Project Report
On

Stock-Price Prediction Using Various Regression Techniques

Course Project, of

DS 303 - Introduction to Machine Learning

under

Prof:Biplab Banerjee

Bhawna Mahur 200100044
Jigmat Chosdol 200100075
Aditya Sugriv Kendre 200100012
Niteesh Singh 200100105

Stock price prediction using KNN model.

This part consists of stock price prediction using the knn model. Important libraries are imported and data is downloaded and loaded into a notebook from yahoo finance website.Data is of TESLA company for 2021-2022 .The Data Is stored in OHLC(Open, High, Low, Close) format in a CSV file. To read a CSV file, we can use the read_csv() method of pandas.
The data needed to be processed before use such that the date column should act as an index .
Explanatory or independent variables are used to predict the value response variable. The X is a dataset that holds the variables which are used for prediction. The X consists of variables such as 'Open – Close' and 'High – Low'. These can be understood as indicators based on which the algorithm will predict tomorrow's trend.
The target variable is the outcome which the machine learning model will predict based on the explanatory variables. y is a target dataset storing the correct trading signal which the machine learning algorithm will try to predict. If tomorrow's price is greater than today's price then we will buy the particular Stock else we will have no position in the. We will store +1 for a buy signal and -1 for a no position in y. We will use where() function from NumPy to do this

We will split data into training(75%) and test data(25%) sets. This is done so that we can evaluate the effectiveness of the model in the test dataset.

We will use a function KNeighborsClassifier from **sklearn** library to create our classifier model using the fit() method on the training data set.

Once the model is fit, we can find the optimal parameter of K ,obtained through GridSearchCV. The value obtained for K is 2 for classification.

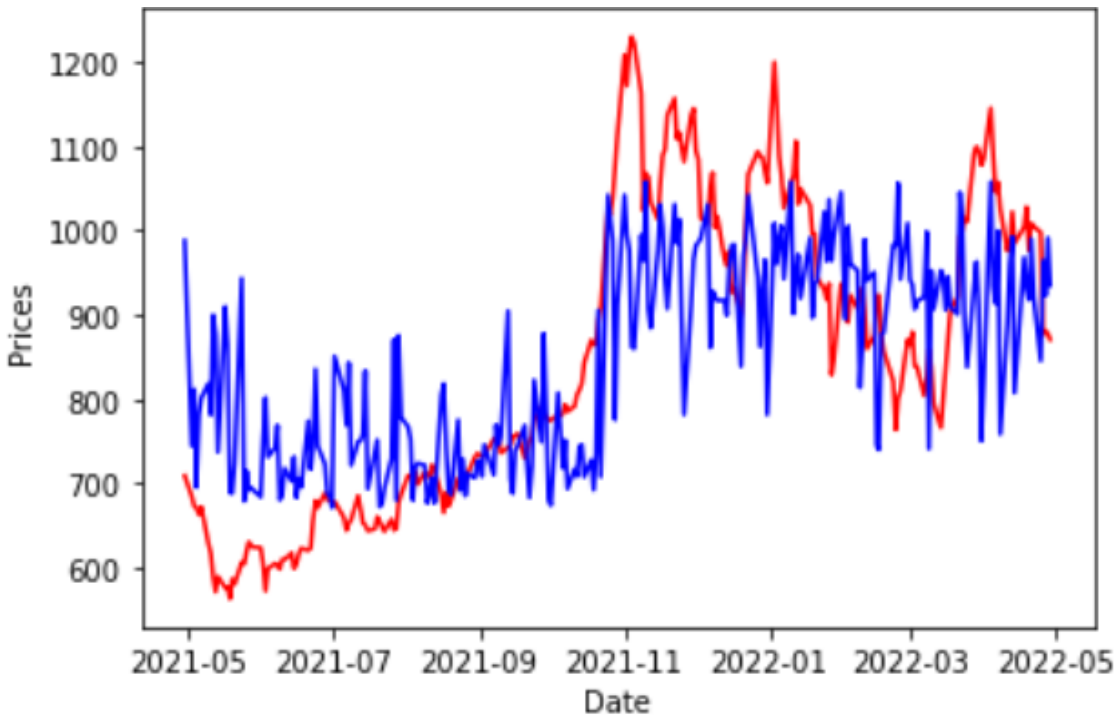
We will compute the accuracy of the algorithm on the train and test the data set by comparing the actual values of the signal with the predicted values of the signal. The function accuracy_score() will be used to calculate the accuracy.

The test data accuracy we obtain after applying Knn classifier is 47 percent and train data accuracy we obtained is 73 percent.

Knn regression model is applied to the data and We will use a function KNeighborsRegressor from **sklearn** library to create our Regression model using the fit() method on the training data set.

Once the model is fit, we can find the optimal parameter of K ,obtained through GridSearchCV. The value obtained for K is 14 for Regression.

Actual and predicted values are compared and Graph is plotted between actual and predicted values for closing stock prices.Given below is the graph that we obtained for actual and predicted closing price.
Blue:model predicted prices
Red:Actual prices



Stock Price Prediction using SVM

This part consists of stock price prediction using the SVM model.

1. Getting our data-

Important libraries are imported. We downloaded the past 1 year data of Reliance Industries Trading in NSE from yahoo finance website. The Data is stored in OHLC(Open, High, Low, Close) format in a CSV file. To read a CSV file, we used the read_csv() method of pandas. We then reset the index and set the index of our dataframe to make sure the dates of our stock prices are a column in our dataframe.

2. Data Visualizations-

We get our adjusted closing prices from our dataframe and we plot a rolling mean on our data to see the long-term trend of the data. We define our moving average window to be 25. We define 25 because we want to see the moving average over the long term in our data. We then create a copy of our dataframe and called it dates_df. We store our original dates in org_dates. We will use org_dates later to plot our predictions and dates.

3. Support Vector Regression

We then convert our dates_df dates to integers by using mdates.date2num. We need the dates as integers because you can't feed dates into Support Vector Machines. We use Sklearn and Support Vector Regression (SVR) to predict the prices on our data. Plotting the predicted price and the original price vs date, we observed that the model fits our data very well, but it is most likely overfit. Thus, the SVR model would have a hard time generalizing on a month of unseen RELIANCE stock data.

4. Support Vector Classifier

We create explanatory or independent variables to predict the value response variable. The X is a dataset that holds the variables which are used for prediction. The X consists of variables such as 'Open – Close' and 'High – Low'. These can be understood as indicators based on which the algorithm will predict tomorrow's trend. The profit or loss calculation is usually determined by the closing price of a stock for the day, hence we will consider the closing price as the target variable. The target variable is the outcome which the machine learning model will predict based on the explanatory variables. y is a target dataset storing the correct trading signal which the machine learning algorithm will try to predict. If tomorrow's price is greater than today's price then we will buy the particular Stock else we will have no position in the. We will store +1 for a buy signal and 0 for a no position in y. We will use where() function from NumPy to do this. We then split the data into training and test datasets. We use SVC() function from **sklearn.svm.SVC** library to create our classifier model using the fit() method on the training data set. We computed the accuracy of the algorithm on the train and test data set by comparing the actual values of the signal with the predicted values of the signal.

The function accuracy_score() was used to calculate the accuracy. We got an accuracy of 53.06% for test data and 57.29% for train data. An accuracy of 50%+ in test data suggests that the classifier model is effective. We predicted the signal (buy or sell) using the cls.predict() function. At last, we calculated the daily returns and strategy returns. Plotting them vs date, we see that our strategy seems to be outperforming the performance of the Reliance stock.

Stock Price Prediction using SARIMA Model

This part consists of stock price prediction using the SARIMA model, this model is a quite famous method used in industries to analyze and model time series data.

Data Collection

The data set was downloaded from yahoo finance for Apple (AAPL) for a time period from Oct 1, 2008, till Sep 30, 2019. The Data Is stored in OHLC(Open, High, Low, Close) format in a CSV file.

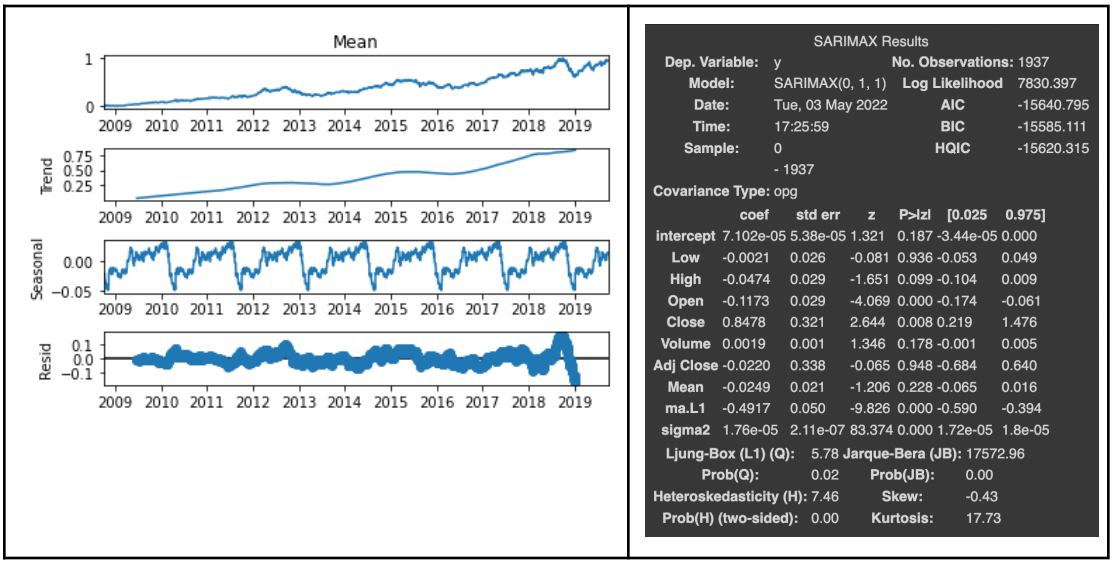
Pre-Processing and Method

The data needs to be processed before we use it . We need to remove any null values and extract features from our data to get the better prediction from our model.
We started by making date as our index column, then removed any null values by dropping them as there were not many non-values.
We then normalize the data which helps the model to train better and increases accuracy of our model by reducing weights of our model during training.
We then extracted useful features like the number of lags to use in our model.
Splitted our data into training and testing data.
Then model parameters p, d, q and P, D, Q were determined.

Framework used to implement this model is:

- Plot the series – to check for outliers
- Transform the data (to make mean and variance constant)
- Apply statistical tests to check if the series is stationary (Both trend and seasonality)
- If non-stationary (has either trend or seasonality), make it stationary by differencing
- Plot ACF of stationary series for MA order, Seasonal MA order at seasonal spikes
- Plot PACF of stationary series for AR order, Seasonal AR order at seasonal spikes
- Run SARIMA with those parameters
- Check for model validity using residual plots

Results



SARIMAX Results

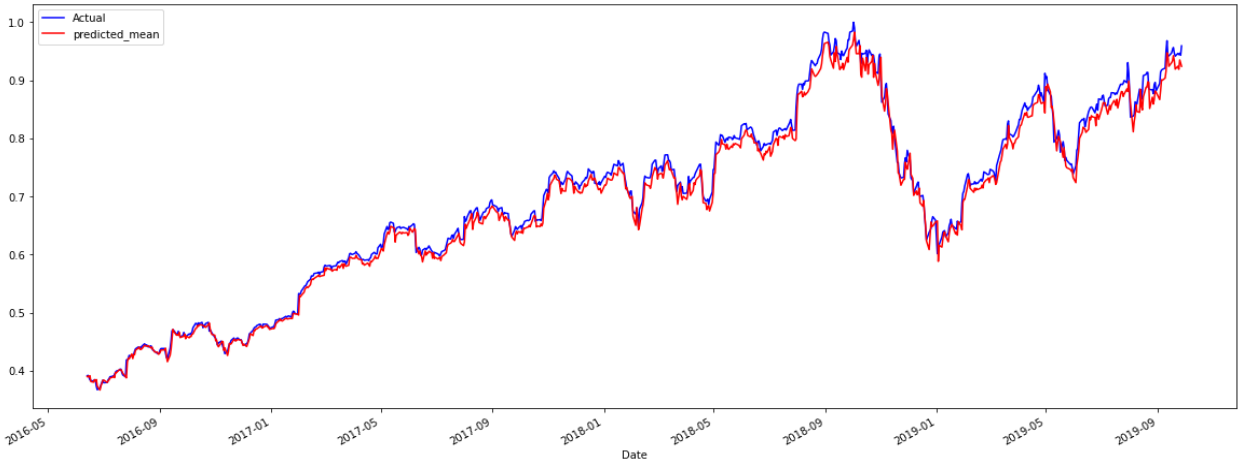
Dep. Variable:	y			**No. Observations:**	1937	
Model:	SARIMAX(0, 1, 1)			**Log Likelihood**	7830.397	
Date:	Tue, 03 May 2022			**AIC**	-15640.795	
Time:	17:25:59			**BIC**	-15585.111	
Sample:	0			**HQIC**	-15620.315	
	- 1937					
Covariance Type:	opg					

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	7.102e-05	5.38e-05	1.321	0.187	-3.44e-05	0.000
Low	-0.0021	0.026	-0.081	0.936	-0.053	0.049
High	-0.0474	0.029	-1.651	0.099	-0.104	0.009
Open	-0.1173	0.029	-4.069	0.000	-0.174	-0.061
Close	0.8478	0.321	2.644	0.008	0.219	1.476
Volume	0.0019	0.001	1.346	0.178	-0.001	0.005
Adj Close	-0.0220	0.338	-0.065	0.948	-0.684	0.640
Mean	-0.0249	0.021	-1.206	0.228	-0.065	0.016
ma.L1	-0.4917	0.050	-9.826	0.000	-0.590	-0.394
sigma2	1.76e-05	2.11e-07	83.374	0.000	1.72e-05	1.8e-05

Ljung-Box (L1) (Q):	5.78	**Jarque-Bera (JB):**	17572.96
Prob(Q):	0.02	**Prob(JB):**	0.00
Heteroskedasticity (H):	7.46	**Skew:**	-0.43
Prob(H) (two-sided):	0.00	**Kurtosis:**	17.73

We will split the data into training(70%) and test data(30%) sets. This is done so that we can evaluate the effectiveness of the model in the test dataset.

The following graph was obtained:



Evaluating the model

We will use **root mean square error** for evaluating the model's performance

It turns out to be 0.014650051337007123 on calculation

Stock Price Prediction Using FNN and CNN

Overview

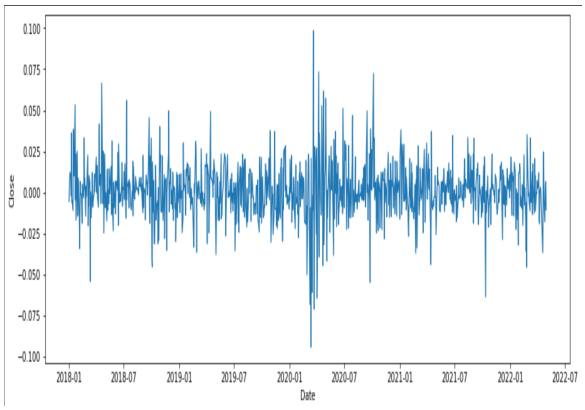
We have used the **Feed Forward Neural Network (FNN)** and **1-D Convolutional Neural Network (CNN)** model to predict the **closing stock price**.
The historical stock data was accumulated, cleaned, new features were added and normalized, then split into training, testing and cross validation dataset. Then both the models were trained and mean squared error is used for calculating error.
The mean squared error (mse) of FNN and CNN on the validation set were **0.0078** and **0.00027** respectively, while on the test set it was 0.0118 , 0.000176 on normalized prices.

Data Collection and Preprocessing

Here I have used stock price data of **Tata Consultancy Services Limited (TCS.NS)** obtained from yahoo! Finance website.
This dataset consists of observations from *2018-01-01* to *2022-04-29* a total of **1069** numbers of observational columns of respective *Date*, *Opening Price*, *Highest Price*, *Lowest price*, *Closing Price*, *Adj Close*, and *Volume* sold.

Stationarity of Data

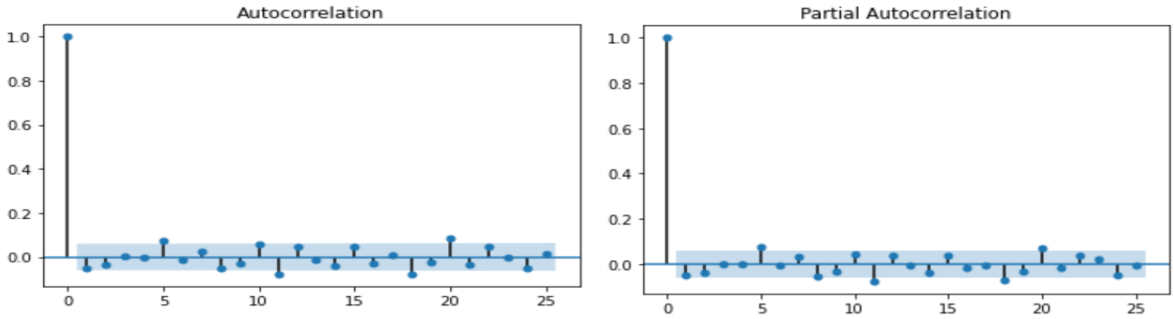
Our Time Series data of closing price of stock was not stationary with time, it has an upward trend so to make it stationary percent change with previous day price was taken.
Dicky-Fuller Test for time series stationarity was used to insure the stationarity of our data results of this test are shown below.




```
Results of Dickey-Fuller Test:
Test Statistic                  -1.038938e+01
p-value                          2.039856e-18
# Lags Used                      1.000000e+01
Number of Observations Used      1.056000e+03
Critical Value (1%)             -3.436558e+00
Critical Value (5%)             -2.864281e+00
Critical Value (10%)            -2.568229e+00
dtype: float64
```

Lag Features

Then we used Auto Correlation Function (ACF) and Partial AutoCorrelation Function (PACF) plots to determine the number of **Lags** to use for our data.



Then from that observation we have used **10 lags** for each of our feature columns Open, Close, High, Low, Adj Close, Volume. After dropping the Nan value columns generated during Lagging of features we finally get our processed dataset of size - *1057 rows(observations) and 66 columns (features)*.

Train, Test, Validation Data Split

Our dependent variable Closing Price was extracted from processed data and then whole data was splitted into 90% training data and the other 10% was further splitted into 60% validation and 40% testing data. After splitting the final dimension of our data was as follows-
Training - 951 x 65
Validation - 64 x 65
Testing - 42 x 65

FNN - Feed Forward Neural Network

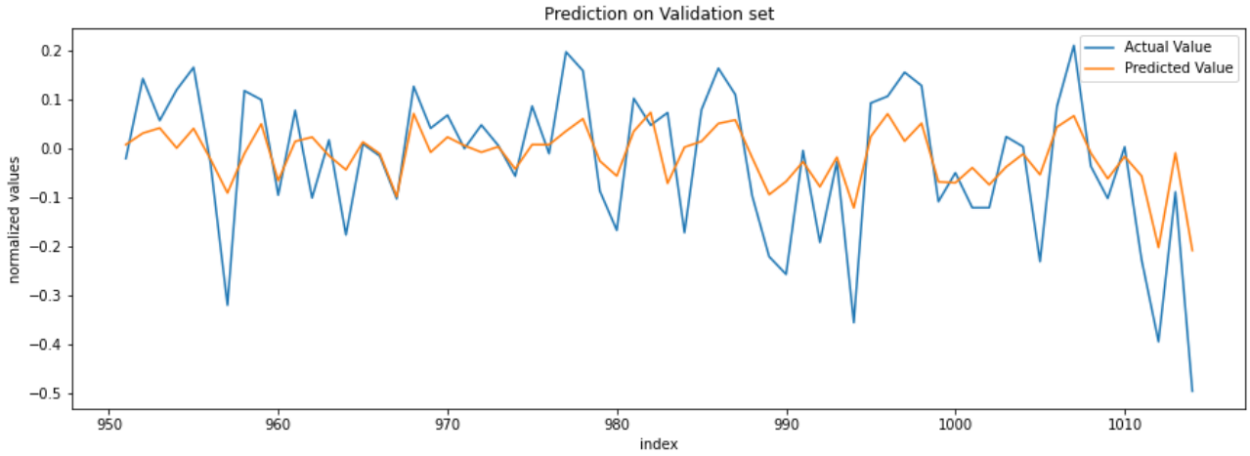
Model parameters

We used feed forward neural network implemented using pytorch library, with tanh as activation function our model parameters are as follows -
Input layer - 65 nodes = number of features
First hidden layer = 30 nodes
Second hidden layer = 10 nodes
Output layer = 1 node //regression task

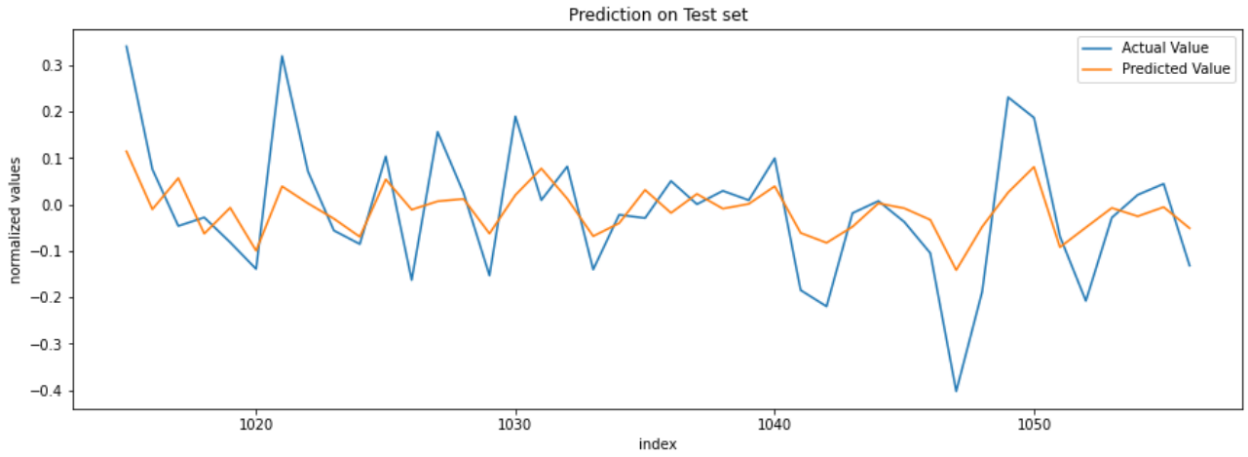
```
<bound method Module.parameters of Model(
  (linear_stack): Sequential(
    (0): Linear(in_features=65, out_features=30, bias=True)
    (1): Tanh()
    (2): Linear(in_features=30, out_features=10, bias=True)
    (3): Tanh()
    (4): Linear(in_features=10, out_features=1, bias=True)
  )
)>
```

Performance

The performance on cross validation set is as follows-
mse loss = 0.010695028814331535



The performance on test set is as follows -
Mse loss = 0.011798989589535801



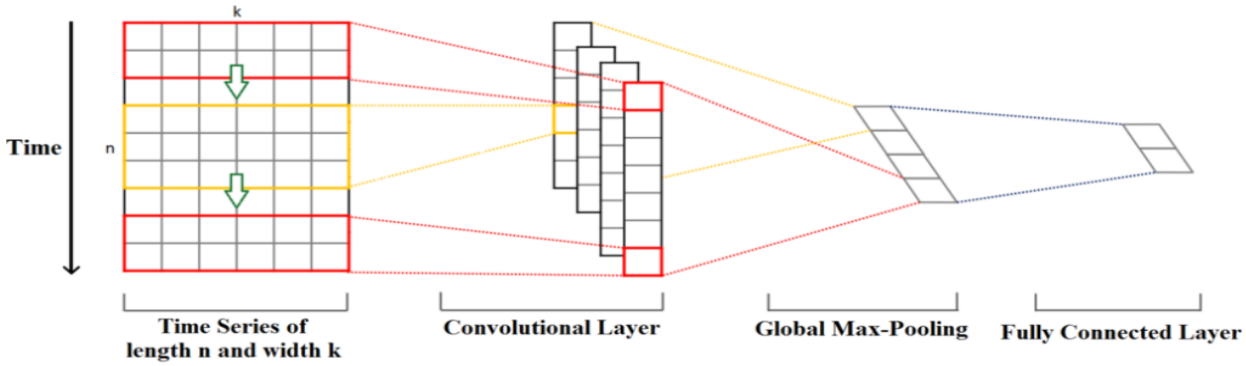
Hyper-parameters

One can tune the following Hyper-parameters to best fit the model-

1. Number of hidden layers - we used 2
2. Number of nodes in hidden layers
3. Batch size - we used 50
4. Number of Epochs - we used 100 epochs
5. Learning rate - we used 0.01
6. Optimizer - we used Stochastic Gradient Descent
7. Loss function - we used MSE loss

1-D CNN - 1 Dimensional Convolutional Neural Network

Till now our FNN model was only seeing the data cross-sectionally. What that means is that we were only using data from the previous day to predict today's value.
But now we will now be using a specific time step so values from the past, “timesteps” number of observations will be used to condense the information and predict it for tomorrow.
We are using *timestep of 30 days* so final shape of our dataset is -
Training set - previously was 951x65 - now 921x30x65
Validation set - previously was 64x65 - now 34x30x65
Training set - previously was 42x65 - now 12x30x65



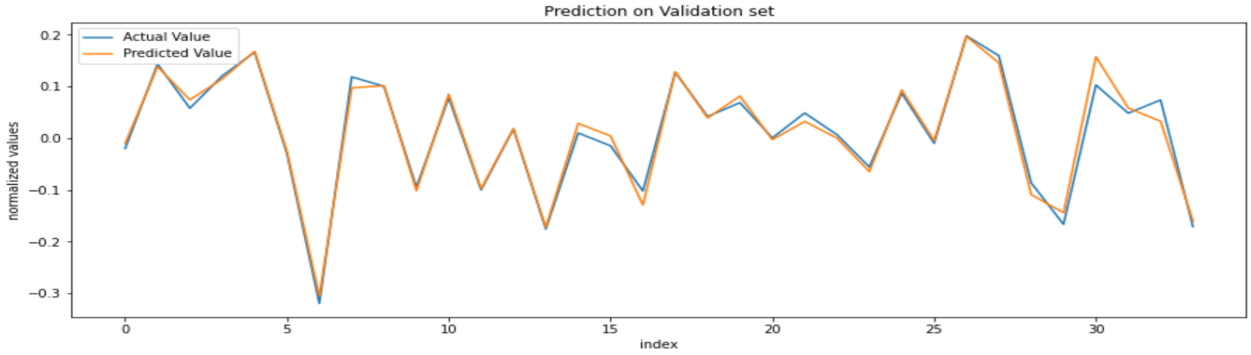
Model parameters

Here n=timestep=30 and k=number of features = 65
We are using *5 number of filters*, *kernel of 2x65* and *stride of 2* so,
our 1st convolutional layer has dimension 15x5
We max pool all the features so after pooling we have 15x5
No dropout gives 15x5
Flatten layer dimension 75
Dense layer/ output layer dimension 1

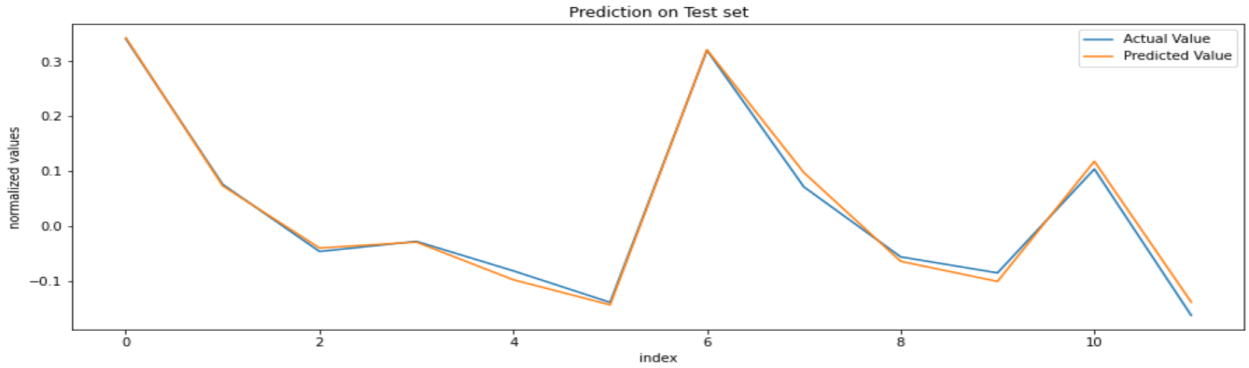
```
 Layer (type)                 Output Shape              Param #
=================================================================
 conv1d_3 (Conv1D)            (None, 15, 5)             655

 max_pooling1d_3 (MaxPooling  (None, 15, 5)             0
 1D)

 dropout_3 (Dropout)          (None, 15, 5)             0

 flatten_3 (Flatten)          (None, 75)                0

 dense_3 (Dense)              (None, 1)                 76

=================================================================
Total params: 731
Trainable params: 731
Non-trainable params: 0
```

Performance

The performance on cross validation set is as follows-
Mse loss = 0.0002761477009136897



The performance on test set is as follows -
Mse loss = 0.00017609081942614242



Hyper-parameters

One can tune the following Hyper-parameters to best fit the model-

1. Value of timestep - we used 30 days
2. Number of filters in convolutional layer - we used 5
3. Kernel size - we use 2x65
4. Stride - we used stride of 2
5. Batch size - we used 20
6. Learning rate - we used 0.001
7. Number of Epochs - we used 1000 epochs
8. Optimizer - we used Stochastic Gradient Descent (SGD)
9. Loss function - we used MSE loss

References Used

KNN Model-

1. https://www.geeksforgeeks.org/predicting-stock-price-direction-using-support-vector-machines/?ref=rp
2. https://www.geeksforgeeks.org/videos/stock-price-prediction-in-machine-learning/?ref=gcse

SVM Model-

1. https://www.geeksforgeeks.org/predicting-stock-price-direction-using-support-vector-machines/?ref=rp
2. Walking through Support Vector Regression and LSTMs with stock price prediction | by Drew Scatterday | Towards Data Science

SARIMA Model-

1. https://towardsdatascience.com/understanding-sarima-955fe217bc77
2. https://www.wisdomgeek.com/development/machine-learning/sarima-forecast-seasonal
3. https://medium.datadriveninvestor.com/step-by-step-time-series-analysis-d2f117554d7e
4. https://medium.datadriveninvestor.com/time-series-prediction-using-sarimax-a6604f258c56

FNN and CNN Model-

1. https://www.youtube.com/watch?v=Rr-ztgKuaSA&t=3932s
2. https://visualstudiomagazine.com/articles/2021/02/11/pytorch-define.aspx
3. https://www.youtube.com/watch?v=JzoIHdkFcQU
4. https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Link of reference is attached as comment wherever code was taken from the internet otherwise the code was written by us.