting the market or low momentum stocks). Compared to the market, value or size risk factors, momentum has offered investors the highest Sharpe ratio. However, momentum has also had the worst crashes, making the strategy unappealing to investors with reasonable risk aversion. We find that the risk of momentum is highly variable over time and quite predictable. The major source of predictability does not come from systematic risk but specific risk. Managing this time-varying risk eliminates crashes and nearly doubles the Sharpe ratio of the momentum strategy.

After understanding about momentum trading we did an assignment related to it in which we Simulated a portfolio from scratch which purchases stocks based on the momentum trading strategy and achieved 20% average annual returns backtested on 10 years of stock market data of 30 unique stocks.

4.3 Paired switching

The basic idea behind Paired Switching is to select two stocks which are negatively co-related, so that if one falls then the other should be rising shortly afterwards and vice versa. We first learned about correlation. The correlation coefficient () is a measure that determines the degree to which the movement of two different variables is associated. The most common correlation coefficient, generated by the Pearson product-moment correlation, is used to measure the linear relationship between two variables. However, in a non-linear relationship, this correlation coefficient may not always be a suitable measure of dependence. The possible range of values for the correlation coefficient is -1.0 to 1.0. In other words, the values cannot exceed 1.0 or be less than -1.0. A correlation of -1.0 indicates a perfect negative correlation, and a correlation of 1.0 indicates a perfect positive correlation. If the correlation coefficient is greater than zero, it is a positive relationship. Conversely, if the value is less than zero, it is a negative relationship. A value of zero indicates that there is no relationship between the two variables. In the financial markets, the correlation coefficient is used to measure the correlation between two securities. For example, when two stocks move in the same direction, the correlation coefficient is positive. Conversely, when two stocks move in opposite directions, the correlation coefficient is negative.

Correlation coefficient can be calculated using covariance of the two variables and the standard deviation of each variable.
Then we learned about paired switching. A paired switching strategy is a subset of rotational asset strategies. The simplest form uses two assets (or stocks), which have a negative correlation. Investors then invest in one pair and periodically switch position based on relative performance (or some other criterion). The strategy is based upon the idea that it is easier to exploit a negative correlation by switching between two assets than by traditional asset mixing. As both assets are negatively correlated, there is a high probability that portfolio performance is lower in the case of a mix than return for individual assets. Academic research shows that if the criterion for switching is even minimally accurate, there is a probability of improving the performance over the portfolio wherein the two assets are statically weighted. The simplicity of this idea is attractive; therefore, we decided to create an independent entry for the strategy as we think that some elements of it could be used in other more complex constructions. There is a high probability that portfolio performance is lower in the case of a mix than return for individual assets if two assets are negatively correlated. The strategy's performance then depends on the accuracy of the timing rule.

After understanding about paired switching we did an assignment related to it in which we Simulated a portfolio from scratch which purchases stocks based on the paired switching strategy and achieved 25% average annual returns backtested on 10 years of stock market data of 30 unique stocks.

5 Deep learning

Deep learning is a subset of machine learning in which artificial neural networks, algorithms inspired by the structure and function of the human brain, are used to analyze and model complex patterns in data. These models can be used for a variety of tasks such as image and speech recognition, natural language processing, and decision making. Deep learning models are called "deep" because they typically have many layers, or levels of abstraction, in their architecture.

5.1 Neural networks

We learned and understood the concept behind neural networks using the 3B1B playlist on youtube. A neural network is a type of machine learning model that is inspired by the structure and function of the human brain. It is composed of layers of interconnected "neurons," which process and transmit information. The input data is passed through the network, where it is transformed and processed by the neurons in each layer, until it reaches the output layer, where a prediction or decision is made. One of the key features of neural networks is their ability to learn from data. This is achieved through a process called "training," where the network is presented with a set of input-output pairs, and the weights and biases of the neurons are adjusted to minimize the error between the network's predictions and the true outputs. To determine the value on each element of a layer the weighted sum of each element of previous layer is taken and a bias is added at the end. A sigmoid or relu function is applied on the value we get. If we want to keep the value between 0 and 1 then sigmoid function is applied. So we have different set of weights and biases for each element and of each layer. Then using a particular set of weights and biases we find the cost function which is kind of estimate of error and we try to minimize it. We learnt about the gradient descent algorithm to minimize the cost function and get the best values of weights and biases. We find the gradient at particular point and go in the negative direction to reach the minima. By this we also get to know that which of the weights and biases have significant effect on the cost function and which have negligible effect and we change our weights and biases according to that to get the best possible result. Neural networks can be used for a wide range of tasks, including image recognition, natural language processing, and prediction. They are particularly useful for tasks that involve large and complex data sets, as well as for tasks where the relationship between inputs and outputs is not well understood. In summary, neural networks are powerful machine learning models that are inspired by the structure and function of the human brain. They can learn from data and can be used for a wide range of tasks, including image recognition, natural language processing, and prediction. There are various types of neural networks, each with their own strengths and weaknesses, and they are widely used in many areas.

5.2 optimizers

Optimizers are algorithms used in machine learning to adjust the parameters of a model in order to minimize a loss function. The loss function measures the difference between the model's predictions and the true output values for a given set of inputs. Optimizers update the model's parameters in order to reduce the value of the loss function. This is done by computing the gradient of the loss function with respect to the model parameters, and then updating the parameters in the direction of the negative gradient. The magnitude of the update is controlled by a learning rate, which determines the step size at which the optimizer makes progress towards finding the minimum of the loss function. There are several different types of optimizers, each with its own strengths and weaknesses. The optimizers which we learned are:

Stochastic Gradient Descent : Stochastic Gradient Descent (SGD) is a commonly used optimization algorithm for training machine learning models. The idea behind SGD is simple. Given a loss function that measures the difference between the model's predictions and the true outputs, the algorithm updates the model parameters in the direction of the negative gradient of the loss function. The gradient of the loss function with respect to the model parameters represents the direction of maximum increase in the loss, and moving in the opposite direction is expected to decrease the loss. In SGD, the update of the model parameters is performed one sample at a time, rather than using the gradient computed from the entire training set. This makes the algorithm computationally efficient, as it can make progress using only a single example at a time. It also adds some randomness to the optimization process, which can help the algorithm escape from local minima and converge faster. The magnitude of the update is controlled by a learning rate, which determines the step size at which the optimizer makes progress towards finding the minimum of the loss function. Setting the learning rate too high can result in overshooting the minimum, while setting it too low can result in slow convergence. The learning rate is often set using a schedule that decreases over time, which helps the optimizer converge to the global minimum. The randomness and efficiency of SGD make it a popular choice for many machine learning tasks.

Mini batch gradient descent : In MBGD, instead of updating the model parameters using the gradient of a single sample at a time, as in SGD, the

update is performed using the gradient of a small random subset, or "mini-batch," of the training samples. The size of the mini-batch is a hyperparameter that can be adjusted to control the trade-off between the accuracy of the update and the computational efficiency of the algorithm. Using mini-batches of data rather than individual samples has several advantages. First, it allows the optimization algorithm to make more efficient use of the available computational resources. Second, it introduces some randomness into the optimization process, which can help the algorithm escape from local minima and converge faster. Finally, it provides a more stable estimate of the gradient, as the average of the gradients of multiple samples is less prone to the noise that can occur when computing the gradient of a single sample. Even in this learning rate is used. MBGD is a more computationally efficient and stable optimization method than SGD, and is commonly used in training large-scale machine learning models.

Momentum : Momentum is a popular optimization algorithm used to train machine learning models, especially deep neural networks. Momentum optimization is a variation of the gradient descent optimization algorithm that uses the concept of momentum to speed up convergence. In gradient descent, the model parameters are updated in the direction of the negative gradient of the loss function with respect to the parameters. Momentum optimization adds a moving average of the previous updates to the current update, allowing the optimizer to overcome obstacles and converge more quickly. The intuition behind momentum optimization is that, when the optimizer is moving in a direction, it tends to continue moving in that direction, even if the gradient changes. This is because the optimizer is influenced by its previous updates, which are stored in the moving average. The size of the moving average is controlled by a hyperparameter called the "momentum" that determines how much the previous updates influence the current update. Momentum optimization can be seen as adding a "friction" term to the gradient update, which slows down the optimizer when it is oscillating, and speeds it up when it is moving in a stable direction. This helps the optimizer escape from local minima and converge to the global minimum more quickly. The momentum term allows the optimizer to overcome obstacles and converge more quickly, and is particularly useful for training deep neural networks, where the optimization surface is often complex and non-convex.

Adagrad : Adagrad (Adaptive Gradient Algorithm) is a popular optimization

algorithm used to train machine learning models, especially deep neural networks. Adagrad adapts the learning rate for each parameter in the model based on its historical gradient information, making it well suited for problems with sparse gradients, where some parameters may require much larger learning rates than others. In Adagrad, the learning rate for each parameter is maintained separately, and updated over time based on the historical gradient information for that parameter. Specifically, the learning rate for each parameter is divided by the square root of the sum of the squares of its historical gradient information. This means that parameters that have been updated frequently will have a smaller learning rate, while parameters that have been updated infrequently will have a larger learning rate. The intuition behind Adagrad is that the learning rate should be larger for parameters that have been updated infrequently, since these parameters are less likely to be close to their optimal values. On the other hand, the learning rate should be smaller for parameters that have been updated frequently, since these parameters are more likely to be close to their optimal values, and larger updates could overshoot the optimal values. Adagrad is an effective optimization algorithm that has been used to train many state-of-the-art deep learning models. However, it has some limitations, such as the accumulation of the squared gradients, which can cause the learning rate to become very small, leading to slow convergence. To address these limitations, Adagrad has been extended and improved in several ways, including Adadelta and RMSprop. Adagrad is well suited for problems with sparse gradients, where some parameters may require much larger learning rates than others, and has been used to train many state-of-the-art deep learning models.

Adam : Adam (Adaptive Moment Estimation) is a popular optimization algorithm used to train machine learning models, especially deep neural networks. Adam combines the ideas of Momentum optimization and Adagrad to adapt the learning rate for each parameter in the model based on its historical gradient information, making it well suited for a wide range of optimization problems. In Adam, the learning rate for each parameter is maintained separately, and updated over time based on the historical gradient information for that parameter. Specifically, Adam maintains moving averages of the first and second moments of the gradients, which are used to estimate the mean and variance of the gradients, respectively. The learning rate for each parameter is then divided by the square root of the estimated variance, multiplied by a bias correction term. The intuition behind Adam is that the

learning rate should be larger for parameters with high variance gradients, since these parameters are less likely to be close to their optimal values. On the other hand, the learning rate should be smaller for parameters with low variance gradients, since these parameters are more likely to be close to their optimal values, and larger updates could overshoot the optimal values. Adam is an effective optimization algorithm that has been used to train many state-of-the-art deep learning models. It has been shown to be more robust and efficient than other optimization algorithms, such as Stochastic Gradient Descent (SGD), Momentum optimization, and Adagrad. Adam has been used to train many state-of-the-art deep learning models, and is well suited for a wide range of optimization problems.

After learning the theory about neural networks and various optimizers, we learned how to implement them using pytorch library and then we learned the implementation of neural networks using keras. Although pytorch might be more difficult than keras, it is more pythonic and is very useful for making custom neural nets. While learning the implementation, we did two assignments, one which involved hand written digit recognition and other one was the cat dog classifier. In the digit recognition assignment, we got an accuracy of 97.9% which is very good. And in the catdogclassifier we got an accuracy of 73.3%.

6 Understanding CNN-LSTM Model

Finally in the last week we learned about the CNN-LSTM model. We learned it through a research paper by which we learned about model layers, activating functions, filters, filter size, etc.

6.1 CNN-LSTM Model

CNN(*Convolutional Neural Network*) helps in understanding and *finding patterns and features* using convolution layer and pooling layer.
LSTM(*Long Short-Term Memory*) networks are well-suited to classifying , processing and making predictions based on time series data.

According to the characteristics of CNN and LSTM, a stock forecasting model based on CNN-LSTM is established. The model structure (refer Figure 1) includes input layer, one-dimensional convolution layer, pooling

layer, LSTM hidden layer, and full connection layer.

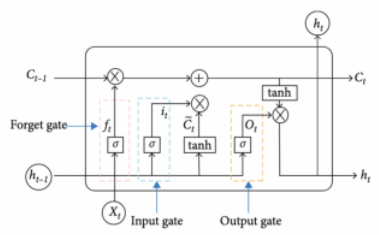

Figure 1: CNN-LSTM structure diagram.

Figure 1: CNN-LSTM structure diagram.

6.2 CNN

CNN is mainly composed of two parts: **convolution layer** and **pooling layer**. Each convolution layer contains a plurality of convolution kernels, and its calculation formula is shown in formula (1). After the convolution operation of the convolution layer, the features of the data are extracted, but the extracted feature dimensions are very high, so in order to solve this problem and reduce the cost of training the network, a pooling layer is added after the convolution layer to reduce the feature dimension:

$$l_t = tanh(x_t * k_t + b_t) \tag{1}$$

where l_t represents the output value after convolution, $tanh$ is the activation function, x_t is the input vector, k_t is the weight of the convolution kernel, and b_t is the bias of the convolution kernel.

6.3 LSTM

LSTM is a network model designed to solve the longstanding problems of gradient explosion and gradient disappearance in RNN. The LSTM memory cell consists of three parts: **the forget gate, the input gate, and the output gate**, as shown in Figure 2.

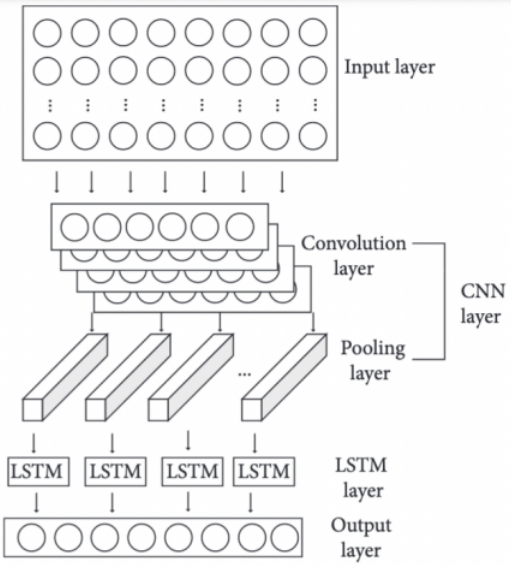

Figure 2: Architecture of LSTM memory cell.

7 CNN-LSTM Training and Prediction Process

The CNN-LSTM process of training and prediction is shown in Figure 3.

The main steps are:

- Input data: input the data required for model training.
- Data standardisation: to train the model better, the *z-score* standardisation method is applied to standardise the input data according to the following formula:

$$y_i = \frac{x_i - \bar{x}}{s},$$

$$x_i = y_i * s + \bar{x}$$

  where y_i is the standardised value, x_i is the input data , $\bar{x}$ is the average of the input data and s is the standard deviation of the input data.

- Initialise network: initialise the weights and biases of each layer of the CNN-LSTM.

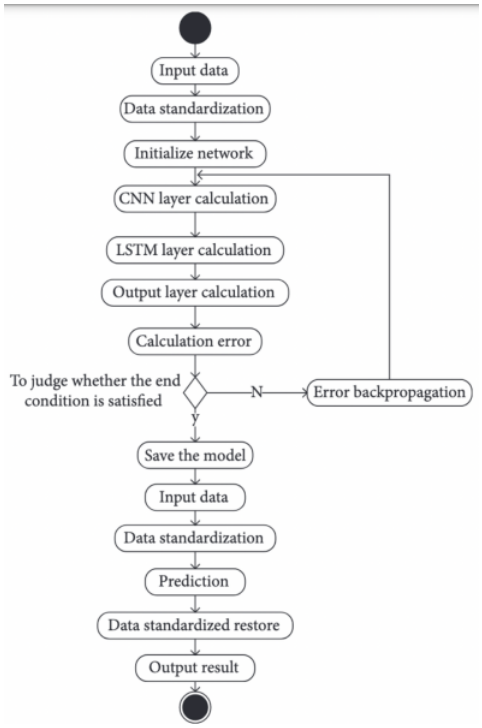

Figure 3: Activity diagram of CNN-LSTM training and prediction process.

- CNN layer calculation: the input data are successively passed through the convolution layer and pooling layer in the CNN layer, the feature extraction of the input data is carried out, and the output value is obtained.
- LSTM layer calculation: the output data of the CNN layer are calculated through the LSTM layer, and the output value is obtained.
- Output layer calculation: the output value of the LSTM layer is input into the full connection layer to get the output value.
- Calculation error: the output value calculated by the output layer is compared with the real value of this group of data, and the corresponding error is obtained.
- To judge whether the end condition is satisfied

- Error backpropagation: propagate the calculated error in the opposite direction, update the weight and bias of each layer, and go to step 4 to continue to train the network.
- Save the model
- Input Data
- Data standardisation
- Forecasting
- Data standardized restore: the output value obtained through the model of CNN-LSTM is restored to the original value.
- Output result: output the restored results to complete the forecasting process.

7.1 Model Implementation

In order to evaluate the forecasting effect of CNN-LSTM, *the mean absolute error (MAE)*, *root mean square error (RMSE)*, and *R-square (R2)* are used as the evaluation criteria of the methods.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

$$RMSE = \sqrt[2]{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$R^2 = 1 - \frac{(\sum_{i=1}^{n}(y_i - \hat{y}_i)^2)/n}{(\sum_{i=1}^{n}(\bar{y}_i - t\hat{y}_i)^2)/n}$$

The closer the value of MAE and $RMSE$ to 0, of R^2 to 1, the smaller the error between the predicted value and the real value, the higher the forecasting accuracy.

8 Implementation of CNN-LSTM

8.1 Parameter setting

The parameter setting of the CNN-LSTM for this experiment is shown in Figure 4.

Parameters	Value
Convolution layer filters	32
Convolution layer kernel_size	1
Convolution layer activation function	tanh
Convolution layer padding	Same
Pooling layer pool_size	1
Pooling layer padding	Same
Pooling layer activation function	Relu
Number of hidden units in LSTM layer	64
LSTM layer activation function	tanh
Time_step	10
Batch_size	64
Learning rate	0.001
Optimizer	Adam
Loss function	mean_absolute_error
Epochs	100

Figure 4: Parameter setting of CNN-LSTM.

8.2 Model Structure and Implementation

According to the parameter setting of CNN-LSTM network, we can know that the specific model is constructed as follows: the input training set data is a three-dimensional data vector (None, 10, 6), in which 10 is the size of the time-step and 8 is the 8 features of the input dimension. First, the data enter the one-dimensional convolution layer to further extract features and obtain a three-dimensional output vector (None, 10, 32), in which 32 is the size of the convolution layer filters. Next, the vector enters the pooling layer, and a three-dimensional output vector (None, 10, 32) is also obtained. And then, the output vector enters the LSTM layer for training, and the output data (None, 64) after training enter another layer of full connection layer to

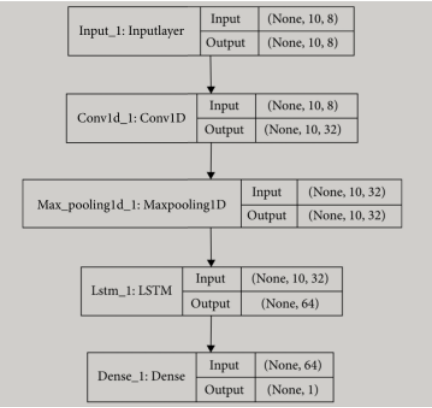

Figure 5: The model structure of CNN-LSTM.

get the output value; 64 is the number of hidden units in the LSTM layer.

After understanding the CNN and LSTM model, we did an assignment to implement these models in which we had to predict the stock market prices of TATA motors. In the assignment we also learnt the z score technique while applying it for data preprocessing. In the assignment, we predicted the adjusted closing price of the next day given the 6 features of the 10 previous days.

9 Acknowledgement

I would like to thank the mentors for providing such good resources and assignments. It was a great learning experience doing the assignments and learning new things. And also I would like to thank Analytics club for doing Winter In Data Science.