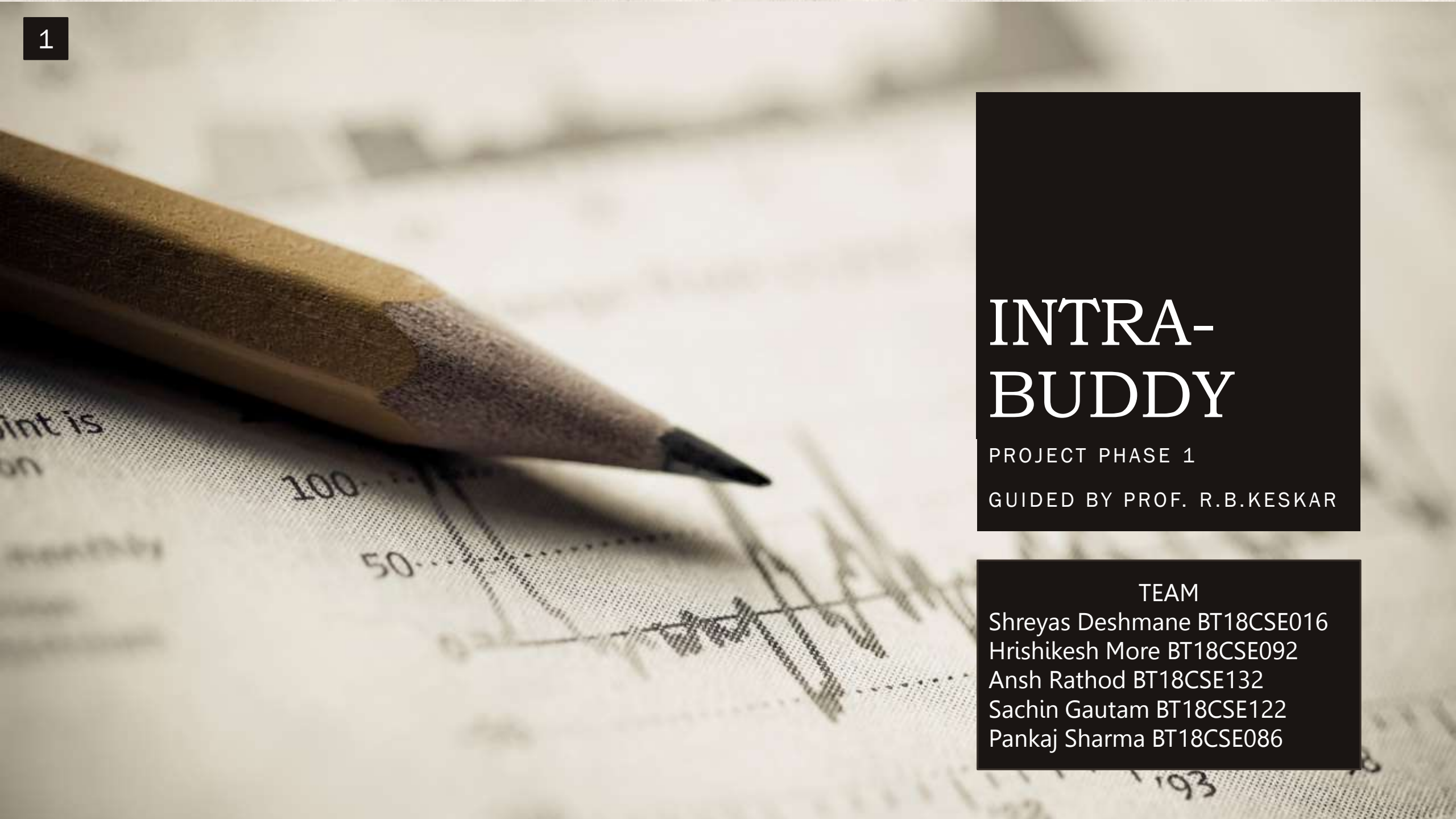


INTRA-BUDDY

PROJECT PHASE 1

GUIDED BY PROF. R.B.KESKAR

TEAM
Shreyas Deshmane BT18CSE016
Hrishikesh More BT18CSE092
Ansh Rathod BT18CSE132
Sachin Gautam BT18CSE122
Pankaj Sharma BT18CSE086

Key Points:

- What is the problem that we are going to solve?
- How we are going to solve the Problem
- Input parameters and what will be the output?
- How will It facilitate the users?

User with the source chart will approach our team

Our team with 200+ Manually examined chart patterns till now

What is the problem that we are going to solve?

- Every challenge has a solution. What matters is the effectiveness of the solution
- Every person once in a lifetime decides to invest but doesn't know either the proper platform or if someone finds a platform doesn't know Where to invest? When to invest? And How much to Invest ?
- So we are trying to come up with a product to help the user to find the best trading opportunity based on technical analysis in the intra-day market.

How it will work?

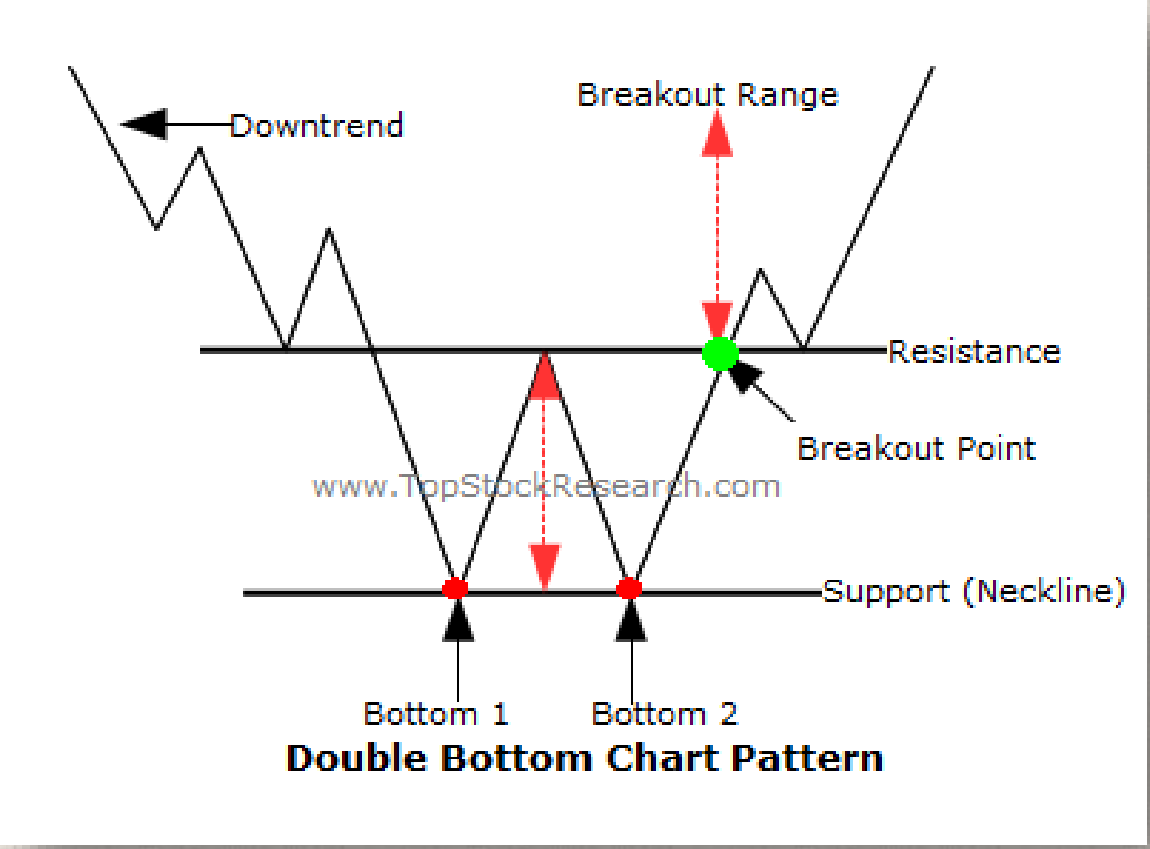
- When The user inputs the source chart our product will compare the charts with all the charts available in the dataset to find if there is a possible optimum match in the dataset.
- After comparing if there is a match present it will check whether it is a bullish double top or Ascending triangle. For example As we know these are bullish Trends so the output for the users will be

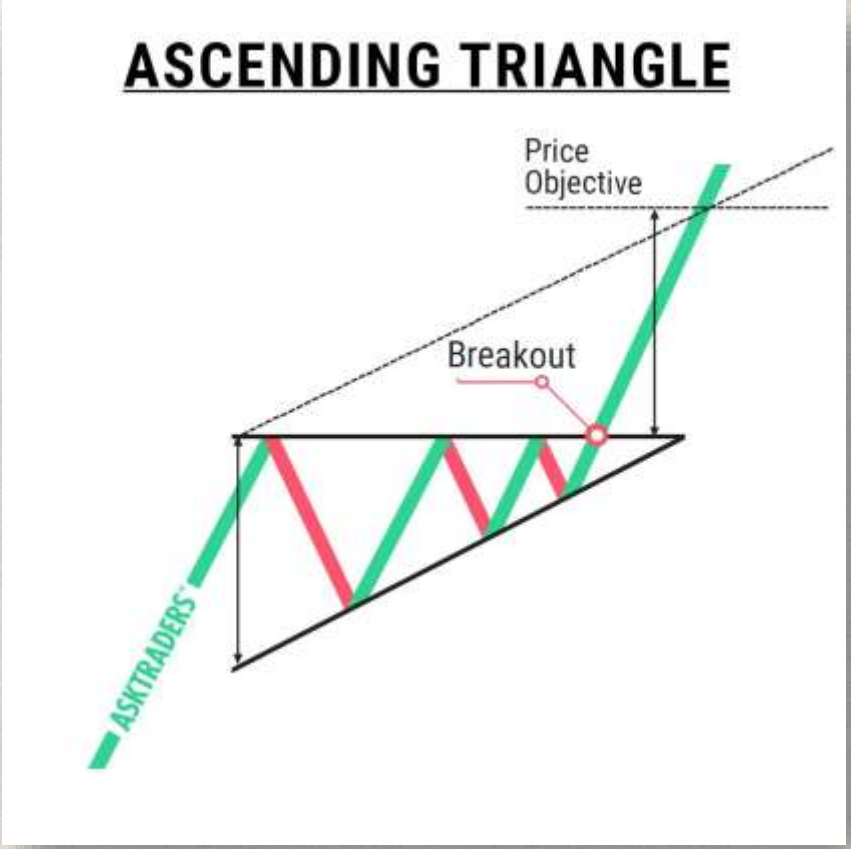
“Bullish Nature Detected”

If after comparing there is no match then –

“No pattern detected”

We have Already Created a dataset with 200+ chart patterns manually on two patterns :
(i) Bullish double top(Double bottom chart pattern)
(ii) Ascending Triangle





Input and Output Parameters

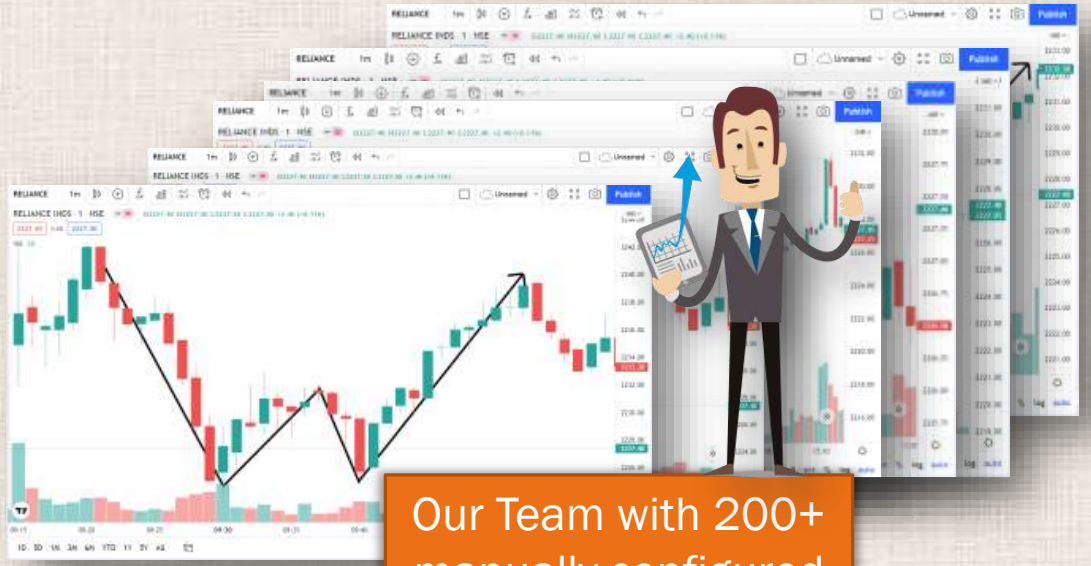
Input :

Taking source image of previous 45mins candlesticks chart in one day(intra-day) column for example

User comes up with a Chart pattern in a 45 min window

~45mins

User with the source chart will approach our team

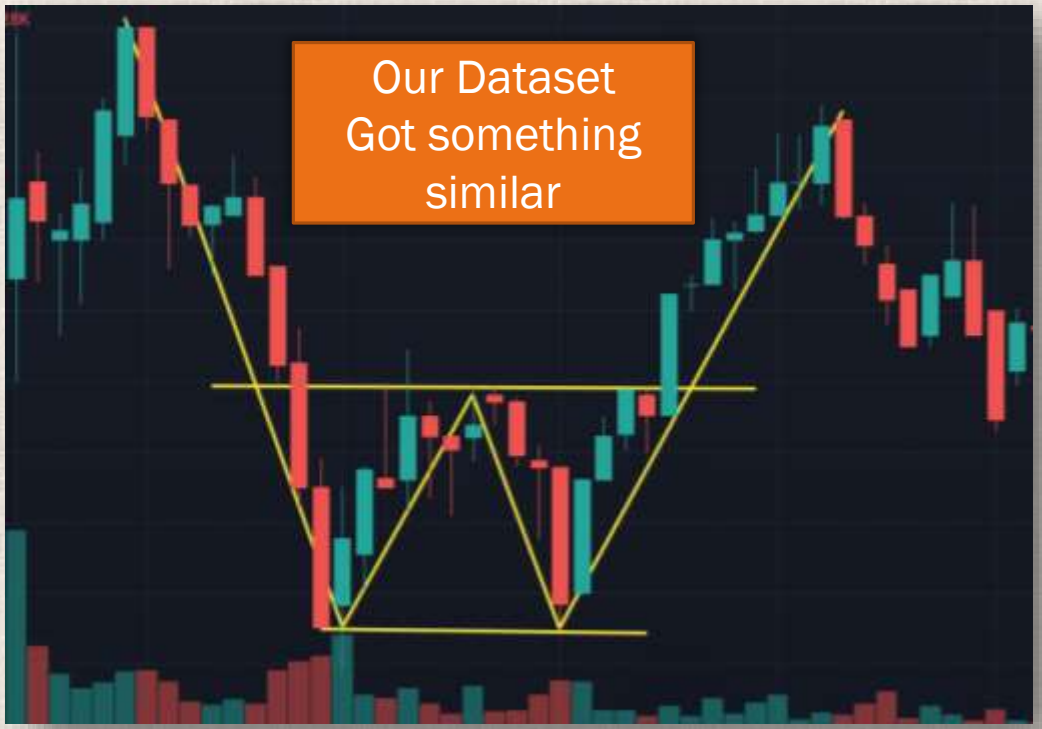


Our Team with 200+ manually configured dataset

Input and Output Parameters

Output :

Whether the chart is present or not , and if present which chart it is



How will It facilitate the users?

- Our Product will basically help the users to find a proper trade opportunity.
- As it is completely based on Technical analysis we can expect a high success rate above 90%.
- To overcome with the remaining 10% the user can allot a stop loss ranging from ± 0.25% to ±1% According to the trade the user wants to execute and the maximum risk he can afford.

How we are going to solve the Problem?

- The problem with estimating the stock price will remain a problem if a better stock market prediction algorithm is not proposed.
- So here We are contemplating towards the study of stock market using technical analysis.
- This is how we are going to Execute:

Step 1: Adding Lots of Analysed charts in training set

Step 2: Taking input from users for target chart

Step 3: Comparing target chart with all charts available in training sets

Step 4: Finding most optimal amongst them

Step 5: Desired output to the user

Possible Algorithms which we are going to work on to find the most optimum solution

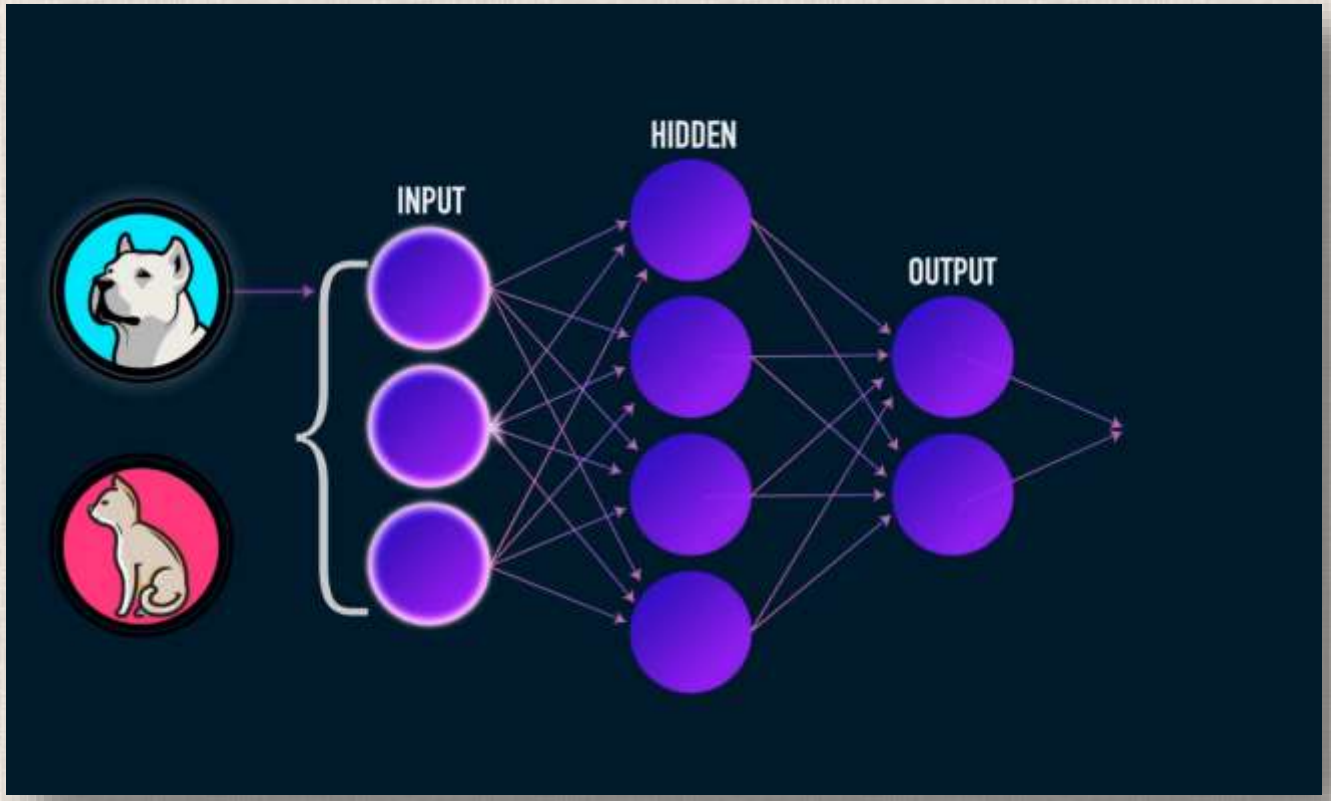
1. CNN
2. Linear Regression
3. Time Series
4. Reinforcement Learning

CNN(convolutional Neural Network)

- CNN takes input,assign importance to various objects in image and differentiate between them.
- 2 types of result :

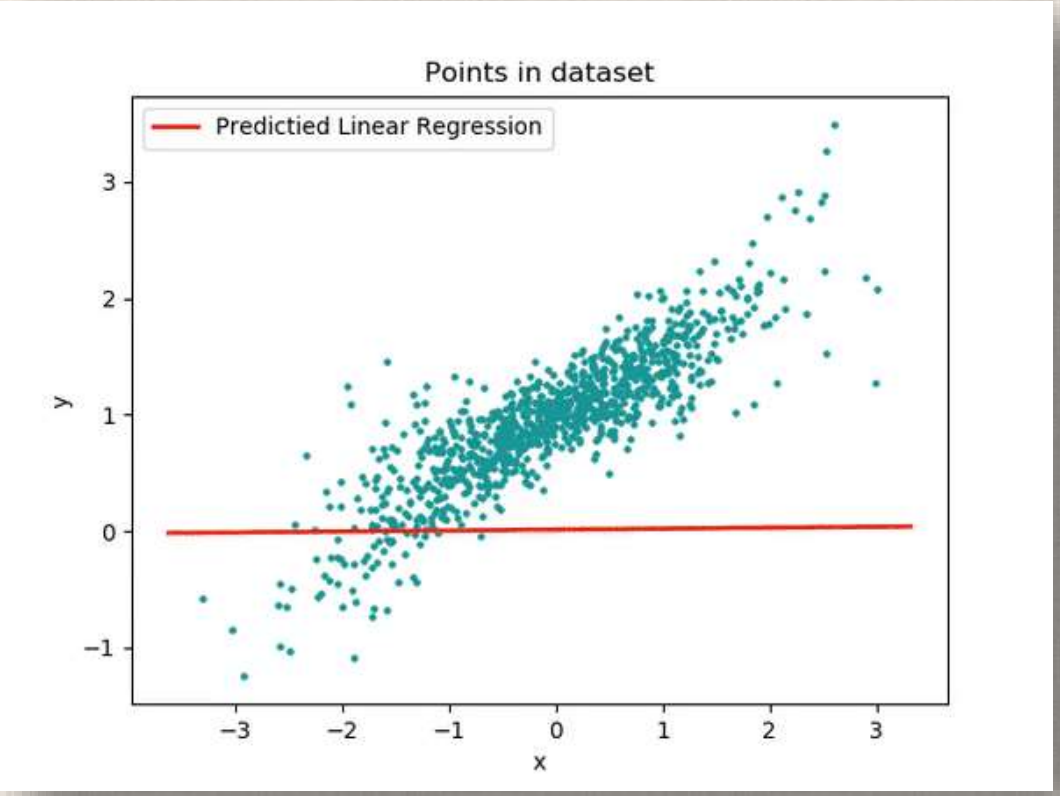
  1.convolved feature is reduced in dimension
  2.convolved feature increases or remains same in dimension

- key architectures-VGGNEt , GoogleNET

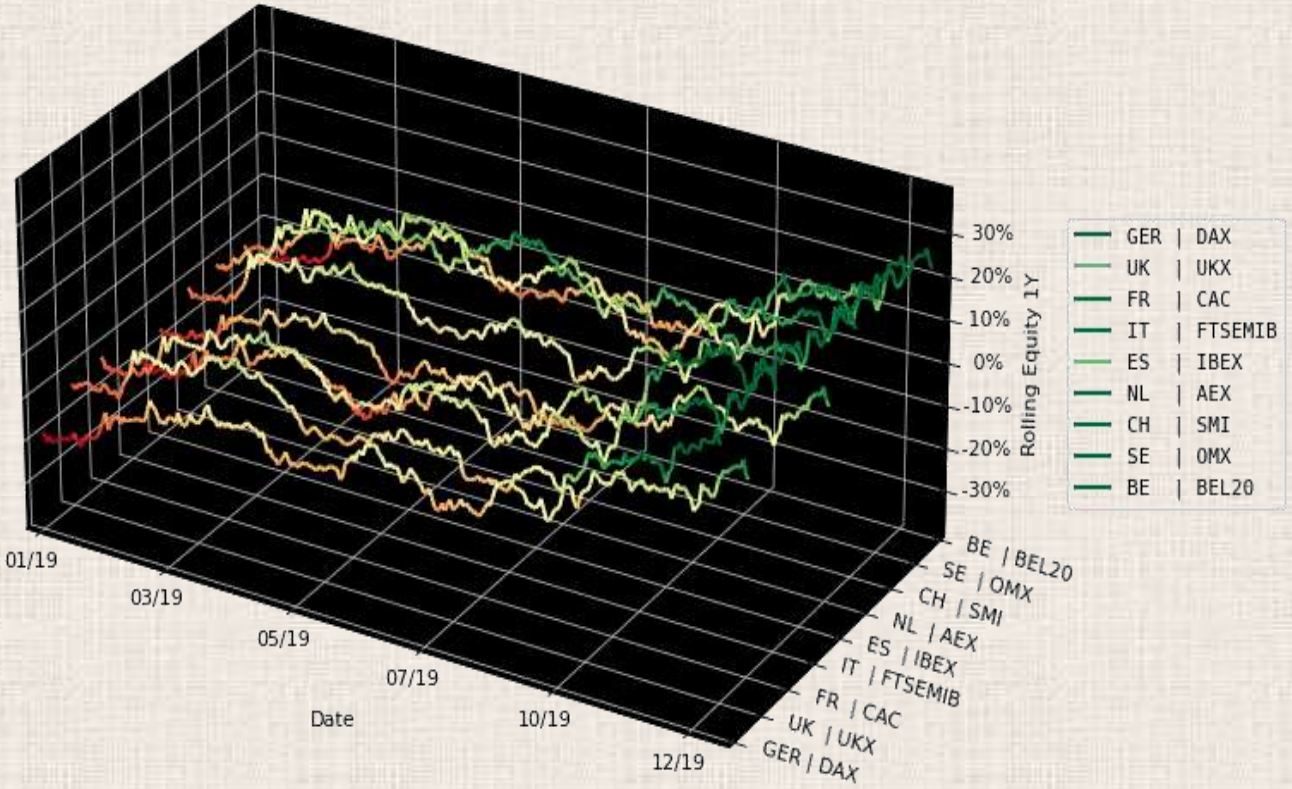


Linear Regression

- Linear regression is used for predictions with data that has numeric target variable.
- During prediction we use some variables as dependent variables and few considered as independent variables.
- In situation when there is one dependent and one independent variable, we prefer to use linear regression methodologies.
- Regression can be single variable or multi variable, it depends upon situation named as single variable or multi variable regression.



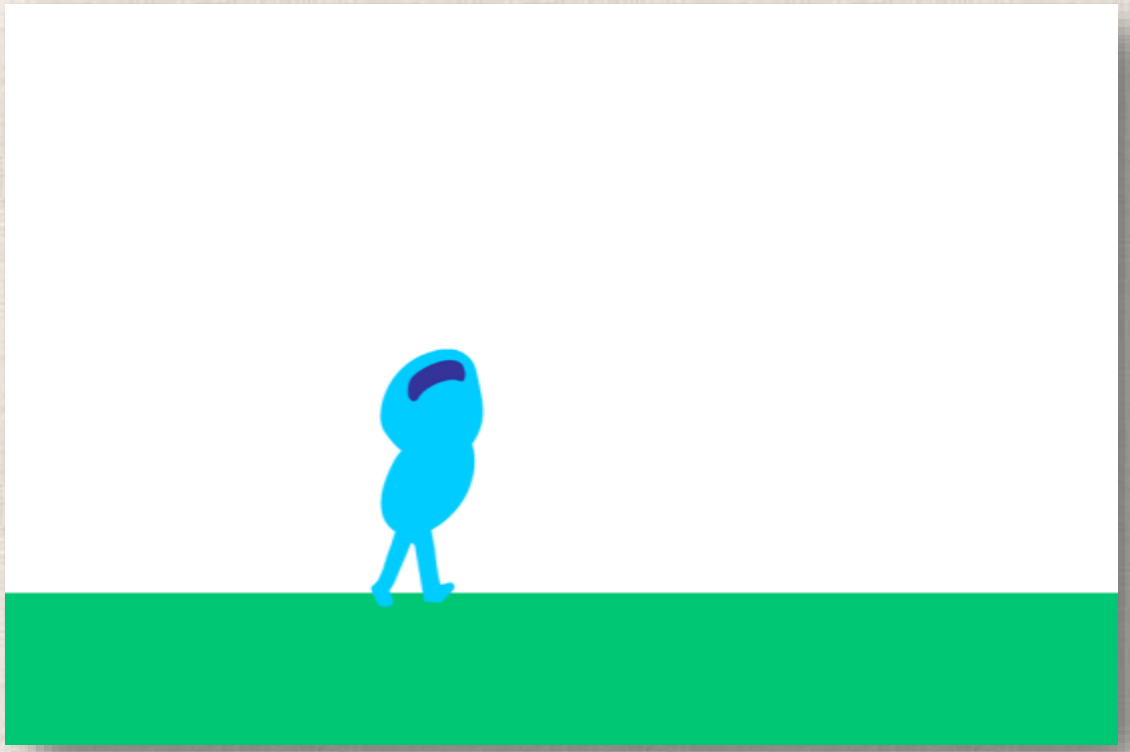
Time Series



- A time series is a data set that tracks a sample over time.
- In particular, a time series allows one to see what factors influence certain variables from period to period.
- Time series analysis can be useful to see how a given asset, security, or economic variable changes over time.
- Forecasting methods using time series are used in both fundamental and technical analysis.
- Although cross-sectional data is seen as the opposite of time series, the two are often used together in practice.

Reinforcement learning (RL)

- It is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward.
- It is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.
- The main focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge)



Literature Survey

- **Chavan and Patil (2013)** contribute to our understanding of ANN stock market prediction by surveying different model input parameters found in nine published articles. They attempt to find the most important input parameters that produce better model prediction accuracy.

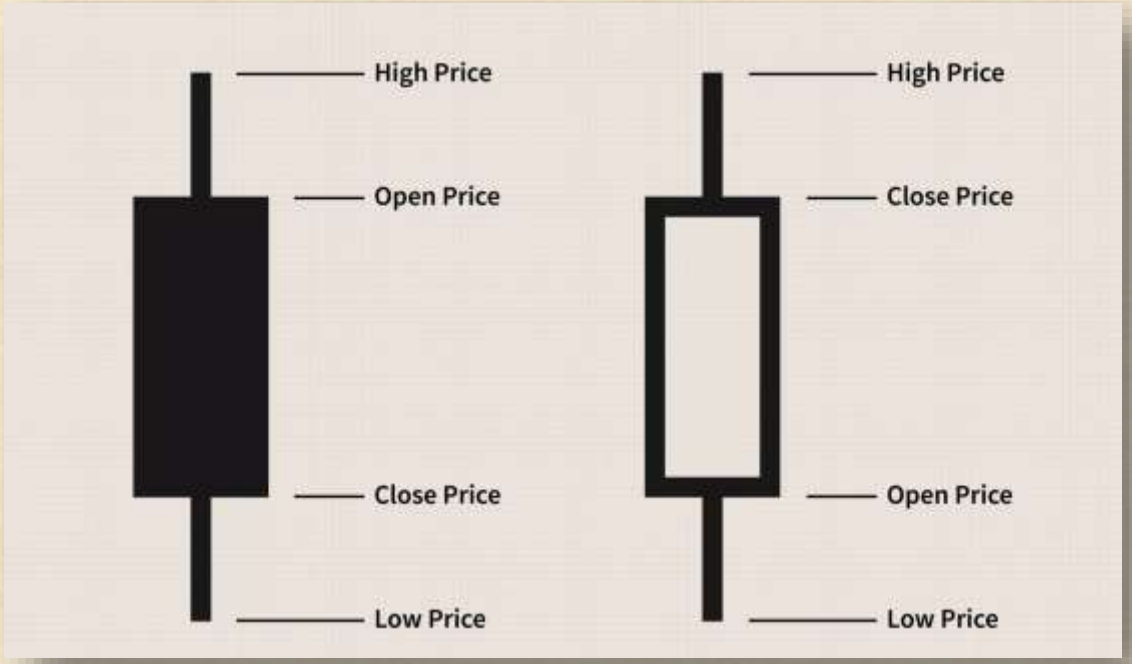
- **Art Paspanthong ,Nick Tantivasadakarn ,Will Vithayapalert(2019)** Convolutional Neural Network model,. RNN models , Support Vector Machine, Baseline Model.This paper explored the usage of multiple machine learning models to predict future prices of stocks. We first simplified our problem to a binary classification problem. Then we narrowed down our data, which consists of multiple indicators, to a smaller and more statistically significant subset.

- **Things used :** Exponential Smoothing Model(ESM), ANN and Support Vector Regression (SVR), Support Vector Machine(SVM), Decision trees, Long Short-Term Memory(LSTM).**Conclusion:** Hybrid Approach applies a combination of multiple different approaches. One such set-based classifier was used which yields an accuracy of 92.1%.

- **Things Used :** Support Vector Machine (SVM) & Radial Basis Function(RBF)Working: Fetching data from the CSV file,With SVM it will select the dataset value File.After Selecting the dataset file it will show graph befor and after Mapping.After that Predict the value of select stock.**Conclusion:** SVM algorithm works on the large dataset value which is collected from different global financial markets. Also, SVM does not give a problem of over fitting.

Key Points:

- Candle Chart
- API used
- How will API Facilitate in Project
- Comparison of API driven charts with Data Set



What is a Candlestick?

This real body represents the price range between the open and close of that day's trading. When the real body is filled in or black, it means the close was lower than the open. If the real body is empty, it means the close was higher than the open.

What is a Candlestick Chart?

Candlestick charts are used to determine possible price movement based on past patterns. Candlesticks show that emotion by visually representing the size of price moves with different colors. Traders use the candlesticks to make trading decisions based on regularly occurring patterns that help forecast the short-term direction of the price.

API Used : ALPHA - VANTAGE

- Alpha Vantage provides enterprise-grade financial market data through a set of powerful and developer-friendly APIs.
- From traditional asset classes (e.g., stocks and ETFs) to forex and cryptocurrencies, from fundamental data to technical indicators, Alpha Vantage is your one-stop-shop for global market data delivered through cloud-based APIs, Excel, and Google Sheets.

How will API(Alpha-Vantage)Facilitate in Project

- API will provide us with the Candlestick Chart of Desired Stock.
- API gathers data of real-time stocks.
- It Helps us to have a chart of any time limit ,as mentioned earlier we'll have a chart of 45min time span in which each candle will be of 5 min

	date	1. open	2. high	3. low	4. close	5. volume
0	2021-10-28 20:00:00	325.04	325.04	324.6600	324.71	2064.0
1	2021-10-28 19:55:00	324.94	325.00	324.8101	325.00	2283.0
2	2021-10-28 19:50:00	324.81	324.81	324.8100	324.81	197.0
3	2021-10-28 19:45:00	324.93	324.94	324.9300	324.94	617.0
4	2021-10-28 19:40:00	324.80	324.80	324.8000	324.80	202.0
...	...	...	...	...	...	...
3387	2021-09-30 04:35:00	285.84	285.84	285.8400	285.84	133.0
3388	2021-09-30 04:30:00	285.96	285.96	285.9600	285.96	302.0
3389	2021-09-30 04:20:00	285.90	285.90	285.9000	285.90	298.0
3390	2021-09-30 04:15:00	286.29	286.29	286.2900	286.29	287.0
3391	2021-09-30 04:05:00	286.00	286.00	286.0000	286.00	115.0

API Gathered data displayed in Charts format using Plotly

Live Charts formed by API

DATASET Charts

Comparison of API driven charts with Data Set

Key Points:

- Dataset Overview
- Classification Using CNN
- Model Training
- Model Testing
- Final Obtained Results

Dataset Overview

An ascending triangle is a chart pattern used in technical analysis. It is **created by price moves that allow for a horizontal line to be drawn along the swing highs**, and a rising trendline to be drawn along the swing lows. The two lines form a triangle. Traders often watch for breakouts from triangle patterns.

Dataset Overview

A double top pattern is a technical analysis charting pattern that describes a change in trend and a momentum reversal from prior leading price action. It describes the drop of a stock or index, a rebound, another drop to the same or similar level as the original drop, and finally another rebound.

Classification Using CNN

```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

```
cv2.imread("/content/drive/MyDrive/Colab Notebooks/Project/basedata/train/AT/1.png")
```

```
train = ImageDataGenerator(rescale= 1/255)
validation = ImageDataGenerator(rescale = 1/255)
```

```
training_dataset = train.flow_from_directory("/content/drive/MyDrive/Colab Notebooks/Project/basedata/train", target_size= (200,200) , batch_size = 3 )
validation_dataset = validation.flow_from_directory("/content/drive/MyDrive/Colab Notebooks/Project/basedata/validation", target_size= (200,200) , batch_size = 3)
```

```
Found 62 images belonging to 2 classes.
Found 16 images belonging to 2 classes.
```

```
training_dataset.class_indices
```

```
{'AT': 0, 'BDT': 1}
```

```
training_dataset.classes
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int32)
```

Model Training

Model Implementation using Keras

```
[ ] model = tf.keras.models.Sequential([ tf.keras.layers.Conv2D(16,(3,3),activation = 'relu' , input_shape = (200,200,3)) ,
                                         tf.keras.layers.MaxPool2D(2,2) ,
                                         #
                                         tf.keras.layers.Conv2D(32,(3,3),activation = 'relu'),
                                         tf.keras.layers.MaxPool2D(2,2),
                                         #
                                         tf.keras.layers.Conv2D(64,(3,3),activation = 'relu'),
                                         tf.keras.layers.MaxPool2D(2,2),
                                         ##
                                         tf.keras.layers.Flatten(),
                                         ##
                                         tf.keras.layers.Dense(512 ,activation = 'relu'),
                                         ##
                                         tf.keras.layers.Dense(1,activation = 'sigmoid')
                                         ])
```

```
[ ] model.compile(loss= 'binary_crossentropy',
                  optimizer = RMSprop(learning_rate = 0.001) ,
                  metrics =['accuracy'])
```

Model Training with 30 Epochs

```
model_fit = model.fit(training_dataset ,
                      steps_per_epoch = 3,
                      epochs = 30,
                      validation_data = validation_dataset)
```

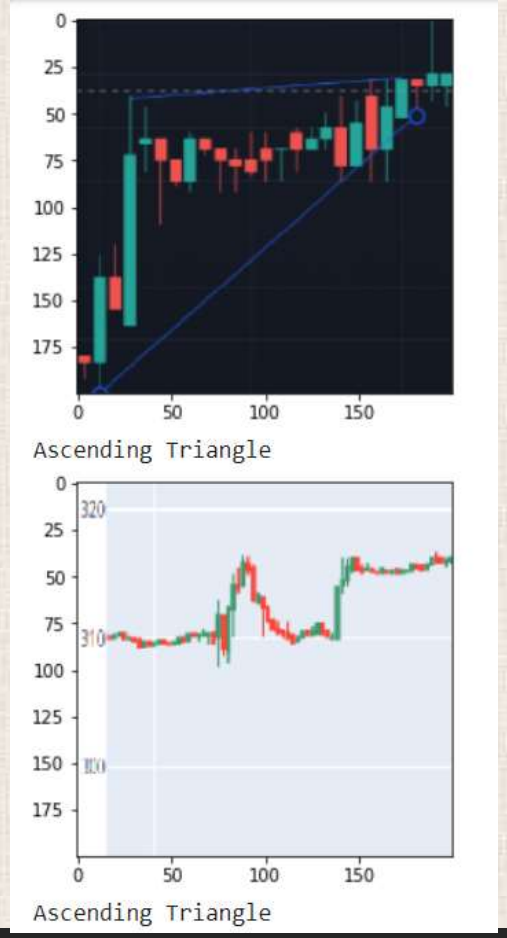
```
Epoch 1/30
3/3 [==============================] - 10s 4s/step - loss: 0.5041 - accuracy: 0.7500 - val_loss: 25.3912 - val_accuracy: 0.3750
Epoch 2/30
3/3 [==============================] - 4s 1s/step - loss: 9.4113 - accuracy: 0.3333 - val_loss: 0.6271 - val_accuracy: 0.3750
Epoch 3/30
3/3 [==============================] - 3s 931ms/step - loss: 0.9086 - accuracy: 0.4444 - val_loss: 0.5741 - val_accuracy: 0.8125
```

Model Testing

Taking Test images at “test” folder and testing all using the CNN Algorithm

```
dir_path = "/content/drive/MyDrive/Colab Notebooks/Project/basedata/test"

for i in os.listdir(dir_path):
    img = image.load_img(dir_path +'//'+ i,target_size=(200,200))
    plt.imshow(img)
    plt.show()

    X = image.img_to_array(img)
    X = np.expand_dims(X,axis =0)
    images = np.vstack([X])
    val = model.predict(images)
    if (val != 1 and val != 0) :
        print("No Pattern")
    if val == 0 :
        print(" Ascending Triangle")
    if val == 1 :
        print("Bullish Double top")
```

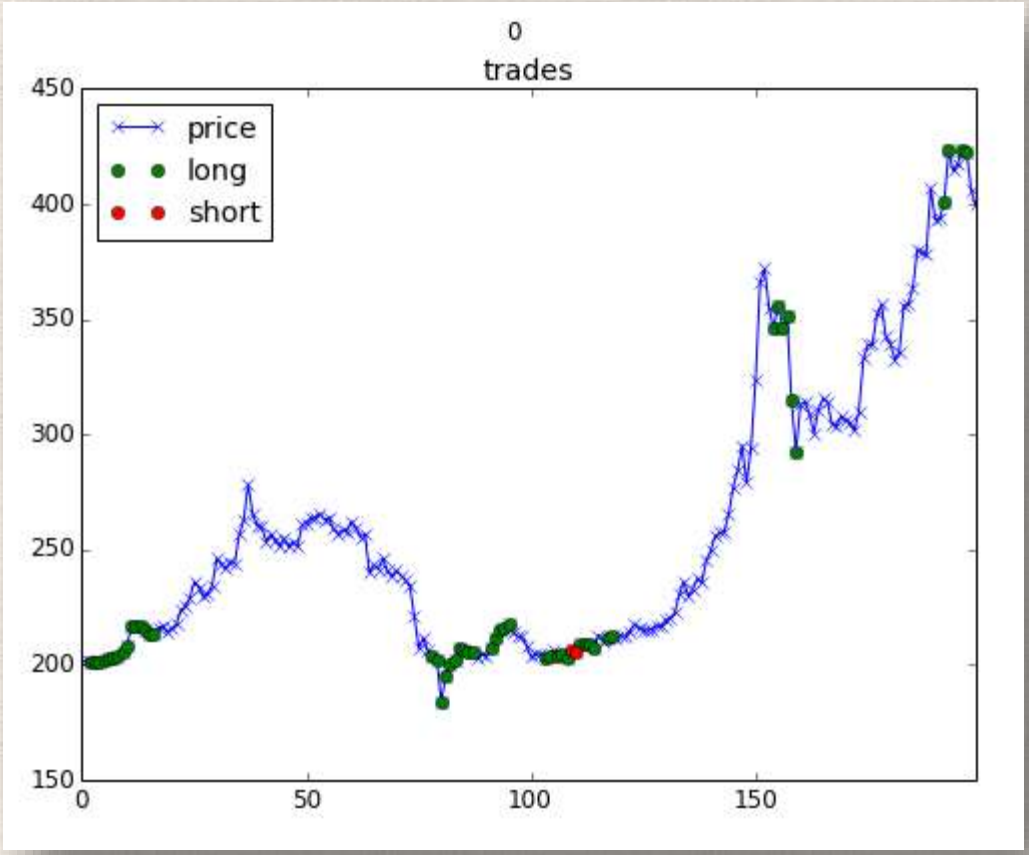
Final Obtained Results



Predicted Results
With 75%+ accuracy

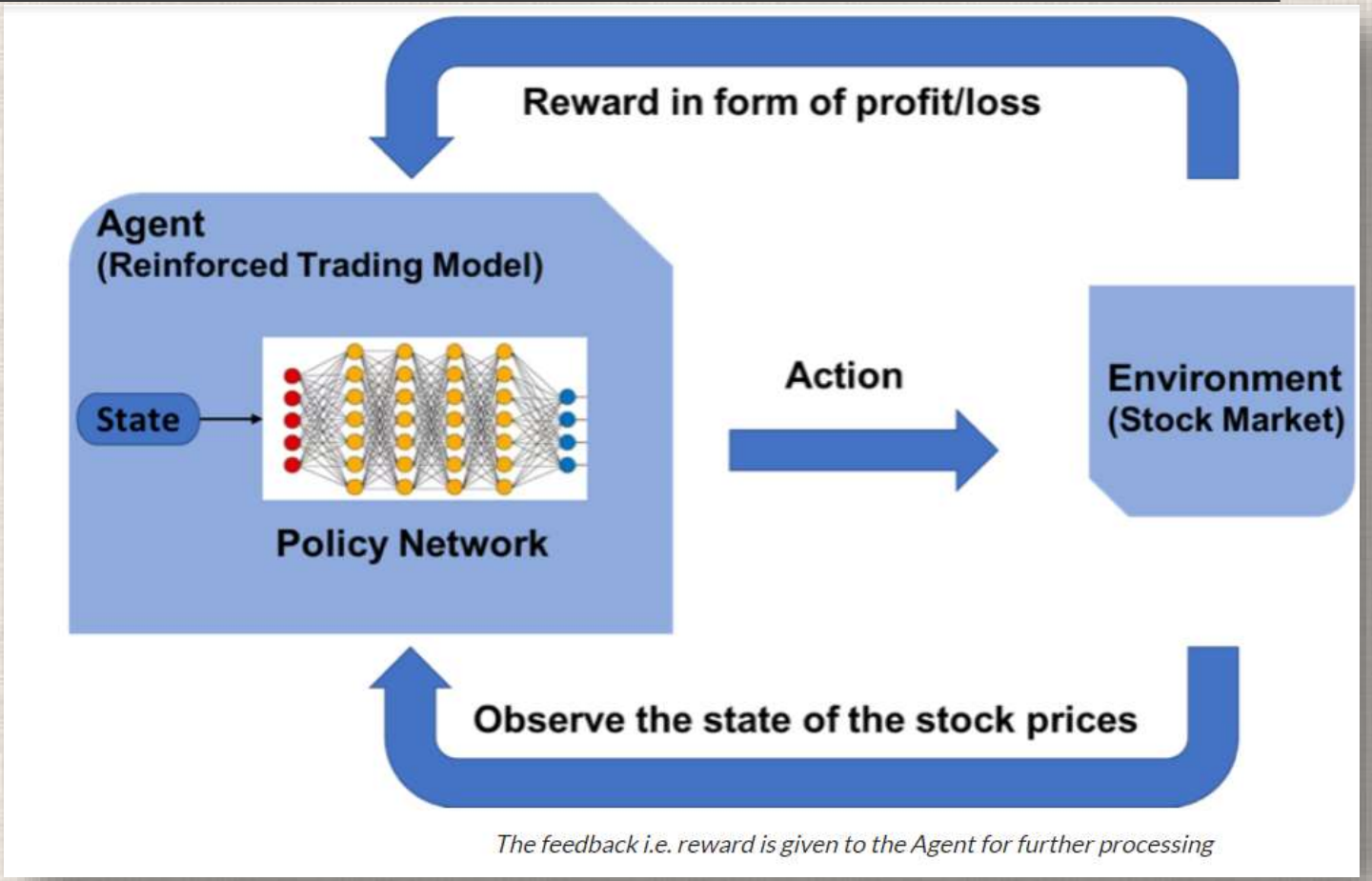
Reinforcement Learning

- The concept of reinforcement learning can be applied to the stock price prediction for a specific stock as it uses the same fundamentals of requiring lesser historical data, working in an agent-based system to predict higher returns based on the current environment.
- We will see an example of stock price prediction for a certain stock by following the reinforcement learning model. It makes use of the concept of Q learning explained further.



Steps for designing a reinforcement learning model is –

- Importing Libraries
- Create the agent who will make all decisions
- Define basic functions for formatting the values function, reading the data file, etc
- Train the agent
- Evaluate the agent performance
- Creating The Environment
- Evaluating the model



Define the Reinforcement Learning Environment

- Agent – An Agent A that works in Environment E
- Action – Buy/Sell/Hold
- States – Data values
- Rewards – Profit / Loss

The Role of Q – Learning : Q-learning is a model-free reinforcement learning algorithm to learn the quality of actions telling an agent what action to take under what circumstances. Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over any successive steps, starting from the current state.

Obtaining Data(5min Timeframe)

- We're Feeding the Algorithm with Live Data Gathered through API –Vantage

```
[ ] pip install alpha_vantage
```

```
import pandas as pd
import matplotlib.pyplot as plt
from alpha_vantage.timeseries import TimeSeries
```

```
[ ] API_key = 'OFYE2132300HHV4I'
```

```
[ ] ts = TimeSeries(key = API_key,output_format='pandas')
    data, meta = ts.get_intraday('MSFT', interval = '5min' , outputsize = 'full')
```

```
df = data
df.dtypes
```

```
date          datetime64[ns]
1. open              float64
2. high              float64
3. low               float64
4. close             float64
5. volume            float64
dtype: object
```

```
df.set_index('date', inplace=True)
df.head()
```

date	1. open	2. high	3. low	4. close	5. volume
2022-02-18 20:00:00	285.78	285.8500	285.70	285.80	8385.0
2022-02-18 19:55:00	285.80	285.8500	285.80	285.85	2682.0
2022-02-18 19:50:00	285.80	285.8500	285.80	285.85	2008.0
2022-02-18 19:40:00	285.75	285.7500	285.75	285.75	1738.0
2022-02-18 19:35:00	285.80	285.8201	285.75	285.75	1585.0

Creating our environment

NOTE : Ideally, the data that you use should mimic the frequency that you want to trade. For example, if you want the RL agent to trade daily data, use the daily data to train the agent and not hourly data.
(we're using 5 min Timeframe)

```
[ ]  #passing the data and creating our environment
     env = gym.make('stocks-v0', frame_bound=(5,100), window_size=5)
```

```
     env.signal_features
```

```
array([[ 1.96946945e+02,  0.00000000e+00],
       [ 2.02382385e+02,  5.43544000e+00],
       [ 2.02982986e+02,  6.00601000e-01],
       [ 2.05405411e+02,  2.42242500e+00],
       [ 2.08823822e+02,  3.41841100e+00],
       [ 2.13493500e+02,  4.66967800e+00],
       [ 2.14414413e+02,  9.20913000e-01],
       [ 2.16041046e+02,  1.62663300e+00],
       [ 2.20360367e+02,  4.31932100e+00],
       [ 2.22382385e+02,  2.02201800e+00],
       [ 2.19604599e+02, -2.77778600e+00],
       [ 2.18028030e+02, -1.57656900e+00],
       [ 2.16516510e+02, -1.51152000e+00],
       [ 2.14714722e+02, -1.80178800e+00],
       [ 2.12632629e+02, -2.08209300e+00],
       [ 2.08593597e+02, -4.03903200e+00]
```

window_size = previous timesteps our trading bot has as reference points

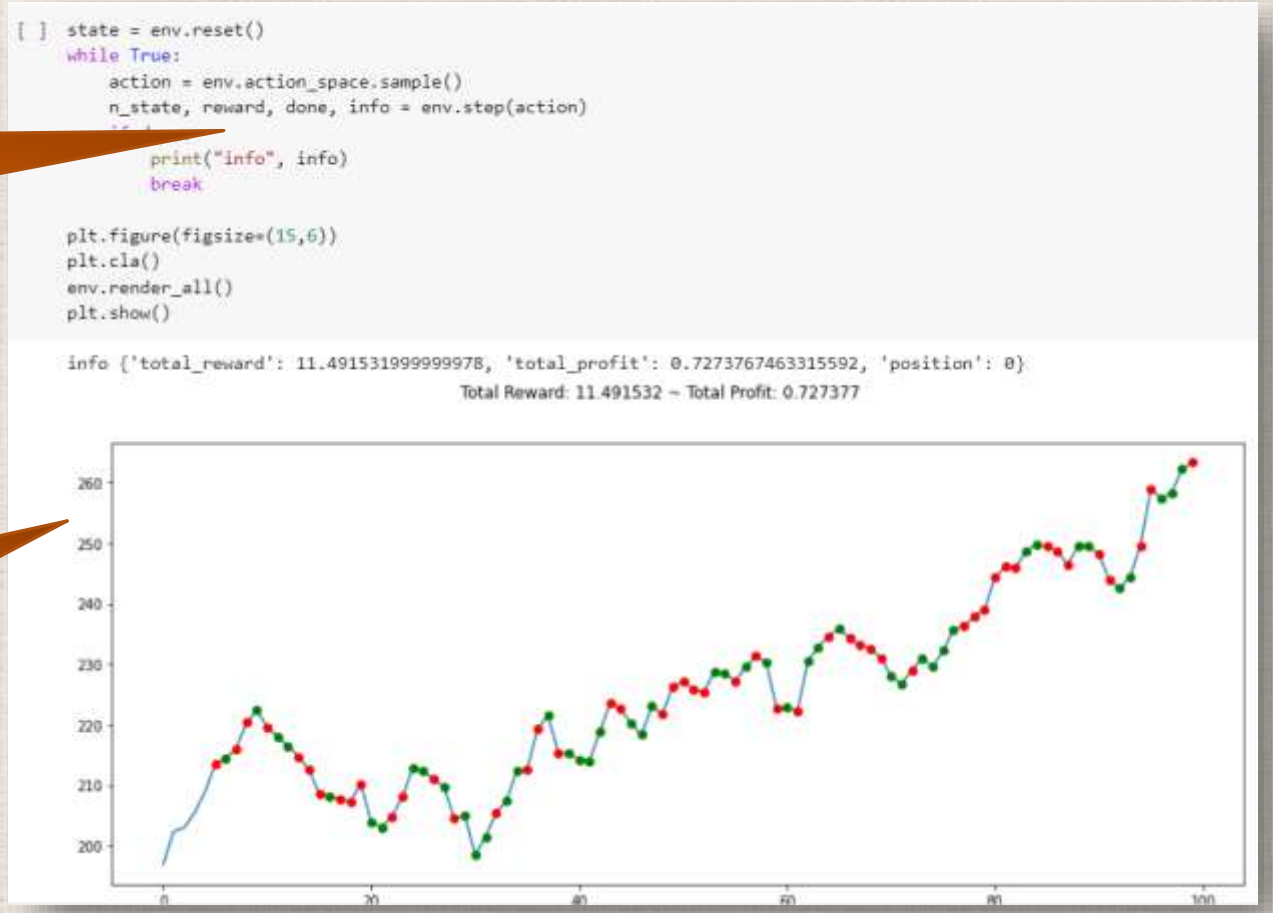
Building the test environment

If we look at the actions we can take using `environment.action_space`, we'll notice that we only have two actions we can take. We can only `Short` or `Long`. In other algorithms, you can `Hold`.

visualizing the environment using `matplotlib`

```
[ ] state = env.reset()
    while True:
        action = env.action_space.sample()
        n_state, reward, done, info = env.step(action)
        if done:
            print("info", info)
            break

    plt.figure(figsize=(15,6))
    plt.cla()
    env.render_all()
    plt.show()
```

```
info {'total_reward': 11.491531999999978, 'total_profit': 0.7273767463315592, 'position': 0}
```



Training an RL agent to trade using the Gym environment

1. We begin by wrapping our environment inside the dummy vectorized environment wrapper, ***DummyVecEnv.***
2. creating an env_build function. We are taking that function and putting it inside the ***DummyVecEnv.***
3. Finally, we save the result inside the env variable so that when we start building our training model. We'll now use the env variable.

```
env_maker = lambda: gym.make('stocks-v0',  frame_bound=(5,100), window_size=5)
env = DummyVecEnv([env_maker])
```

```
model = A2C('MlpLstmPolicy', env, verbose=1)
model.learn(total_timesteps=10000)
```

```
---------------------------------
| explained_variance | -0.909   |
| fps                | 21       |
| nupdates           | 1        |
| policy_entropy     | 0.693    |
| total_timesteps    | 5        |
| value_loss         | 7.41     |
---------------------------------
---------------------------------
| explained_variance | -23.4    |
| fps                | 381      |
| nupdates           | 100      |
| policy_entropy     | 0.693    |
| total_timesteps    | 500      |
| value_loss         | 0.000359 |
---------------------------------
---------------------------------
| explained_variance | -0.0101  |
| fps                | 406      |
| nupdates           | 200      |
| policy_entropy     | 0.693    |
| total_timesteps    | 1000     |
```

Evaluation

```
env = gym.make('stocks-v0', frame_bound=(90,110), window_size=5)
obs = env.reset()
while True:
    obs = obs[np.newaxis, ...]
    action, _states = model.predict(obs)
    obs, rewards, done, info = env.step(action)
    if done:
        print("info", info)
        break
```

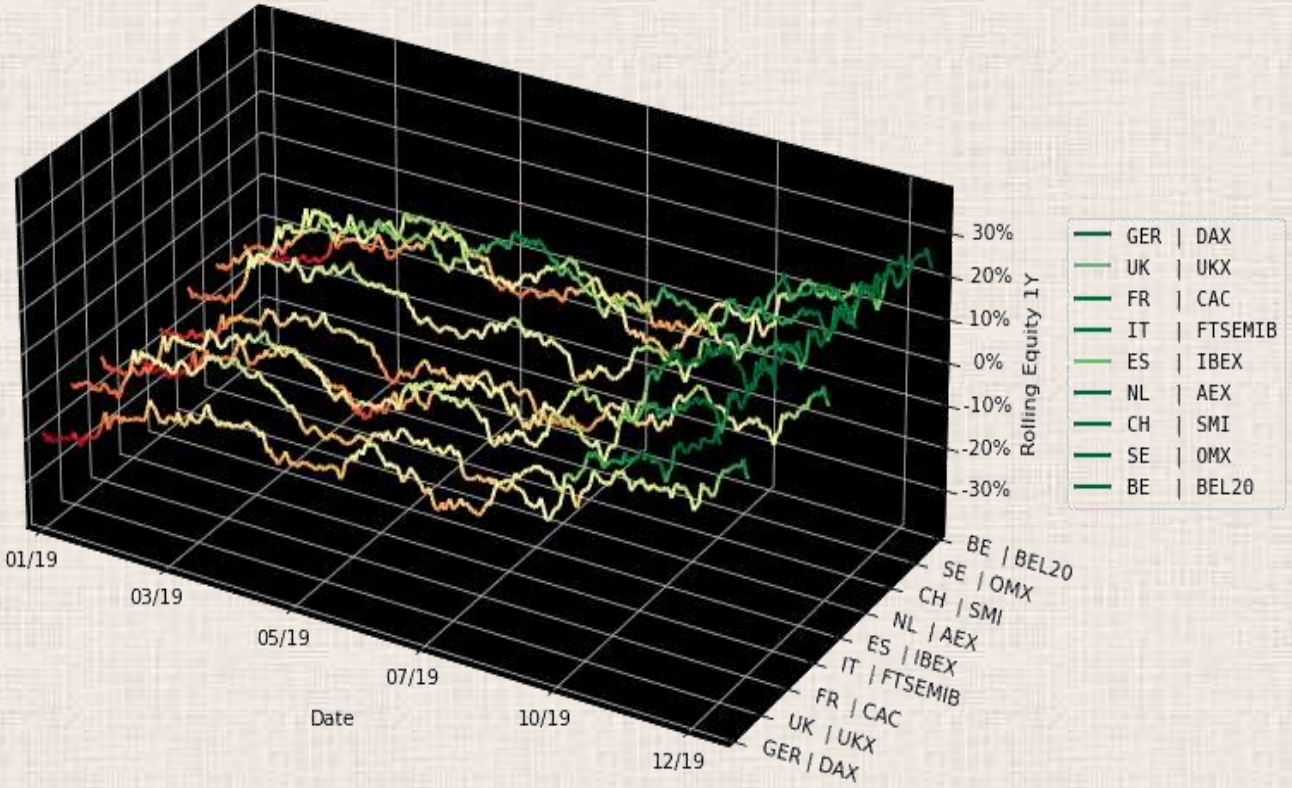
```
info {'total_reward': 4.104111000000046, 'total_profit': 0.9723945651271646, 'position': 1}
```

As we can see, our agent RL bought and sold stocks at random. Our profit margin appears to be greater than 1, so we can determine that our bot has made us profit from the trades it has made. But these were random steps, now let's properly train our model to get better trades.

The model isn't perfect. It has made some long and short trades. Some good and bad trades .With a few tweaks, this model can be trained to trade with stocks, forex, equities, and securities.

```
plt.figure(figsize=(15,6))
plt.cla()
env.render_all()
plt.show()
```

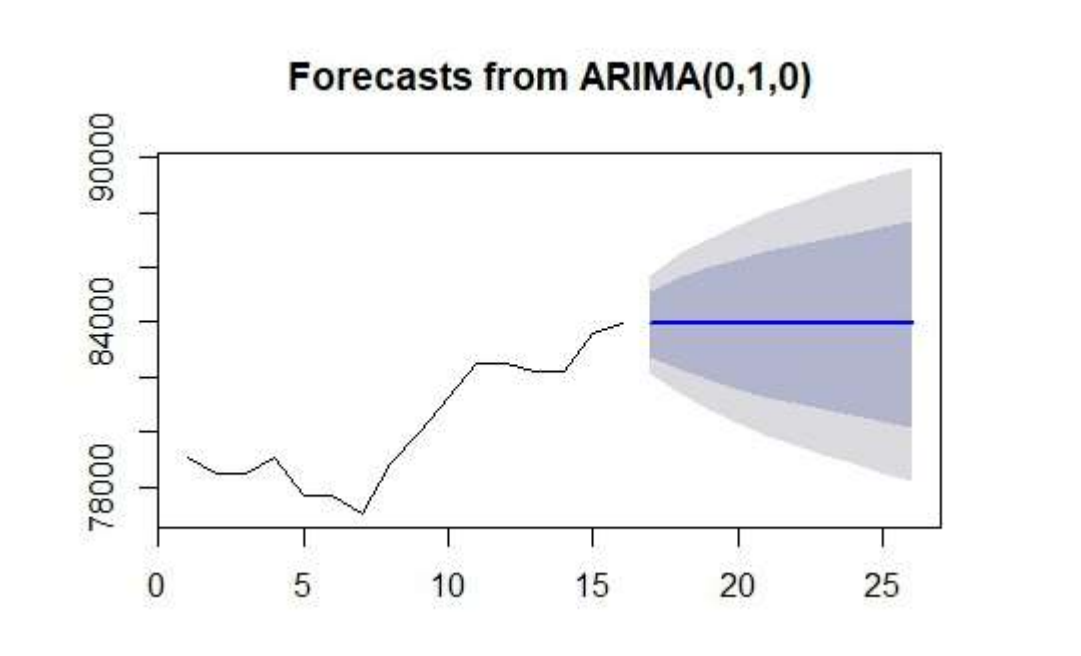
Time Series



- A time series is a data set that tracks a sample over time.
- In particular, a time series allows one to see what factors influence certain variables from period to period.
- Time series analysis can be useful to see how a given asset, security, or economic variable changes over time.
- Forecasting methods using time series are used in both fundamental and technical analysis.
- Although cross-sectional data is seen as the opposite of time series, the two are often used together in practice.

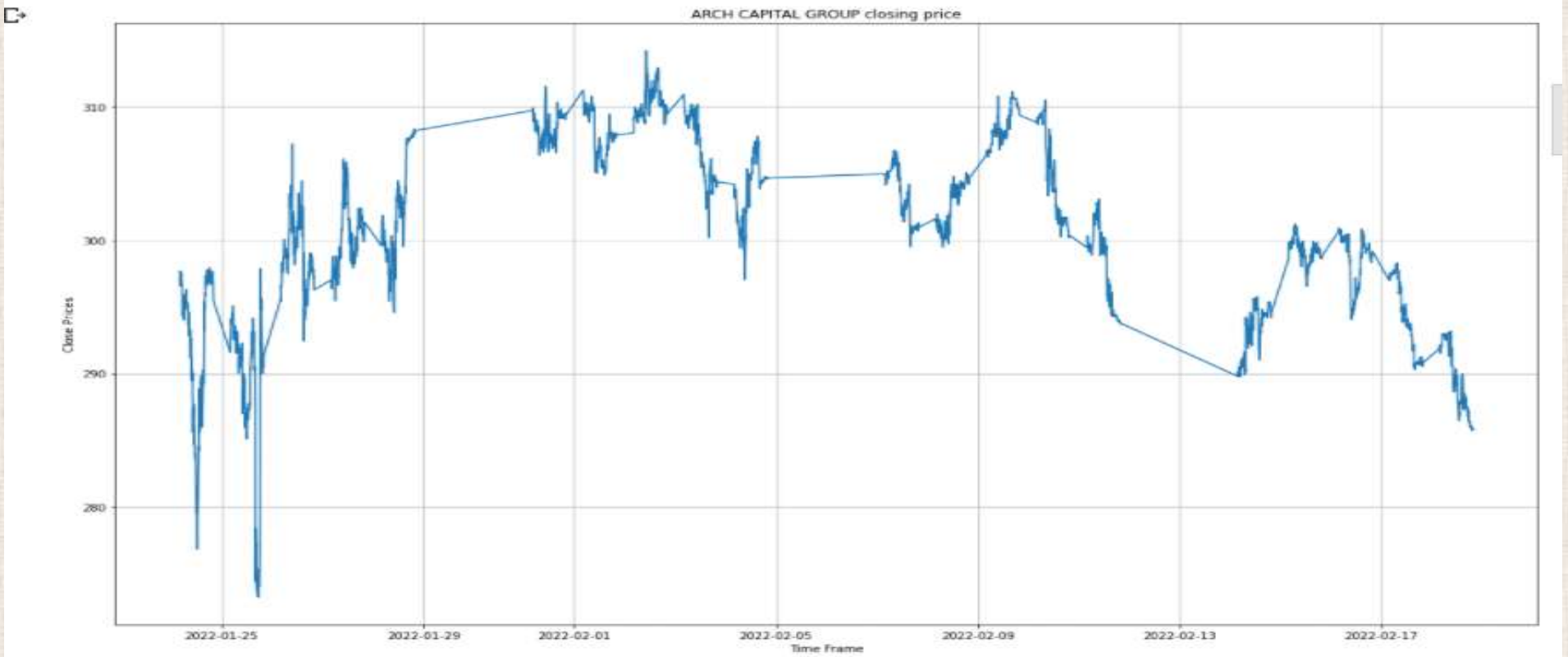
What is ARIMA?

- Before working with non-stationary data, the Autoregressive Integrated Moving Average (ARIMA) Model converts it to stationary data. One of the most widely used models for predicting linear time series data is this one.
- The ARIMA model has been widely utilized in banking and economics since it is recognized to be reliable, efficient, and capable of predicting short-term share market movements.



Plot close price

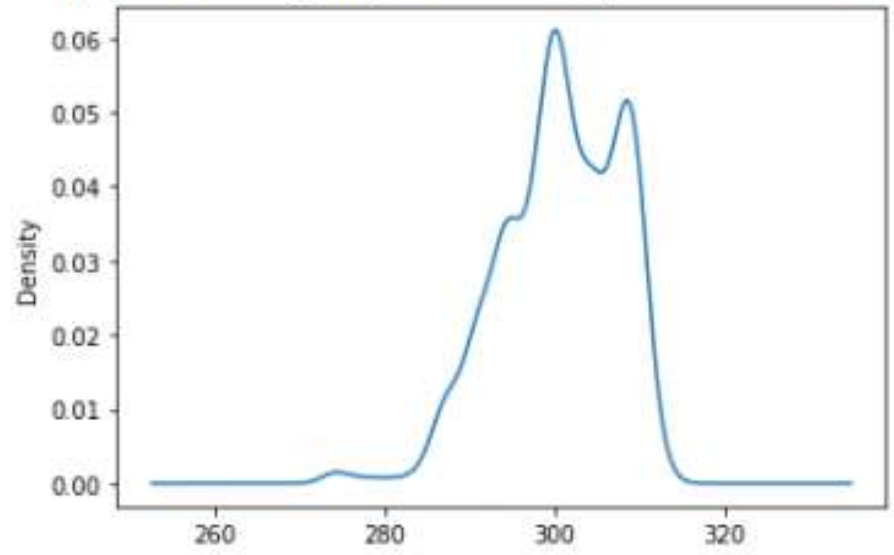
```
#plot close price
plt.figure(figsize=(20,12))
plt.grid(True)
plt.xlabel('Time Frame')
plt.ylabel('Close Prices')
plt.plot(stock_data['4. close'])
plt.title('ARCH CAPITAL GROUP closing price')
plt.show()
```

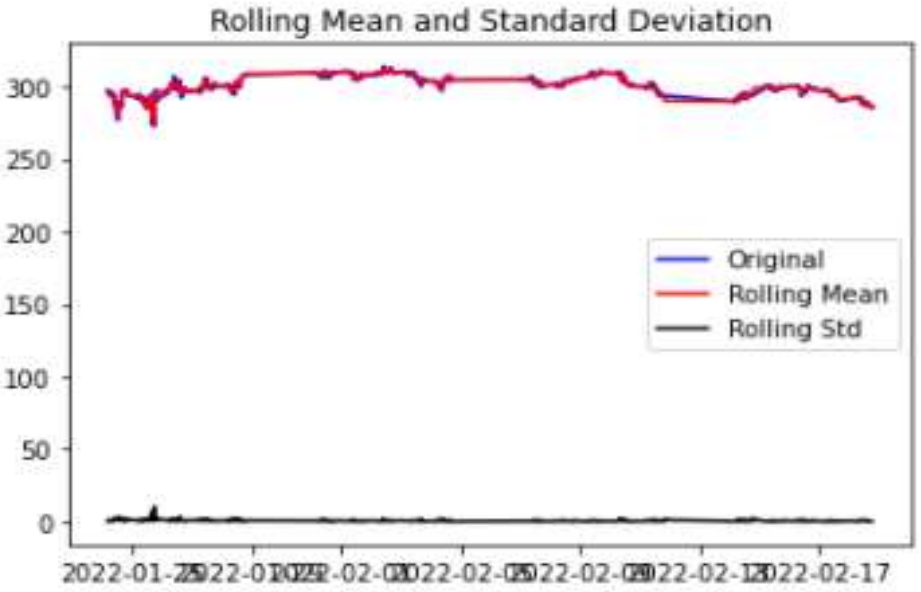


Test for stationarity

```
#Distribution of the dataset
df_close = stock_data['4. close']
df_close.plot(kind='kde')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f80ca5ced50>
```



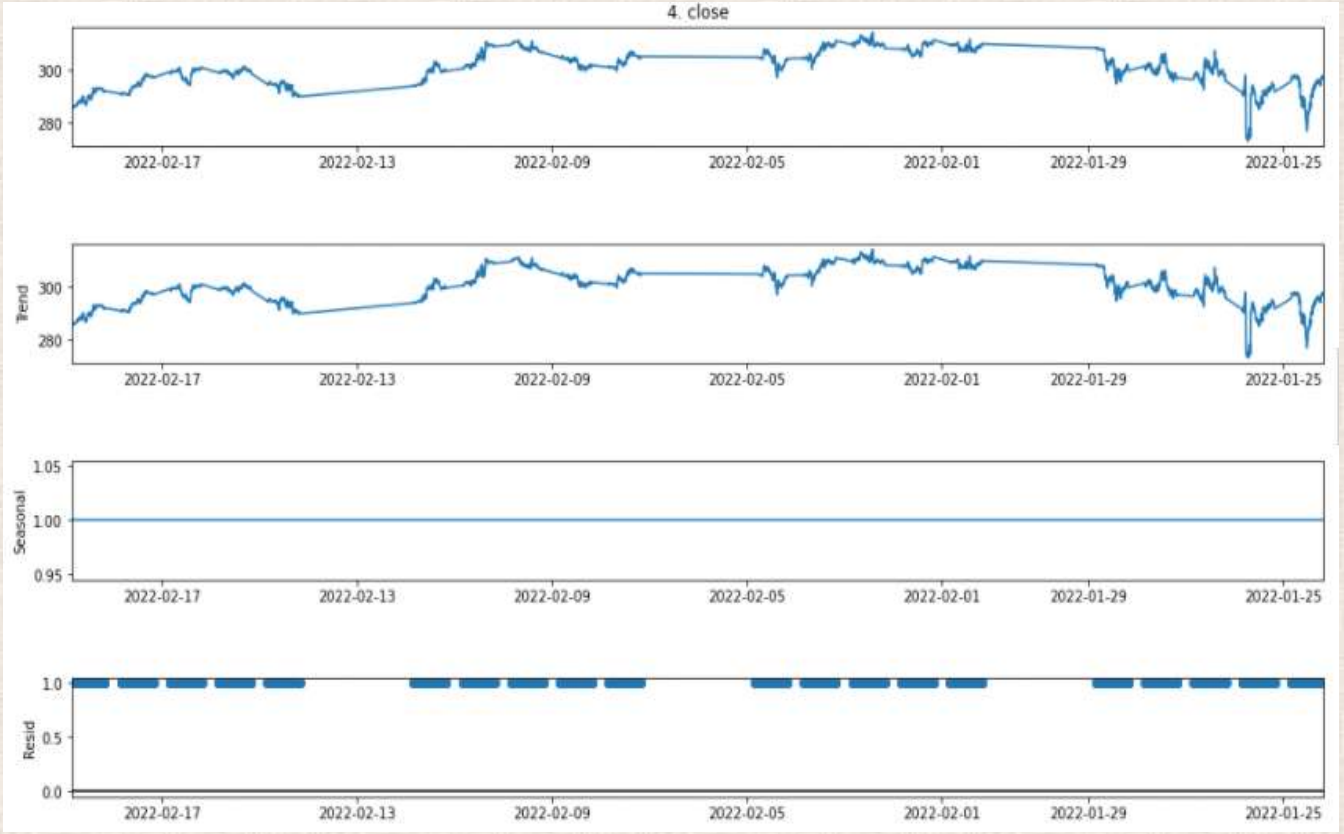



```
Results of dickey fuller test
Test Statistics                  -2.874560
p-value                           0.048387
No. of lags used                 27.000000
Number of observations used    3556.000000
critical value (1%)              -3.432190
critical value (5%)              -2.862353
critical value (10%)             -2.567203
dtype: float64
```

Separating the trend and the seasonality from a time series

we can decompose the series using the following code.

```
from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(df_close, model='multiplicative', period=1)
fig = plt.figure()
fig = result.plot()
fig.set_size_inches(16, 9)
```



Split data into train and training set

ARIMA Model

```
model_autoARIMA = auto_arima(train_data, start_p=0, start_q=0,
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=3, max_q=3, # maximum p and q
                      m=1,              # frequency of series
                      d=None,           # let model determine 'd'
                      seasonal=False,   # No Seasonality
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)
print(model_autoARIMA.summary())
model_autoARIMA.plot_diagnostics(figsize=(15,8))
plt.show()
```

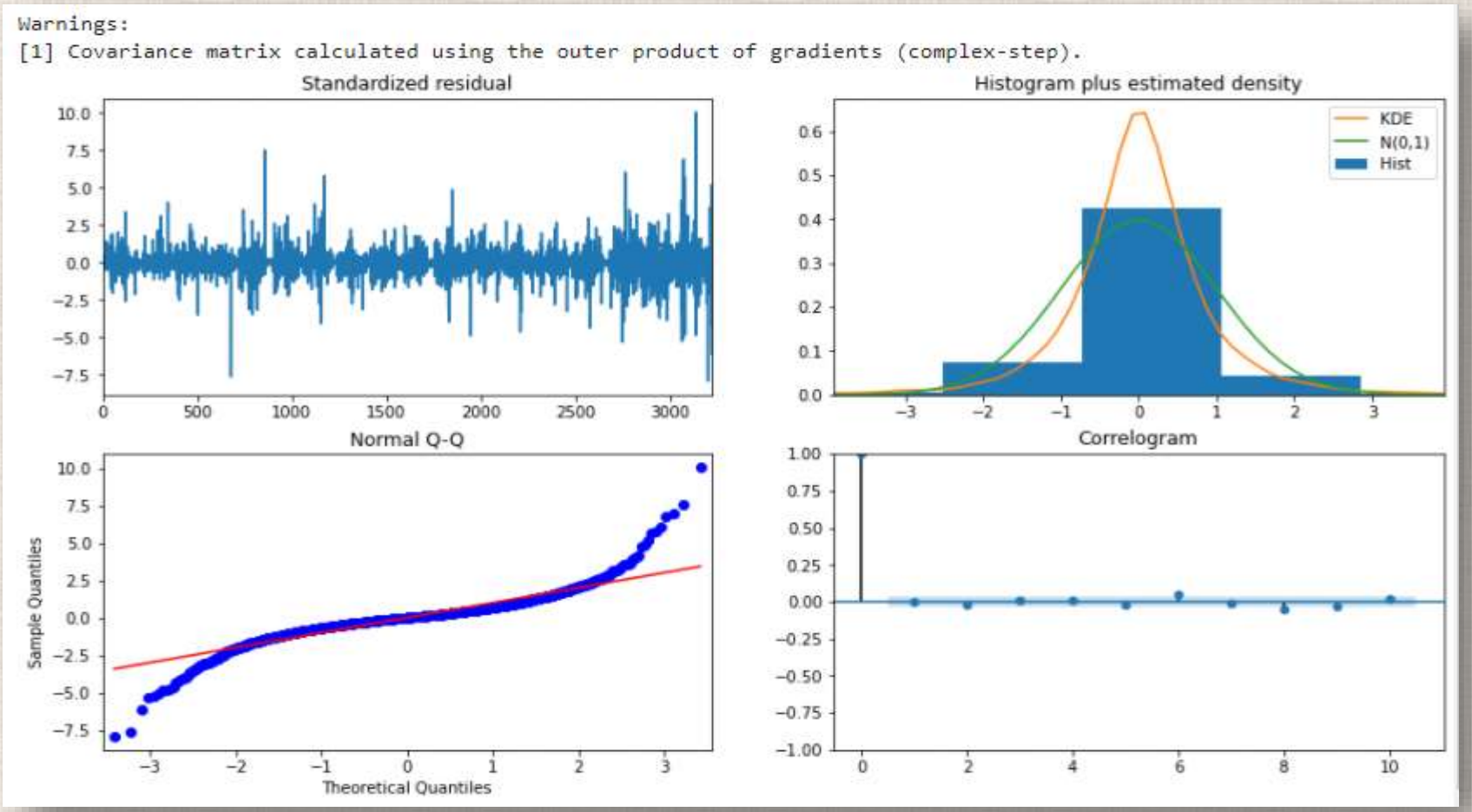
SARIMAX Results

Dep. Variable:	y	No. Observations:	3222
Model:	SARIMAX(0, 1, 1)	Log Likelihood	15768.718
Date:	Mon, 21 Feb 2022	AIC	-31533.436
Time:	14:21:31	BIC	-31521.282
Sample:	0	HQIC	-31529.080
	- 3222		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.0884	0.010	-9.065	0.000	-0.108	-0.069
sigma2	3.275e-06	3.09e-08	105.815	0.000	3.21e-06	3.34e-06

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	19111.30
Prob(Q):	0.99	Prob(JB):	0.00
Heteroskedasticity (H):	2.34	Skew:	0.24
Prob(H) (two-sided):	0.00	Kurtosis:	14.92

Visual Summary Of ARIMA Model



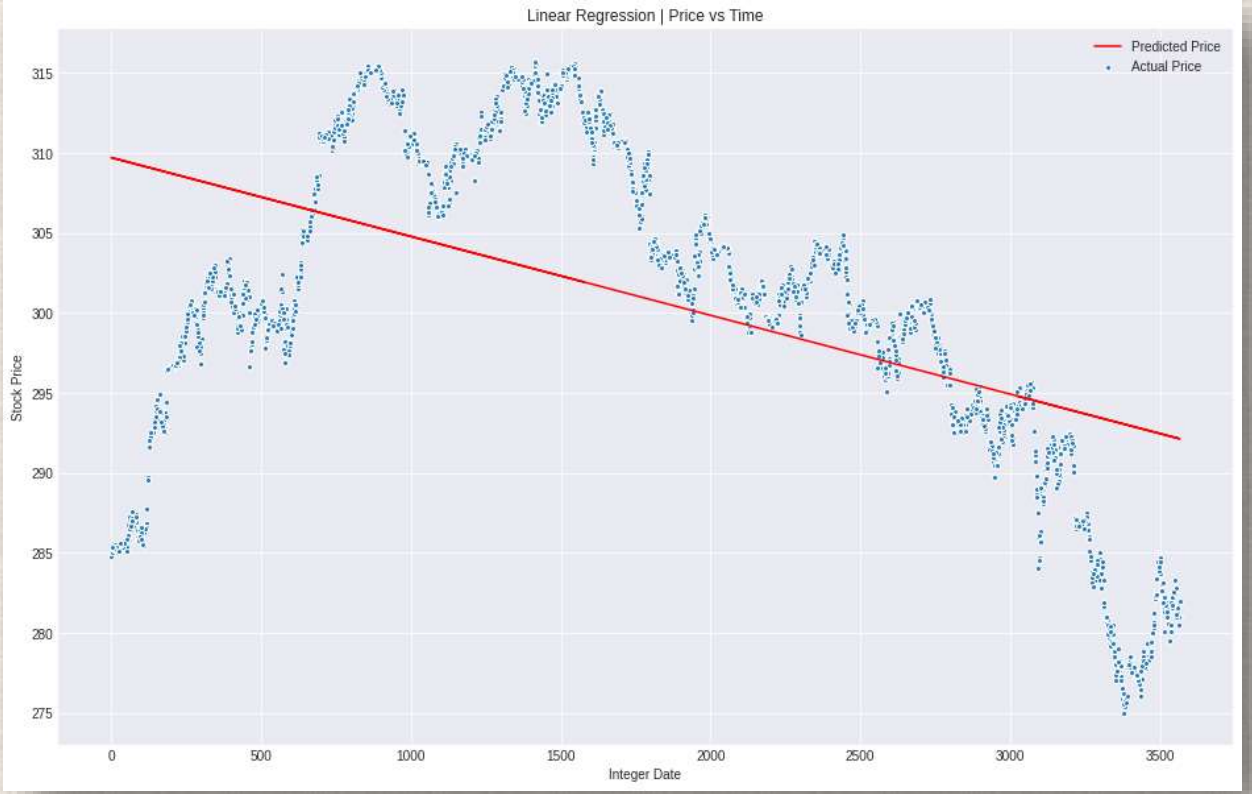
Trained Arima Model

```
import statsmodels.api as smapi

model = smapi.tsa.arima.ARIMA(train_data, order=(0,1,1))

result = model.fit()
print(result.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                 4. close   No. Observations:                 3222
Model:                 ARIMA(0, 1, 1)   Log Likelihood               15768.718
Date:                Mon, 21 Feb 2022   AIC                         -31533.436
Time:                        15:38:55   BIC                         -31521.282
Sample:                             0   HQIC                        -31529.080
                                - 3222
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.0884      0.010     -9.065      0.000      -0.108      -0.069
sigma2      3.275e-06   3.09e-08    105.815      0.000    3.21e-06    3.34e-06
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):             19111.30
Prob(Q):                              0.99   Prob(JB):                         0.00
Heteroskedasticity (H):               2.34   Skew:                             0.24
Prob(H) (two-sided):                  0.00   Kurtosis:                        14.92
===================================================================================
```

Linear Regression

- Linear regression is used for predictions with data that has numeric target variable.
- During prediction we use some variables as dependent variables and few considered as independent variables.
- In situation when there is one dependent and one independent variable, we prefer to use linear regression methodologies.
- Regression can be single variable or multi variable, it depends upon situation named as single variable or multi variable regression.



Select Subset with relevant features

We use the 5 min Timeframe closing price **Close** as the value to predict, so we can discard the other features.

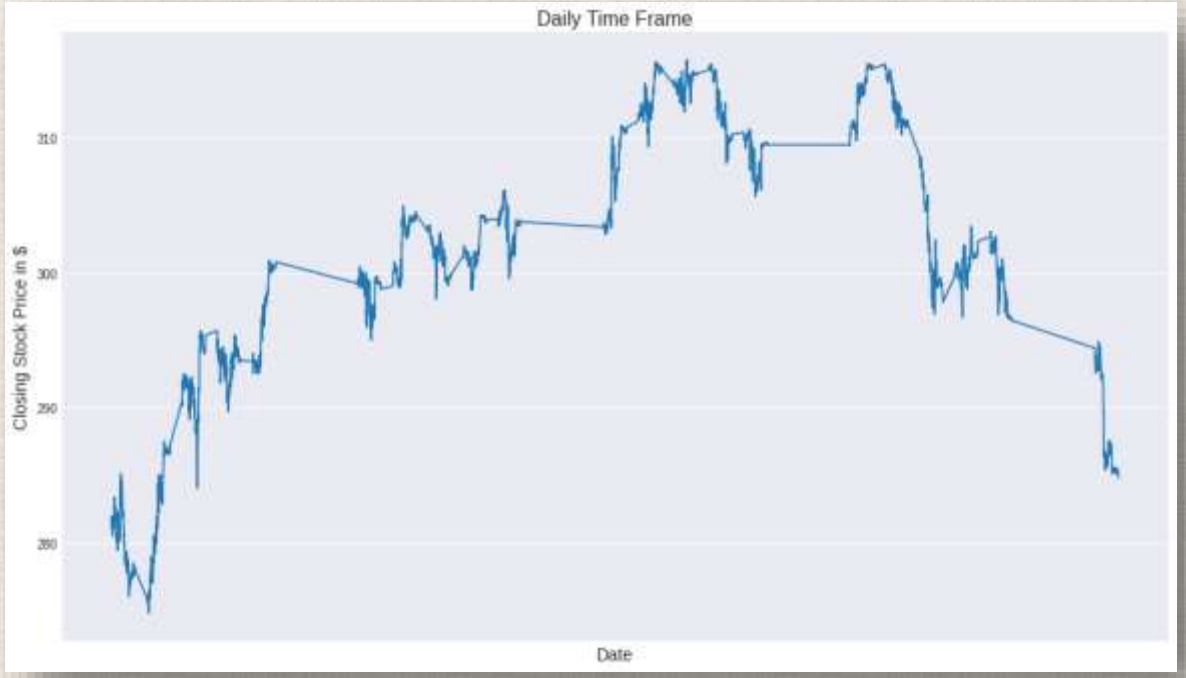
- ‘Close' column has numerical data type
- The 'Date' is the index column and contains datetime values

```
df = pd.DataFrame(data, columns=['date','4. close'])
df
```

	date	4. close
0	2022-04-11 20:00:00	284.80
1	2022-04-11 19:55:00	285.01
2	2022-04-11 19:50:00	285.05
3	2022-04-11 19:45:00	285.24
4	2022-04-11 19:40:00	285.15
...	...	...

Explore the Data

When we take a look at the price movement over time by simply plotting the *Closing price* vs *Time*, we can already see, that the price continously increases over time and we can also estimate that trend could be linear.




```
import matplotlib.dates as mdates

years = mdates.YearLocator() # Get every year
yearsFmt = mdates.DateFormatter('%Y') # Set year format

# Create subplots to plot graph and control axes
fig, ax = plt.subplots()
ax.plot(df['date'], df['4. close'])

# Format the ticks
ax.xaxis.set_major_locator(years)
ax.xaxis.set_major_formatter(yearsFmt)

# Set figure title
plt.title('Daily Time Frame', fontsize=16)
# Set x label
plt.xlabel('Date', fontsize=14)
# Set y label
plt.ylabel('Closing Stock Price in $', fontsize=14)

# Rotate and align the x labels
fig.autofmt_xdate()

# Show plot
plt.show()
```

Linear Regression Formulation

Our data contains only one **independent variable (X)** which represents the *date* and the **dependent variable (Y)** we are trying to predict is the *Stock Price*. To fit a line to the data points, which then represents an estimated relationship between X and Y, we can use a **Simple Linear Regression**.

The best fit line can be described with $Y = \beta_0 + \beta_1 X$ where,

- Y is the predicted value of the dependent variable
- β_0 is the y-intercept
- β_1 is the slope
- X is the value of the independent variable

The goal is to find such coefficients β0 and β1 that the **Sum of Squared Errors**, which represents the difference between each point in the dataset with it's corresponding predicted value outputted by the model, is minimal.

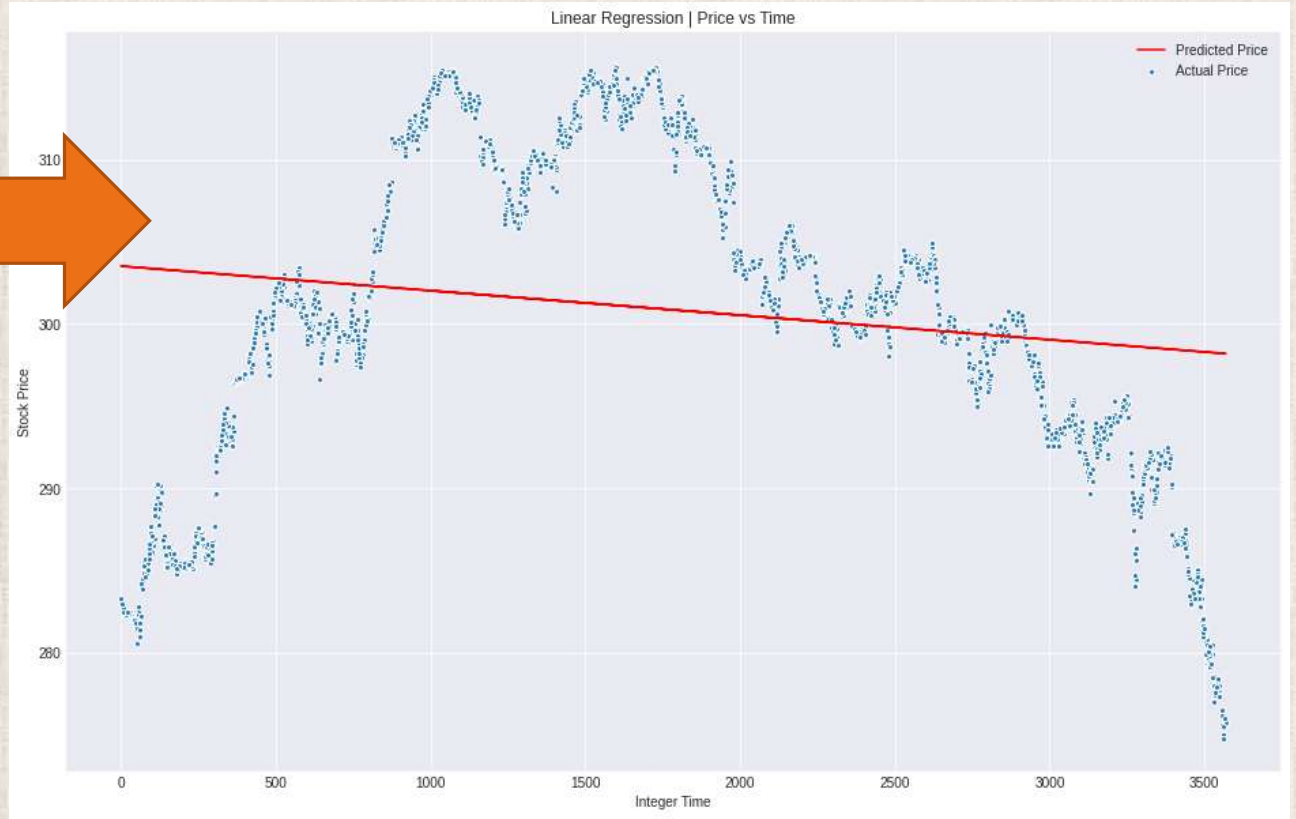
Training Linear Regression Model

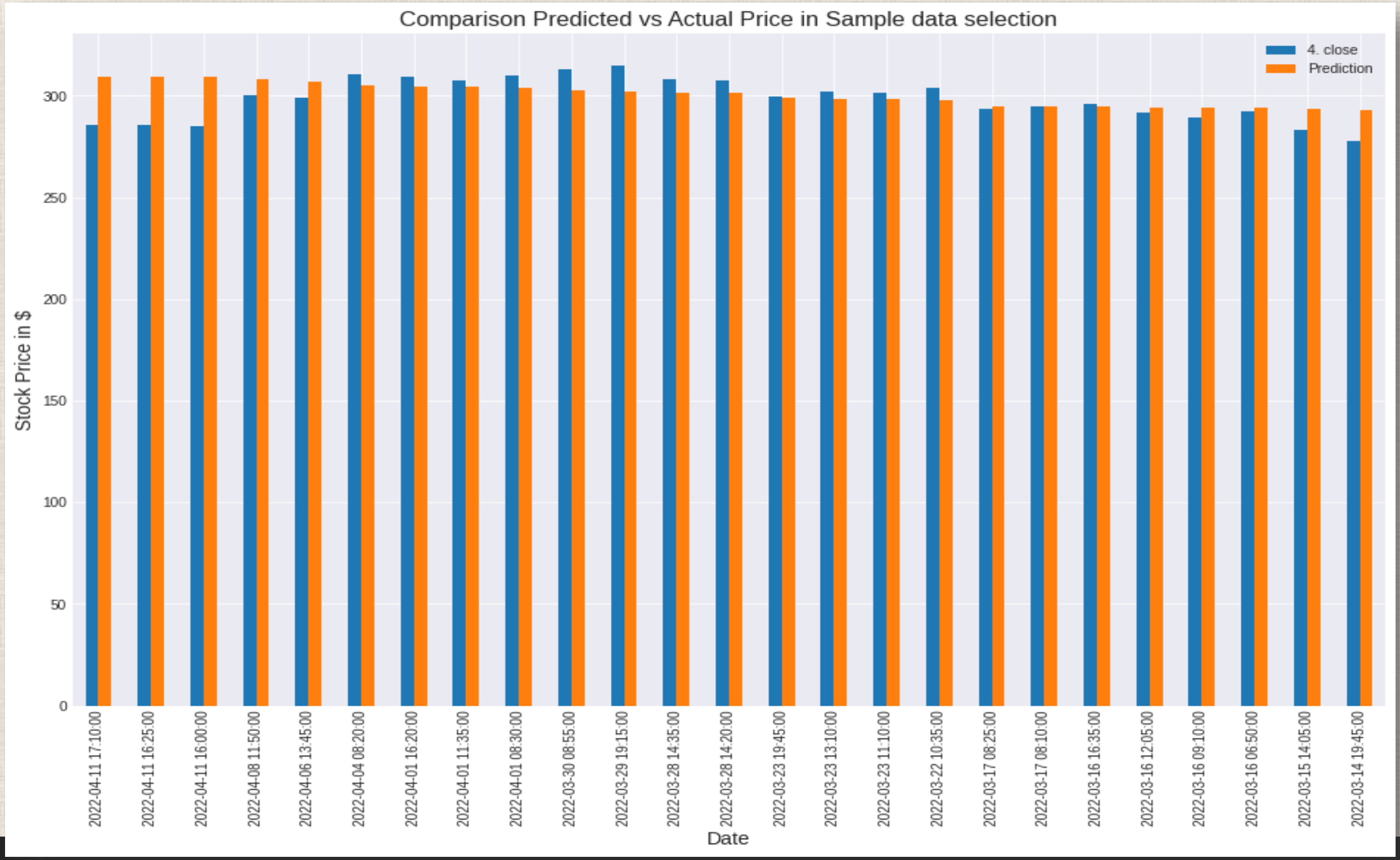
```
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size=0.20)
from sklearn.linear_model import LinearRegression
X_train = np.array(train.index).reshape(-1, 1)
y_train = train['4. close']
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
print('Slope: ', np.asscalar(np.squeeze(model.coef_)))
print('Intercept: ', model.intercept_)
```

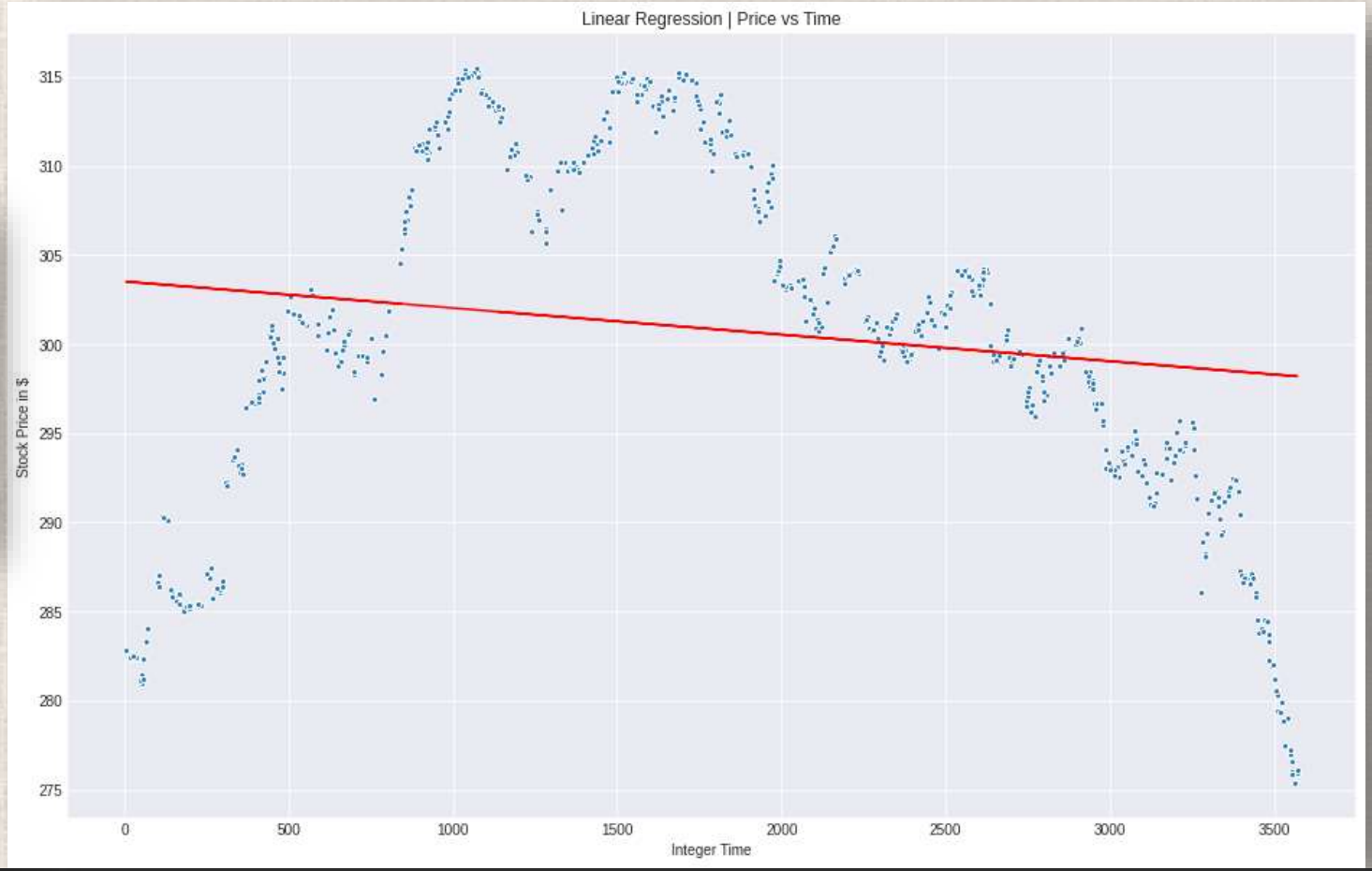
```
plt.figure(1, figsize=(16,10))
plt.title('Linear Regression | Price vs Time')
plt.scatter(X_train, y_train, edgecolor='w', label='Actual Price')
plt.plot(X_train, model.predict(X_train), color='r', label='Predicted Price')
plt.xlabel('Integer Date')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```




Comparison Predicted vs Actual Price in Sample data selection
4. close
Prediction
Stock Price in $
0
50
100
150
200
250
300
2022-04-11 17:10:00
2022-04-11 16:25:00
2022-04-11 16:00:00
2022-04-08 11:50:00
2022-04-06 13:45:00
2022-04-04 08:20:00
2022-04-01 16:20:00
2022-04-01 11:35:00
2022-04-01 08:30:00
2022-03-30 08:55:00
2022-03-29 19:15:00
2022-03-28 14:35:00
2022-03-28 14:20:00
2022-03-23 19:45:00
2022-03-23 13:10:00
2022-03-23 11:10:00
2022-03-22 10:35:00
2022-03-17 08:25:00
2022-03-17 08:10:00
2022-03-16 16:35:00
2022-03-16 12:05:00
2022-03-16 09:10:00
2022-03-16 06:50:00
2022-03-15 14:05:00
2022-03-14 19:45:00
Date

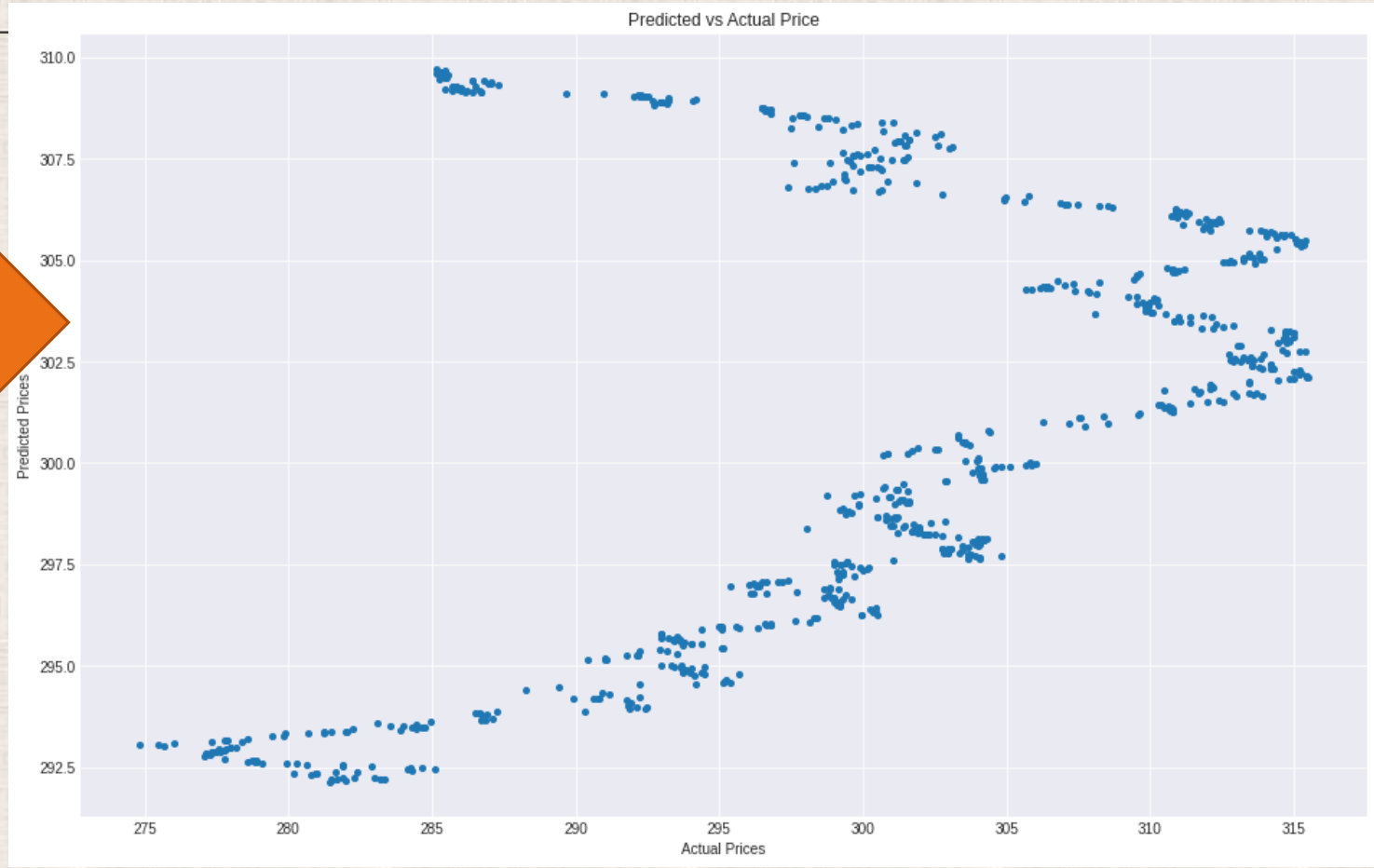
Linear Regression | Price vs Time

```
plt.figure(1, figsize=(16,10))
plt.title('Linear Regression | Price vs Time')
plt.plot(X_test, model.predict(X_test), color='r', label='Predicted Price')
plt.scatter(X_test, y_test, edgecolor='w', label='Actual Price')

plt.xlabel('Integer Time')
plt.ylabel('Stock Price in $')

plt.show()
```



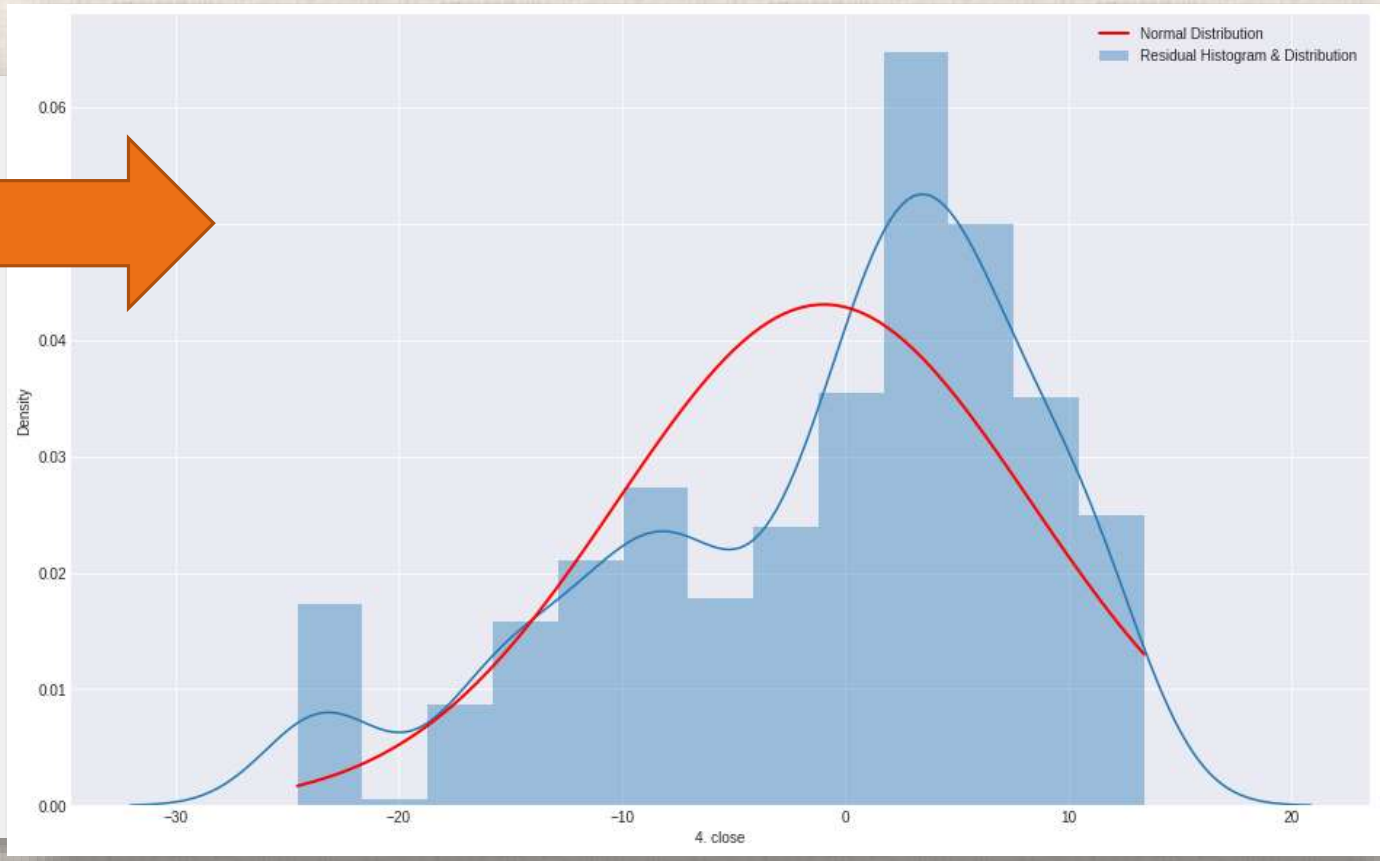
Predicted Vs Actual Price

```
plt.scatter(y_test, y_pred)

plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')

plt.title('Predicted vs Actual Price')

plt.show()
```



Normal v/s Residual Histogram Distribution

```
from scipy.stats import norm

# Fit a normal distribution to the data:
mu, std = norm.fit(y_test - y_pred)

ax = sns.distplot((y_test - y_pred), label='Residual Histogram & Distribution')

# Calculate the pdf over a range of values
x = np.linspace(min(y_test - y_pred), max(y_test - y_pred), 100)
p = norm.pdf(x, mu, std)

# And plot on the same axes that seaborn put the histogram
ax.plot(x, p, 'r', lw=2, label='Normal Distribution')

plt.legend()
plt.show()
```



Evaluation

```
df.head()
```

	date	4. close	Prediction
0	2022-04-11 20:00:00	284.80	309.717133
1	2022-04-11 19:55:00	285.01	309.712202
2	2022-04-11 19:50:00	285.05	309.707271
3	2022-04-11 19:45:00	285.24	309.702341
4	2022-04-11 19:40:00	285.15	309.697410

```
from sklearn import metrics
df['4. close'].describe()
```

```
count    3567.000000
mean      300.734479
std         9.930445
min       274.800000
25%       294.345000
50%       301.065000
75%       309.750000
max       315.745100
Name: 4. close, dtype: float64
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 7.4177592938312715
Mean Squared Error: 86.76950403412903
Root Mean Squared Error: 9.315014977665308
```

```
print('R2: ', metrics.r2_score(y_test, y_pred))
```

```
R2:  0.19000735920220158
```

```
from sklearn.metrics import explained_variance_score
explained_variance_score(y_test, y_pred)
```

```
0.19854457163726957
```

References

- Chavan, P. S., & Patil, S. T. (2013). Parameters for stock market prediction. International Journal of Computer Technology and Applications, 4(2), 337.
- Machine Learning in Intraday Stock Trading ,Name : Art Paspanthong , Nick Tantivasadakarn , Will Vithayapalert Institute : Stanford University Date of Publication: Spring 2019
- School of Computing, Queen's University, Kingston, ON K7L 2N8, Canada; dshah@cs.queensu.ca (D.S.); farhana@cs.queensu.ca (F.Z.) Correspondence: isah@cs.queensu.ca Received: 4 March 2019; Accepted: 15 May 2019; Published: 27 May 2019
- Stock Market Prediction using Machine Learning Name : Kranthi Sai Reddy Vanukuru Institute : Srineedhi Institue Of Science & Technology Date of Publication: November 2018
- Stock Market Prediction Using Machine Learning(ML)Algorithms M Umer Ghani, M Awais and Muhammad Muzammulla Department of Software Engineering, Government College University Faisalabad 2019

Timeline

Work Done Till 1st Presentation

1. Idea Discussion
2. Literature Survey
3. Manual Dataset Creation
4. Finding Possible Algorithms

Work Done Till 2nd Presentation

1. Finding API(Alpha-Vantage) required for gathering real-time stocks Data
2. Studied about Library(**PLOTLY**) that help us to plot Candlestick Chart

Work Done Till 3rd Presentation

1. Dataset Overview
2. Classification Using CNN
3. Model Training
4. Model Testing
5. Final Obtained Results

Timeline

Work done till 4th Presentation	Work done till Now	Work to be done till next Presentation
1. Worked On Reinforcement Learning and Obtained The Output 2. Worked On Time Series Trained the Model	1. Worked On Linear Regression Model,	1. Comparing All the output results from all four Machine learning Algorithms. 2. Creating A final report.