CSE 519 -- Data Science (Fall 2023)
Prof. Steven Skiena
Homework 3: Data Integration and Modeling
Due: Thursday, October 19, 2023

This homework will investigate data integration and model building in IPython. It is based on the Optiver Trading at the Close Kaggle challenge, revolving around predicting how stock prices will change at the closing auction in the last ten minutes of the Nasdaq trading day.

Files

The training data contains records with the following fields:

- `row_id` - A unique string f"{date_id}_{time_in_bucket}_{row_id}"
- `time_id` - A sequential indicator for the time buckets.
- `stock_id` - A unique identifier for the stock. Not all stock IDs exist in every time bucket.
- `date_id` - A unique identifier for the date. Date IDs are sequential & consistent across all stocks.
- `imbalance_size` - The amount unmatched at the current reference price (in USD).
- `imbalance_buy_sell_flag` - An indicator reflecting the direction of auction imbalance.
    - buy-side imbalance; 1
    - sell-side imbalance; -1
    - no imbalance; 0
- `reference_price` - The price at which paired shares are maximized, the imbalance is minimized and the distance from the bid-ask midpoint is minimized, in that order. Can also be thought of as being equal to the near price bounded between the best bid and ask price. Additionally, it might be worth mentioning that the prices are all given as price movement in points (0.01% == one point) from the previous price instead of being given as absolute prices in dollars.
- `matched_size` - The amount that can be matched at the current reference price (in USD).
- `far_price` - The crossing price that will maximize the number of shares matched based on auction interest only. This calculation excludes continuous market orders. Note: FAR Price (Far-from-the-close price): FAR Price is one of the features provided in the competition dataset. It represents the average price of trades that occurred significantly before the closing auction. It's often used as a reference price to determine how "far" the closing price is from earlier trading activity.
- `near_price` - The crossing price that will maximize the number of shares matched based auction and continuous market orders. Note: NEAR Price is another feature provided in the competition dataset. It represents the average price of trades that occurred closer to the closing auction, as opposed to those that occurred far from the close. NEAR Price is another reference price used to gauge the stock's behavior as it approaches the closing auction.
- `[bid/ask]_price` - Price of the most competitive buy/sell level in the non-auction book.

- `[bid/ask]_size` - The dollar notional amount on the most competitive buy/sell level in the non-auction book.
- `wap` - The weighted average price in the non-auction book.

$$\frac{BidPrice * AskSize + AskPrice * BidSize}{BidSize + AskSize}$$

-
- `seconds_in_bucket` - The number of seconds elapsed since the beginning of the day's closing auction, always starting from 0.
- `target` - The 60 second future move in the wap of the stock, less the 60 second future move of the synthetic index. Only provided for the train set.
    - The synthetic index is a custom weighted index of Nasdaq-listed stocks constructed by Optiver for this competition.
    - The unit of the target is basis points, which is a common unit of measurement in financial markets. A 1 basis point price move is equivalent to a 0.01% price move, Where t is the time at the current observation, we can define the target:

$$Target = (\frac{StockWAP_{t+60}}{StockWAP_t} - \frac{IndexWAP_{t+60}}{IndexWAP_t}) * 10000$$

Some of the tasks mirror those of the previous assignment, as practice makes perfect. As in the previous assignment, you will need to submit all your results in a single google form and your code files in three different format (.ipynb, .pdf and .py). Make sure to have your code documented with proper comments and the exact sequence of operations you needed to produce the resulting tables and figures.

Data downloading

First of all, you need to join the challenge and download the data here. The description of the data can also be found at this page.

Python Installation

Instead of installing python and other tools manually, we suggest installing **Anaconda**, which is a Python distribution with a package and environment manager. It simplifies a lot of common problems when installing tools for data science. More introduction can be found here. Installation instructions can be found here.

Another option can be using **Google Colaboratory.** This is another option for those who want to run their Jupyter notebook remotely instead of installing the required packages locally. Colab allows you to write and execute Python in your browser, with

- Zero configuration required (for simple cases)
- Free access to GPUs
- Easy sharing

If you are an expert of Python and data science, what you need to do is install some packages relevant to data science. Packages that I believe you will use for this homework include:

- pandas
- scikit-learn
- numpy
- Matplotlib
- seaborn (maybe)

The Google colab notebook (Use your @cs.stonybrook.edu email to access the file. If you do not have one, use the form that comes up to request access) contains boilerplate code to download the data to your google drive and a dictionary containing the features along with its data type. **Make a copy of the notebook before you start your HW.**

Tasks (100 pts)

1. Take a look at the training data. There may be anomalies in the data that you may need to factor in before you start on the other tasks. Make a note of the anomalies that you notice. Clean the data first to handle these issues **if it will improve your analysis**. Explain what you did to clean the data (in bulleted form) or explain why you did not change it to address these anomalies. (15 points)

2. Construct a pairwise correlation table of the given variables. Use Pearson correlation. For pairs of variables with high absolute correlations, explain **why** they are so highly correlated. (10 points)

3. Define an "average" or "consensus" record for each stock id *s* on a particular day *d*, measuring how the stock *s* performs on day *d*'s close. Then define a distance function between pairs of "stock-day" distance records, measuring how similarly they are.
    a. For each stock, measure the autocorrelation of the average distance between day *i* and day *i+k* for *-10* <= *k* <= *+10*. On average (over all stocks), is there a statistically significant degree of autocorrelation in the market? Present your evidence for or against? Are there particular stocks whose performance is **unusually** autocorrelated, in a statistically significant way? (10 points)

b. For each pair of stocks *a* and *b*, measure the distance between *a* on day *i* and *b* on day *i* for every day *i*. Are there pairs of stocks which are unusually similar on a consistent basis? If a for-loop proves too slow, investigate broadcastable numpy/pandas operations. (5 points)
c. Construct one "average" or "consensus" record for each stock. Now cluster the stocks using a clustering algorithm like k-means. How many big clusters do you find? Create a TSNE-plot of the stocks where you color each stock according to its cluster ID. Do the colors seem visually coherent to you or not? (10 points)

4. Is the closing trajectory of stocks on each day highly correlated ("there are up days and down days in the market") or is it essentially random (say, "supply and demand cause distinct fluctuations on individual stocks each day")?
    a. Make three plots that convince you of which is the right answer, and will convince me as well. If there is a formal statistical test to help you do this, do it. (10 points)
    b. Perform a permutation test to determine the statistical confidence that you believe your answer. In particular, for each stock randomly permute the day index to construct an artificial time series for it. Now measure the consistency of daily performance in this permuted data set. Run enough permutations per variable to establish a *p*-value of how confident you are in your conclusion (15 points)

5. Finally, build the best prediction model you can to solve the Kaggle task. Use any data, ideas, and approach that you like, but describe all the approaches you try. For each model you try, report the average absolute error using 5-fold cross-validation. (20 points)

6. Submit the results of your best models on Kaggle. When submitting to Kaggle, understand that you must use the API provided by the competition host to retrieve data or you will not receive a score. Report the rank, score, number of entries, for your highest rank. Include a snapshot of your best score on the leaderboard as confirmation. (5 points)

Rules of the Game

1. This assignment must be done **individually by each student**. It is not a group activity.
2. This project is on a particular technical domain (closing auctions in the stock market) and understanding as much as you can about the domain is important to doing well on it. Make sure you understand what the variables mean and what their impact on the target should be.
3. Get started on this as early as possible! As we have seen, it helps to start simple and then iterate to better solutions.

4. I will soon distribute semester project descriptions, and the period of this assignment will overlap that of the time you are preparing your project proposals. Be prepared to work on both simultaneously.
5. I have one other question I almost added to the assignment. If you are a hot shot who wants another task to do (for little or no credit), ask me to release it after you have finished the real assignment.
6. All of your written responses should be put in the appropriate place in your notebook template. Get the template notebook form from here. *Use your @cs.stonybrook.edu email to access the file. If you do not have one, use the form that comes up to request access.*

   You are allowed to add more cells, but definitely fill out the cells we give.
7. I intend to have covered all relevant material before the assignment is due. But I look forward to discussing things in class and on Piazza, so please ask. And feel free to read ahead in the book and/or lecture slides.
8. To ensure that you are who you are when submitting your models, have your Kaggle profile show your face as well as a Stony Brook affiliation.
9. There are some public discussions and demos relevant to this problem on Kaggle. It is okay for students to read these discussions, but they must write the code and analyze the data by themselves.