Mabrains Coding Test

We have the AAPL stock prices snapshots taken for the last 5 years. I would like you to do the following set of tasks. Please provide the code for all the steps. All coding should be done in python in a Jupyter notebook. In this coding test, you are basically required to build a couple of models that would help stock traders do trades every day. You will be required to build 2 models: “same_day_strategy” and “next_day_strategy”. For "same_day_strategy", we run prediction at 9:30PM in morning of every trading day to predict if the close price will be higher than the open price of that same day or not. Meaning, we know today's open price at that point. Bare in mind, we don't know anything about today's price fluctuations, news or anything. We buy the stock in the morning and we sell again at the end of the trading day. We only do a single day prediction at a time. Meaning, that we do know the past history of the stock perfectly at that point. As for "next_day_strategy", we run prediction at 3:30PM of every trading day, before the trading day ends to predict if the close price of tomorrow will be higher than the close price of today. Meaning that we don’t know the close price of today. Please keep that in mind when building your model.

Percentage of change is defined as: (last_price - start_price) / start_price.

Task 1: Create github account if you doing have one and create a repo for this project.

1. If you don’t have a GitHub account, please create one.
2. Create a new repo and call it whatever you want.
3. Create a readme.md and write some documentation about the project.
4. Commit all those changes and create the following folders underneath the project: input, scripts, notebooks, output
5. Create a small readme.md in each folder to be able to commit keep this folder structure.

Task 2: Analysis of AAPL stock.

1. Download the AAPL.csv file.

2. Create a new Jupyter notebook and read “AAPL.csv”. Make sure to parse the “date” column correctly.

3. Create a new column that has the date name like as: “mon”,”tue”,”wed”, etc... Name that column: “day_of_week”.

4. Create a new column that has the percentage of change between the “Open” and “Close” on the same day with the name “same_day_delta”. Mathematically is defined as: (“Today's Close Price” - “Today’s Open Price”) / “Today Open Price”.

5. Create a new column that is either: 0 or 1. The value is set to 0 when “same_day_delta” is less than or equal to 0.5, and it’s set to 1 when it’s larger than that value. Name that column “same_day_strategy”.

6. Create a new column that has the percentage of change between “Close” and last trading day “Close”. Name that column “next_close_delta”. Mathematically is defined as: (“Today’s Close Price” - “Yesterday’s Open Price”) / “Yesterday’s Open Price”.

7. Create a new column that is either: 0 or 1. The value is set to 0 when “next_close_delta” is less than or equal to 0.5, and it’s set to 1 when it’s larger than that value. Name that column “next_close_strategy”.

8. Find the following information per month and save it to a file: “monthly_analysis.csv”:

    1. Average close price per month.
    2. Average open price per month.
    3. Highest close price per month.
    4. Lowest open price per month.
    5. Highest high and low price month.
    6. Lowest high and low price month.

9. In the notebook, plot “open”, “close” price in the same plot.
10. Plot the distribution of “same_day_delta”.
11. (Optional Extra points: Plot candle chart for the stock.)

Task 3: Machine learning. Accuracy/Performance of those models is not important.

1. Split the data as follows:
    - Training dataset: Start date of the dataset till 1st of November, 2018
    - Validation dataset: 1st of November, 2018 till 1st of April, 2018
    - Testing dataset: 1st of April, 2018 till the end of the dataset.

2. Create a model that predicts:“same_day_strategy” use whatever features you think gets you the best results. Please note that we run this prediction as discussed at the beginning of document when the market opens, we don’t know anything about the close price of that same day. You can’t use anything that is related to the close price of that same day. Please make sure to document everything you do and how you built the model and your choices in a separate markdown page “same_day_model_documentation.md”. Document the accuracy score, precision and F1 score. Also document the criteria used in creating this model. (Optional [Extra Points]: Plot the confusion matrix for that classifier)

3. Create a model that predicts:“next_close_strategy” use whatever features you think gets you the best results. Please note that we run this model as discussed at the end of the trading day before the trading ends. We don’t know anything about tomorrow’s information like: “open”, “close”, etc… and we don’t know the “close”, “low”, “high” of today. Please make sure to document everything you do and how you built the model and your choices in a separate markdown page “next_day_model_documentation.md”. Document the accuracy score, precision and F1 score. Also document the criteria used in creating this model. (Optional [Extra Points]: Plot the confusion matrix for that classifier)

4. Save those models to disk and submit them to repo.

Task 4: Optimize the inference time of the model. (Please don’t waste a lot of time on this, Just try your best)

1. Measure the execution time for processing the data and the inference time of the models.

2. Try to optimize the time above.

Task 5 (Optional [Extra Points] Challenge Task):

1. Collect extra text data like historical tweets about Apple, news, google trends, tech industry news.

2. Use historical text data to create a document classifier that could be used to predict both: “same_day_strategy” and “next_day_strategy”. Please note, that you can’t see the tweets or text of the future, meaning for example, if you are working on May 1st, 2017, then you could see all the tweets, news, trends, etc.. April 28th, 2017 moving backwards.

3. Ensemble the new text models with the models created in Task 3 and create a better model that has better classification performance.