

# Task-3

# Stock Prediction

# **Samritha A.R**



# Importing all necessary Libraries

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** Stock Prediction.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved
- Right Panel:** Comment, Share, Settings, User profile, RAM/Disk status, Colab AI
- Sidebar:** Code (+), Text (+), Task-3 (expanded), Stock Prediction (selected), Import all the required libraries
- Code Cell 1:** Imports pandas, datetime, date, matplotlib.pyplot, yfinance, numpy, and tensorflow. A FutureWarning message is displayed at the bottom of the cell.
- Code Cell 2:** Defines START as "2010-01-01" and TODAY as date.today().strftime("%Y-%m-%d"). It also defines a function to load the dataset.
- Bottom Status Bar:** 0s completed at 11:31

# Fetching TCS Dataset from Yahoo Finance Library

The screenshot shows a Jupyter Notebook interface with the title "Stock Prediction.ipynb". The notebook contains the following code:

```
[2]: START = "2010-01-01"
TODAY = date.today().strftime("%Y-%m-%d")

# Define a function to load the dataset

def load_data(ticker):
    data = yf.download(ticker, START, TODAY)
    data.reset_index(inplace=True)
    return data

[4]: data = load_data('TCS.NS')
df=data
df.head()
```

The output of cell [4] shows the first five rows of the fetched dataset:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2010-01-04	377.399994	379.450012	374.524994	375.825012	284.856262	1963682
1	2010-01-05	377.500000	379.774994	373.274994	375.924988	284.932098	2014488
2	2010-01-06	375.924988	376.100006	366.500000	367.424988	278.489594	3349176
3	2010-01-07	367.750000	369.700012	355.500000	357.200012	270.739532	6474892
4	2010-01-08	358.000000	359.250000	348.250000	349.899994	265.206360	6048178

```
[5]: df = df.drop(['Date', 'Adj Close'], axis = 1)
df.head()
```

The status bar at the bottom indicates "0s completed at 11:31".

# Dropping 'Date', 'Adj Close' column & Visualizing Closing Price of TCS

The screenshot shows a Google Colab notebook titled "Stock Prediction.ipynb". The interface includes a toolbar with file operations, a sidebar with search and filter icons, and a main workspace with code cells and output.

**Code Cells:**

- [4] 0s `2 2010-01-06 375.924988 376.100006 366.500000 367.424988 278.489594 3349176  
3 2010-01-07 367.750000 369.700012 355.500000 357.200012 270.739532 6474892  
4 2010-01-08 358.000000 359.250000 348.250000 349.899994 265.206360 6048178`
- [5] 0s `df = df.drop(['Date', 'Adj Close'], axis = 1)  
df.head()`

The output of cell [5] displays a Pandas DataFrame with columns: Open, High, Low, Close, Volume. The data is as follows:

	Open	High	Low	Close	Volume
0	377.399994	379.450012	374.524994	375.825012	1963682
1	377.500000	379.774994	373.274994	375.924988	2014488
2	375.924988	376.100006	366.500000	367.424988	3349176
3	367.750000	369.700012	355.500000	357.200012	6474892
4	358.000000	359.250000	348.250000	349.899994	6048178

**Section Header:**

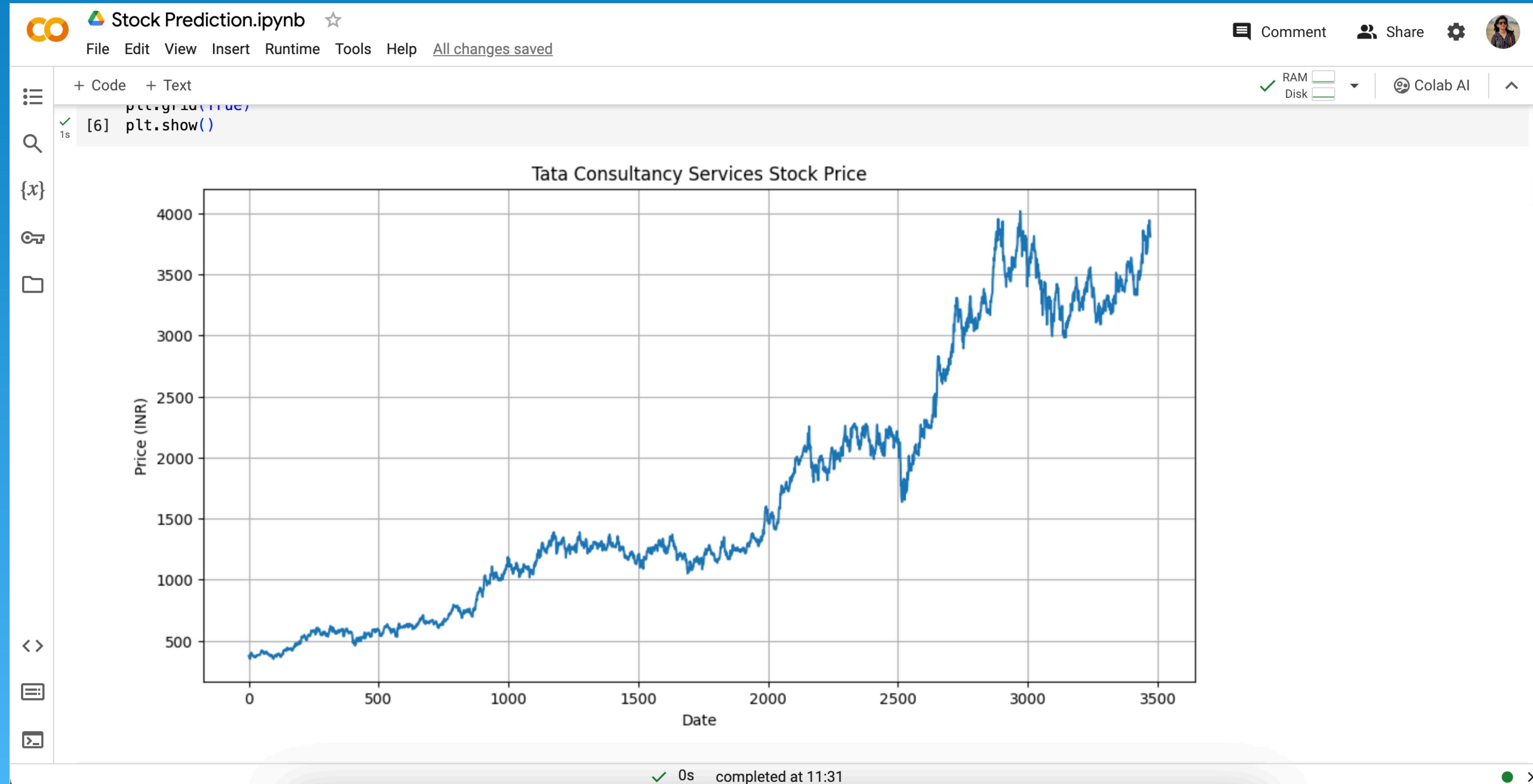
## Visualizing Closing Price

**Code Cell:**

```
1s ▶ plt.figure(figsize=(12, 6))  
plt.plot(df['Close'])  
plt.title("Tata Consultancy Services Stock Price")  
plt.xlabel("Date")  
plt.ylabel("Price (INR)")  
plt.grid(True)  
plt.show()
```

The status bar at the bottom indicates "0s completed at 11:31".

# Visualizing Closing Price of TCS



# Dataframe and Dimensions of the TCS dataset

The screenshot shows a Jupyter Notebook interface with the title "Stock Prediction.ipynb". The notebook contains the following content:

**Code Cell 1:**

```
df
```

The output of this cell is a DataFrame named "df" with the following structure:

	Open	High	Low	Close	Volume
0	377.399994	379.450012	374.524994	375.825012	1963682
1	377.500000	379.774994	373.274994	375.924988	2014488
2	375.924988	376.100006	366.500000	367.424988	3349176
3	367.750000	369.700012	355.500000	357.200012	6474892
4	358.000000	359.250000	348.250000	349.899994	6048178
...	...	...	...	...	...
3468	3945.000000	3963.550049	3915.050049	3943.050049	2941975
3469	3943.050049	3943.050049	3943.050049	3943.050049	0
3470	3900.000000	3933.899902	3842.750000	3858.250000	2618854
3471	3880.000000	3883.649902	3805.600098	3841.800049	2657709
3472	3839.899902	3861.000000	3778.699951	3810.300049	2205154

3473 rows × 5 columns

**Text Cell 2:**

Plotting moving averages of 100 day

**Code Cell 3:**

```
[8] ma100 = df.Close.rolling(100).mean()  
ma100
```

The cell was completed at 11:31.

# Plotting moving average of 100 days

Stock Prediction.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ Colab AI ^

RAM Disk

+ Code + Text

**Plotting moving averages of 100 day**

```
{x} 0s ma100 = df.Close.rolling(100).mean()
ma100
```

0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
..	
3468	3569.765002
3469	3575.382502
3470	3580.209502
3471	3584.866003
3472	3589.059504
Name:	Close, Length: 3473, dtype: float64

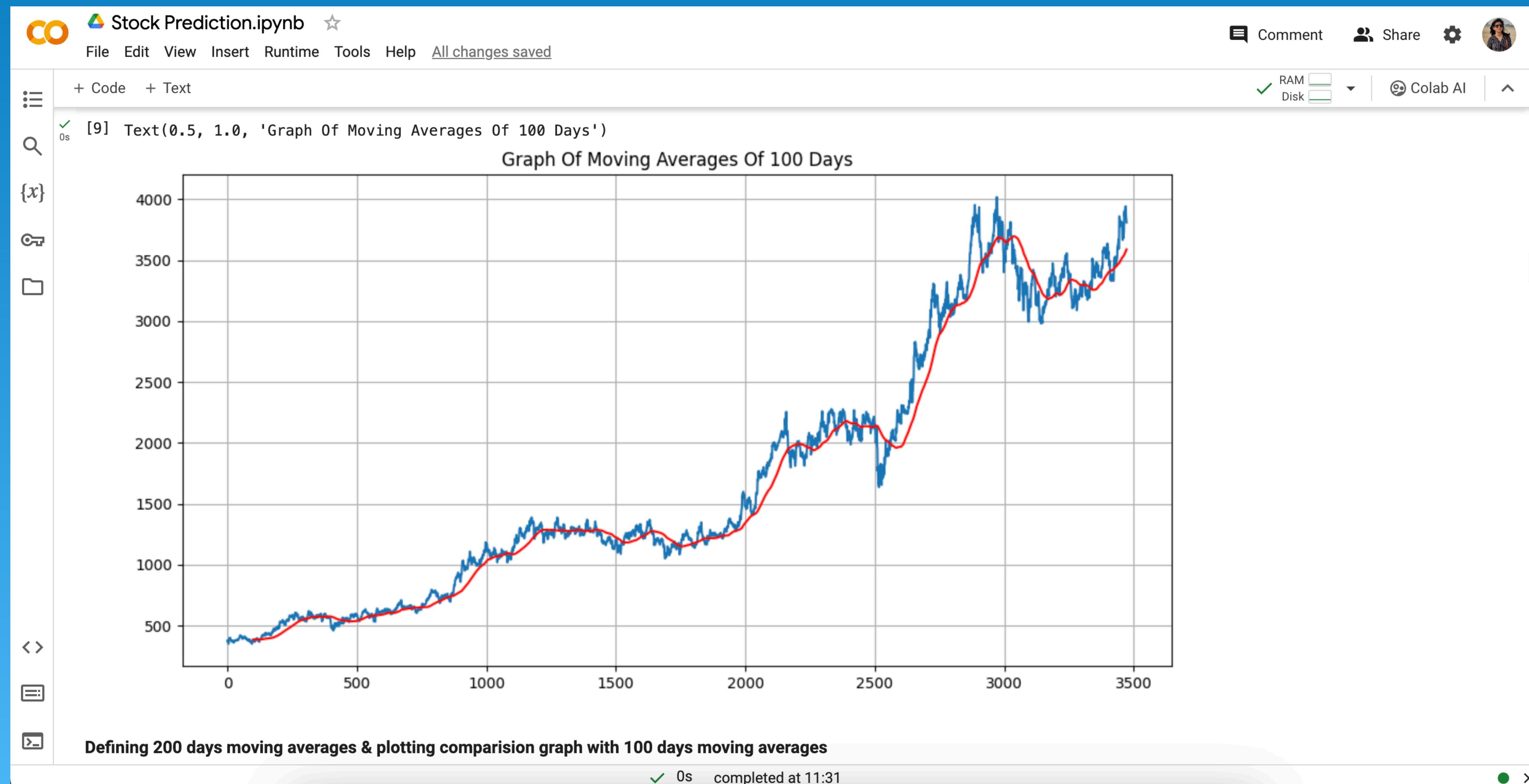
```
[9] plt.figure(figsize = (12,6))
plt.plot(df.Close)
plt.plot(ma100, 'r')
plt.grid(True)
plt.title('Graph Of Moving Averages Of 100 Days')
```

Text(0.5, 1.0, 'Graph Of Moving Averages Of 100 Days')

Graph Of Moving Averages Of 100 Days

0s completed at 11:31

# Plotting moving average of 100 days



# Plotting comparison graph with 100 & 200 days moving averages of TCS

Stock Prediction.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

RAM Disk Colab AI

+ Code + Text

Defining 200 days moving averages & plotting comparision graph with 100 days moving averages

```
[10] ma200 = df.Close.rolling(200).mean()
      ma200
```

0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
..	..
3468	3432.585748
3469	3436.606499
3470	3439.868250
3471	3443.077250
3472	3445.929750

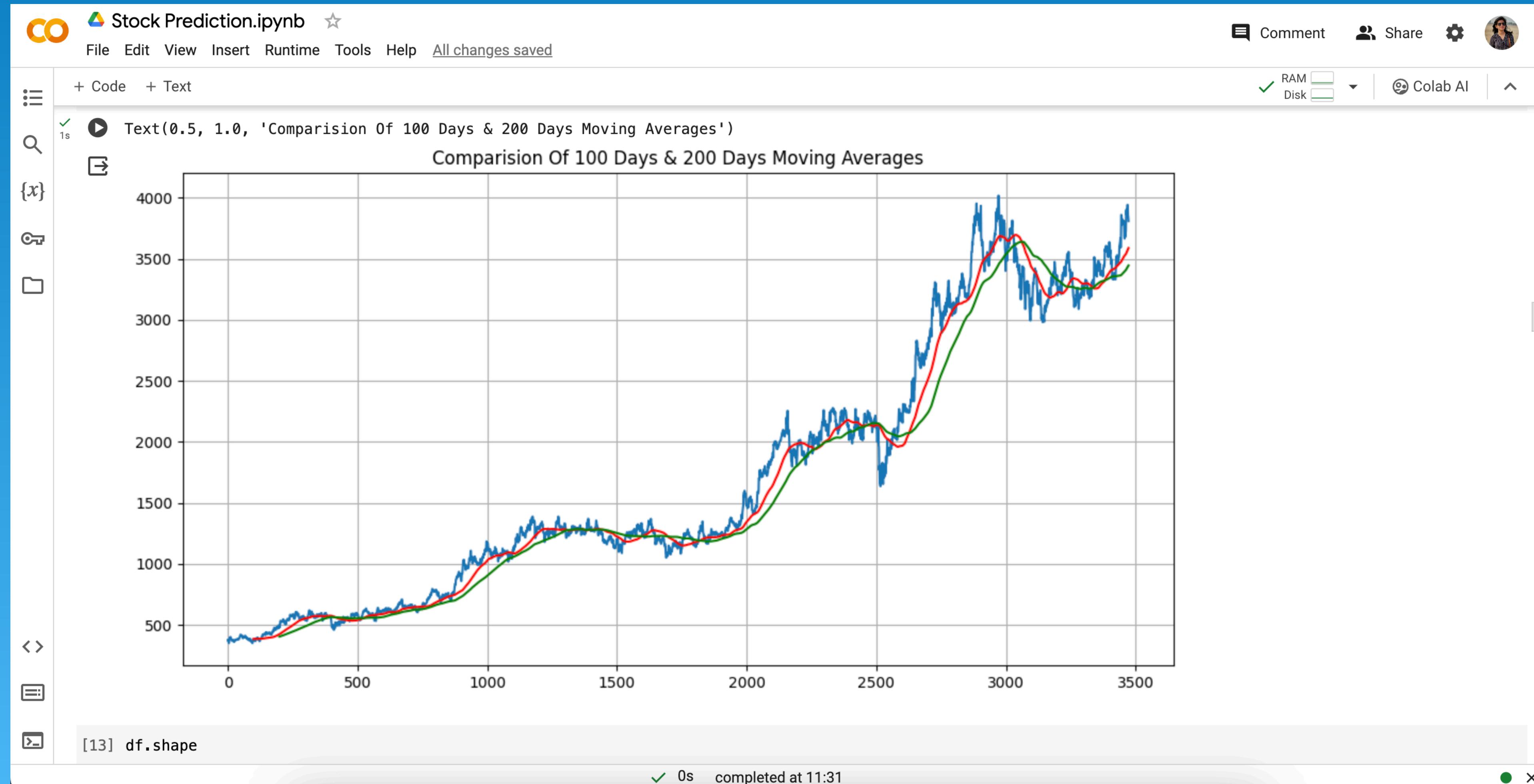
Name: Close, Length: 3473, dtype: float64

```
plt.figure(figsize = (12,6))
plt.plot(df.Close)
plt.plot(ma100, 'r')
plt.plot(ma200, 'g')
plt.grid(True)
plt.title('Comparision Of 100 Days & 200 Days Moving Averages')
```

Text(0.5, 1.0, 'Comparision Of 100 Days & 200 Days Moving Averages')

Comparision Of 100 Days & 200 Days Moving Averages

# Plotting comparison graph with 100 & 200 days moving averages of TCS



## Splitting the dataset into training & testing sets

Stock Prediction.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings User Picture

+ Code + Text RAM Disk Colab AI

df.shape  
(3473, 5)

Splitting the dataset into training (70%) & testing (30%) set

[14] # Splitting data into training and testing

```
train = pd.DataFrame(data[0:int(len(data)*0.70)])
test = pd.DataFrame(data[int(len(data)*0.70): int(len(data))])

print(train.shape)
print(test.shape)

(2431, 7)
(1042, 7)
```

[15] train.head()

	Date	Open	High	Low	Close	Adj Close	Volume
0	2010-01-04	377.399994	379.450012	374.524994	375.825012	284.856262	1963682
1	2010-01-05	377.500000	379.774994	373.274994	375.924988	284.932098	2014488
2	2010-01-06	375.924988	376.100006	366.500000	367.424988	278.489594	3349176
3	2010-01-07	367.750000	369.700012	355.500000	357.200012	270.739532	6474892
4	2010-01-08	358.000000	359.250000	348.250000	349.899994	265.206360	6048178

0s completed at 11:31

## Fetching the train & test dataset's 1st five rows

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** Stock Prediction.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved
- Code Cells:**
  - [14] `print(train.shape)`  
Output: (2431, 7)  
(1042, 7)
  - [15] `train.head()`  
Output:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2010-01-04	377.399994	379.450012	374.524994	375.825012	284.856262	1963682
1	2010-01-05	377.500000	379.774994	373.274994	375.924988	284.932098	2014488
2	2010-01-06	375.924988	376.100006	366.500000	367.424988	278.489594	3349176
3	2010-01-07	367.750000	369.700012	355.500000	357.200012	270.739532	6474892
4	2010-01-08	358.000000	359.250000	348.250000	349.899994	265.206360	6048178
  - [16] `test.head()`  
Output:

	Date	Open	High	Low	Close	Adj Close	Volume
2431	2019-11-18	2178.399902	2187.750000	2148.750000	2152.600098	1985.483276	1611089
2432	2019-11-19	2153.000000	2154.649902	2105.000000	2108.800049	1945.084106	2975313
2433	2019-11-20	2121.949951	2135.000000	2094.500000	2108.550049	1944.852783	2894129
2434	2019-11-21	2112.000000	2126.600098	2107.399902	2118.100098	1953.661377	2052574
2435	2019-11-22	2097.000000	2107.000000	2060.500000	2071.699951	1910.863770	3742049
- User Interface:** Includes sidebar icons for file operations, search, and help, and a top bar with Comment, Share, and settings.

# Normalization of the dataset

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** Stock Prediction.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved
- Right Sidebar:** Comment, Share, Settings, User profile, RAM/Disk status, Colab AI
- Code Cells:**
  - [17] from sklearn.preprocessing import MinMaxScaler  
scaler = MinMaxScaler(feature\_range=(0,1))
  - [18] train\_close = train.iloc[:, 4:5].values  
test\_close = test.iloc[:, 4:5].values
  - [19] data\_training\_array = scaler.fit\_transform(train\_close)  
data\_training\_array  
array([[0.01351019],  
 [0.01356204],  
 [0.00915373],  
 ...,  
 [0.94847467],  
 [0.95791362],  
 [0.9463223 ]])
  - [20] x\_train = []  
y\_train = []  
  
for i in range(100, data\_training\_array.shape[0]):  
 x\_train.append(data\_training\_array[i-100: i])  
 y\_train.append(data\_training\_array[i, 0])  
  
x\_train, y\_train = np.array(x\_train), np.array(y\_train)
  - [21] x\_train.shape
- Bottom Status Bar:** ✓ 0s completed at 11:31

# LSTM Machine Learning Model

The screenshot shows a Google Colab notebook titled "Stock Prediction.ipynb". The notebook interface includes a toolbar at the top with File, Edit, View, Insert, Runtime, Tools, Help, and a status bar indicating "All changes saved". On the left, there are sidebar icons for code, text, search, variables, and cell controls. The main workspace displays the following Python code for an LSTM model:

```
from tensorflow.keras.layers import Dense, Dropout, LSTM
from tensorflow.keras.models import Sequential

model = Sequential()
model.add(LSTM(units = 50, activation = 'relu', return_sequences=True
               ,input_shape = (x_train.shape[1], 1)))
model.add(Dropout(0.2))

model.add(LSTM(units = 60, activation = 'relu', return_sequences=True))
model.add(Dropout(0.3))

model.add(LSTM(units = 80, activation = 'relu', return_sequences=True))
model.add(Dropout(0.4))

model.add(LSTM(units = 120, activation = 'relu'))
model.add(Dropout(0.5))

model.add(Dense(units = 1))

[24] model.summary()
```

The code defines an LSTM model with four layers. The first layer has 50 units, the second has 60, the third has 80, and the fourth has 120. Each layer uses the ReLU activation function and returns sequences. Between the first and second layers, there is an input shape specification. Each layer is followed by a 20% dropout layer. Finally, a dense layer with one unit is added. The last cell, [24], shows the model summary.

# Model Summary of LSTM

The screenshot shows a Jupyter Notebook interface with the title "Stock Prediction.ipynb". The code cell at index 23 contains the command `model.add(Dense(units = 1))` and the command `model.summary()`. The output cell displays the model summary for a sequential model, listing the layers, their types, output shapes, and parameter counts:

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 100, 50)	10400
dropout (Dropout)	(None, 100, 50)	0
lstm_1 (LSTM)	(None, 100, 60)	26640
dropout_1 (Dropout)	(None, 100, 60)	0
lstm_2 (LSTM)	(None, 100, 80)	45120
dropout_2 (Dropout)	(None, 100, 80)	0
lstm_3 (LSTM)	(None, 120)	96480
dropout_3 (Dropout)	(None, 120)	0
dense (Dense)	(None, 1)	121

At the bottom of the summary, it states:

```
Total params: 178761 (698.29 KB)
Trainable params: 178761 (698.29 KB)
Non-trainable params: 0 (0.00 Byte)
```

The notebook interface includes a sidebar with various icons for file operations, a search bar, and a runtime status bar indicating "0s completed at 11:31".

# Training the Model

The screenshot shows a Google Colab interface with a Jupyter notebook titled "Stock Prediction.ipynb". The notebook has a single cell containing Python code for training a machine learning model. The code imports TensorFlow, compiles a model with Adam optimizer, mean\_squared\_error loss, and MeanAbsoluteError metric, and fits it with 50 epochs. The output of the cell shows the training progress for 15 epochs, with each epoch taking approximately 20-30 seconds and achieving a mean absolute error between 0.039 and 0.046.

```
[26] import tensorflow as tf
model.compile(optimizer = 'adam', loss = 'mean_squared_error', metrics=[tf.keras.metrics.MeanAbsoluteError()])
model.fit(x_train, y_train, epochs = 50)

Epoch 1/50
73/73 [=====] - 28s 304ms/step - loss: 0.0046 - mean_absolute_error: 0.0463
Epoch 2/50
73/73 [=====] - 22s 300ms/step - loss: 0.0041 - mean_absolute_error: 0.0446
Epoch 3/50
73/73 [=====] - 21s 287ms/step - loss: 0.0045 - mean_absolute_error: 0.0469
Epoch 4/50
73/73 [=====] - 22s 300ms/step - loss: 0.0039 - mean_absolute_error: 0.0441
Epoch 5/50
73/73 [=====] - 22s 301ms/step - loss: 0.0037 - mean_absolute_error: 0.0417
Epoch 6/50
73/73 [=====] - 22s 302ms/step - loss: 0.0032 - mean_absolute_error: 0.0399
Epoch 7/50
73/73 [=====] - 23s 311ms/step - loss: 0.0032 - mean_absolute_error: 0.0404
Epoch 8/50
73/73 [=====] - 21s 291ms/step - loss: 0.0033 - mean_absolute_error: 0.0400
Epoch 9/50
73/73 [=====] - 22s 302ms/step - loss: 0.0032 - mean_absolute_error: 0.0408
Epoch 10/50
73/73 [=====] - 22s 302ms/step - loss: 0.0032 - mean_absolute_error: 0.0407
Epoch 11/50
73/73 [=====] - 22s 303ms/step - loss: 0.0031 - mean_absolute_error: 0.0393
Epoch 12/50
73/73 [=====] - 21s 289ms/step - loss: 0.0031 - mean_absolute_error: 0.0393
Epoch 13/50
73/73 [=====] - 22s 301ms/step - loss: 0.0030 - mean_absolute_error: 0.0392
Epoch 14/50
73/73 [=====] - 23s 314ms/step - loss: 0.0032 - mean_absolute_error: 0.0396
Epoch 15/50
```

# Training the Model

Stock Prediction.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share 

+ Code + Text RAM Disk Colab AI ^

Training the model

```
{x} 18m import tensorflow as tf
model.compile(optimizer = 'adam', loss = 'mean_squared_error', metrics=[tf.keras.metrics.MeanAbsoluteError()])
model.fit(x_train, y_train, epochs = 50)

Epoch 15/50
73/73 [=====] - 22s 300ms/step - loss: 0.0030 - mean_absolute_error: 0.0396
Epoch 16/50
73/73 [=====] - 22s 299ms/step - loss: 0.0028 - mean_absolute_error: 0.0372
Epoch 17/50
73/73 [=====] - 21s 288ms/step - loss: 0.0026 - mean_absolute_error: 0.0366
Epoch 18/50
73/73 [=====] - 22s 303ms/step - loss: 0.0026 - mean_absolute_error: 0.0373
Epoch 19/50
73/73 [=====] - 22s 302ms/step - loss: 0.0028 - mean_absolute_error: 0.0375
Epoch 20/50
73/73 [=====] - 22s 300ms/step - loss: 0.0028 - mean_absolute_error: 0.0384
Epoch 21/50
73/73 [=====] - 21s 293ms/step - loss: 0.0025 - mean_absolute_error: 0.0361
Epoch 22/50
73/73 [=====] - 22s 303ms/step - loss: 0.0027 - mean_absolute_error: 0.0373
Epoch 23/50
73/73 [=====] - 22s 302ms/step - loss: 0.0025 - mean_absolute_error: 0.0363
Epoch 24/50
73/73 [=====] - 22s 298ms/step - loss: 0.0026 - mean_absolute_error: 0.0366
Epoch 25/50
73/73 [=====] - 22s 297ms/step - loss: 0.0025 - mean_absolute_error: 0.0369
Epoch 26/50
73/73 [=====] - 21s 286ms/step - loss: 0.0025 - mean_absolute_error: 0.0368
Epoch 27/50
73/73 [=====] - 22s 301ms/step - loss: 0.0024 - mean_absolute_error: 0.0358
Epoch 28/50
73/73 [=====] - 22s 302ms/step - loss: 0.0024 - mean_absolute_error: 0.0357
Epoch 29/50
```

# Training the Model

Stock Prediction.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings User profile

RAM Disk Colab AI

+ Code + Text

18m

Epoch 29/50  
73/73 [=====] - 22s 302ms/step - loss: 0.0024 - mean\_absolute\_error: 0.0358

Epoch 30/50  
73/73 [=====] - 23s 316ms/step - loss: 0.0025 - mean\_absolute\_error: 0.0363

Epoch 31/50  
73/73 [=====] - 21s 290ms/step - loss: 0.0025 - mean\_absolute\_error: 0.0368

Epoch 32/50  
73/73 [=====] - 22s 302ms/step - loss: 0.0029 - mean\_absolute\_error: 0.0386

Epoch 33/50  
73/73 [=====] - 22s 301ms/step - loss: 0.0022 - mean\_absolute\_error: 0.0345

Epoch 34/50  
73/73 [=====] - 22s 302ms/step - loss: 0.0024 - mean\_absolute\_error: 0.0359

Epoch 35/50  
73/73 [=====] - 21s 291ms/step - loss: 0.0023 - mean\_absolute\_error: 0.0351

Epoch 36/50  
73/73 [=====] - 22s 298ms/step - loss: 0.0023 - mean\_absolute\_error: 0.0349

Epoch 37/50  
73/73 [=====] - 22s 302ms/step - loss: 0.0023 - mean\_absolute\_error: 0.0354

Epoch 38/50  
73/73 [=====] - 23s 319ms/step - loss: 0.0023 - mean\_absolute\_error: 0.0357

Epoch 39/50  
73/73 [=====] - 22s 300ms/step - loss: 0.0024 - mean\_absolute\_error: 0.0356

Epoch 40/50  
73/73 [=====] - 21s 288ms/step - loss: 0.0023 - mean\_absolute\_error: 0.0346

Epoch 41/50  
73/73 [=====] - 22s 302ms/step - loss: 0.0025 - mean\_absolute\_error: 0.0361

Epoch 42/50  
73/73 [=====] - 22s 302ms/step - loss: 0.0024 - mean\_absolute\_error: 0.0355

Epoch 43/50  
73/73 [=====] - 22s 301ms/step - loss: 0.0023 - mean\_absolute\_error: 0.0350

Epoch 44/50  
73/73 [=====] - 21s 291ms/step - loss: 0.0025 - mean\_absolute\_error: 0.0360

Epoch 45/50  
73/73 [=====] - 23s 309ms/step - loss: 0.0024 - mean\_absolute\_error: 0.0352

Epoch 46/50  
73/73 [=====] - 22s 302ms/step - loss: 0.0024 - mean\_absolute\_error: 0.0345

Epoch 47/50

## Saving the trained model

The screenshot shows a Jupyter Notebook interface with the title "Stock Prediction.ipynb". The notebook contains the following code:

```
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[26] Epoch 43/50
    73/73 [=====] - 22s 301ms/step - loss: 0.0023 - mean_absolute_error: 0.0350
Epoch 44/50
    73/73 [=====] - 21s 291ms/step - loss: 0.0025 - mean_absolute_error: 0.0360
Epoch 45/50
    73/73 [=====] - 23s 309ms/step - loss: 0.0024 - mean_absolute_error: 0.0352
Epoch 46/50
    73/73 [=====] - 22s 302ms/step - loss: 0.0024 - mean_absolute_error: 0.0345
Epoch 47/50
    73/73 [=====] - 22s 303ms/step - loss: 0.0022 - mean_absolute_error: 0.0346
Epoch 48/50
    73/73 [=====] - 22s 305ms/step - loss: 0.0022 - mean_absolute_error: 0.0335
Epoch 49/50
    73/73 [=====] - 22s 298ms/step - loss: 0.0025 - mean_absolute_error: 0.0364
Epoch 50/50
    73/73 [=====] - 22s 296ms/step - loss: 0.0022 - mean_absolute_error: 0.0347
<keras.src.callbacks.History at 0x7fb9c4778e0>

[27] model.save('keras_model.h5')
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This f
saving_api.save_model()

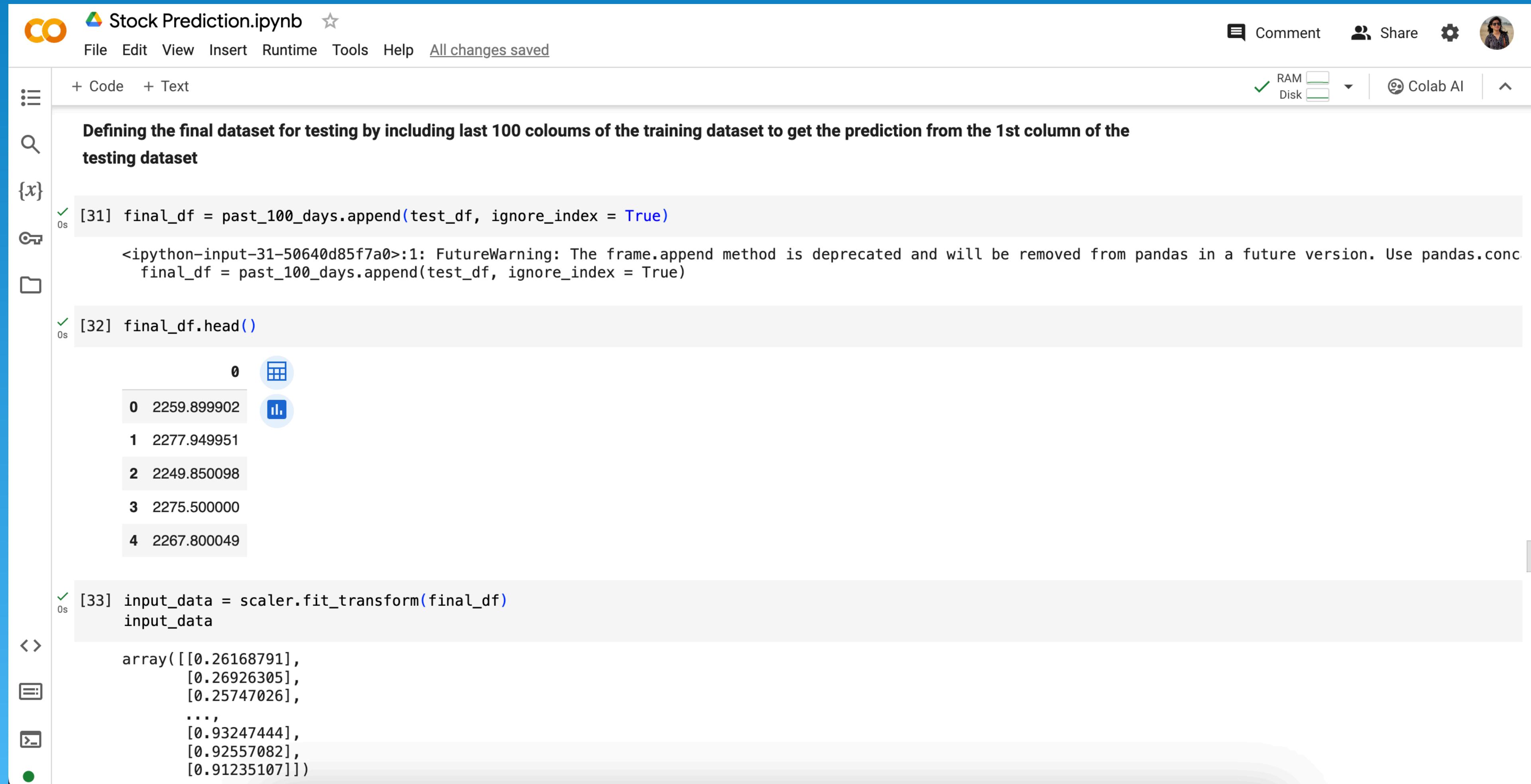
[28] test_close.shape
(1042, 1)

[29] past_100_days = pd.DataFrame(train_close[-100:])
[30] test_df = pd.DataFrame(test_close)

Dfining the final dataset for testing by including last 100 columns of the training dataset to get the prediction from the 1st column of the
```

The notebook also includes a sidebar with various icons for file operations, search, and sharing.

## Defining the final dataset by including last 100 columns of training dataset to get prediction



The screenshot shows a Google Colab notebook titled "Stock Prediction.ipynb". The notebook interface includes a toolbar with file operations, a sidebar with various icons, and a main workspace with code cells and output.

**Code Cells:**

- [31] `final_df = past_100_days.append(test_df, ignore_index = True)`  
A warning message follows: `<ipython-input-31-50640d85f7a0>:1: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat`  
`final_df = past_100_days.append(test_df, ignore_index = True)`
- [32] `final_df.head()`  
Output:

0
2259.899902
2277.949951
2249.850098
2275.500000
2267.800049
- [33] `input_data = scaler.fit_transform(final_df)`  
`input_data`  
Output:

```
array([[0.26168791],  
       [0.26926305],  
       [0.25747026],  
       ...,  
       [0.93247444],  
       [0.92557082],  
       [0.91235107]])
```

# Ploting graph

Stock Prediction.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ Colab AI ^

RAM Disk

+ Code + Text

[33] [0.93247444],  
[0.92557082],  
[0.91235107])

{x}

[34] input\_data.shape  
(1142, 1)

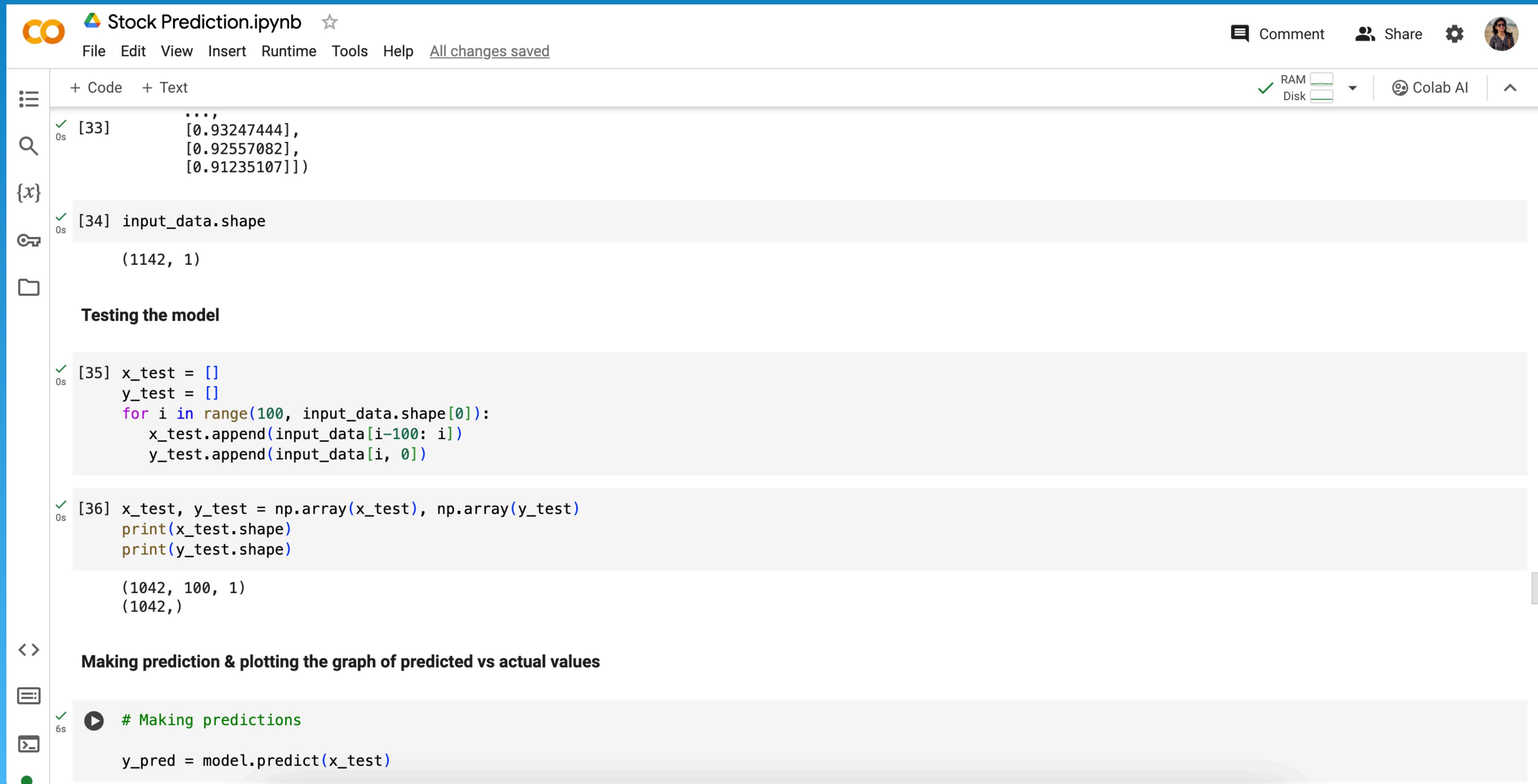
Testing the model

[35] x\_test = []  
y\_test = []  
for i in range(100, input\_data.shape[0]):  
 x\_test.append(input\_data[i-100: i])  
 y\_test.append(input\_data[i, 0])

[36] x\_test, y\_test = np.array(x\_test), np.array(y\_test)  
print(x\_test.shape)  
print(y\_test.shape)  
(1042, 100, 1)  
(1042,)

Making prediction & plotting the graph of predicted vs actual values

# Making predictions  
y\_pred = model.predict(x\_test)



# Making Prediction

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** Stock Prediction.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved
- Right Sidebar:** Comment, Share, Settings, User profile
- Resource Monitor:** RAM (green checkmark), Disk (green checkmark), Colab AI
- Code Cells:**
  - [37] # Making predictions  
y\_pred = model.predict(x\_test)  
33/33 [=====] - 3s 87ms/step
  - [38] y\_pred.shape  
(1042, 1)
  - [40] y\_test  
array([0.21665693, 0.19827518, 0.19817026, ..., 0.93247444, 0.92557082, 0.91235107])
  - [42] y\_pred  
array([[0.22601974],  
[0.2279132 ],  
[0.22942823],  
...,  
[0.8817482 ],  
[0.8862219 ],  
[0.8885009 ]], dtype=float32)
  - [43] scaler.scale\_  
array([0.00041967])

## Plotting graph of Predicted and Actual values

Stock Prediction.ipynb

File Edit View Insert Runtime Tools Help All changes saved

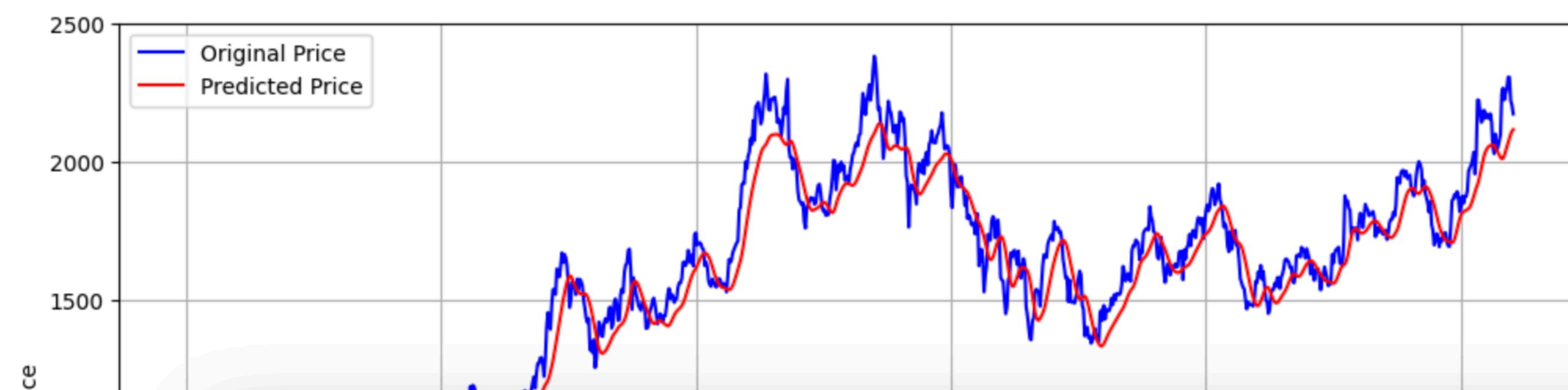
Comment Share

RAM Disk Colab AI

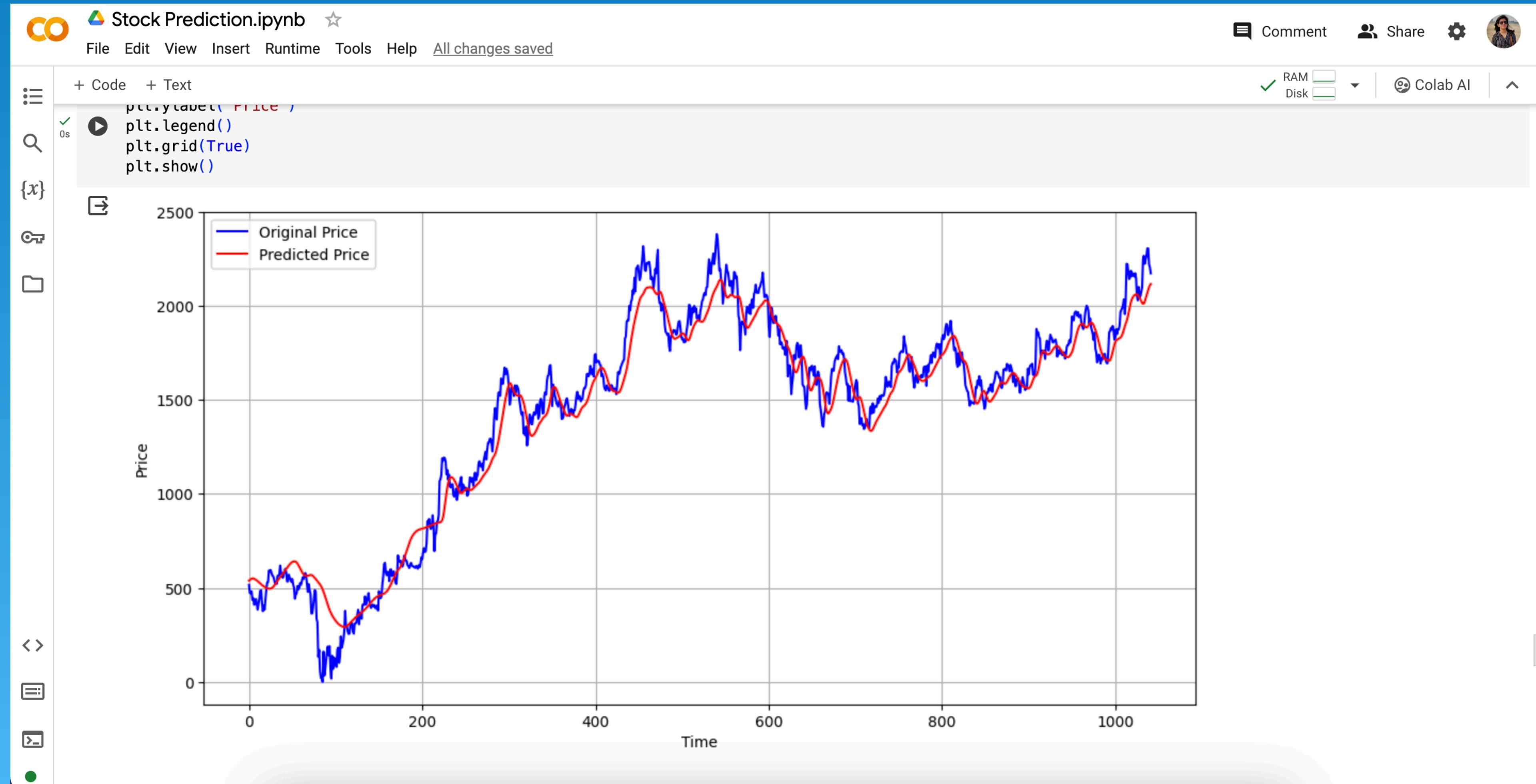
```
[43] scaler.scale_
0s {x} array([0.00041967])

[44] scale_factor = 1/0.00041967
y_pred = y_pred * scale_factor
y_test = y_test * scale_factor

[45] plt.figure(figsize = (12,6))
plt.plot(y_test, 'b', label = "Original Price")
plt.plot(y_pred, 'r', label = "Predicted Price")
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.show()
```



## Plotting graph of Predicted and Actual values



# Model Evaluation

The screenshot shows a Jupyter Notebook interface with the title "Stock Prediction.ipynb". The notebook has a sidebar on the left containing icons for file operations, search, and cell types. The main area displays two code cells for calculating model performance metrics.

**Model evaluation**

Calculation of mean absolute error

```
[46] from sklearn.metrics import mean_absolute_error  
  
mae = mean_absolute_error(y_test, y_pred)  
mae_percentage = (mae / np.mean(y_test)) * 100  
print("Mean absolute error on test set: {:.2f}%".format(mae_percentage))
```

Mean absolute error on test set: 6.08%

Calculation of R2 score

```
[47] from sklearn.metrics import r2_score  
  
# Actual values  
actual = y_test  
  
# Predicted values  
predicted = y_pred  
  
# Calculate the R2 score  
r2 = r2_score(actual, predicted)  
  
print("R2 score:", r2)
```

R2 score: 0.960947104280743

# Model Evaluation

Stock Prediction.ipynb ★

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙️ Colab AI ^

+ Code + Text RAM Disk

Plotting the R2 score

```
{x} 0s
▶ fig, ax = plt.subplots()
ax.barh(0, r2, color='skyblue')
ax.set_xlim([-1, 1])
ax.set_yticks([])
ax.set_xlabel('R2 Score')
ax.set_title('R2 Score')

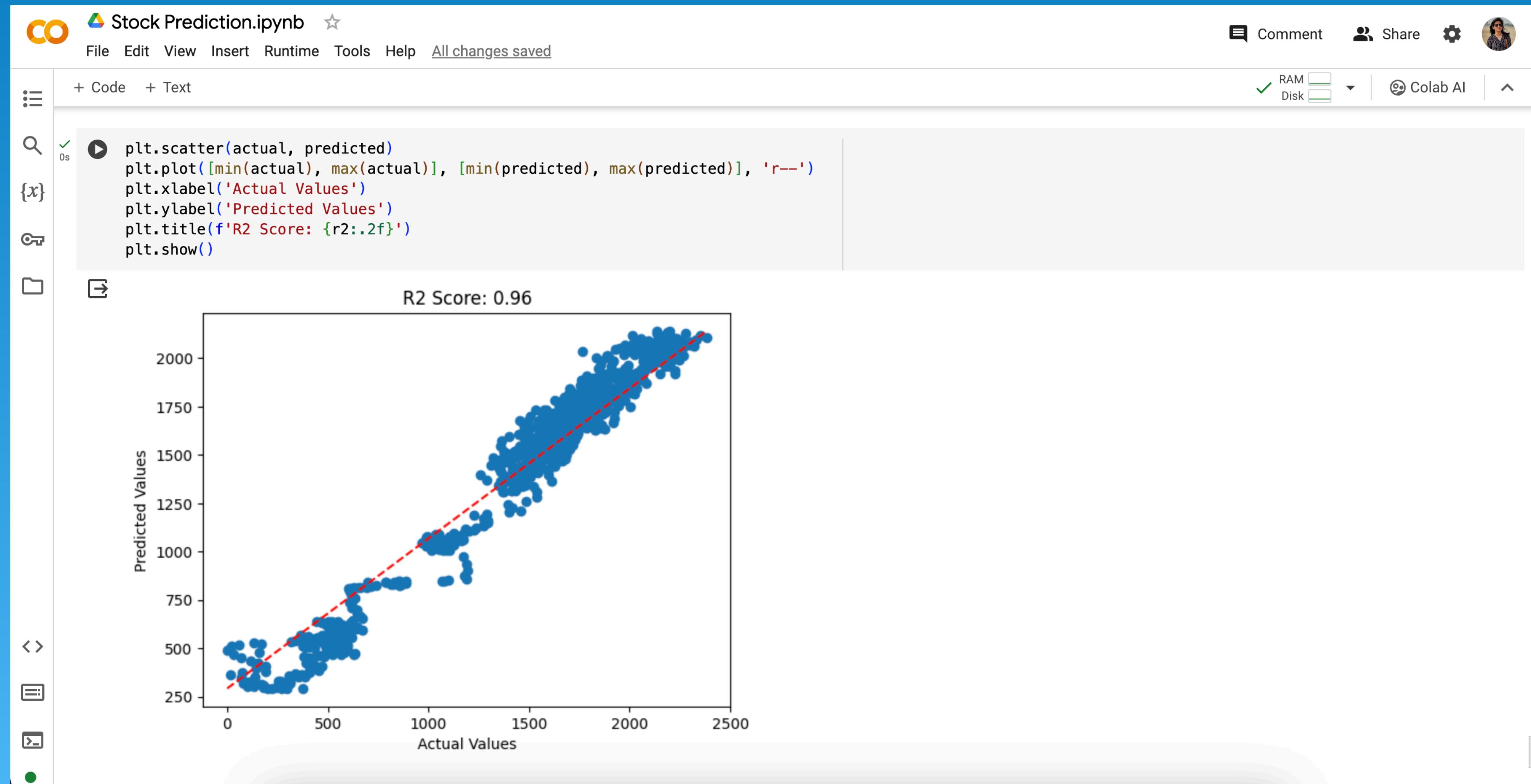
# Adding the R2 score value on the bar
ax.text(r2, 0, f'{r2:.2f}', va='center', color='black')

plt.show()
```

R2 Score

The figure is a horizontal bar chart titled "R2 Score". It has a single blue bar centered at the value 0.96 on the x-axis. The x-axis ranges from -1.00 to 1.00 with ticks at -1.00, -0.75, -0.50, -0.25, 0.00, 0.25, 0.50, 0.75, and 1.00. The y-axis has a single tick labeled "0.96". The chart is set against a white background with a black border.

# Plotting the R2 Graph



# Thank you