Development Finance Models & Fintech:

Stocks Liquidity Analysis with Machine Learning Models

I. Background

Overview of Stock Markets:

The stock market serves as a dynamic and complex platform where investors buy and sell financial instruments, primarily stocks, representing ownership in companies. This market is essential for capital formation, enabling businesses to raise funds for expansion and innovation, while providing investors with opportunities for wealth creation.

Transaction Mechanism

In the stock market, transactions occur through a well-established mechanism. Investors execute buy or sell orders through intermediaries known as brokers. These orders are matched electronically on a stock exchange, facilitating the transfer of ownership. The execution of transactions involves considerations such as market orders, limit orders, and stop orders, each influencing the price and quantity of securities traded.

Transaction Book: Supply and Demand Dynamics

At the heart of stock market transactions lies the concept of a transaction book, a record of buy and sell orders awaiting execution. The transaction book reflects the supply and demand for a particular security at various price levels. The highest bid and the lowest ask prices in this book establish the current market price, while the order book continuously evolves as new orders are placed or existing orders are canceled.

II. Problem Description:

The Tel Aviv stock market is grappling with a liquidity challenge in its stocks, leading to widened margins in the buying and selling processes.

This means that the actual value of a share is not accurately reflected in the execution of transactions, resulting in additional costs for both buyers and sellers. These unforeseen costs could be mitigated if the securities exhibited higher liquidity. In cases of low liquidity and limited trading activity, securities may fail to meet the necessary criteria for buying or selling, exacerbating the overall decline in the liquidity of the stock market.

This issue not only hinders the efficiency of transactions but also potentially deters market participants due to increased uncertainty and associated costs. Our project aims to address these liquidity concerns through the application of machine learning models, providing valuable insights to enhance decision-making and improve overall market liquidity.

III. Project's Objective:

The primary objective of our project is to design and implement a machine learning algorithm to predict whether an order will be executed or not, with the goal of minimizing spreads.

This algorithm will further endeavor to forecast if it is the optimal time for buying and selling shares, categorized by specific months, weeks, days, and hours. The overarching aim is to minimize the margin ('spread') between transaction prices and the actual value of a share, thereby mitigating additional costs for market participants.

In addition, we will develop a user-friendly query interface. This interface will facilitate seamless interaction with the stock exchange's data, empowering users to filter transactions based on criteria such as paper number, quantity, date, and time.

This collaborative effort aims to enhance the transparency and efficiency of stock market transactions, providing users with a valuable tool to navigate the complexities of the market while optimizing their investment decisions.

IV. Target Audience:

ML Algorithm:

Investment houses, pension funds, professional traders in the high-scale capital market, venture capital funds.

Query interface:

The Stock Exchange - Tel Aviv Stock Exchange employees.

V. Key Features:

ML Algorithm

The end user will enter a share for which he wishes to perform a buy/sell transaction. The algorithm will make a prediction based on the information and will give a recommendation as to whether this is currently the correct timing to carry out the transaction (at the level of liquidity only), or whether it is better to wait with the sell / buy order - and this is to maintain a minimum margin between the value of the stock and the price of the transaction.

- Predicts whether a buy/sell order will be executed based on liquidity.
- Helps minimize the margin between the order price and execution price, optimizing transactions.

Query interface:

- Filters securities by paper number, quantity, date, and time.
- Queries the duration for completing transactions at predefined prices.
- Determines the actual margin from the execution price of specific transactions.

VI. Implementation

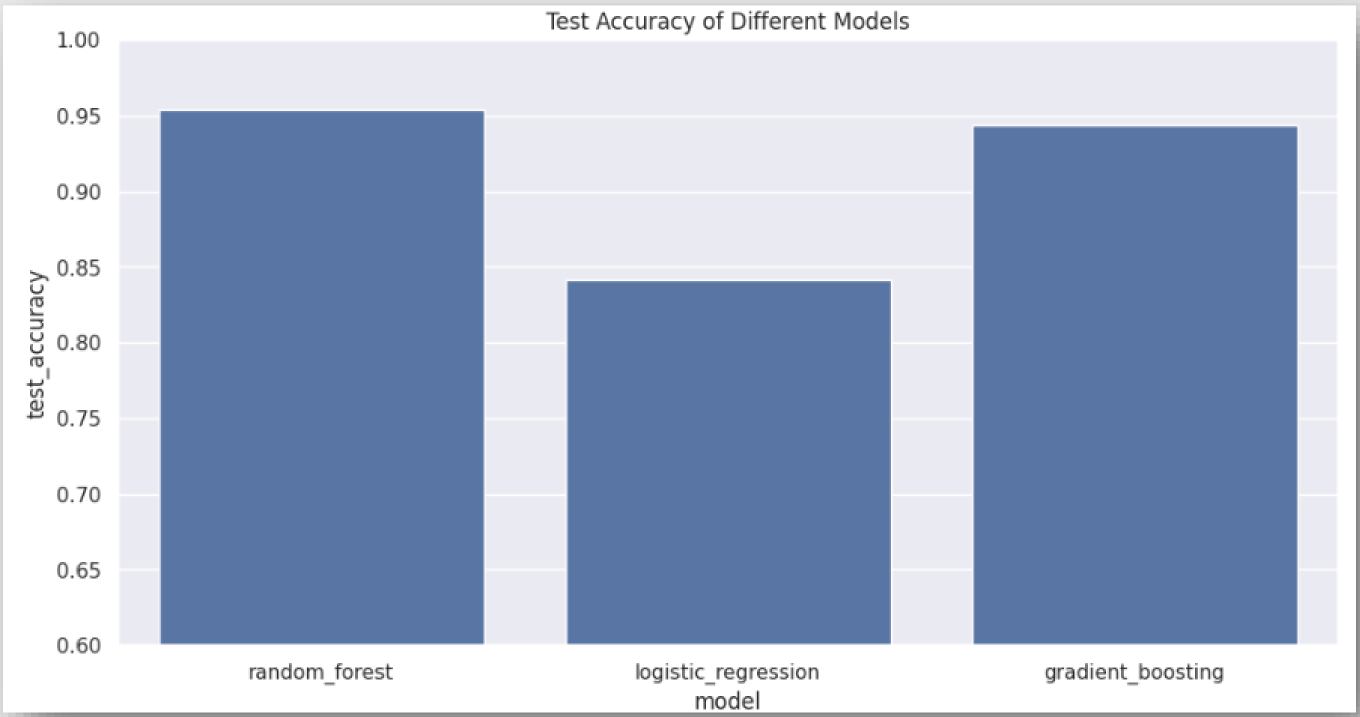
Data Collection and Preprocessing

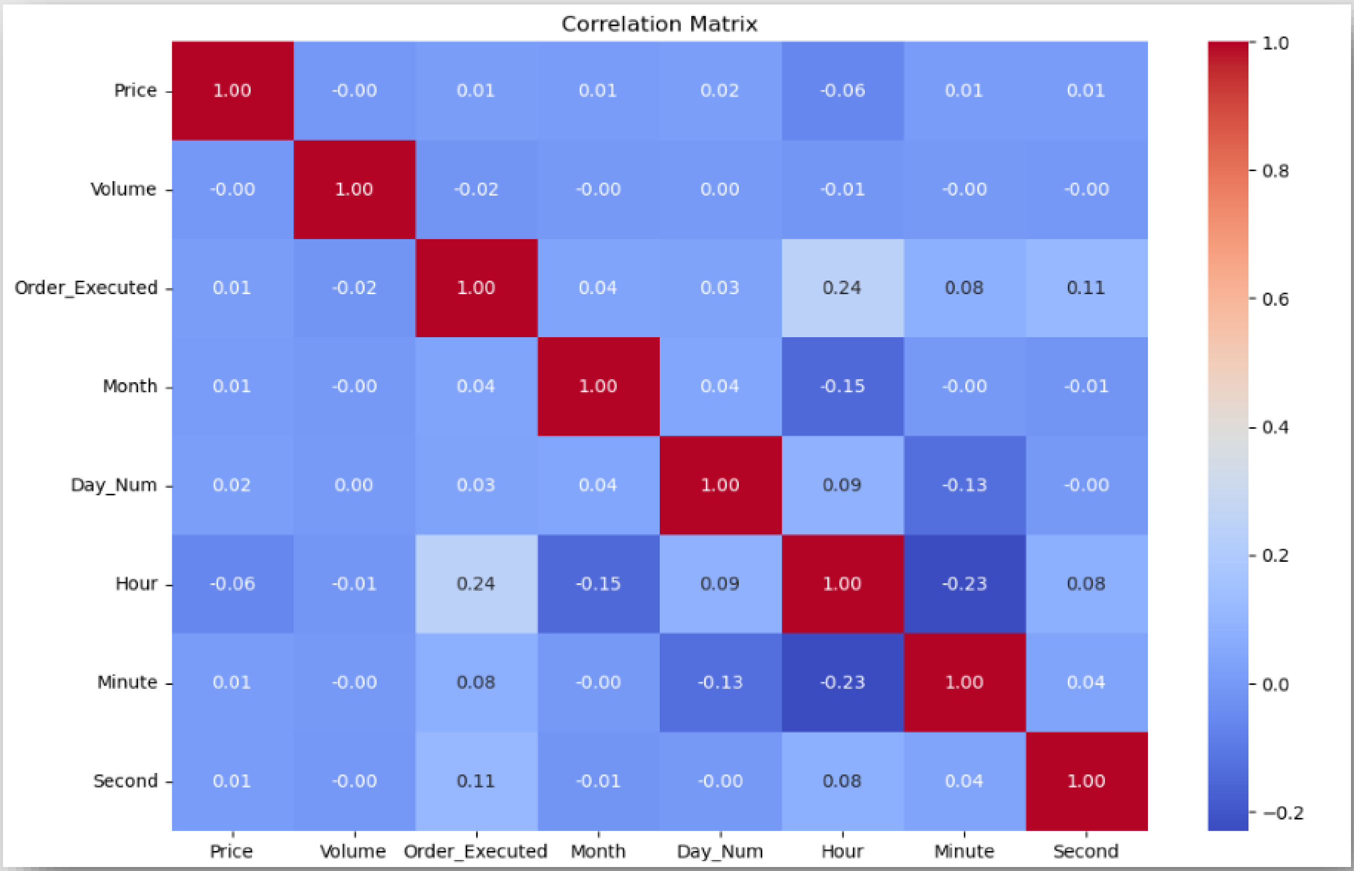
- Interfacing with the Tel Aviv Stock Exchange API to download daily encrypted trading data.
- Building a parser script to decode and decompress the encrypted data, preparing it for machine learning analysis.
- As part of the data preprocessing process, we utilized a specialized library provided by stock exchange employees due to the unique and proprietary encryption of the files, implemented for security reasons.
- Consequently, we had to perform numerous integrations. The output of this preparation phase is a csv file containing approximately 26 million records, covering around six months of trading activity.

Model Development

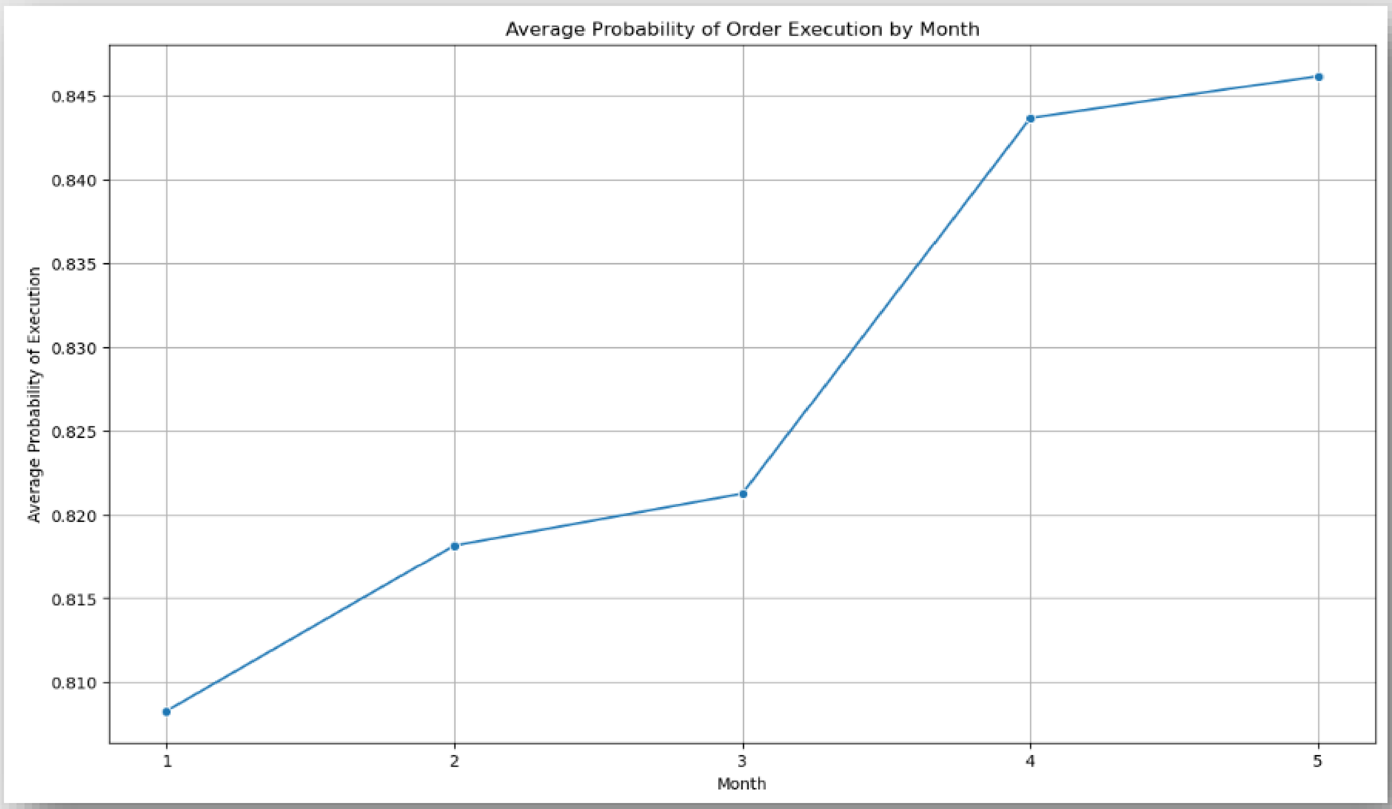
- Experimenting with several machine learning models, including Random Forest Classifier, to determine the most accurate one.
- Given the large volume of data and our limited computing resources, we tested several models on a single trading day. The "Random Forest Classifier" model demonstrated the highest accuracy for our data. Consequently, we applied this model to the entire dataset and achieved an accuracy rate exceeding 96%.
- We encountered several challenges due to the substantial data size and limited computational resources, including crashes in Google Collab due to insufficient RAM. To mitigate these issues, we employed strategies such as reading the data in chunks and limiting RAM usage in Jupyter Notebook. As a result, the model training process took an entire day.

Data Visualizations and Model Accuracy Outputs

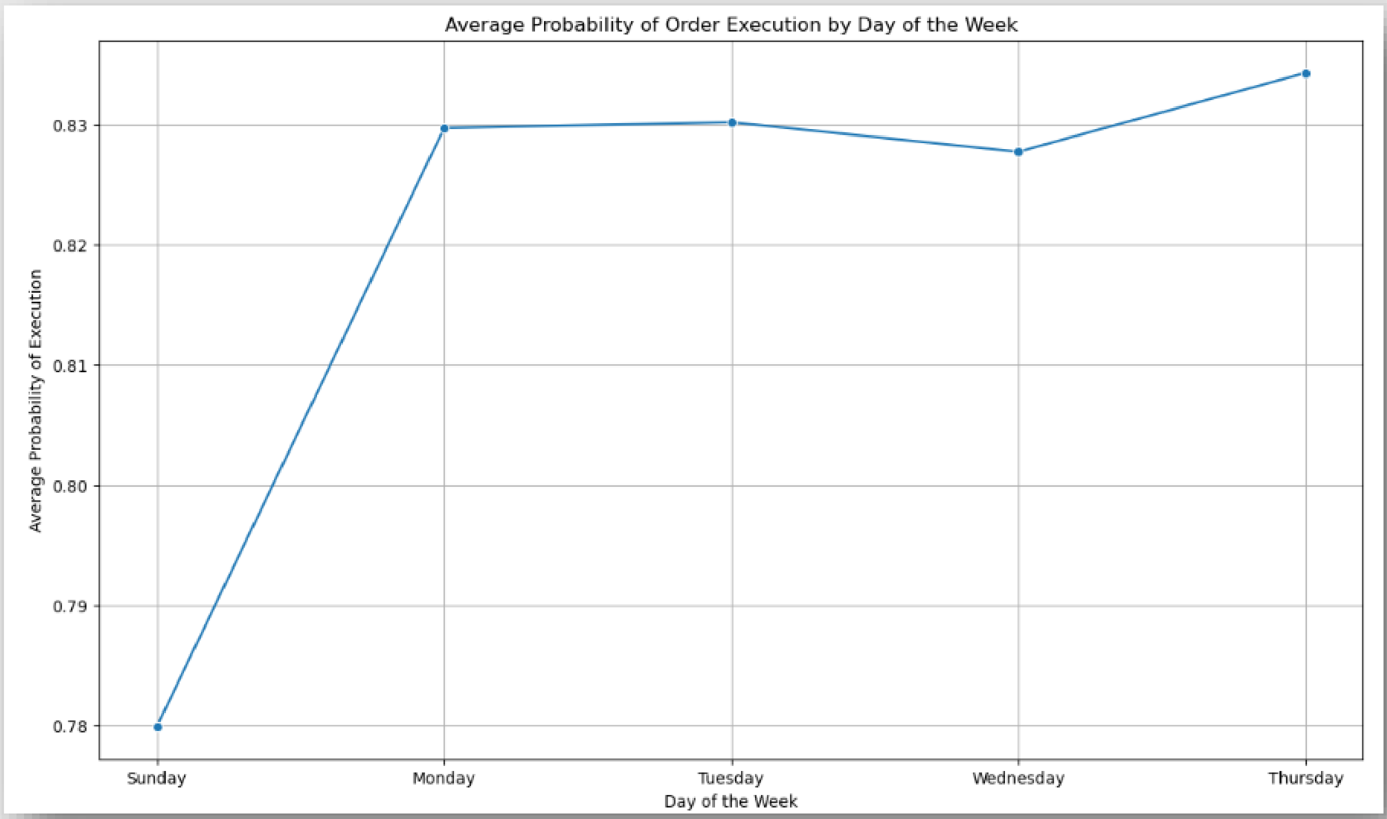




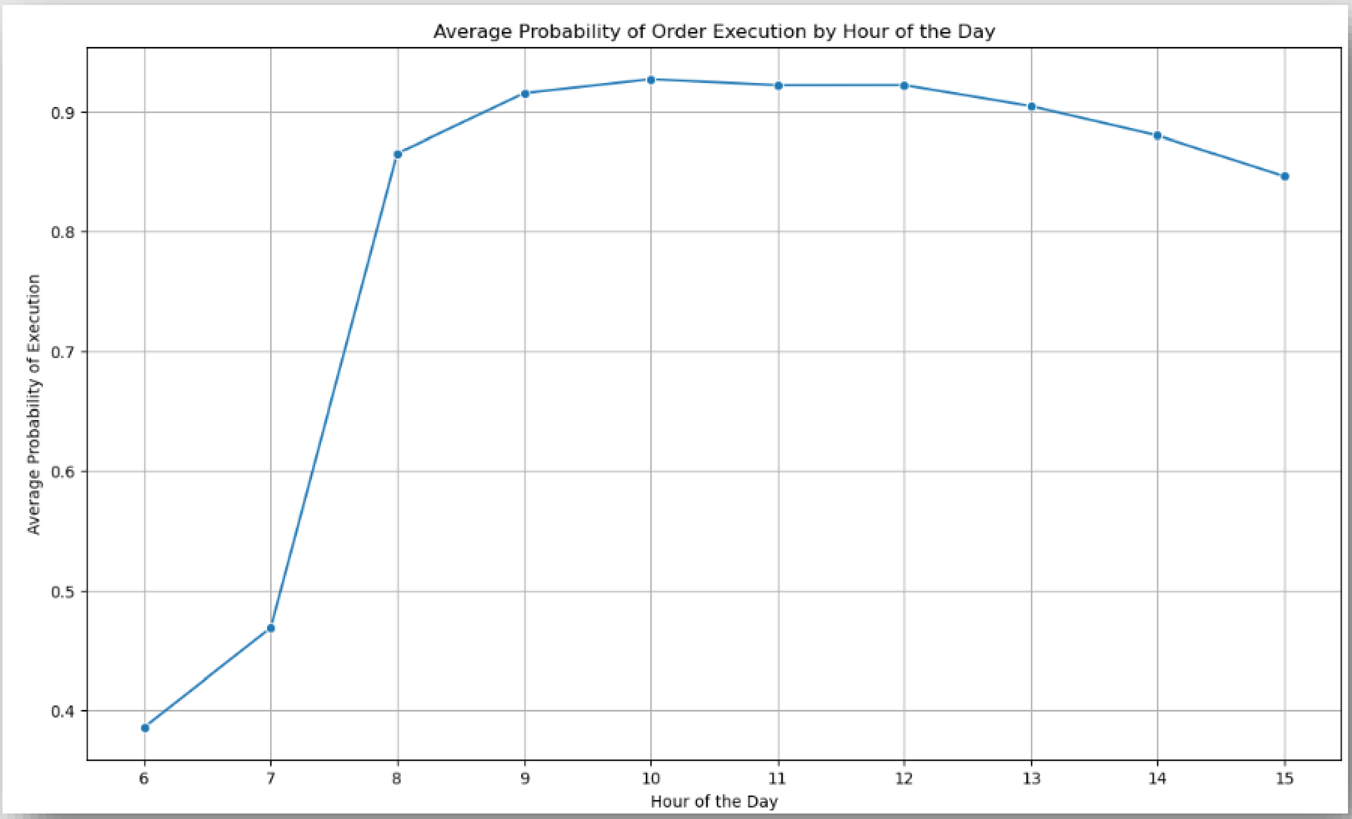
	Price	Volume	Order_Executed	Month	Day_Num	Hour	Minute	Second
Price	1.00	-0.00	0.01	0.01	0.02	-0.06	0.01	0.01
Volume	-0.00	1.00	-0.02	-0.00	0.00	-0.01	-0.00	-0.00
Order_Executed	0.01	-0.02	1.00	0.04	0.03	0.24	0.08	0.11
Month	0.01	-0.00	0.04	1.00	0.04	-0.15	-0.00	-0.01
Day_Num	0.02	0.00	0.03	0.04	1.00	0.09	-0.13	-0.00
Hour	-0.06	-0.01	0.24	-0.15	0.09	1.00	-0.23	0.08
Minute	0.01	-0.00	0.08	-0.00	-0.13	-0.23	1.00	0.04
Second	0.01	-0.00	0.11	-0.01	-0.00	0.08	0.04	1.00

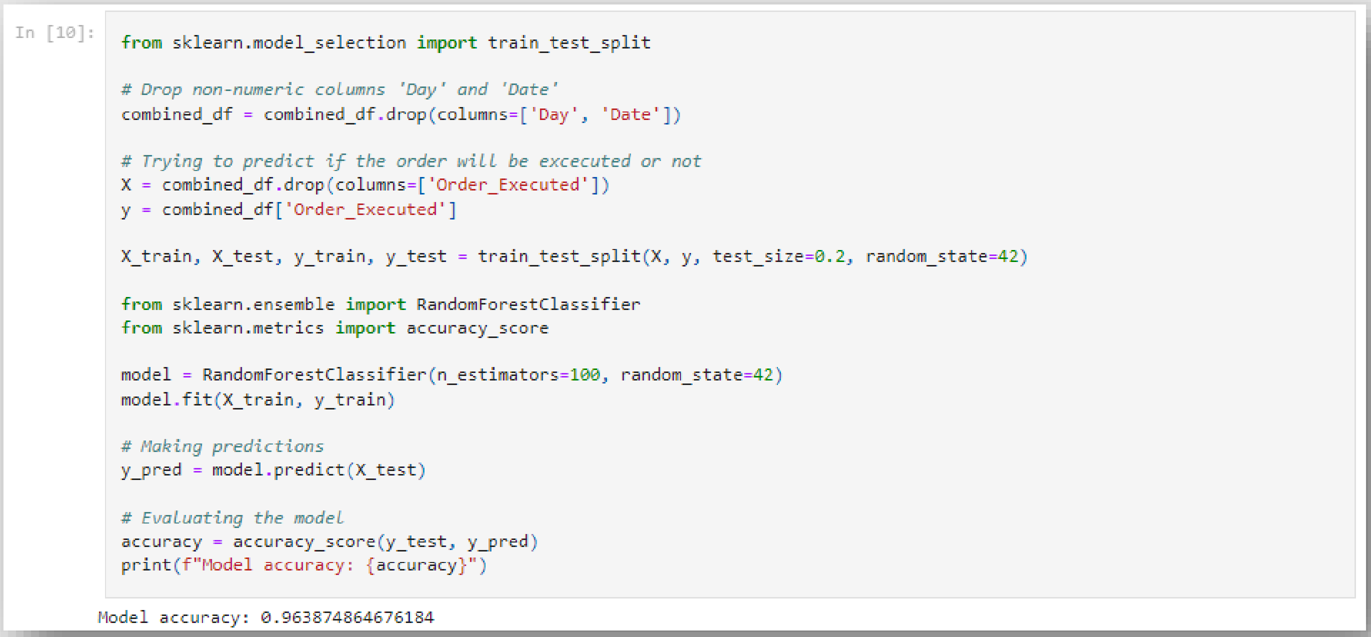


Average Probability of Order Execution by Month
Average Probability of Execution
0.845
0.840
0.835
0.830
0.825
0.820
0.815
0.810
1
2
3
4
5
Month



Average Probability of Order Execution by Day of the Week
Average Probability of Execution
0.83
0.82
0.81
0.80
0.79
0.78
Sunday
Monday
Tuesday
Wednesday
Thursday
Day of the Week




In [10]:

```
from sklearn.model_selection import train_test_split

# Drop non-numeric columns 'Day' and 'Date'
combined_df = combined_df.drop(columns=['Day', 'Date'])

# Trying to predict if the order will be excecuted or not
X = combined_df.drop(columns=['Order_Executed'])
y = combined_df['Order_Executed']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Making predictions
y_pred = model.predict(X_test)

# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Model accuracy: {accuracy}")
```

```
Model accuracy: 0.963874864676184
```

Dashboard Development

In collaboration with the Tel Aviv Stock Exchange, we developed a dashboard to analyze and display findings on transactions occurring in the stock market daily. This part of the project focused on calculating and visualizing data for specific securities based on the exchange's guidelines.

Dashboard Components

Based on the CSV files we created, we performed daily calculations for each security.

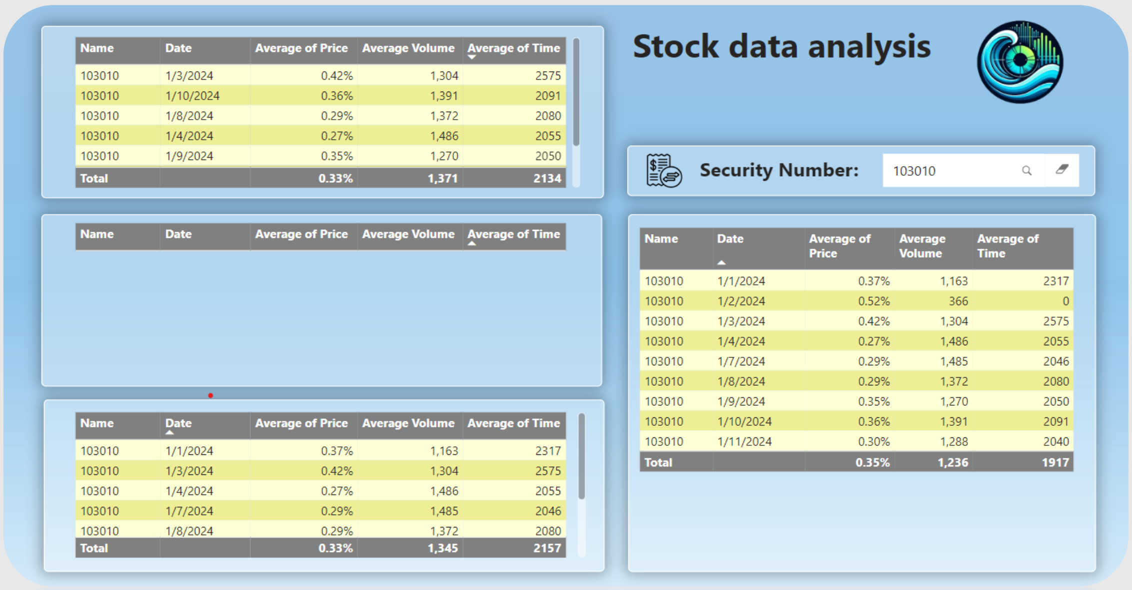
Tables:

The metrics calculated included:

- <u>Percentage of the Bid-Ask Spread:</u> This indicates the difference between the highest bid and the lowest ask prices calculated as a percentage.
- <u>Average Quoted Quantity per Day:</u> This represents the average number of shares quoted for buying and selling each day.
- <u>Average Quote Time in Seconds:</u> The average duration for which quotes remain valid each day.

The dashboard includes a table that filters data based on the requested security number, displaying the calculation results and summarizing the data.
Total Average Quoted Quantity: This parameter is displayed in the first table on the right as a key metric for the next three tables that represent on the left. In these tables, we calculated and displayed data with restrictions on the minimum quoted amount.
In each table, only instructions were taken into account which are:

- At least the average amount.
- At least twice the average amount.
- At least half the average amount.



Name	Date	Average of Price	Average Volume	Average of Time
103010	1/3/2024	0.42%	1,304	2575
103010	1/10/2024	0.36%	1,391	2091
103010	1/8/2024	0.29%	1,372	2080
103010	1/4/2024	0.27%	1,486	2055
103010	1/9/2024	0.35%	1,270	2050
Total		0.33%	1,371	2134

Name	Date	Average of Price	Average Volume	Average of Time

Name	Date	Average of Price	Average Volume	Average of Time
103010	1/1/2024	0.37%	1,163	2317
103010	1/3/2024	0.42%	1,304	2575
103010	1/4/2024	0.27%	1,486	2055
103010	1/7/2024	0.29%	1,485	2046
103010	1/8/2024	0.29%	1,372	2080
Total		0.33%	1,345	2157

Stock data analysis

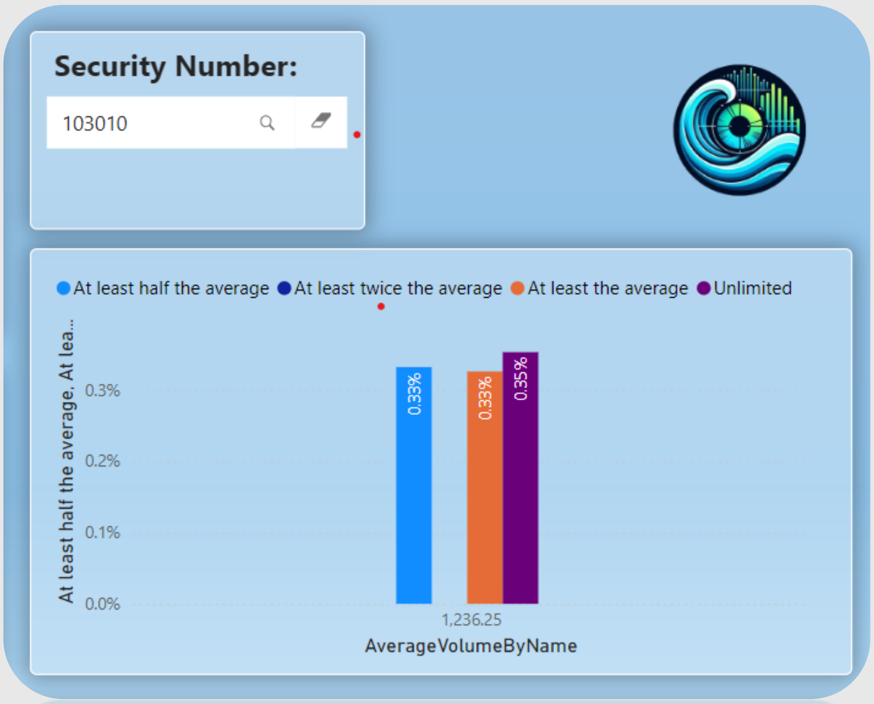
Security Number: 103010

Name	Date	Average of Price	Average Volume	Average of Time
103010	1/1/2024	0.37%	1,163	2317
103010	1/2/2024	0.52%	366	0
103010	1/3/2024	0.42%	1,304	2575
103010	1/4/2024	0.27%	1,486	2055
103010	1/7/2024	0.29%	1,485	2046
103010	1/8/2024	0.29%	1,372	2080
103010	1/9/2024	0.35%	1,270	2050
103010	1/10/2024	0.36%	1,391	2091
103010	1/11/2024	0.30%	1,288	2040
Total		0.35%	1,236	1917

Column Graph:

This graph illustrates the relationship between the minimum quoted amount and the percentage of the bid-ask spread. It includes data from all four tables on the first page: (No minimum quantity limit, At least the average amount, At least twice the average amount, At least half the average amount)

The graph helps understand how a minimum quoted amount affects the profit margin percentage, and subsequently, the level of liquidity.



Insights and Learnings:

- **Economic and Stock Market Understanding:** Through this part of the project, we deepened our knowledge of economic and stock market concepts. We learned how transactions are conducted and how to analyze information to derive meaningful conclusions.
- **Technical Skills:** We developed skills in using new calculation tools to analyze data. We focused on creating an inviting and visually appealing user interface to present our findings effectively**.**

VII. External dependencies:

The project is carried out in collaboration with the Tel Aviv Stock Exchange, and for the sake of the project we get access to real data of transactions in the stock exchange. The data is in binary format in MBO protocol, that developed by them. We had to decipher and process the information, and cast the relevant database for the algorithm, as well as for the query interface.

VIII. Technologies:

ML Algorithm

1. Preparing and processing the data and saving it in a database in the cloud - creating a script in the Python programming language, which will receive the data in binary format, decode the data, arrange the data in a file suitable for reading, and store them in a database in the cloud.

2. Prediction algorithm - we will combine several well-known models in the Python language and check their level of accuracy, in the end we will recommend using the model with the highest level of accuracy.

3. Libraries in Python that we will work with - Pandas, NumPy, Scikit-learn, TensorFlow, Seaborn, Matplotlib

4. Use of a log library for internal monitoring - Logging

Query interface:

- User interface – Power BI