Columbia University in the City of New York
MATHGR5430 Machine Learning for Finance, Spring 2022

Homework 1

Prof. Silva

Due by 02/07/2022

Please submit a single .pdf-file generated from your Jupyter-Notebook on CourseWorks.

1 Setup and Data Retrieval

1. Install Anaconda `https://docs.anaconda.com/anaconda/install/index.html`.
2. Obtain the data of an MSCI World ETF (”XWD.TO”) as a Pandas DataFrame using the yfinance package starting from 01/01/2010 with daily frequency.

2 Data Analysis and Exploration in Pandas

1. Plot the Adj Close column of the MSCI ETF
2. Gather information about the data frame by looking at key statistics, i.e. the moments (mean and variance), data types, statistics...
3. Get all rows where the ”High” value is > 40
4. Drop the volume column
5. Get the adjusted close price of the 01/17/2022
6. Create a new column that stores the daily return
7. How are the returns distributed?
8. Calculate the cumulative return (Hint: cumsum()) and add it to the dataframe
9. Add a 10-day exponential moving-average ”EMA_10” column to the MSCI data frame based on the ”Adj Close” column (Hint: you can use pandas_ta using pip install pandas_ta)

3 Linear Regression usign Scikit-Learn

In this part we will build a linear regression model, that can infer the value of the MSCI ETF using the EMA.

3.1 Handle Missing Values

Adding the EMA has created missing values for the first n indexes, where n is your window size (9, in our case).

3.1.1 Find and print out the number of missing values grouped by column.

3.1.2 Replace missing values

Often times, if statistically reasonable (esp. in time series), we can replace missing values with their mean or mode. This can especially be done if the missing values make up only a little percentage of the data set. Another approach would be to drop the missing entries, which will however often times decrease our performance. Try filling the missing rows of the explanatory variable with "reasonable" values (Hint: Look at the distribution of the beginning of the EMA_10 time series). Make sure there are no missing values left.

3.2 Write the function split_data

The function split_data takes explanatory and response time series as parameter to and splits it into X_train, X_test, y_train and y_test. In the context of linear regression, y denotes the response variable (=label) and X (=features) the explanatory variable. Other than the series, it takes the size of your train split as parameter.
Also, be sure to add docstring documentation as learnt in the Python Refresher. Furthermore print out useful information in the function, like the shape of the initial time series as well as the subsets created.

For example: the first 80% of the time series as train, the last 20% as test.

3.3 Run a linear regression model on the EMA and the Adj Close

1. Split the data into train and test using your custom function using 20% test size
2. Train the model using scikit-learn
3. Retrieve and present statistical measures and key numbers i.e. r^2, intercept and coefficient

3.4 Visualize the results in one plot using Matplotlib.Pyplot

Create a combined plot that shows

1. the actual output
2. the predicted output
3. the training data

Make sure to add a legend and labels to each line and that every series is at the "correct" x coordinates.

4 Optional: Basic Data Structures in Python

1. Create an empty dictionary
2. Update the new dictionary with any key values
3. List comprehension: iterate through the following dictionaries values and return all values as string

   ```
   new_dict = {0: 1, 1: 'a', 2: 0.34, 3: True}
   ```

4. Convert the two lists to sets and then return the union of the two

   ```
   L1 = ["A","B","C","D"]
   L2 = ["B","D","E","F"]
   ```

5. Iterate through the first list and check if it is in the second by element-wise comparison