TECHNOCOLABS DATA ANALYSIS INTERNSHIP

PROJECT REPORT

TITLE: Machine-Learning-in-Intraday-Stock-Trading

AIM:

In this project, we aim aims to explore how the predictive power of machine learning models can reap financial benefits for investors who trade based on future price prediction.

ABSTRACT:

Stock market prediction is an attempt of determining the future value of a stock traded on a stock exchange. This project focuses on classification problem, predicting the next-second price movement and acting upon the insights generated from our models. We implemented multiple machine learning algorithms including: logistic regression, support vector machines (SVM), Long-Short Term memory (LSTM), and Convolutional Neural Networks (CNN) to determine the trading action in the next minute. Using the predicted results from our models to generate the portfolio value over time, support vector machine with polynomial kernel performs the best among all of our models.

INTRODUCTION:

The ability to precisely predict the price movement of stocks is the key to profitability in trading. Many investors spend time actively trading stocks in hope of outperforming the market, colloquially referred to as a passive investment. In light of the increasing availability of financial data, prediction of price movement in the financial market with machine learning has become a

topic of interests for both investors and researchers alike. Insights about price movements from the models could help investors make more educated decisions. In this project, we aim to focus on making short term price movements prediction using the timeseries data of stock price, commonly used technical-analysis indicators, and trading volume. Such predictions will then be used to generate short-term trading strategies to capitalize on small price movements in highly liquid stocks.

Exploratory Data Analysis Report

Historical Dataset with 1 day time interval

The Eda team downloaded the dataset of Microsoft with 1 day interval by using the yfinance api.

Samples: 8861

Start Date: 1986-03-13

End Date: 2021-04-29

Date Open High Low Close Adj Close Volume

1986-03-13, 0.088542, 0.101563, 0.088542, 0.097222, 0.061751, 1031788800

1986-03-14, 0.097222, 0.102431, 0.097222, 0.100694, 0.063956, 308160000

1986-03-17, 0.100694, 0.103299, 0.100694, 0.102431, 0.065059, 133171200

1986-03-18, 0.102431, 0.103299, 0.098958, 0.099826, 0.063405, 67766400

1986-03-19, 0.099826, 0.100694, 0.097222, 0.098090, 0.062302, 47894400

Code:

```
data = yfinance.download("MSFT", interval='1d')
data.head()
data.tail()
print(len(data))
```

Historical Dataset with 1 min time interval

The Eda team downloaded the dataset of Microsoft and NFTY with 1 min interval by using the yfinance api and online dataset platforms.

NFTY DATASET

NFTY Samples: 22806

Start Date: 2021-01-01

Start Time: 09:16

End Date: 2021-03-31

End Time: 15:31

This Dataset was used for classification Change amount was calculated using python method pct_change

```
Code: dataframe["change"] = dataframe[["open",
"close"]].pct_change(axis=1)["close"]
```

After calculating change. We used the newly added feature(change) to classify the data into profit or loss.

Code:

```
clas = []


for col in dataframe["change"]:
    if col <= 0:
        clas.append("-1")
    elif col > 0:
        clas.append("1")
    else:
        clas.append("NA")

dataframe["classification"] = clas
```

After adding classification column the change column was removed.

We used MinMaxscaler to scale the data.

Microsoft Dataset 1 min interval

The Eda team downloaded the dataset of Microsoft with 1 minute interval by using the yfinance api.

```
data = yfinance.download("MSFT", interval='1m',
period='7d')
data.head()
data.tail()
print(len(data))
```

samples: 2726

Start Date: 2021-04-23

Start Time: 09:30:00-4:00

End Date: 2021-04-30

End Time: 15:55:00-04:00

This Dataset was used to predict the stock price of Microsoft for the next 1 day

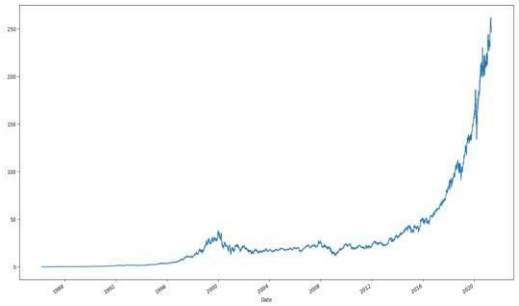
Code:

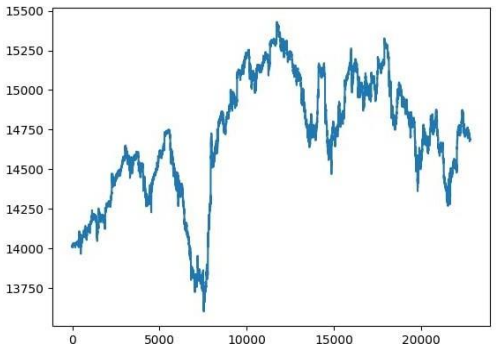
```
train = pd.read_csv("1mindata/MSFT1Min.csv")
test = pd.read_csv("1mindata/1MSFT-Test.csv")
len(train)
len(test)

train = train.sort_values('Datetime')
test = test.sort_values("Datetime")

train.reset_index(inplace=True)
train.set_index("Datetime", inplace=True)

test.reset_index(inplace=True)
test.set_index("Datetime", inplace=True)
"""
```

```
plt.figure(figsize=(12, 6))
plt.plot(train["Adj Close"])
plt.xlabel('Date', fontsize=15)
plt.ylabel('Adjusted Close Price', fontsize=15)
plt.show()


# Rolling mean
close_px = train['Adj Close']
mavg = close_px.rolling(window=100).mean()

plt.figure(figsize=(12, 6))
close_px.plot(label='MSFT')
mavg.plot(label='mavg')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
"""

dates_df = train.copy()
dates_df = dates_df.reset_index()
org_dates = dates_df['Datetime']
```

```
# convert to ints
dates_df['Datetime'] =
dates_df['Datetime'].map(mdates.date2num)

dates_df.tail()

dates = dates_df['Datetime'].to_numpy()
prices = train['Adj Close'].to_numpy()

test_df = test.copy()
test_df = test_df.reset_index()
org_dates_test = test_df['Datetime']
test_df['Datetime'] =
test_df['Datetime'].map(mdates.date2num)
test_dates = test_df['Datetime'].to_numpy()
test_prices = test['Adj Close'].to_numpy()

test_dates = np.reshape(test_dates, (len(test_dates),
1))
test_prices = np.reshape(test_prices,
(len(test_prices), 1))
```

Output:





Model we are going to build for stock price prediction are as follows:

1. baseline(logistic)
2. SVM linear
3. SVM poly
4. SVM RBF
5. SVM Sigmoid
6. GRU
7. LSTM single layer
8. LSTM multi-layer
9. CNN

Logistic Regression:

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

SVM:

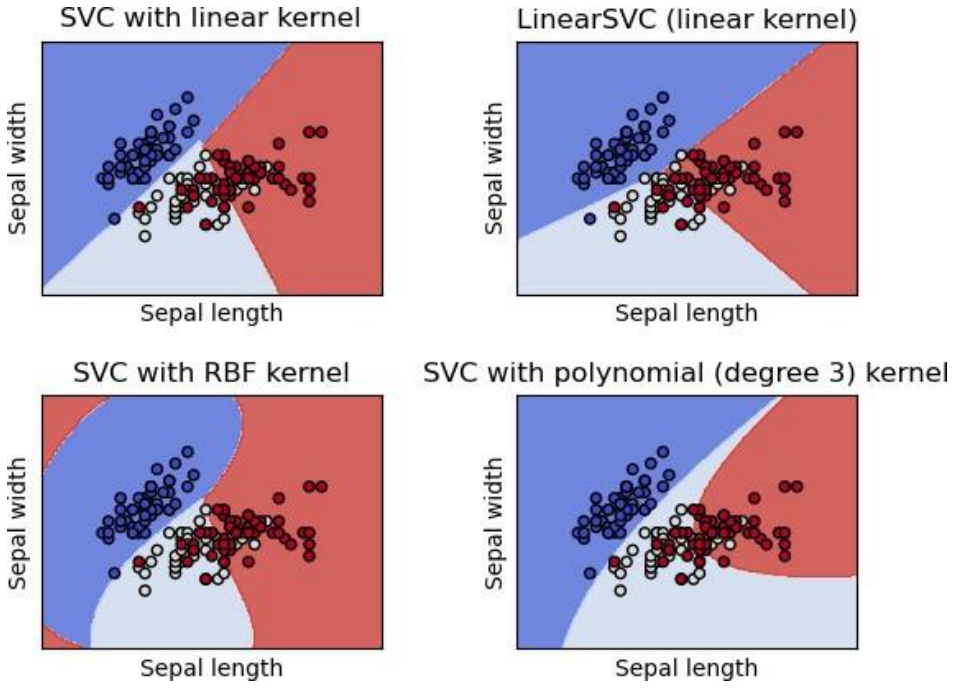
Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outlier's detection.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).



GRU:

Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks, introduced in 2014 by Kyunghyun Cho et al. The GRU is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate

- Gated RNNs can better capture dependencies for sequences with large time step distances.
- Reset gates help capture short-term dependencies in sequences.
- Update gates help capture long-term dependencies in sequences.
- GRUs contain basic RNNs as their extreme case whenever the reset gate is switched on. They can also skip subsequences by turning on the update gate.

LSTM:

- Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work.1 They work tremendously well on a large variety of problems, and are now widely used.
- LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behaviour, not something they struggle to learn!
- All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

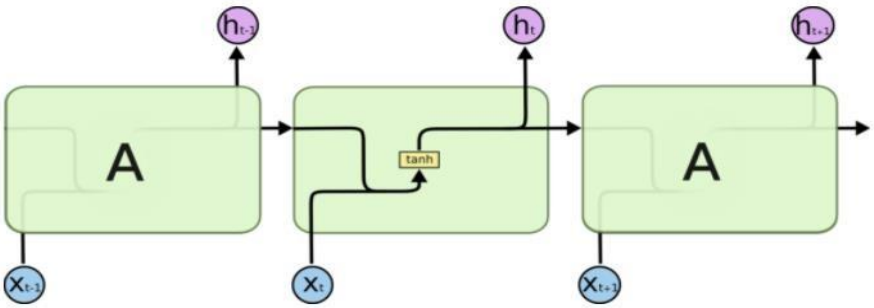


Fig: The repeating module in a standard RNN contains a single layer.

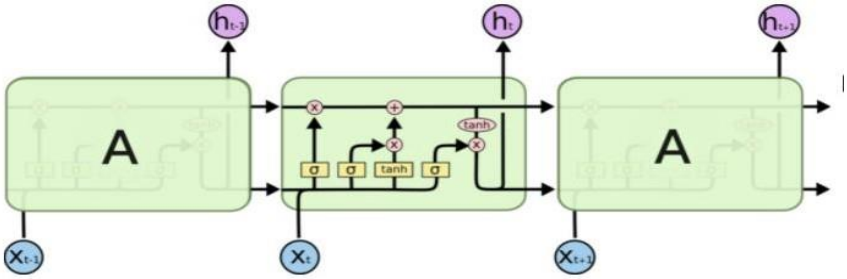
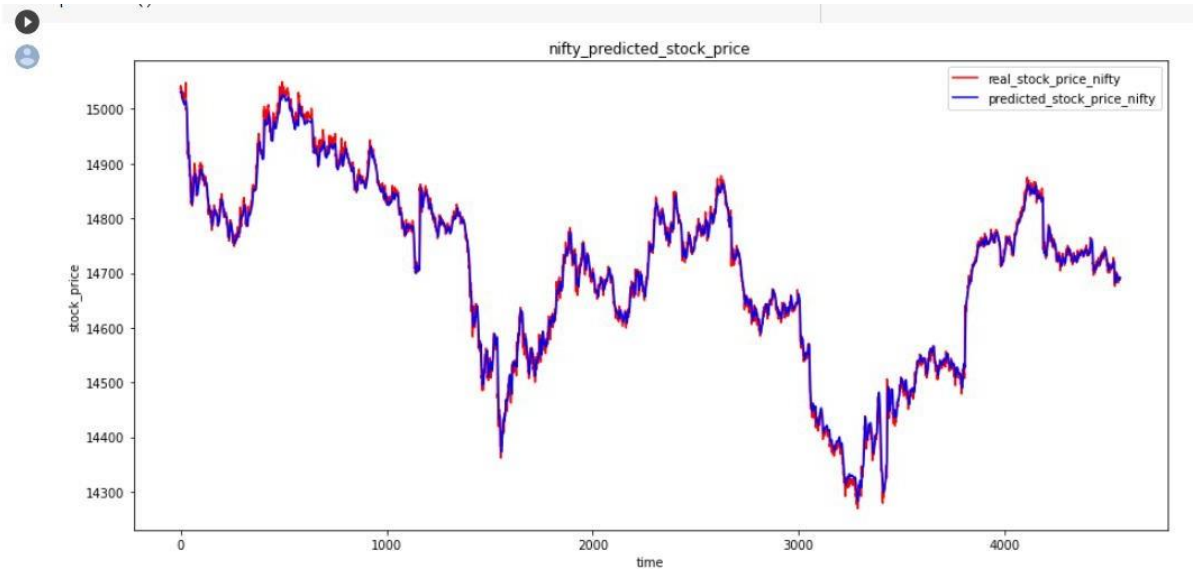
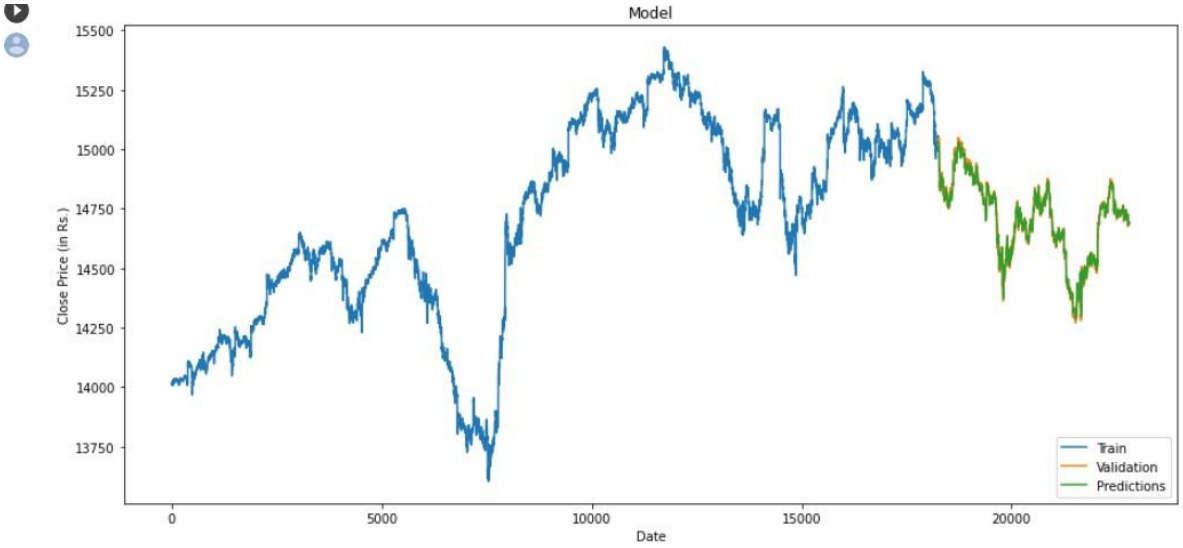


Fig: The repeating module in an LSTM contains four interacting layers.

Output:



Plot 1: For test dataset



Plot 2: For whole dataset

Accuracy level	Model name
Highest (55 above)	LSTM (Multi-layer) SVM (poly)
Midrange (50-55)	SVM (linear), SVM (RBF), SVM (Sigmoid), GRU, CNN, LSTM(Single-Layer)
Lowest (>50%)	Logistic

Model Deployment:

Deployed Link: https://stockprice2.herokuapp.com

Part A:

Importing the required libraries

```
app.py > stock_analysis
1   import numpy as np
2   from flask import Flask, request, jsonify, render_template
3   import pickle
4   import pandas as pd
5   from sklearn.preprocessing import StandardScaler
```

Importing the preprocessed data

```
6     df = pd.read_csv('Sanskar.csv')
7     X = df.iloc[:, :-1].values
8     y = df.iloc[:, -1].values
9     sc = StandardScaler()
10    X_train = sc.fit_transform(X)
```

Importing the trained model

```
11
12    app = Flask(__name__)
13    mp = pickle.load(open('model_pickle.pkl', 'rb'))
14
```

Routing and creating variable rules

```
16  @app.route('/')
17  def home():
18      return render_template("index.html")
19
20
21  @app.route('/predict', methods=['POST'])
22  def stock_analysis():
23      df = pd.read_csv('Sanskar.csv')
24      X = df.iloc[:, :-1].values
25      y = df.iloc[:, -1].values
26
27      sc = StandardScaler()
28      X_train = sc.fit_transform(X)
```

```python

    features = [x for x in request.form.values()]
    final_features = [np.array(features)]
    inp = sc.transform(final_features)

    output = mp.predict(inp)
    if output == -1:
        output = "LOSS"
    else:
        output = "PROFIT"

    return render_template('index.html', prediction_text='  {}'.format(output))
```

Part B:

Creating the HTML template

```html
<body>
    <div class="login">
        <h1>Machine Learning in Intraday Stock Trading</h1>
        <h2>Formats to Enter Data</h2>
        <h3>Date:  20210101 YearMonthDay</h3>
        <h4>Hour: (Number)1-24 int</h4>
        <h5>Min: (Number) 1-60 int</h5>
        <form action="{{ url_for('stock_analysis')}}" method="POST">
            <input type="number" name="Date" placeholder="Date" required='required'>
            <input type="number" placeholder="Open" step="0.01" name="open" required='required'>
            <input type="number" name="hour" placeholder="Hour" required='required'>
            <input type="number" name="min" placeholder="Minute" required='required'>

            <button type="submit" class="button">Predict</button>

        </form>

        <br>
        <br>
       <span class="sp"> {{ prediction_text }} </span>
        </div>
</body>
```

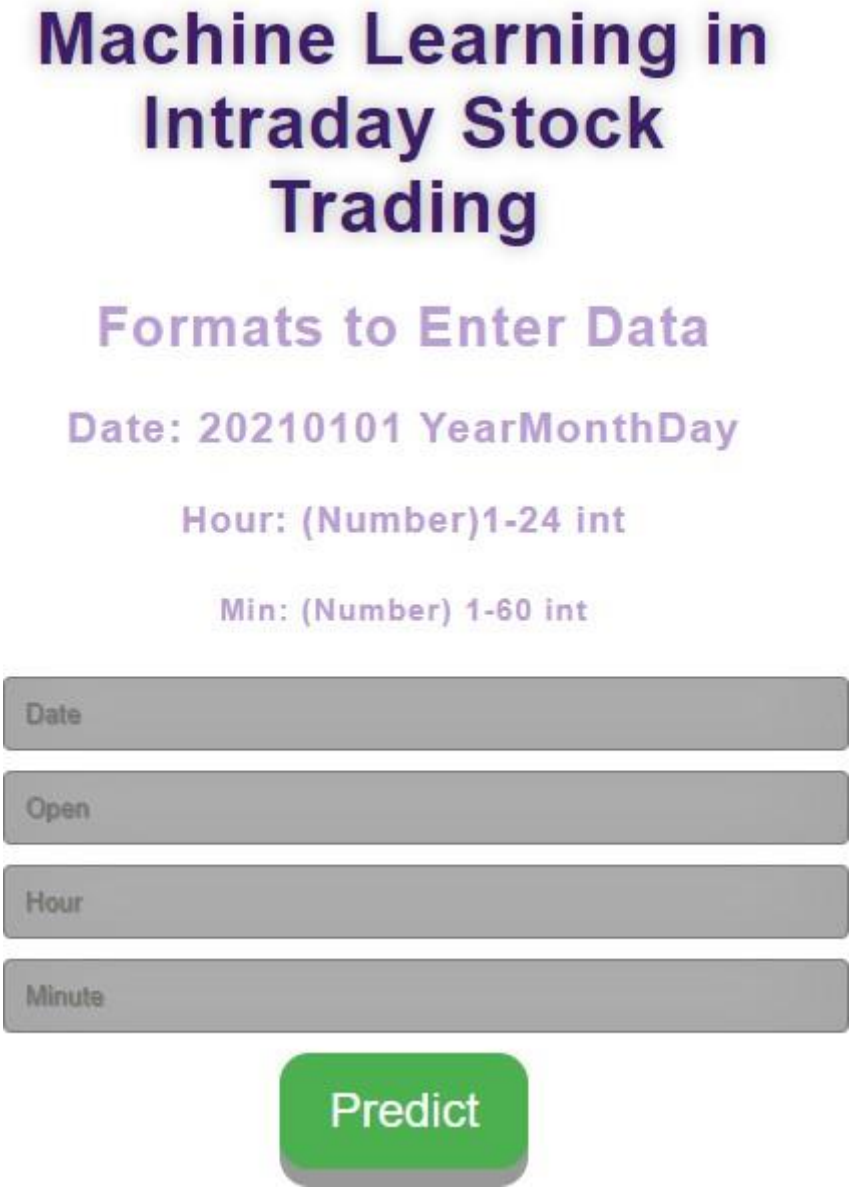
Requirements.txt

requirements.txt

```
click==7.1.2
Flask>=1.1.2
Cython>= 0.28.5
gunicorn==20.1.0
itsdangerous==1.1.0
Jinja2==2.11.2
joblib==1.0.1
MarkupSafe==1.1.1
numpy==1.19.3
pandas==1.2.0
python-dateutil==2.8.1
pytz==2021.1
scipy==1.6.3
six==1.15.0
sklearn==0.0
threadpoolctl==2.1.0
Werkzeug==1.0.1
pickles==0.1.1
pickleshare==0.7.5
```

Part C:

1. Deployed App



Link: https://stockprice2.herokuapp.com/

2. Deployed App (using SVM Poly)

Link: http://svm-stock-price.herokuapp.com/

TEAM MEMBERS:

Name	Task
Shrey Soni	Model Generation
Sanskar Sharma	Model Deployment
Manoranjan Kumar Thakur	Model Generation
Mehul Garg	Model Deployment (Team Leader)
Asmita Pritam	Exploratory Data Analysis (Team Leader)
Umesh Mohite	Model Generation (Team Leader)
C Murali Gopal	Model Generation
Sayali Ghadage	Exploratory Data Analysis
Varshika Malabanti	Model Generation
Nikitha peddi	Exploratory Data Analysis
Amaan Saxena	Model Generation
Joshua John	Model Deployment
Bavitha Reddy	Model Generation
Umar Hajam	Project Lead (Including All Task)
Rahul Barchha	Exploratory Data Analysis