In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

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

In [ ]:

```
data = pd.read_csv("D:\\Workshops\\W 17 - AI Masterclass Workshop\\data\\Mastercard_stock.CSV", index_col="Date", parse_dates=["Date"])
```

In [ ]:

```
data.head()
```

In [ ]:

```
data.drop(["Dividends", "Stock Splits"], axis=1,inplace=True)
```

In [ ]:

```
data.head()
```

In [ ]:

```
tstart = 2016
tend = 2020
data.loc[f"{tstart}":f"{tend}", "High"].plot(figsize=(16, 4), legend=True)
data.loc[f"{tend+1}":, "High"].plot(figsize=(16, 4), legend=True)
plt.legend([f"Train (Before {tend+1})", f"Test ({tend+1} and beyond)"])
plt.title("MasterCard stock price")
plt.show()
```

Split the data to train and test

In [ ]:

```
train=data.loc[f"{tstart}":f"{tend}", "High"].values
test=data.loc[f"{tend+1}":, "High"].values
```

Scaling data

In [ ]:

```
sc = MinMaxScaler(feature_range=(0, 1))
training_set = train.reshape(-1, 1)
training_set_scaled = sc.fit_transform(training_set)
```

In [ ]:

```
training_set_scaled
```

Getting training x and training y

In [ ]:

```
n_steps = 60
features = 1

x=[]
y=[]
for i in range(len(training_set_scaled)):
    end_ix = i + n_steps
    if end_ix > len(training_set_scaled) - 1:
        break
    seq_x, seq_y = training_set_scaled[i:end_ix], training_set_scaled[end_ix]
    x.append(seq_x)
    y.append(seq_y)

x_train=np.array(x)
y_train=np.array(y)
```

In [ ]:

```
x_train = x_train.reshape(x_train.shape[0],x_train.shape[1],features)
```

In [ ]:

```
x_train
```

In [ ]:

```
y_train
```

Building the LSTM model

In [ ]:

```
model_lstm = Sequential()
model_lstm.add(LSTM(units=125, activation="tanh", input_shape=(n_steps, featur
es)))
model_lstm.add(Dense(units=1))
# Compiling the model
model_lstm.compile(optimizer="RMSprop", loss="mse")
model_lstm.summary()
```

Training the model

In [ ]:

```
model_lstm.fit(x_train, y_train, epochs=50)
```

Getting testing x and testing y

In [ ]:

```
dataset_total = data.loc[:,"High"]
inputs = dataset_total[len(dataset_total) - len(test) - n_steps :].values
inputs = inputs.reshape(-1, 1)
#scaling
inputs = sc.transform(inputs)


n_steps = 60
features = 1

x=[]
y=[]
for i in range(len(inputs)):
    end_ix = i + n_steps
    if end_ix > len(inputs) - 1:
        break
    seq_x, seq_y = inputs[i:end_ix], inputs[end_ix]
    x.append(seq_x)
    y.append(seq_y)

x_test=np.array(x)
y_test=np.array(y)
```

Get the predictions

In [ ]:

```
# reshape
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], features)
#prediction
predicted_stock_price = model_lstm.predict(x_test)
#inverse transform the values
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

Prediction accuracy

In [ ]:

```
plt.plot(data.loc[f"{tend+1}":, "High"].values, color="gray", label="Real")
plt.plot(predicted_stock_price, color="red", label="Predicted")
plt.title("MasterCard Stock Price Prediction")
plt.xlabel("Time")
plt.ylabel("MasterCard Stock Price")
plt.legend()
plt.show()
```

In [ ]:

```
np.sqrt(mean_squared_error(data.loc[f"{tend+1}":, "High"].values, predicted_stock_price))
```

In [ ]: