Stock price prediction

Report link: [Stock Price Prediction using Machine Learning with Source Code](https://www.projectpro.io/article/stock-price-prediction-using-machine-learning-project/571)

* Downloaded data of Netflix listed on ---, we have taken data from march 2019 to march 2022, This will also challenge our model to work with the unpredictable changes caused by the COVID-19 pandemic.

**🧠 Key Concepts Explained**

**1. Time Series Data**

* **Definition**: A sequence of data points collected or recorded at regular time intervals.
* **In this context**: Stock prices recorded daily over a period.

**2. Moving Averages**

* **Simple Moving Average (SMA)**: Calculates the average of a selected range of prices, usually closing prices, by the number of periods in that range.
* **Exponential Moving Average (EMA)**: Similar to SMA but gives more weight to recent prices, making it more responsive to new information.

**3. LSTM (Long Short-Term Memory)**

* **Definition**: A type of Recurrent Neural Network (RNN) capable of learning long-term dependencies.
* **Why use LSTM?**: It's effective for time series prediction because it can remember previous data points, which is crucial for predicting stock prices.[projectpro.io](https://www.projectpro.io/article/stock-price-prediction-using-machine-learning-project/571?utm_source=chatgpt.com)

**4. Evaluation Metrics**

* **RMSE (Root Mean Squared Error)**: Measures the average magnitude of the errors between predicted and actual values. Lower RMSE indicates better performance.
* **MAPE (Mean Absolute Percentage Error)**: Expresses accuracy as a percentage, showing how far off predictions are from actual values on average.[projectpro.io](https://www.projectpro.io/article/stock-price-prediction-using-machine-learning-project/571?utm_source=chatgpt.com)

**🛠️ Step-by-Step Implementation**

**Step 1: Import Necessary Libraries**

python

Copy code

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

* **pandas**: For data manipulation and analysis.
* **numpy**: For numerical operations.
* **matplotlib**: For data visualization.
* **MinMaxScaler**: Scales data to a specified range, often [0,1], which helps in faster convergence during training.
* **Sequential**: A linear stack of layers in Keras.
* **LSTM, Dense**: Layers used to build the neural network.[projectpro.io](https://www.projectpro.io/article/stock-price-prediction-using-machine-learning-project/571?utm_source=chatgpt.com)

**Step 2: Load and Preprocess the Data**

python

Copy code

# Load the dataset

df = pd.read\_csv('Netflix\_Dataset.csv')

# Sort by date

df = df.sort\_values('Date')

# Use 'Close' price for prediction

data = df['Close'].values.reshape(-1,1)

# Scale the data

scaler = MinMaxScaler(feature\_range=(0,1))

scaled\_data = scaler.fit\_transform(data)

* **reshape(-1,1)**: Reshapes the data into a 2D array, which is required for the scaler.
* **fit\_transform**: Fits the scaler to the data and then transforms it.

**Step 3: Create Training and Testing Datasets**

python

Copy code

# Define training data length

training\_data\_len = int(np.ceil( len(scaled\_data) \* .8 ))

# Create the training data

train\_data = scaled\_data[0:int(training\_data\_len), :]

# Split the data into x\_train and y\_train datasets

x\_train = []

y\_train = []

for i in range(60, len(train\_data)):

x\_train.append(train\_data[i-60:i, 0])

y\_train.append(train\_data[i, 0])

# Convert to numpy arrays

x\_train, y\_train = np.array(x\_train), np.array(y\_train)

# Reshape the data to be 3D for LSTM [samples, time\_steps, features]

x\_train = np.reshape(x\_train, (x\_train.shape[0], x\_train.shape[1], 1))

* **60**: Number of previous time steps to use for predicting the next value.
* **Reshaping**: LSTM networks expect input in the form of [samples, time\_steps, features].

**Step 4: Build the LSTM Model**

python

Copy code

# Build the LSTM model

model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape= (x\_train.shape[1], 1)))

model.add(LSTM(units=50, return\_sequences=False))

model.add(Dense(units=25))

model.add(Dense(units=1))

# Compile the model

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Train the model

model.fit(x\_train, y\_train, batch\_size=1, epochs=1)

* **units=50**: Number of neurons in the layer.
* **return\_sequences**: Determines whether to return the last output in the output sequence or the full sequence.
* **Dense layer**: A regular fully-connected neural network layer.
* **optimizer='adam'**: An optimization algorithm that adjusts the weights to minimize the loss.
* **loss='mean\_squared\_error'**: The function to minimize during training.[projectpro.io+3arxiv.org+3arxiv.org+3](https://arxiv.org/abs/2009.10819?utm_source=chatgpt.com)[arxiv.org](https://arxiv.org/abs/2202.05702?utm_source=chatgpt.com)

**Step 5: Create the Testing Dataset**

python

Copy code

# Create the testing data

test\_data = scaled\_data[training\_data\_len - 60: , :]

# Create the x\_test and y\_test datasets

x\_test = []

y\_test = data[training\_data\_len:, :]

for i in range(60, len(test\_data)):

x\_test.append(test\_data[i-60:i, 0])

# Convert to a numpy array

x\_test = np.array(x\_test)

# Reshape the data to be 3D

x\_test = np.reshape(x\_test, (x\_test.shape[0], x\_test.shape[1], 1))

# Get the model's predicted price values

predictions = model.predict(x\_test)

predictions = scaler.inverse\_transform(predictions)

* **inverse\_transform**: Converts the scaled data back to the original scale.

**Step 6: Evaluate the Model**

python

Copy code

# Calculate RMSE

rmse = np.sqrt(np.mean(((predictions - y\_test) \*\* 2)))

print(f'RMSE: {rmse}')

* **RMSE**: Provides a measure of how accurately the model predicts the response.[en.wikipedia.org](https://en.wikipedia.org/wiki/Stock_market_prediction?utm_source=chatgpt.com)

**Step 7: Visualize the Data**

python

Copy code

# Plot the data

train = df[:training\_data\_len]

valid = df[training\_data\_len:]

valid['Predictions'] = predictions

# Visualize the data

plt.figure(figsize=(16,8))

plt.title('Model')

plt.xlabel('Date')

plt.ylabel('Close Price USD ($)')

plt.plot(train['Close'])

plt.plot(valid[['Close', 'Predictions']])

plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')

plt.show()

* **Visualization**: Helps in understanding how well the model is performing by comparing actual vs. predicted prices.[projectpro.io](https://www.projectpro.io/article/stock-price-prediction-using-machine-learning-project/571?utm_source=chatgpt.com)

**🎤 How to Explain This Project in an Interview**

**1. Introduction**

*"In this project, I developed a model to predict stock prices using historical data. The goal was to forecast future prices to aid in investment decisions."*

**2. Data Collection and Preprocessing**

*"I used historical stock data, focusing on the 'Close' price. The data was scaled to a range between 0 and 1 to improve the model's performance."*

**3. Model Selection**

*"I chose an LSTM neural network because it's well-suited for time series data, allowing the model to learn from sequences of past data points."*

**4. Training and Testing**

*"The dataset was split into training and testing sets. The model was trained on the training set and evaluated on the testing set using RMSE as the performance metric."*

**5. Results**

*"The model achieved an RMSE of [insert RMSE value], indicating [interpretation of RMSE]. The predictions closely followed the actual stock prices, as visualized in the plotted graph."*

**6. Challenges and Improvements**

*"One challenge was ensuring the model didn't overfit the training data. To address this, I could implement techniques like dropout layers or use more data for training. Additionally, incorporating other features like trading volume or technical indicators might improve accuracy."*

**🛠️**

[**https://chatgpt.com/share/68402ff5-64cc-8009-b73d-42f66bbdacff**](https://chatgpt.com/share/68402ff5-64cc-8009-b73d-42f66bbdacff)