PSYLIQ

DATA SCIENCE (PYTHON) ASSESSMENT QUESTIONS

Note: These assessment questions cater to interns at various skill levels, from beginners to experienced analysts. If you find any questions challenging, feel free to search for solutions or contact us at intern@psyliq.com for assistance. Good luck with the assessment!

1) Stock Prediction:

Question: How can I predict the stock price of a company using LSTM in a Jupyter notebook?

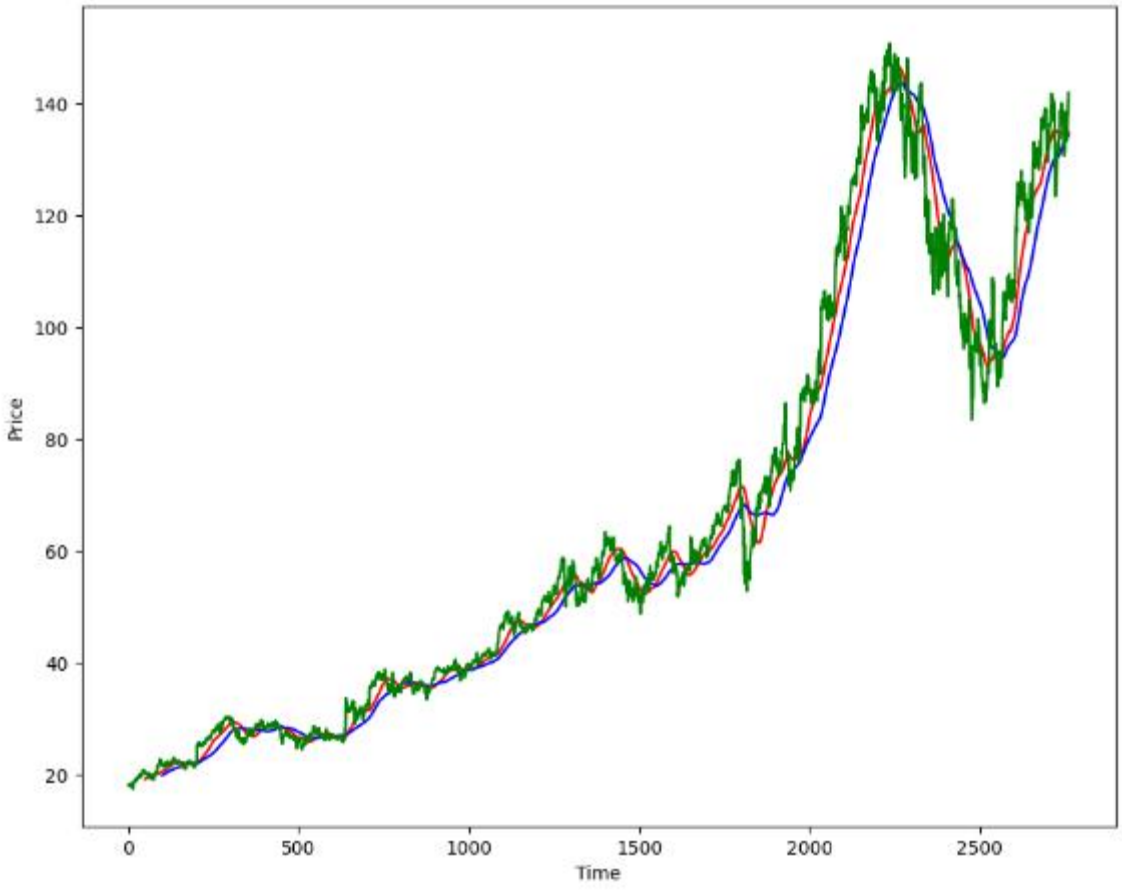
Answer:

1. **Data Loading:** The first part of the code imports the necessary libraries and downloads the stock data using the Yahoo Finance API. The stock data is stored in a pandas DataFrame.
2. **Moving Average:** The next section calculates the moving average (MA) of the stock data over different periods, such as 50 days, 100 days and 200 days. These moving averages are plotted using the matplotlib library to visualize the trends in the stock price.
3. **Preprocessing:** The data is then preprocessed by removing any rows with missing values (NaN) using the dropna() function. The stock data is then split into a training set and a test set using the split() function from the pandas library. The data is scaled to a range of [0,1] using the MinMaxScaler from the sklearn.preprocessing module.
4. **Model Building:** The model is built using the Keras library in Python. The model architecture consists of a stack of 4 LSTM layers with dropout layers between them to prevent overfitting. The LSTM layers have varying numbers of units (50, 60, 80, and 120), which determines the number of features each layer can learn. The input to the model is of shape (x.shape[1],1)), where x.shape[1] is the number of timesteps used for prediction.
5. **Model Training:** The model is trained on the training data using the fit() function. The data is first converted into a 3D array (samples, timesteps, features) to match the expected input shape for the LSTM model.
6. **Model Testing:** The model's performance is evaluated on the test data. The predicted stock prices are obtained by feeding the scaled test data into the model. The predicted prices are then rescaled to their original range and compared to the actual stock prices to evaluate the model's performance.
7. **Model Summary and Visualization:** Finally, the model's architecture is displayed using the summary() function. The actual stock prices, predicted stock prices, and moving averages are plotted using the matplotlib library to visualize the trends in the stock price.

Google Stock Price Prediction using data from 2013-01-01 to 2023-12-32

Price Vs MA(50):

Price Vs MA(50) Vs MA(100):



Price Vs MA(100) VS MA(200):

Original Price Vs Predicted Price:

2) Titanic Classification:

Question: How can I create a system to predict if a person will survive the Titanic sinking and identify key factors influencing survival like socio-economic status, age, and gender?

Answer:

1. First, we import the necessary libraries.

2. Next, we load the dataset from the specified file path using pandas.read_csv() function.

3. We check the basic details of the dataset such as its shape, data types, and null values using the functions head(), shape, info(), and isnull().sum().

4. We handle the missing values in the 'Cabin' and 'Age' columns by dropping the 'Cabin' column and filling the missing values in the 'Age' column with the mean value of the column. We also handle the missing values in the 'Embarked' column by filling them with the most common value in the column.

5. After handling the missing values, we check for null values again using isnull().sum() and ensure that all the missing values have been taken care of.

6. We describe the dataset using the describe() function to obtain summary statistics.

7. We visualize the data using seaborn's countplot function. This function provides a quick overview of the data by displaying the count of occurrences for each category.

8. We preprocess the data by replacing the categorical values of the 'Sex' and 'Embarked' columns with numerical values using the replace() function.

9. We then separate the features and the target variable from the dataset using the drop() and the get_item() functions.

10. We split the data into training and testing sets using the train_test_split() function from sklearn.model_selection.

11. We create a logistic regression model using the LogisticRegression() function from sklearn.linear_model.

12. We train the model on the training data using the fit() function.

13. We predict the target variable for both the training and testing data.

14. We calculate the accuracy of the model's predictions on the training and testing data.

3) Number Recognition:

Question: How can I create a Handwritten Digit Recognition system using the MNIST dataset with a Neural Network in a Jupyter notebook?

Answer:

1. First, we import the necessary libraries, which are TensorFlow, Keras, NumPy, Matplotlib, and Seaborn.

2. Next, we load the MNIST dataset using the load_data() function from the Keras library. This dataset consists of 70,000 images of handwritten digits (0-9) in a 28x28 pixel format. The images are pre-processed and divided into training and test sets.

3. We then normalize the pixel values of the images to the range [0, 1] by dividing each pixel value by 255.

4. After normalizing the pixel values, we reshape the 28x28 pixel images into 784-dimensional

vectors (flattened images) using the reshape() function. This is done to prepare the images for feeding into the neural network model.

5. We define the structure of our neural network model using the Keras Sequential class. This model consists of a dense layer with 100 neurons and a ReLU activation function. The output layer also has 10 neurons and a sigmoid activation function.

6. We compile the model using the Adam optimizer, sparse categorical cross-entropy loss function, and accuracy as the evaluation metric.

7. We train the model using the fit() function with the training data (flattened images) and their corresponding labels (actual digit values). The model is trained for 50 epochs.

8. We evaluate the model's performance on the test data using the evaluate() function.

9. To visualize the prediction of the model, we display a sample test image and its predicted digit.

10. We calculate the confusion matrix using the true labels from the test set and the predicted labels from the model. This matrix provides a visual representation of the model's performance by showing the number of correct and incorrect predictions for each digit.

11. Finally, we plot the confusion matrix using a heatmap, which highlights the true positives, false positives, true negatives, and false negatives for each digit.