Stock Price Prediction Analysis

I) Feature Engineering

1 .**Data Scaling:**

Normalize the data using Min-Max scaling.

```
# Normalize the df
scaler = MinMaxScaler()
data = scaler.fit_transform(df)
```

2.**Create Input Sequences:**

Create input sequences for the LSTM model.

```
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i + seq_length])
        y.append(data[i + seq_length])
    return np.array(X), np.array(y)

sequence_length = 10  # You can adjust this hyperparameter
X, y = create_sequences(df, sequence_length)
```

II) Model Creation

LSTM Model Creation:

Build the LSTM model using TensorFlow's Keras API.

```
#Creating a LSTM model for prediction
model = Sequential()
model.add(LSTM(units = 50, return_sequences = True, input_shape = (x_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(units = 50))
model.add(Dropout(0.2))
model.add(Dense(units = 1))

#Printing a overview of the model
model.summary()

#Compiling and Fitting the model
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
model.fit(x_train, y_train, epochs = 50, batch_size = 32)
```

Which would generate the following summary of the model created:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_8 (LSTM)               (None, 60, 50)            10400

 dropout_8 (Dropout)         (None, 60, 50)            0

 lstm_9 (LSTM)               (None, 60, 50)            20200

 dropout_9 (Dropout)         (None, 60, 50)            0

 lstm_10 (LSTM)              (None, 60, 50)            20200

 dropout_10 (Dropout)        (None, 60, 50)            0

 lstm_11 (LSTM)              (None, 50)                20200

 dropout_11 (Dropout)        (None, 50)                0

 dense_2 (Dense)             (None, 1)                 51

=================================================================
Total params: 71051 (277.54 KB)
Trainable params: 71051 (277.54 KB)
Non-trainable params: 0 (0.00 Byte)
```

Following which the training process undergoes as follows:

```
Epoch 40/50
24/24 [==============================] - 3s 114ms/step - loss: 0.0041
Epoch 41/50
24/24 [==============================] - 3s 114ms/step - loss: 0.0042
Epoch 42/50
24/24 [==============================] - 4s 168ms/step - loss: 0.0042
Epoch 43/50
24/24 [==============================] - 3s 114ms/step - loss: 0.0037
Epoch 44/50
24/24 [==============================] - 3s 114ms/step - loss: 0.0036
Epoch 45/50
24/24 [==============================] - 3s 114ms/step - loss: 0.0042
Epoch 46/50
24/24 [==============================] - 4s 155ms/step - loss: 0.0042
Epoch 47/50
24/24 [==============================] - 3s 125ms/step - loss: 0.0041
Epoch 48/50
24/24 [==============================] - 3s 114ms/step - loss: 0.0037
Epoch 49/50
24/24 [==============================] - 3s 115ms/step - loss: 0.0034
Epoch 50/50
24/24 [==============================] - 3s 132ms/step - loss: 0.0032
<keras.src.callbacks.History at 0x798dce6e2980>
```

III) Analysis

Analysis of Predicted Prices:

Inverse transform the predicted prices to their original scale.

```
y_train_actual = scaler.inverse_transform(data_train)
y_train_pred = scaler.inverse_transform(y_train_pred)
y_test_actual = scaler.inverse_transform(data_test)
y_test_pred = scaler.inverse_transform(y_test_pred)
```

Display the number of predicted prices.

```
# Display the number of predicted prices
print("Number of Predicted Prices:", len(predicted_price))
```

Which outputs as follows:

```
Number of Predicted Prices: 710
```

Calculate performance metrics:

Performing RMSE as actual data is available.

```
train_rmse = np.sqrt(mean_squared_error(data_train, y_train_pred))
test_rmse = np.sqrt(mean_squared_error(data_test, y_test_pred))

print("Train RMSE:", train_rmse)
print("Test RMSE:", test_rmse)
```

Which outputs as follows:

```
Train RMSE: 16.815131088550885
Test RMSE: 67.27047773743399
```

Conclusion

This analysis presented a workflow for stock price prediction using an LSTM model. Key steps included data preprocessing, model creation, evaluation, and analysis. Accurate stock price prediction is essential for investment decisions. While the model's results can be valuable, they should be used cautiously, and continuous refinement is crucial in adapting to dynamic markets.

This concise conclusion summarizes the document's main points and the importance of exercising caution when using stock price prediction models.