IE 555 – Programming for Analytics

Homework #8 – OR Applications – Forecasting

Due Date: To Be Announced

Assignment Details

You are asked to provide a single Python script (`.py` file) containing four (4) Python functions, as described below. It is critical that you follow these instructions **exactly**.

1. `importQuotes`. This function will accept three (3) inputs, in the following order:

   (a) `a` – A **string** describing the stock ticker symbol (e.g., "INTC"). Note that the user might provide this as uppercase, lowercase, or a mix; your code should work regardless of case.
   (b) `b` – A **string** describing the start date, of the form "YYYY-MM-DD". You may assume that the user will provide the input in the correct form.
   (c) `c` – A **string** describing the end date, of the form "YYYY-MM-DD". You may assume that the user will provide the input in the correct form. You may also assume that the user is smart enough to provide you with a start date that precedes the end date.

   The `importQuotes(a, b, c)` function should return the stock's **closing** prices between the start and end dates, as a numpy object. These closing prices should come from Quandl's `quandl.get` function, and should use your API key (hard-coded within your function). See `stock_prices_quandl.py` for an example, but be aware that the `quandl.get` function returns more than simply the closing prices.

2. `forecastMA`. This function will accept three (3) inputs:

   - `a` – A **positive integer** indicating the desired number of periods to use for calculating a moving average forecast.
   - `b` – A **numpy object** containing the closing prices for a given stock. This will be of the same form as the output from the `importQuotes` function.
   - `c` – A **positive integer** indicating the number of periods into the future for which a forecast should be provided. A value of 1 would indicate the next period.

   The `forecastMA(a, b, c)` function should return a scalar floating point value, rounded to 2 decimal places, representing the moving average forecast corresponding to the input parameters.

3. `forecastLR`. This function will accept two (2) inputs:

   - `a` – A **positive integer** indicating the number of periods into the future for which a forecast should be provided. A value of 1 would indicate the next period.
   - `b` – A **numpy object** containing the closing prices for a given stock. This will be of the same form as the output from the `importQuotes` function.

The `forecastLR(a, b)` function should return a list, with the following information related to generating a forecast based on linear regression:

- The forecasted closing price of the stock, `a` periods in the future, using linear regression. This should be a floating-point value, rounded to 2 decimal places.
- The slope of the linear regression line.
- The y-intercept of the linear regression line.

NOTE: The three elements in the output list must be in the order described above.

4. `forecastHolt`. This function will accept four (4) inputs:

   - `a` – A **positive integer** indicating the number of periods into the future for which a forecast should be provided. A value of 1 would indicate the next period.
   - `b` – A **floating-point value in the open interval (0,1)** describing the α parameter used by Holt's method.
   - `c` – A **floating-point value in the open interval (0,1)** describing the β parameter used by Holt's method.
   - `d` – A **numpy object** containing the closing prices for a given stock. This will be of the same form as the output from the `importQuotes` function.

   The `forecastHolt(a, b, c, d)` function should return a scalar floating point value, rounded to 2 decimal places, representing the Holt's method forecast corresponding to the input parameters. **NOTE:** Use linear regression to find estimates of S_0 and G_0 using all of the closing prices provided via input parameter `d`.

These functions should be saved within a file named `UPPERCASEUBUSERID_forecasting.py`, where `UPPERCASEUBUSERID` should be replaced with your UB username in ALL CAPS.

Example Function Inputs/Outputs

As an example, consider a student whose UB username is "xyz123". We would execute that student's functions via an IPython terminal as follows:

```
In  [1]:  import XYZ123_forecasting as forecast

In  [2]:  closing_prices = forecast.importQuotes('INTC', '
   2020-01-01', '2020-01-31')

In  [3]:  ma_forecast = forecast.forecastMA(5, closing_prices,
    3)
In  [4]:  print(ma_forecast)
Out [4]:  27.50    # I'm making this number up.

In  [5]:  [lr_forecast, slope, intercept] = forecast.
   forecastLR(3, closing_prices)
In  [6]:  print(lr_forecast)
```

```
Out [6]:  23.29    # I'm making this number up.
In  [7]:  print(slope)
Out [7]:  -0.802091    # I'm making this number up.
In  [8]:  print(intercept)
Out [8]:  29.348272    # I'm making this number up.

In  [9]:  holt_forecast = forecast.forecastHolt(2, 0.2, 0.6,
    closing_prices)
In  [9]:  print(holt_forecast)
Out [9]:  28.98    # I'm making this number up.
```

Notes

- You should experiment with using different inputs to your functions to verify that your code works properly.

Grading

- If you submit properly-working code by the due date, you will have earned 100 points (the maximum score) on this assignment.
- The TA will notify you if your submission has any errors. In such a case, you will need to re-submit your assignment.
  - Each re-submission of your assignment will result in a 10-point deduction.

Submitting Your Assignment

A private GitHub repository will be created for you. Upload your code to the repository and then send an email to the TA when you are finished.