CHANPREET SINGH

02113202717

CSE-1

ML

EXPERIMENT-6

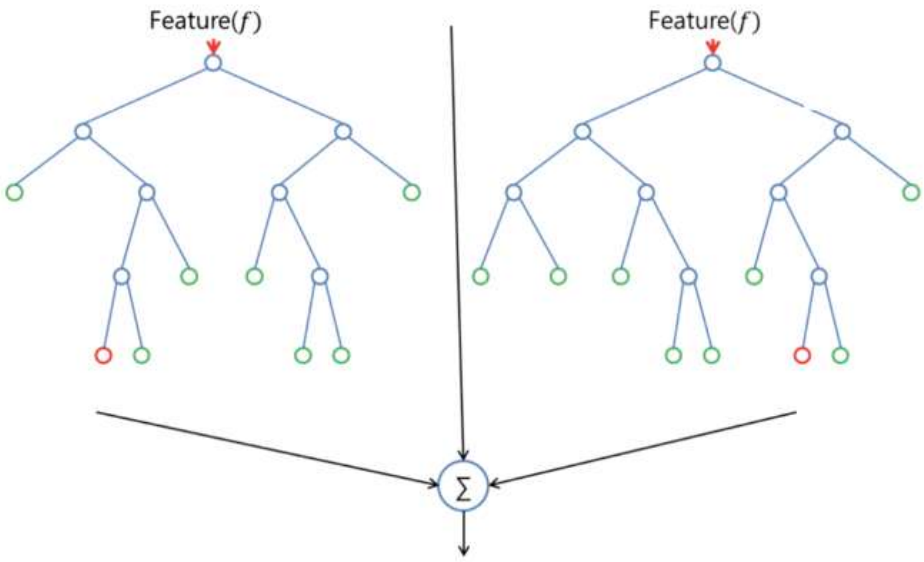
Problem Statement

Develop a machine learning method to predict stock price based on past price variation.

Algorithm

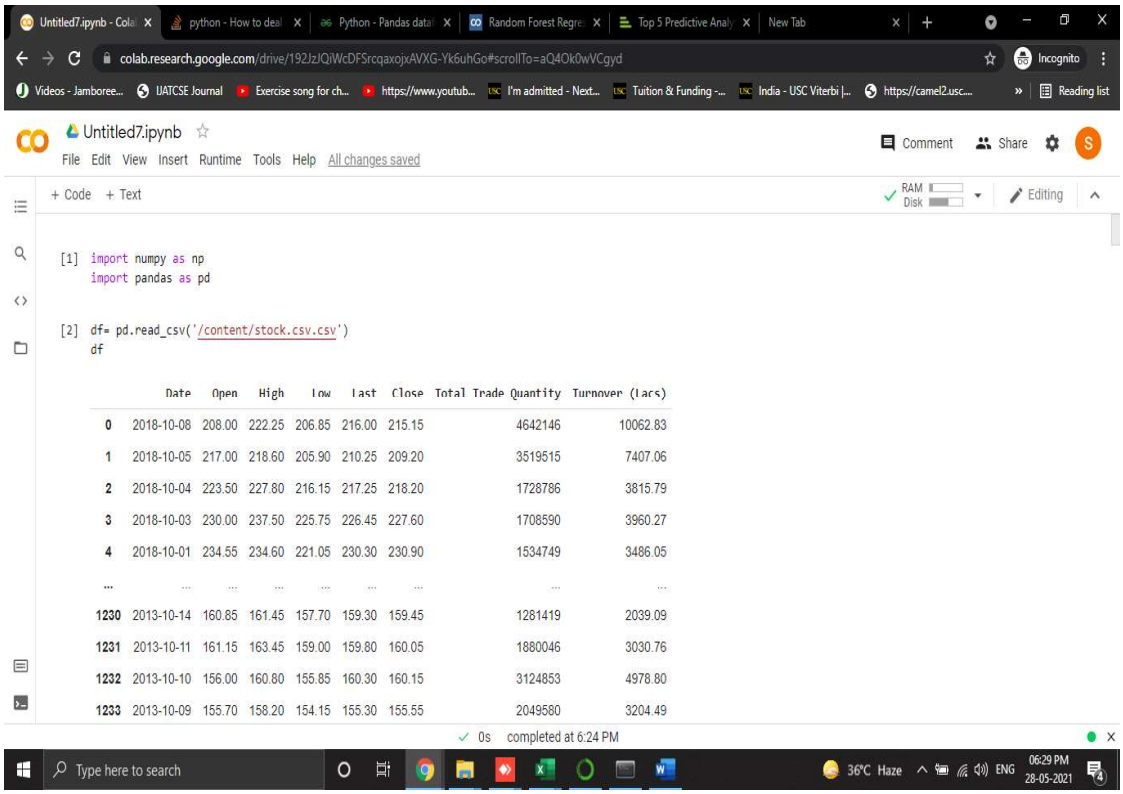
Random forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and diversity (it can be used for both classification and regression tasks). In this post we'll learn how the random forest algorithm works, how it differs from other algorithms and how to use it.

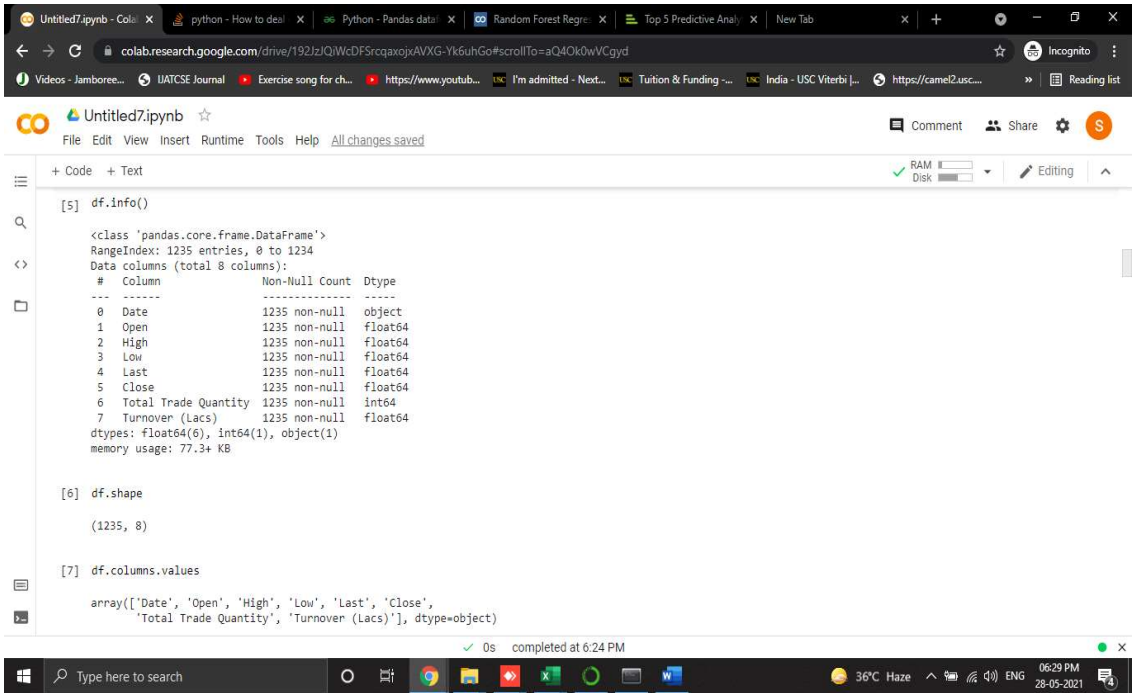
One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Let's look at random forest in classification, since classification is sometimes considered the building block of machine learning. Below you can see how a random forest would look like with two trees:



Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there's no need to combine a decision tree with a bagging classifier because you can easily use the classifier-class of random forest. With random forest, you can also deal with regression tasks by using the algorithm's regressor.

Program Snippet





Untitled7.ipynb

File Edit View Insert Runtime Tools Help All changes saved

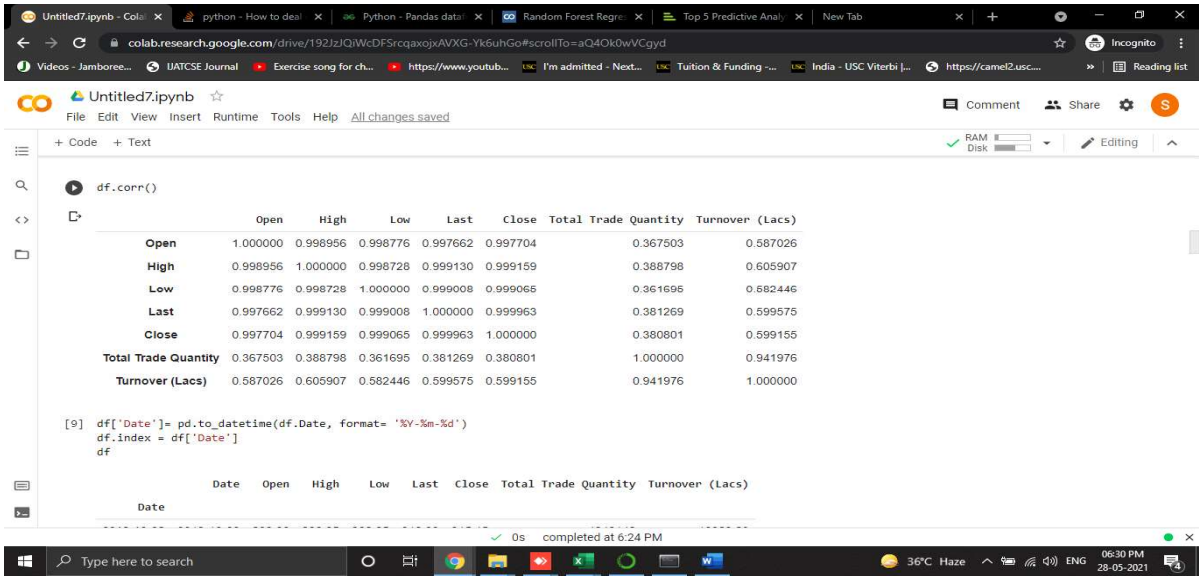
Comment Share

+ Code + Text

RAM Disk Editing

```
[5] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1235 entries, 0 to 1234
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Date                  1235 non-null   object
 1   Open                  1235 non-null   float64
 2   High                  1235 non-null   float64
 3   Low                   1235 non-null   float64
 4   Last                  1235 non-null   float64
 5   Close                 1235 non-null   float64
 6   Total Trade Quantity  1235 non-null   int64
 7   Turnover (Lacs)       1235 non-null   float64
dtypes: float64(6), int64(1), object(1)
memory usage: 77.3+ KB


[6] df.shape

(1235, 8)


[7] df.columns.values

array(['Date', 'Open', 'High', 'Low', 'Last', 'Close',
       'Total Trade Quantity', 'Turnover (Lacs)'], dtype=object)
```

0s completed at 6:24 PM



Untitled7.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

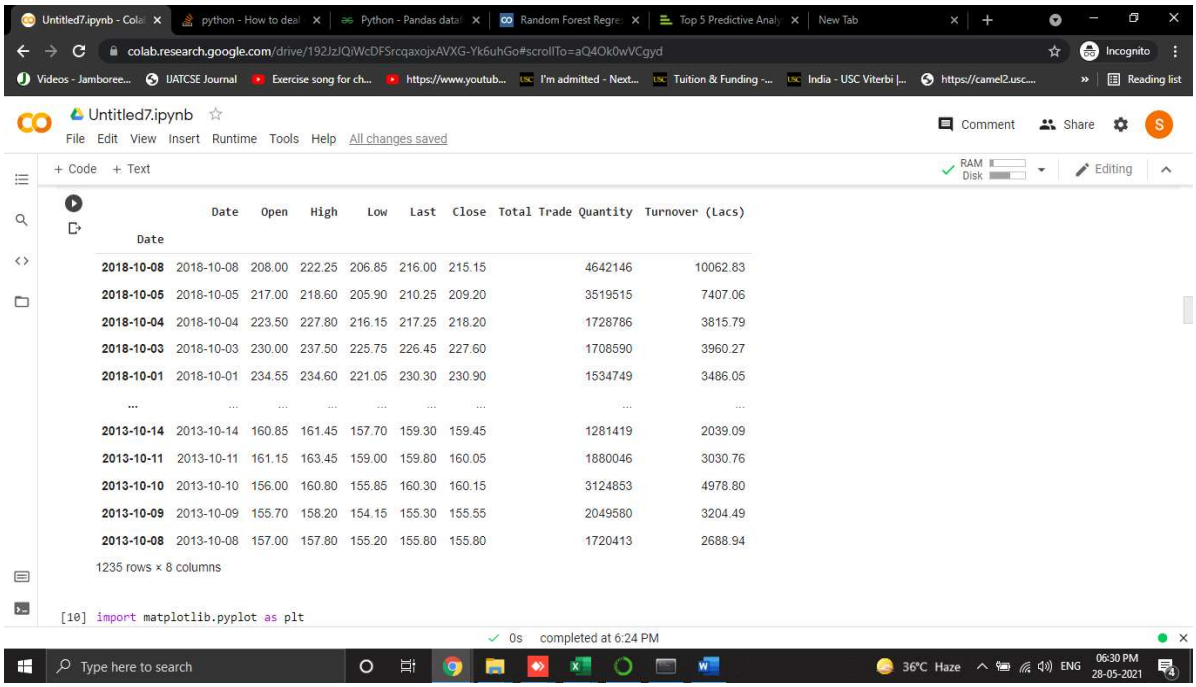
```
df.corr()
```

	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
Open	1.000000	0.998956	0.998776	0.997662	0.997704	0.367503	0.587026
High	0.998956	1.000000	0.998728	0.999130	0.999159	0.388798	0.605907
Low	0.998776	0.998728	1.000000	0.999008	0.999065	0.361695	0.582446
Last	0.997662	0.999130	0.999008	1.000000	0.999963	0.381269	0.599575
Close	0.997704	0.999159	0.999065	0.999963	1.000000	0.380801	0.599155
Total Trade Quantity	0.367503	0.388798	0.361695	0.381269	0.380801	1.000000	0.941976
Turnover (Lacs)	0.587026	0.605907	0.582446	0.599575	0.599155	0.941976	1.000000

```
[9] df['Date']= pd.to_datetime(df.Date, format= '%Y-%m-%d')
    df.index = df['Date']
    df
```

Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
Date							

0s completed at 6:24 PM



Untitled7.ipynb

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
Date								
2018-10-08	2018-10-08	208.00	222.25	206.85	216.00	215.15	4642146	10062.83
2018-10-05	2018-10-05	217.00	218.60	205.90	210.25	209.20	3519515	7407.06
2018-10-04	2018-10-04	223.50	227.80	216.15	217.25	218.20	1728786	3815.79
2018-10-03	2018-10-03	230.00	237.50	225.75	226.45	227.60	1708590	3960.27
2018-10-01	2018-10-01	234.55	234.60	221.05	230.30	230.90	1534749	3486.05
...	...	...	...	...	...	...	...	...
2013-10-14	2013-10-14	160.85	161.45	157.70	159.30	159.45	1281419	2039.09
2013-10-11	2013-10-11	161.15	163.45	159.00	159.80	160.05	1880046	3030.76
2013-10-10	2013-10-10	156.00	160.80	155.85	160.30	160.15	3124853	4978.80
2013-10-09	2013-10-09	155.70	158.20	154.15	155.30	155.55	2049580	3204.49
2013-10-08	2013-10-08	157.00	157.80	155.20	155.80	155.80	1720413	2688.94

1235 rows × 8 columns

```
[10] import matplotlib.pyplot as plt
```

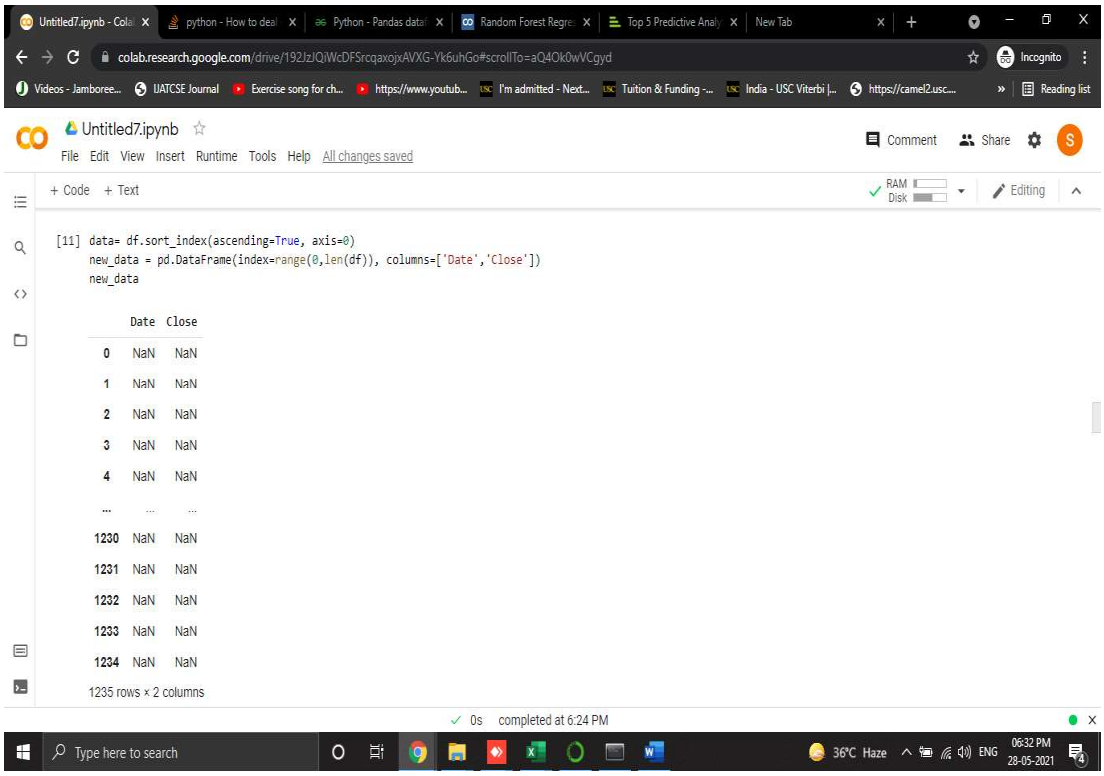
0s completed at 6:24 PM

Untitled7.ipynb

```
import matplotlib.pyplot as plt
plt.figure(figsize=(16,8))
plt.plot(df['Close'],label='CloseHistory')
```

```
[<matplotlib.lines.Line2D at 0x7f774a777310>]
```


0s completed at 6:24 PM



Untitled7.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

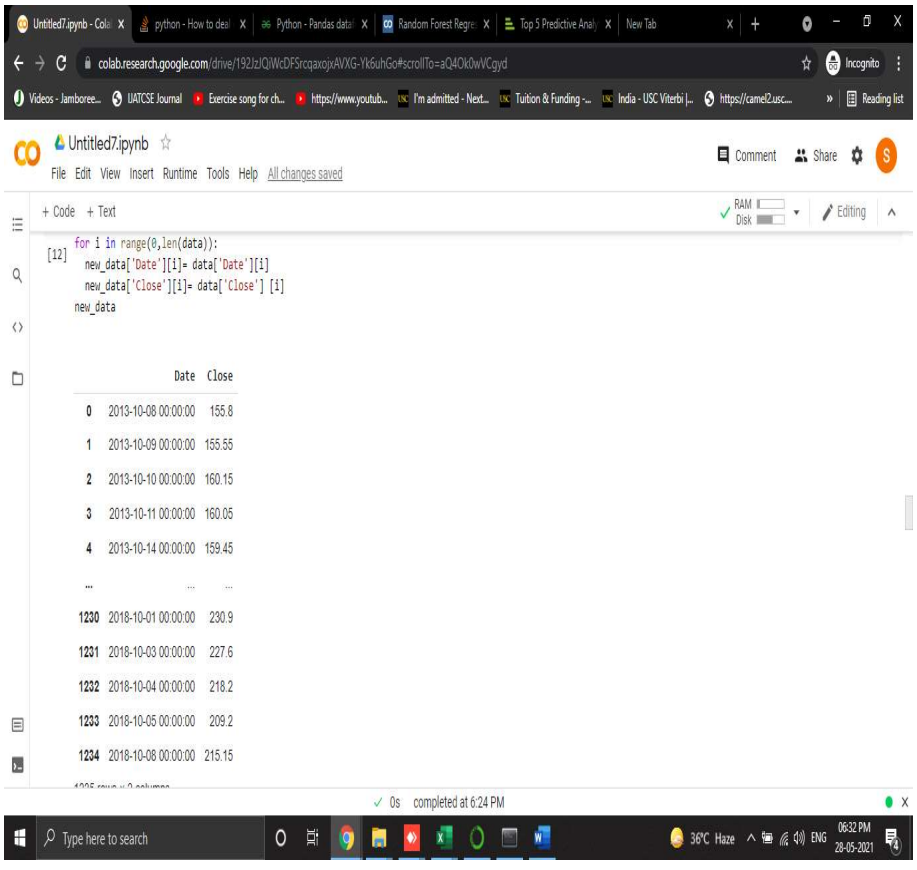
RAM Disk Editing

```
[11] data= df.sort_index(ascending=True, axis=0)
     new_data = pd.DataFrame(index=range(0,len(df)), columns=['Date','Close'])
     new_data
```

	Date	Close
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN
...	...	...
1230	NaN	NaN
1231	NaN	NaN
1232	NaN	NaN
1233	NaN	NaN
1234	NaN	NaN

1235 rows × 2 columns

✓ 0s completed at 6:24 PM



Untitled7.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

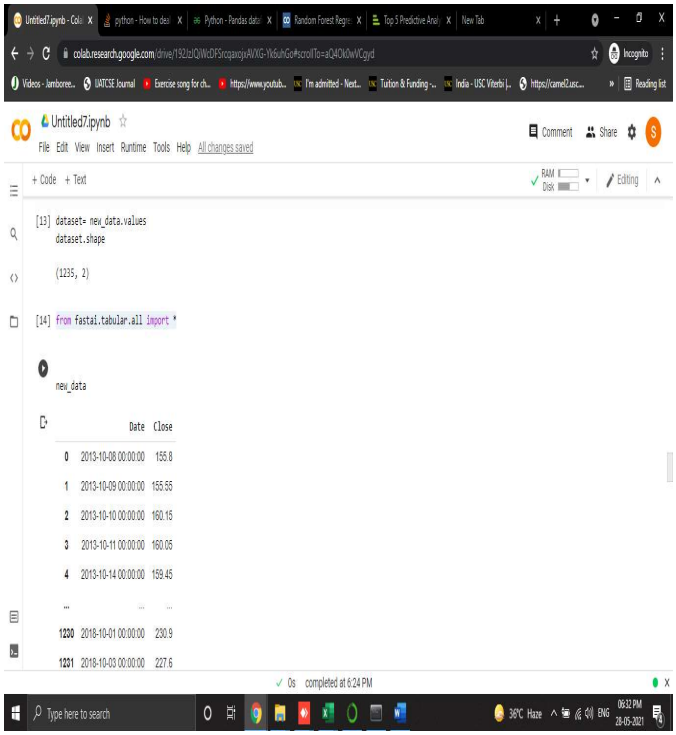
+ Code + Text

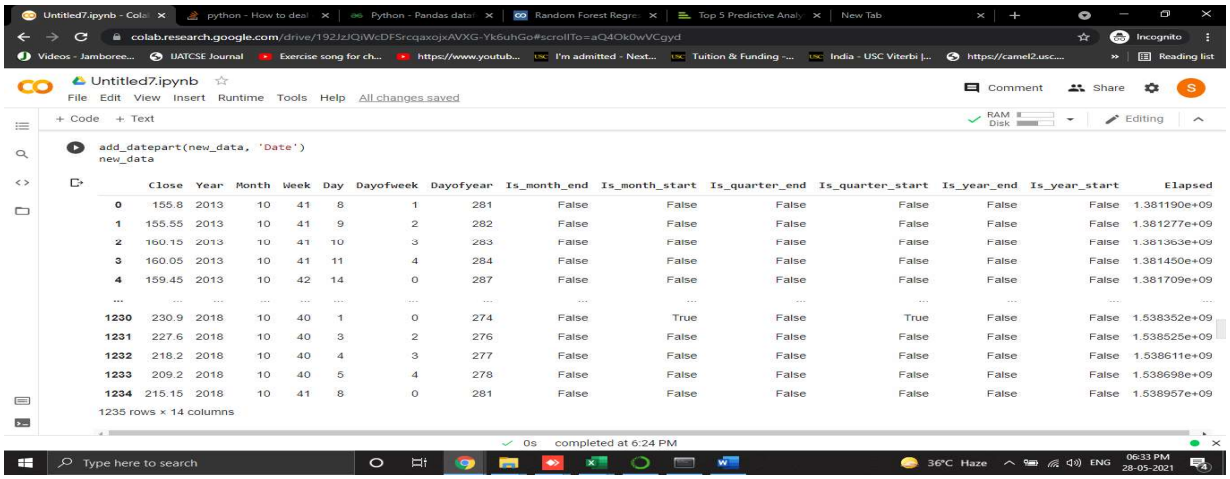
RAM Disk Editing

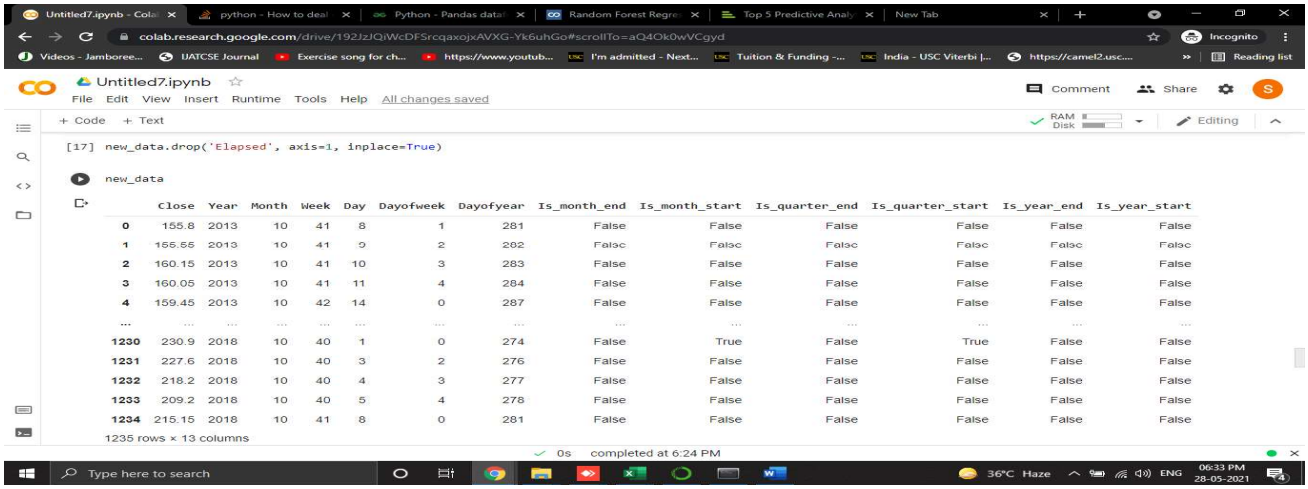
```
[12] for i in range(0,len(data)):
       new_data['Date'][i]= data['Date'][i]
       new_data['Close'][i]= data['Close'] [i]
     new_data
```

	Date	Close
0	2013-10-08 00:00:00	155.8
1	2013-10-09 00:00:00	155.55
2	2013-10-10 00:00:00	160.15
3	2013-10-11 00:00:00	160.05
4	2013-10-14 00:00:00	159.45
...	...	...
1230	2018-10-01 00:00:00	230.9
1231	2018-10-03 00:00:00	227.6
1232	2018-10-04 00:00:00	218.2
1233	2018-10-05 00:00:00	209.2
1234	2018-10-08 00:00:00	215.15

✓ 0s completed at 6:24 PM


Untitled7.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[13] dataset= new_data.values
dataset.shape
(1235, 2)
[14] from fastai.tabular.all import *
new_data
Date Close
0 2013-10-08 00:00:00 155.8
1 2013-10-09 00:00:00 155.55
2 2013-10-10 00:00:00 160.15
3 2013-10-11 00:00:00 160.05
4 2013-10-14 00:00:00 159.45
1230 2018-10-01 00:00:00 230.9
1231 2018-10-03 00:00:00 227.6
0s completed at 6:24 PM
Type here to search
36°C Haze
ENG
06:32 PM
28-05-2021


Untitled7.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
add_datepart(new_data, 'Date')
new_data
Close Year Month Week Day Dayofweek Dayofyear Is_month_end Is_month_start Is_quarter_end Is_quarter_start Is_year_end Is_year_start Elapsed
0 155.8 2013 10 41 8 1 281 False False False False False False 1.381190e+09
1 155.55 2013 10 41 9 2 282 False False False False False False 1.381277e+09
2 160.15 2013 10 41 10 3 283 False False False False False False 1.381363e+09
3 160.05 2013 10 41 11 4 284 False False False False False False 1.381450e+09
4 159.45 2013 10 42 14 0 287 False False False False False False 1.381709e+09
1230 230.9 2018 10 40 1 0 274 False True False True False False 1.538352e+09
1231 227.6 2018 10 40 3 2 276 False False False False False False 1.538525e+09
1232 218.2 2018 10 40 4 3 277 False False False False False False 1.538611e+09
1233 209.2 2018 10 40 5 4 278 False False False False False False 1.538698e+09
1234 215.15 2018 10 41 8 0 281 False False False False False False 1.538957e+09
1235 rows × 14 columns
0s completed at 6:24 PM
Type here to search
36°C Haze
ENG
06:33 PM
28-05-2021

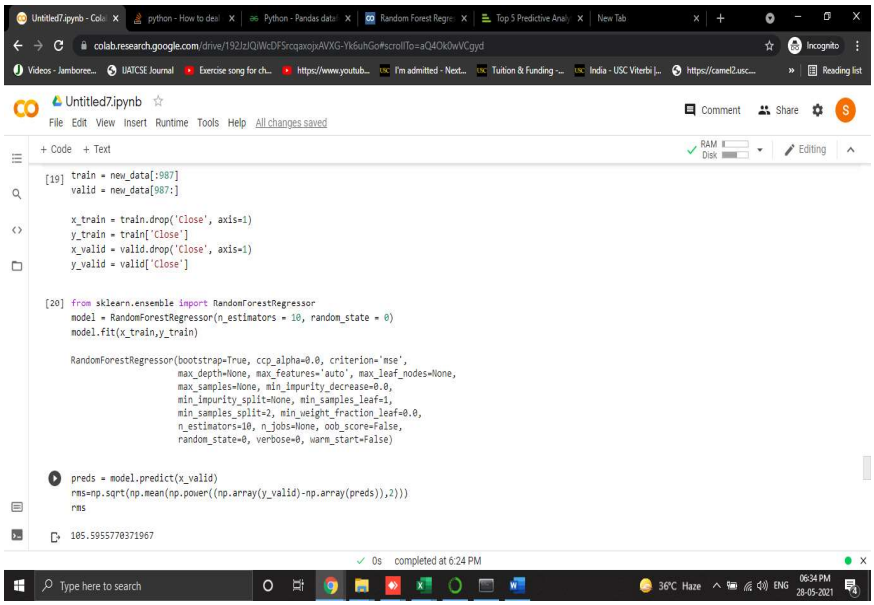


```
[17] new_data.drop('Elapsed', axis=1, inplace=True)
```

```
new_data
```

	Close	Year	Month	Week	Day	Dayofweek	Dayofyear	Is_month_end	Is_month_start	Is_quarter_end	Is_quarter_start	Is_year_end	Is_year_start
0	155.8	2013	10	41	8	1	281	False	False	False	False	False	False
1	155.65	2013	10	41	9	2	282	False	False	False	False	False	False
2	160.15	2013	10	41	10	3	283	False	False	False	False	False	False
3	160.05	2013	10	41	11	4	284	False	False	False	False	False	False
4	159.45	2013	10	42	14	0	287	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1230	230.9	2018	10	40	1	0	274	False	True	False	True	False	False
1231	227.6	2018	10	40	3	2	276	False	False	False	False	False	False
1232	218.2	2018	10	40	4	3	277	False	False	False	False	False	False
1233	209.2	2018	10	40	5	4	278	False	False	False	False	False	False
1234	215.15	2018	10	41	8	0	281	False	False	False	False	False	False

1235 rows × 13 columns

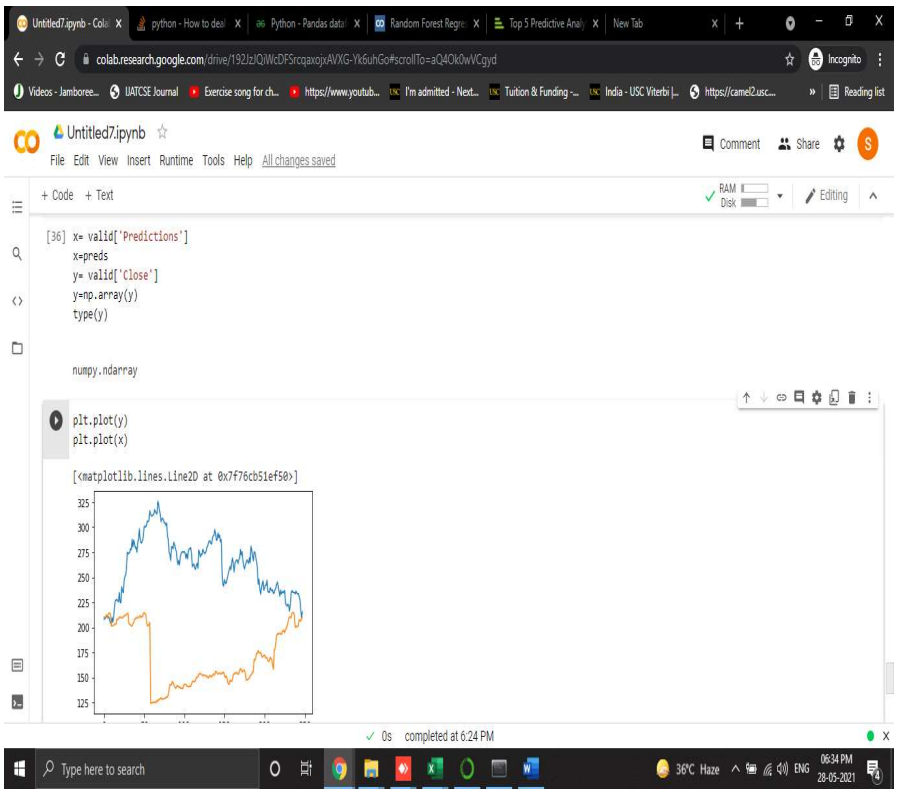


```
[19] train = new_data[:987]
     valid = new_data[987:]

     x_train = train.drop('Close', axis=1)
     y_train = train['Close']
     x_valid = valid.drop('Close', axis=1)
     y_valid = valid['Close']
```

```
[20] from sklearn.ensemble import RandomForestRegressor
     model = RandomForestRegressor(n_estimators = 10, random_state = 0)
     model.fit(x_train,y_train)

     RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                           max_depth=None, max_features='auto', max_leaf_nodes=None,
                           max_samples=None, min_impurity_decrease=0.0,
                           min_impurity_split=None, min_samples_leaf=1,
                           min_samples_split=2, min_weight_fraction_leaf=0.0,
                           n_estimators=10, n_jobs=None, oob_score=False,
                           random_state=0, verbose=0, warm_start=False)
```

```
preds = model.predict(x_valid)
rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
rms
```

```
105.5955770371967
```



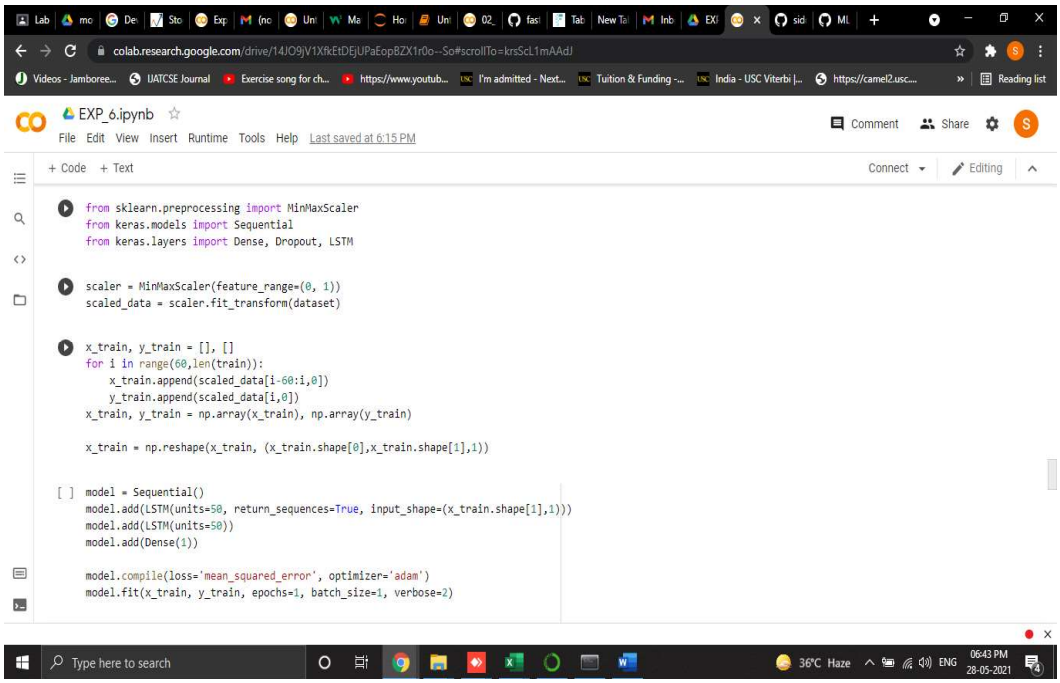
Long Short Term Memory (LSTM)

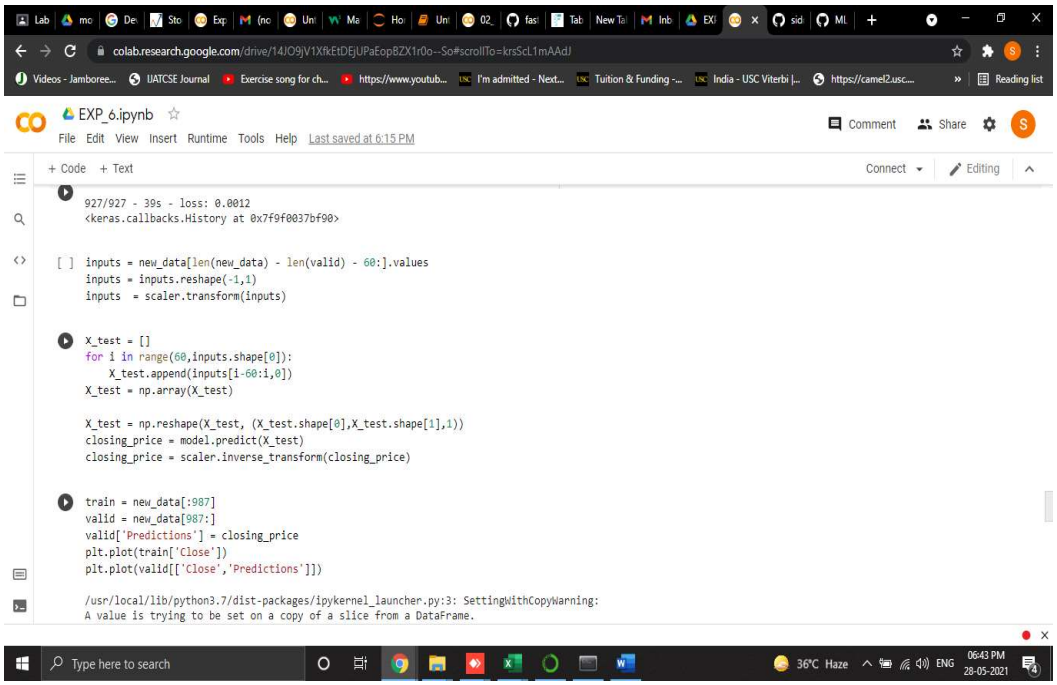
LSTMs are widely used for sequence prediction problems and have proven to be extremely effective. The reason they work so well is because LSTM is able to store past information that is important, and forget the information that is not. LSTM has three gates:

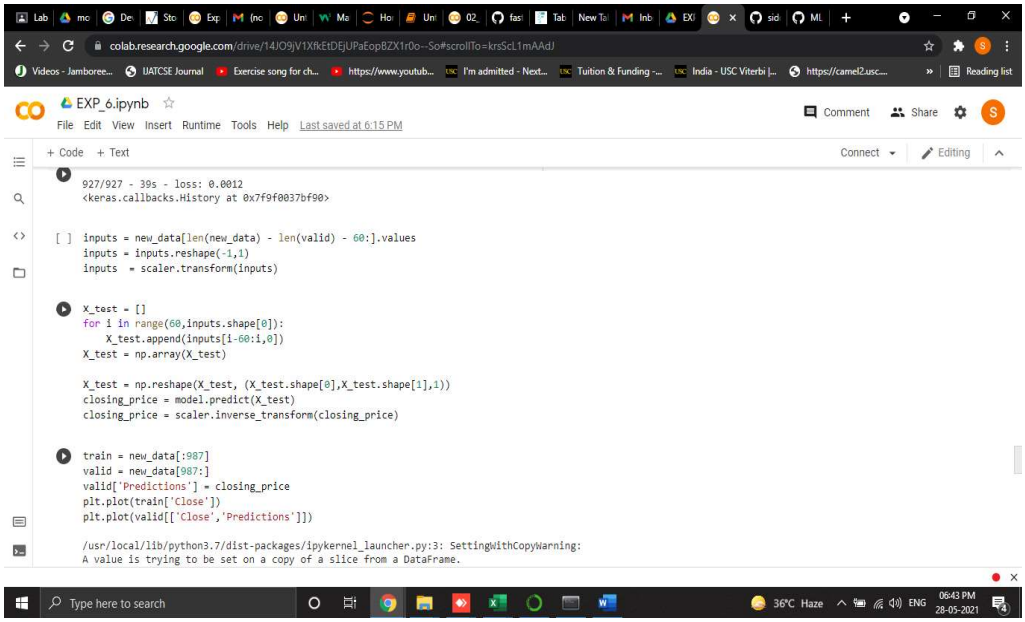
The input gate: The input gate adds information to the cell state

The forget gate: It removes the information that is no longer required by the model

The output gate: Output Gate at LSTM selects the information to be shown as output


colab.research.google.com/drive/14JO9jV1XfkEtDEjUPaEopBZX1r0o--So#scrollTo=krsScL1mAAdJ
EXP_6.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6:15 PM
Comment
Share
+ Code + Text
Connect
Editing
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)
x_train, y_train = [], []
for i in range(60,len(train)):
x_train.append(scaled_data[i-60:i,0])
y_train.append(scaled_data[i,0])
x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=50))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=1, batch_size=1, verbose=2)
Type here to search
36°C Haze
ENG
06:43 PM
28-05-2021


colab.research.google.com/drive/14JO9jV1XfkEtDEjUPaEopBZX1r0o--So#scrollTo=krsScL1mAAdJ
EXP_6.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6:15 PM
Comment
Share
+ Code + Text
Connect
Editing
927/927 - 39s - loss: 0.0012
<keras.callbacks.History at 0x7f9f0037bf90>
inputs = new_data[len(new_data) - len(valid) - 60:].values
inputs = inputs.reshape(-1,1)
inputs = scaler.transform(inputs)
X_test = []
for i in range(60,inputs.shape[0]):
X_test.append(inputs[i-60:i,0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
closing_price = model.predict(X_test)
closing_price = scaler.inverse_transform(closing_price)
train = new_data[:987]
valid = new_data[987:]
valid['Predictions'] = closing_price
plt.plot(train['Close'])
plt.plot(valid[['Close','Predictions']])
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Type here to search
36°C Haze
ENG
06:43 PM
28-05-2021



Github link-

https://github.com/chanpreet1999/ML-Assignment/tree/master/Exp6