Internship Report

At KANINI Software, Sholinganallur, Chennai   Ananth T A

From October 2023 to December 2023

Problem Statement (s):

Task 1:

- Given a dataset of clinic visits of a group of clinics, predict if a particular patient will appear for their appointment or not

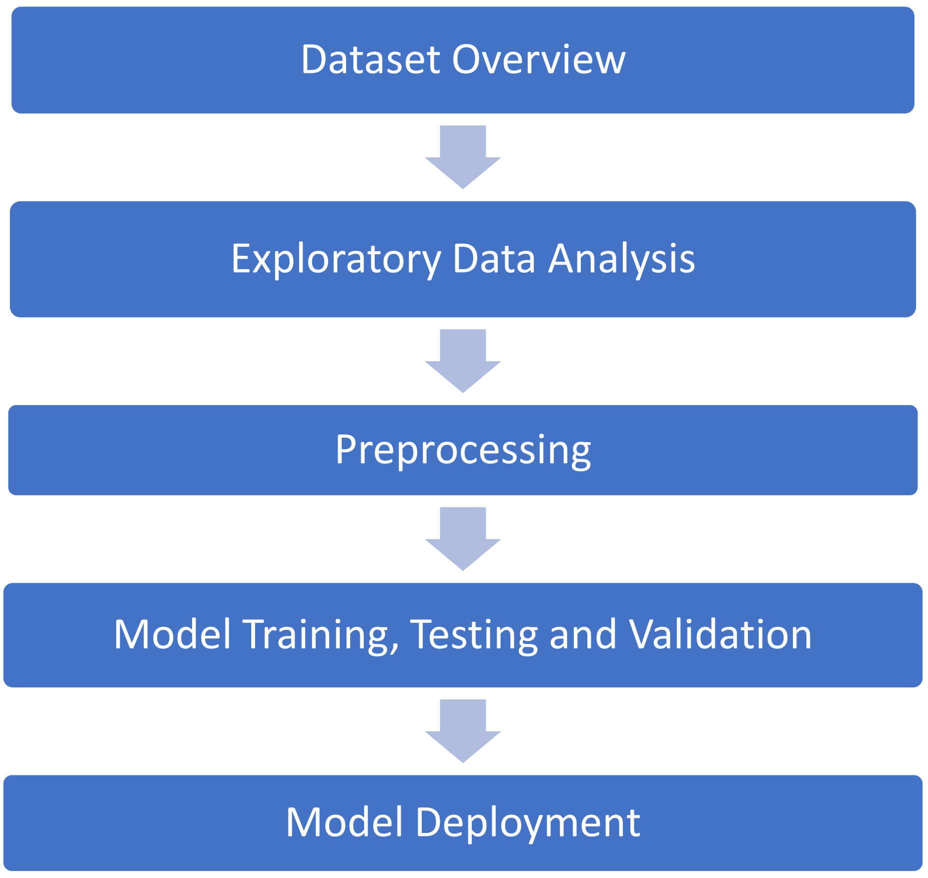
Task 2:

- Given a dataset of clinic visits of a group of clinics, predict at what time the patient will appear for their appointment

Task 3:

- Given a dataset of the trends of a particular stock from 2015 to 2023, predict the Highest, Lowest and Closing price at any given date

Solution Framework Overview:


Dataset Overview
Exploratory Data Analysis
Preprocessing
Model Training, Testing and Validation
Model Deployment

Tools Used:

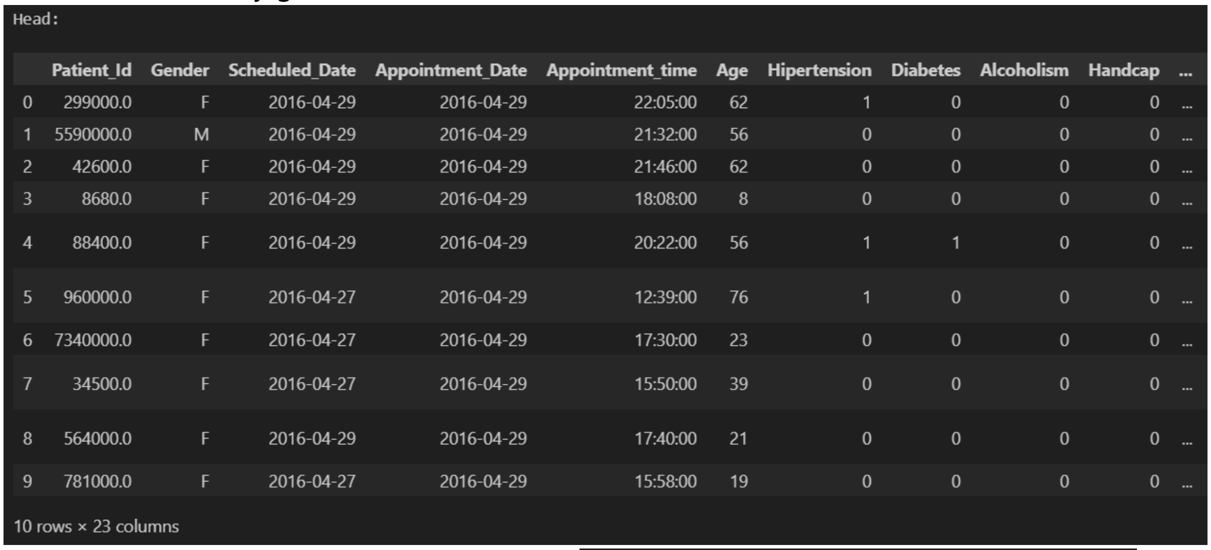
All processing was done locally in python, via Visual Studio Code and relevant extensions.

- Data Overview – Jupyter Notebook, Pandas
- Exploratory Data Analysis - Pandas, Seaborn, Matplotlib
- Preprocessing - Jupyter Notebook, Pandas, Scikit Learn
- Model Training, Testing and Validation – Scikit Learn
- Model Deployment – Joblib, Flask, HTML, CSS, JavaScript

Task 1: No-show Classification

Understanding the dataset:

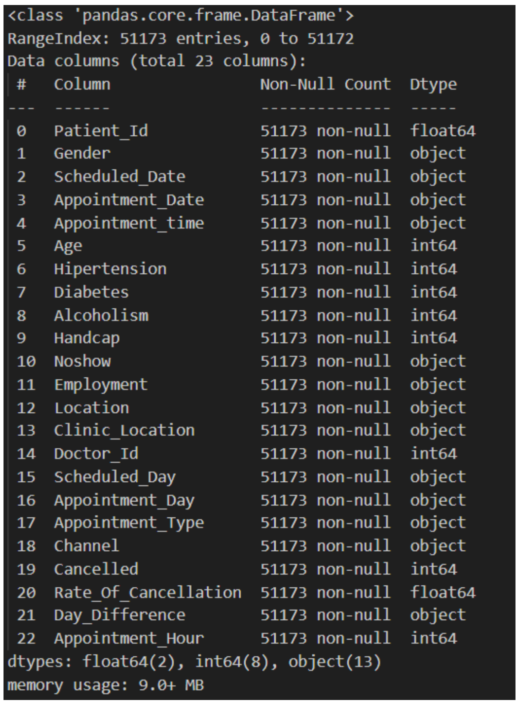
Let us take a cursory glance at our dataset



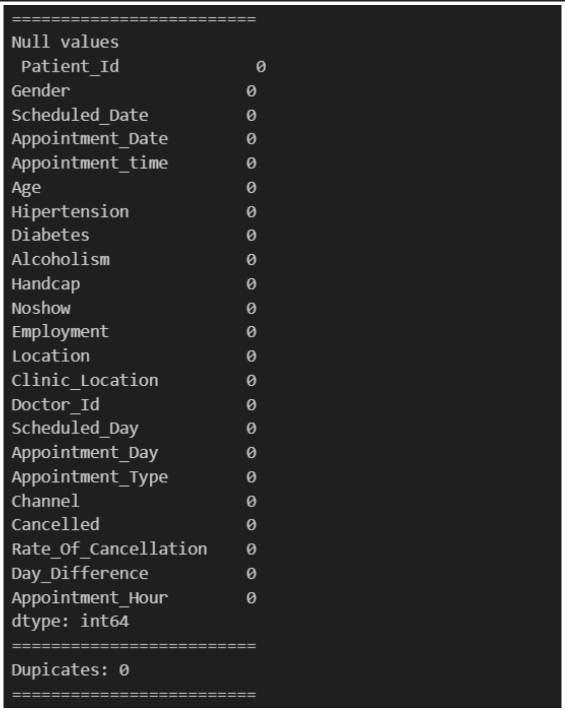
Head:

	Patient_Id	Gender	Scheduled_Date	Appointment_Date	Appointment_time	Age	Hipertension	Diabetes	Alcoholism	Handcap	...
0	299000.0	F	2016-04-29	2016-04-29	22:05:00	62	1	0	0	0	...
1	5590000.0	M	2016-04-29	2016-04-29	21:32:00	56	0	0	0	0	...
2	42600.0	F	2016-04-29	2016-04-29	21:46:00	62	0	0	0	0	...
3	8680.0	F	2016-04-29	2016-04-29	18:08:00	8	0	0	0	0	...
4	88400.0	F	2016-04-29	2016-04-29	20:22:00	56	1	1	0	0	...
5	960000.0	F	2016-04-27	2016-04-29	12:39:00	76	1	0	0	0	...
6	7340000.0	F	2016-04-27	2016-04-29	17:30:00	23	0	0	0	0	...
7	34500.0	F	2016-04-27	2016-04-29	15:50:00	39	0	0	0	0	...
8	564000.0	F	2016-04-29	2016-04-29	17:40:00	21	0	0	0	0	...
9	781000.0	F	2016-04-27	2016-04-29	15:58:00	19	0	0	0	0	...

10 rows × 23 columns



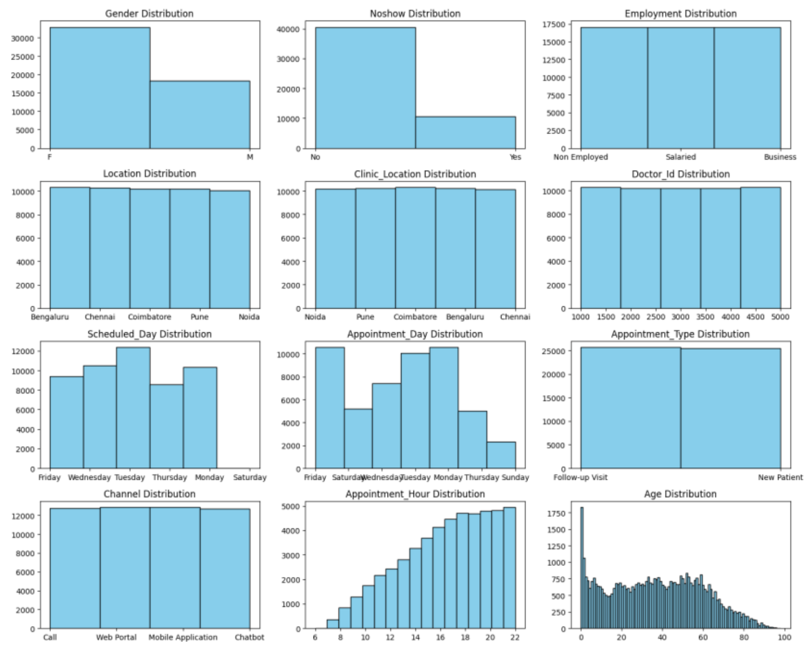
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51173 entries, 0 to 51172
Data columns (total 23 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Patient_Id           51173 non-null  float64
 1   Gender               51173 non-null  object
 2   Scheduled_Date       51173 non-null  object
 3   Appointment_Date     51173 non-null  object
 4   Appointment_time     51173 non-null  object
 5   Age                  51173 non-null  int64
 6   Hipertension         51173 non-null  int64
 7   Diabetes             51173 non-null  int64
 8   Alcoholism           51173 non-null  int64
 9   Handcap              51173 non-null  int64
 10  Noshow               51173 non-null  object
 11  Employment           51173 non-null  object
 12  Location             51173 non-null  object
 13  Clinic_Location      51173 non-null  object
 14  Doctor_Id            51173 non-null  int64
 15  Scheduled_Day        51173 non-null  object
 16  Appointment_Day      51173 non-null  object
 17  Appointment_Type     51173 non-null  object
 18  Channel              51173 non-null  object
 19  Cancelled            51173 non-null  int64
 20  Rate_Of_Cancellation 51173 non-null  float64
 21  Day_Difference       51173 non-null  object
 22  Appointment_Hour     51173 non-null  int64
dtypes: float64(2), int64(8), object(13)
memory usage: 9.0+ MB
```

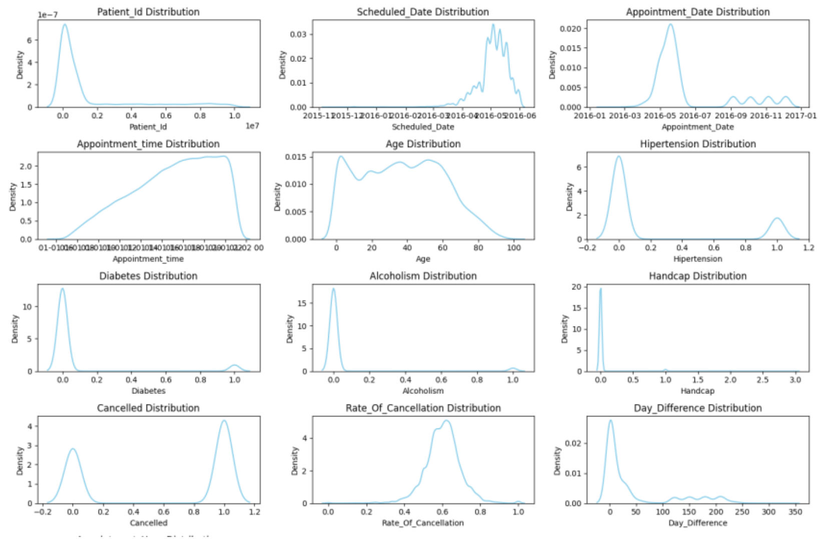


```
========================
Null values
 Patient_Id            0
Gender                0
Scheduled_Date        0
Appointment_Date      0
Appointment_time      0
Age                   0
Hipertension          0
Diabetes              0
Alcoholism            0
Handcap               0
Noshow                0
Employment            0
Location              0
Clinic_Location       0
Doctor_Id             0
Scheduled_Day         0
Appointment_Day       0
Appointment_Type      0
Channel               0
Cancelled             0
Rate_Of_Cancellation  0
Day_Difference        0
Appointment_Hour      0
dtype: int64
========================
Dupicates: 0
========================
```

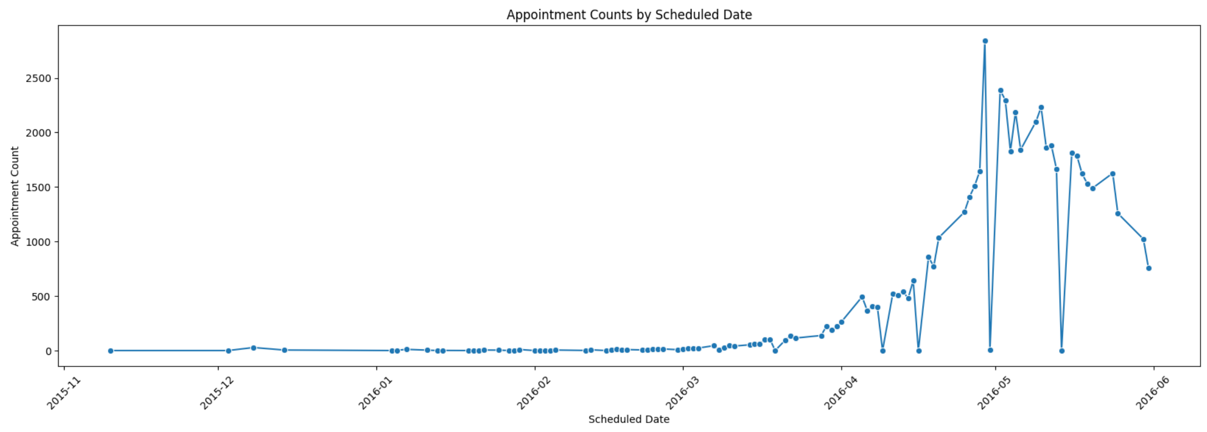
From the above overview of the dataset, we can see that there are 23 columns and 51173 entries. Most columns are non-numerical, non-ordinal and have no duplicates or null entries. Mean age is 36, 20% of patients have hypertension, 6% have diabetes, and 3% have alcoholism.

Exploratory Data Analysis:

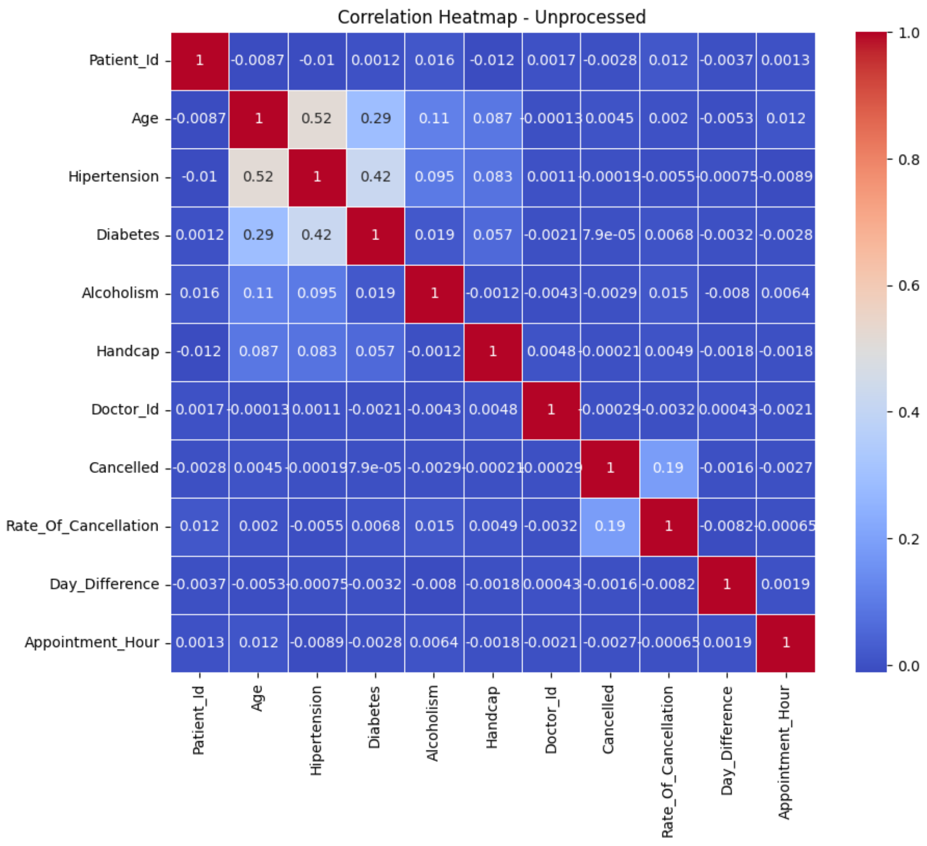




From the above, we can see that Gender is heavily skewed towards female, age has a massive spike in 0, Appointments tend to be higher into the night, there are far more 'no' entries in no-show (i.e., more people who showed up than not). Follow ups are just as much as new visits, there are more appointments on Monday, Tuesday, and Friday. Most people tend to have a rate of cancellation around 0.6, There are more cancelled appointments than non-cancelled ones, and most people have very little difference between their scheduled appointment and actual appointment dates. Very few of our patients have hypertension, and fewer have diabetes. There seems to be a massive spike in appointments at 2016-5, which falls by 2016-7



Here we can notice a gradual increase in appointments as April 2016 begins, with some falls that rise quickly. Towards the end of April, the appointments scheduled see a massive surge to reach its peak, but suddenly falls by the time May 2016 approaches, only to pick right back up. This can be explained by a clinic-wide holiday on May 1st, May Day. The high rate of appointments continues till mid may, where a similar plummet happens. But as June approaches, rates fall to mid-April levels.



The above heatmap shows us that Hypertension and age has a medium correlation of 0.52. Hypertension and diabetes have a relatively weaker correlation of 0.42, and diabetes is weakly correlated with age, at a 0.29 coefficient. There is a weak correlation between rate of cancellation and cancelled appointments, meaning people with higher cancellation rates mildly tend to cancel appointments more. The lack of all variables indicates the need to transform the categorical variables for further analysis.

Preprocessing:

The following preprocessing tasks are done:

1. **Dropping Null Values:** The function first removes any rows that contain missing values. This ensures that the dataset is clean and does not contain any incomplete or unreliable data.
2. **Removing Zero Age Entries:** The function identifies and removes any rows where the age is recorded as zero. This is done because it is unlikely for a patient to have an age of zero, and these entries are likely to be errors or missing data.
3. **Converting Data Types:** The function converts certain columns to the appropriate data types. The 'Patient_Id' column is converted to integer type, while the 'Scheduled_Date', 'Appointment_Date', and 'Appointment_time' columns are converted to datetime format. This allows for easier manipulation and analysis of these columns.
4. **Calculating Day Difference:** The function calculates the number of days between the appointment date and the scheduled date and adds this information as a new column called 'Day_Difference'. This can be useful for analysing the time gap between scheduling and the actual appointment.
5. **Feature Extraction:** The function extracts additional features from the date columns. It extracts the year, month, and day from both the scheduled date and appointment date, as well as the hour and minute from the appointment time. These extracted features can provide more insights into patterns and trends related to appointment scheduling.
6. **Age Grouping:** The function creates a new column called 'Age_Group' by categorizing the age into different groups. The age groups are defined as 'Child' (0-18 years), 'Young Adult' (18-35 years), 'Adult' (35-50 years), and 'Senior' (50-100 years). This grouping allows for easier analysis and comparison based on age ranges.
7. **Feature Encoding:** The function performs ordinal encoding on the categorical features in the dataframe. Ordinal encoding assigns a numerical value to each category, preserving the ordinal relationship between categories. This encoding is applied to features such as gender, hypertension, diabetes, alcoholism, and others.
8. **Removing Duplicates and Null Values:** After the preprocessing steps, the function removes any duplicate rows and any remaining rows with missing values. This ensures that the final dataset is clean and ready for further analysis.
9. **Correlation identification:** We plot a correlation matrix to identify the most correlating variables (at a threshold of 0.3) to use in model training, since we need not use all columns to train the model. ‘Scheduled_Day’, ‘Scheduled_Month’, ‘Rate_Of_Cancellation’, ‘Alcoholism’, ‘Cancelled’, ‘Scheduled_Year’, ‘Appointment_Day’,

'Gender', 'Diabetes', 'Hipertension', 'Appointment_Month', and 'Day_Difference' are all the features that show most correlation. Thus, these are suitable features we select for training.

Model Training, Testing and Validation:

- Data Preparation:
    - Features: All columns in the dataset except the target variable 'Noshow'.
    - Target variable: 'Noshow'.
- Models:
    - Random Forest
    - Gradient Boosting
    - XGBoost
    - Logistic Regression
    - Decision Tree
    - K-Nearest neighbours
    - Neural Network (MLP)
    - AdaBoost
- Undersampling (Random):
    - Applied Random Under Sampling to balance the classes.
    - Best performing model: Gradient Boosting with an accuracy of 62.71%.
- Undersampling (ENN):
    - Applied Edited Nearest Neighbours (ENN) for undersampling.
    - Best performing model: K-Nearest Neighbours with an accuracy of 76.84%.
- Oversampling (SMOTE):
    - Applied Synthetic Minority Oversampling Technique (SMOTE) for oversampling.
    - Best performing model: XGBoost with an accuracy of 85.02%.
- Combination (ENN then SMOTE):
    - Applied ENN for undersampling followed by SMOTE for oversampling.
    - Best performing model: K-Nearest Neighbours with an accuracy of 80.51%.
- Combination (SMOTE then ENN):
    - Applied SMOTE for oversampling followed by ENN for undersampling.
    - Best performing model: K-Nearest Neighbours with an accuracy of 89.48%.
- Hyperparameter Tuning:
    - Used Grid Search and Random Search for hyperparameter tuning on XGBoost.

- Best performing model (Grid Search): XGBoost with an accuracy of 86.87%.
- Best performing model (Random Search): XGBoost with an accuracy of 88.97%.

- Recursive Feature Elimination (RFE):

  - Applied RFE to select the optimal features.
  - Best performing model: Random Forest with an accuracy of 88.68% using the selected features.

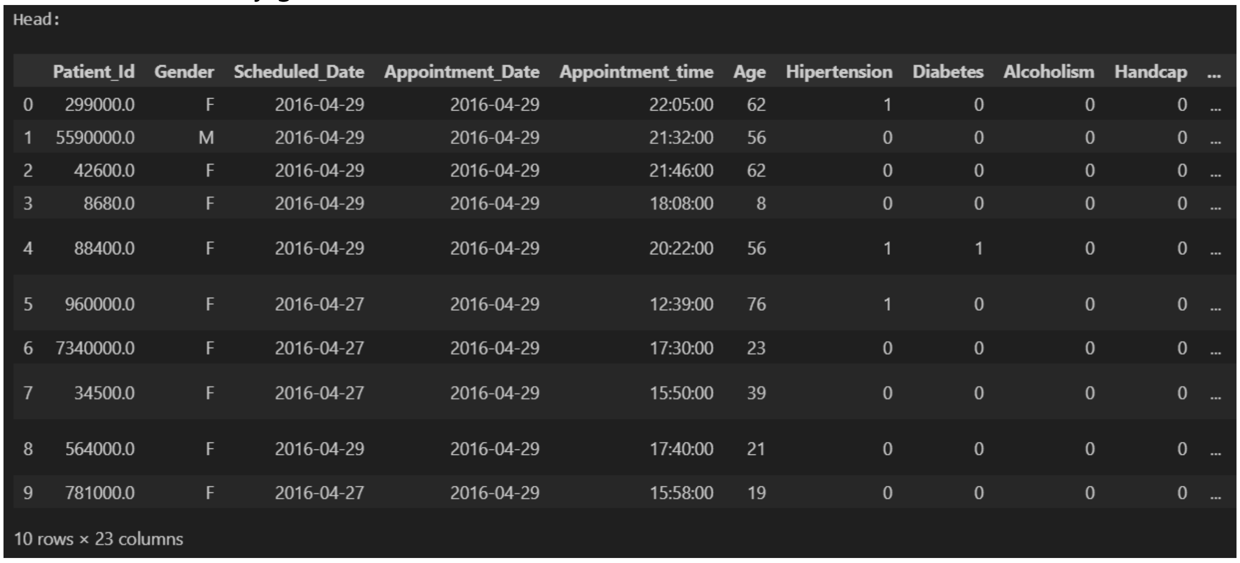
- Inferences and Results:

  - The choice of sampling techniques significantly impacts model performance.
  - The combination of SMOTE for oversampling followed by ENN for undersampling yielded the highest accuracy (89.48%).
  - Hyperparameter tuning improved model accuracy, with both Grid Search and Random Search providing competitive results.
  - Recursive Feature Elimination (RFE) helped identify and use the most relevant features for better model performance.

Task 2: Appointment Time Prediction

Understanding the dataset:

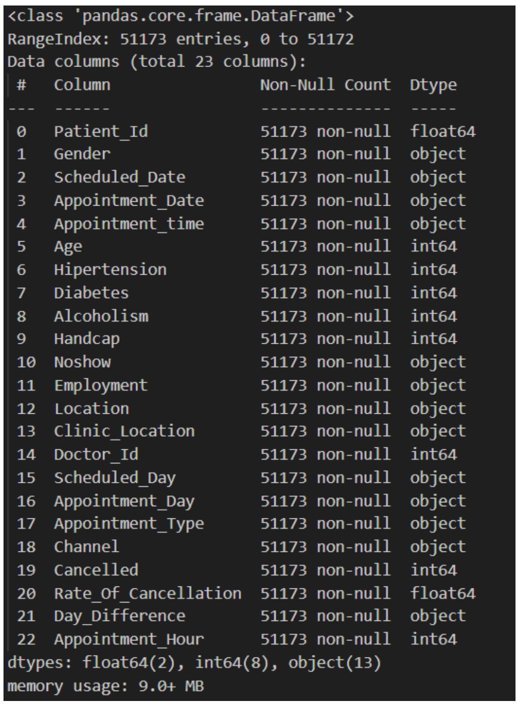
Let us take a cursory glance at our dataset



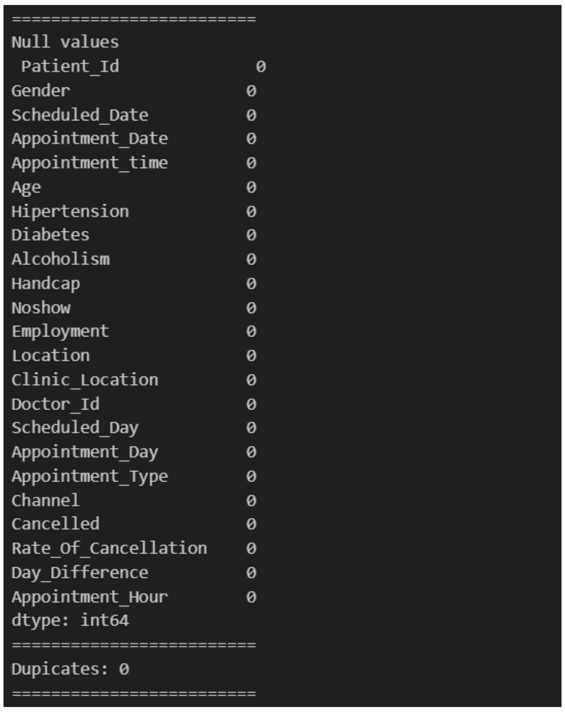
Head:

	Patient_Id	Gender	Scheduled_Date	Appointment_Date	Appointment_time	Age	Hipertension	Diabetes	Alcoholism	Handcap	...
0	299000.0	F	2016-04-29	2016-04-29	22:05:00	62	1	0	0	0	...
1	5590000.0	M	2016-04-29	2016-04-29	21:32:00	56	0	0	0	0	...
2	42600.0	F	2016-04-29	2016-04-29	21:46:00	62	0	0	0	0	...
3	8680.0	F	2016-04-29	2016-04-29	18:08:00	8	0	0	0	0	...
4	88400.0	F	2016-04-29	2016-04-29	20:22:00	56	1	1	0	0	...
5	960000.0	F	2016-04-27	2016-04-29	12:39:00	76	1	0	0	0	...
6	7340000.0	F	2016-04-27	2016-04-29	17:30:00	23	0	0	0	0	...
7	34500.0	F	2016-04-27	2016-04-29	15:50:00	39	0	0	0	0	...
8	564000.0	F	2016-04-29	2016-04-29	17:40:00	21	0	0	0	0	...
9	781000.0	F	2016-04-27	2016-04-29	15:58:00	19	0	0	0	0	...

10 rows × 23 columns



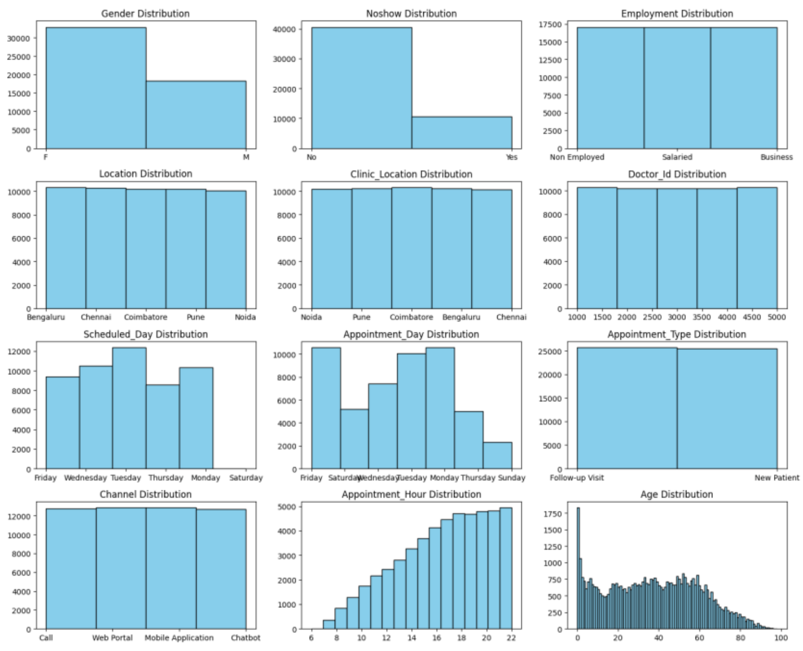
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51173 entries, 0 to 51172
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Patient_Id            51173 non-null  float64
 1   Gender                51173 non-null  object
 2   Scheduled_Date        51173 non-null  object
 3   Appointment_Date      51173 non-null  object
 4   Appointment_time      51173 non-null  object
 5   Age                   51173 non-null  int64
 6   Hipertension          51173 non-null  int64
 7   Diabetes              51173 non-null  int64
 8   Alcoholism            51173 non-null  int64
 9   Handcap               51173 non-null  int64
 10  Noshow                51173 non-null  object
 11  Employment            51173 non-null  object
 12  Location              51173 non-null  object
 13  Clinic_Location       51173 non-null  object
 14  Doctor_Id             51173 non-null  int64
 15  Scheduled_Day         51173 non-null  object
 16  Appointment_Day       51173 non-null  object
 17  Appointment_Type      51173 non-null  object
 18  Channel               51173 non-null  object
 19  Cancelled             51173 non-null  int64
 20  Rate_Of_Cancellation  51173 non-null  float64
 21  Day_Difference        51173 non-null  object
 22  Appointment_Hour      51173 non-null  int64
dtypes: float64(2), int64(8), object(13)
memory usage: 9.0+ MB
```

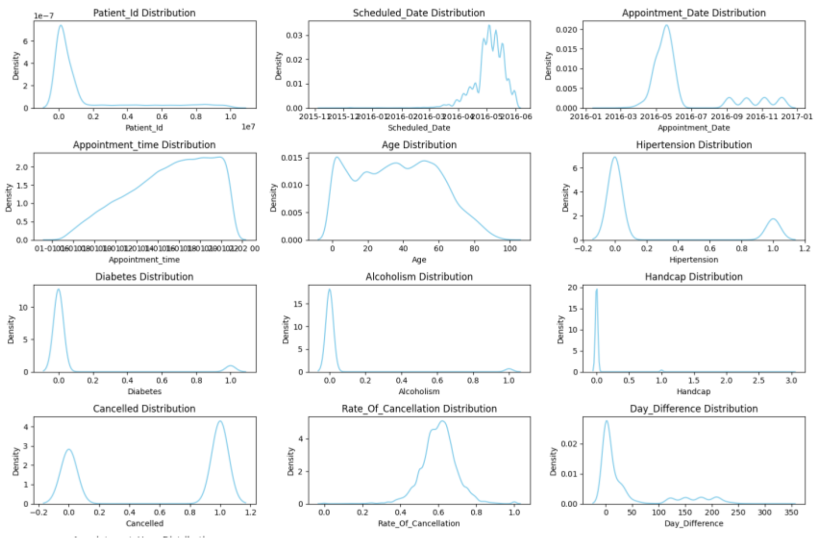


```
=========================
Null values
 Patient_Id            0
Gender                0
Scheduled_Date        0
Appointment_Date      0
Appointment_time      0
Age                   0
Hipertension          0
Diabetes              0
Alcoholism            0
Handcap               0
Noshow                0
Employment            0
Location              0
Clinic_Location       0
Doctor_Id             0
Scheduled_Day         0
Appointment_Day       0
Appointment_Type      0
Channel               0
Cancelled             0
Rate_Of_Cancellation  0
Day_Difference        0
Appointment_Hour      0
dtype: int64
=========================
Dupicates: 0
=========================
```

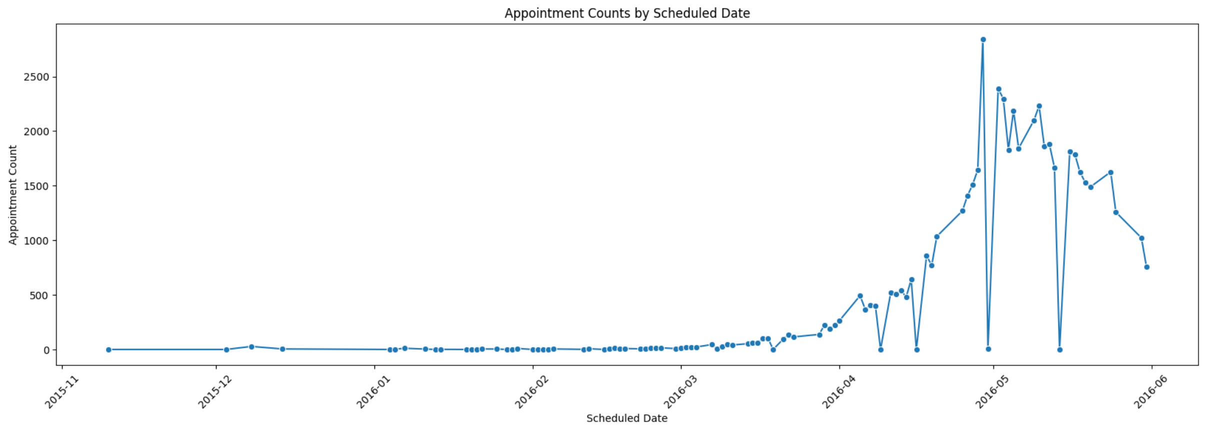
From the above overview of the dataset, we can see that there are 23 columns and 51173 entries. Most columns are non-numerical, non-ordinal and have no duplicates or null entries. Mean age is 36, 20% of patients have hypertension, 6% have diabetes, and 3% have alcoholism.

Exploratory Data Analysis:

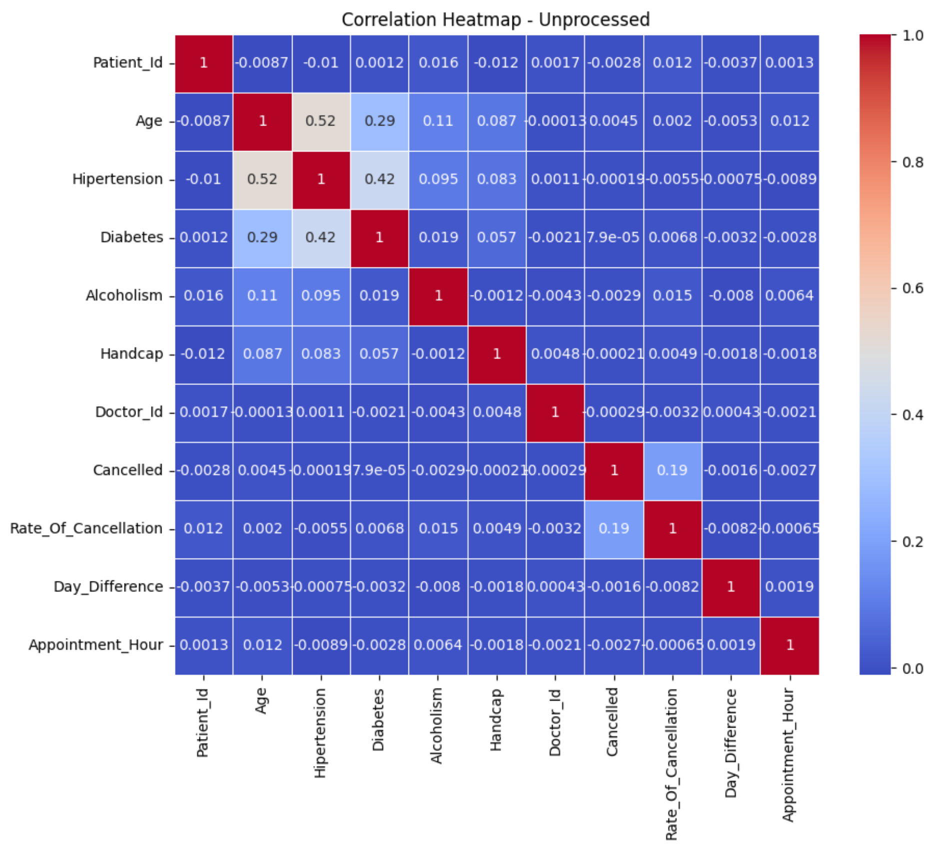




From the above, we can see that Gender is heavily skewed towards female, age has a massive spike in 0, Appointments tend to be higher into the night, there are far more 'no' entries in noshow (i.e., more people who showed up than not). Follow ups are just as much as new visits, there are more appointments on Monday, Tuesday, and Friday. Most people tend to have a rate of cancellation around 0.6, There are more cancelled appointments than non-cancelled ones, and most people have very little difference between their scheduled appointment and actual appointment dates. Very few of our patients have hypertension, and fewer have diabetes. There seems to be a massive spike in appointments at 2016-5, which falls by 2016-7



Here we can notice a gradual increase in appointments as April 2016 begins, with some falls that rise quickly. Towards the end of April, the appointments scheduled see a massive surge to reach its peak, but suddenly falls by the time May 2016 approaches, only to pick right back up. This can be explained by a clinic-wide holiday on May 1st, May Day. The high rate of appointments continues till mid may, where a similar plummet happens. But as June approaches, rates fall to mid-April levels.



The above heatmap shows us that Hypertension and age has a medium correlation of 0.52. Hypertension and diabetes have a relatively weaker correlation of 0.42, and diabetes is weakly correlated with age, at a 0.29 coefficient. There is a weak correlation between rate of cancellation and cancelled appointments, meaning people with higher cancellation rates mildly tend to cancel appointments more. The lack of all variables indicates the need to transform the categorical variables for further analysis.

Preprocessing:

The following preprocessing tasks are done:

1. **Dropping Null Values:** The function first removes any rows that contain missing values. This ensures that the dataset is clean and does not contain any incomplete or unreliable data.
2. **Removing Zero Age Entries:** The function identifies and removes any rows where the age is recorded as zero. This is done because it is unlikely for a patient to have an age of zero, and these entries are likely to be errors or missing data.
3. **Converting Data Types:** The function converts certain columns to the appropriate data types. The 'Patient_Id' column is converted to integer type, while the 'Scheduled_Date', 'Appointment_Date', and 'Appointment_time' columns are converted to datetime format. This allows for easier manipulation and analysis of these columns.

4. **Calculating Day Difference:** The function calculates the number of days between the appointment date and the scheduled date and adds this information as a new column called 'Day_Difference'. This can be useful for analysing the time gap between scheduling and the actual appointment.
5. **Feature Extraction:** The function extracts additional features from the date columns. It extracts the year, month, and day from both the scheduled date and appointment date, as well as the hour and minute from the appointment time. These extracted features can provide more insights into patterns and trends related to appointment scheduling.
6. **Age Grouping:** The function creates a new column called 'Age_Group' by categorizing the age into different groups. The age groups are defined as 'Child' (0-18 years), 'Young Adult' (18-35 years), 'Adult' (35-50 years), and 'Senior' (50-100 years). This grouping allows for easier analysis and comparison based on age ranges.
7. **Combining time:** We create a feature called Combined_Time, which creates a timestamp of the appointment time since the UNIX time epoch (i.e., 01-01-1970, 00:00:00)
8. **Feature Encoding:** The function performs ordinal encoding on the categorical features in the dataframe. Ordinal encoding assigns a numerical value to each category, preserving the ordinal relationship between categories. This encoding is applied to features such as gender, hypertension, diabetes, alcoholism, and others.
9. **Removing Duplicates and Null Values:** After the preprocessing steps, the function removes any duplicate rows and any remaining rows with missing values. This ensures that the final dataset is clean and ready for further analysis.
10. **Correlation identification:** We plot a correlation matrix to identify the most correlating variables (at a threshold of 0.3) to use in model training, since we need not use all columns to train the model. 'Scheduled_Day', 'Scheduled_Month', 'Rate_Of_Cancellation', 'Alcoholism', 'Cancelled', 'Scheduled_Year', 'Appointment_Day', 'Gender', 'Diabetes', 'Hipertension', 'Appointment_Month', and 'Day_Difference' are all the features that show most correlation. Thus, these are suitable features we select for training.

Model Training, Testing and Validation:

- Data Splitting:
  - Features ('X') are defined by dropping the 'Combined_Time' column, and the target variable ('y') is set to the 'Combined_Time' column.
  - The data is split into training and testing sets using the 'train_test_split' function, with 80% for training and 20% for testing.

- Model Initialization:

- Various regression models are initialized, including Linear Regression, Random Forest Regressor, XGBoost Regressor, AdaBoost Regressor, Gradient Boosting Regressor, and Decision Tree Regressor.

- Model Evaluation:

  - Each model is trained on the training set and evaluated on the testing set.
  - Evaluation metrics include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2).
  - Results are printed for each model.

- Best Model Selection:

  - The best model is determined based on the model with the lowest RMSE on the testing set.
    - The best model name and its RMSE are printed.

- Weighted Model Training and Evaluation

  - Class Weights Computation:
  - Class weights for the target column 'y_train' are computed using 'class_weight.compute_sample_weight('balanced', y_train)'.

- Function for Training and Evaluating Models:

  - A function 'train_and_evaluate_model' is defined to train models with sample weights and evaluate them.
  - Sample weights are computed for predictions ('y_pred').

- Weighted Model Evaluation:

  - Each model is trained using the defined function, and results (RMSE, MAE, R2) are printed.
  - The best model is selected and compared against the best model overall

- Hyperparameter Tuning - Grid Search

  - Random Forest Regressor is used with a grid search to find the best hyperparameters.
  - Grid search parameters include the number of estimators, maximum depth, and minimum samples split.
  - The best hyperparameters are used to train a new Random Forest model.

- Hyperparameter Tuning - Random Search

  - Random Forest Regressor is used with a random search to find optimal hyperparameters.
  - Random search parameters are defined using distributions.
  - The best hyperparameters are used to train a new Random Forest model.

- Model Comparison and Selection

- Results of grid search and random search are compared using a DataFrame ('results_df').
- The DataFrame includes metrics for both grid and random search.
- The overall best model is selected based on the lowest RMSE from grid and random search and compared against the best model overall

- Cross-Validation:

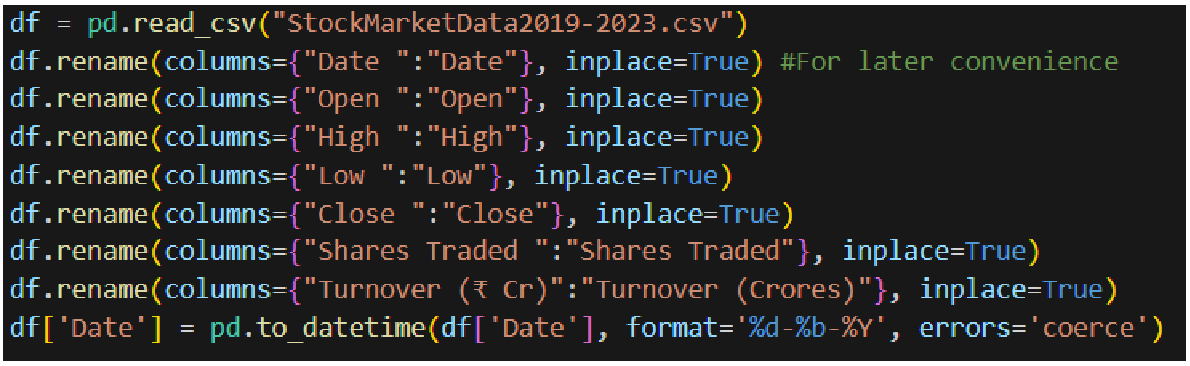
- Models are evaluated using cross-validation with k=2 folds.
- Metrics (RMSE, MAE, R2) are calculated for each model.
- Results for each model are printed.
- The best model is selected and compared against the best model overall.
- In our case, the best model is Gradient Boosting Regressor with RMSE: 0.29.

Task 3: Stock Market Prediction

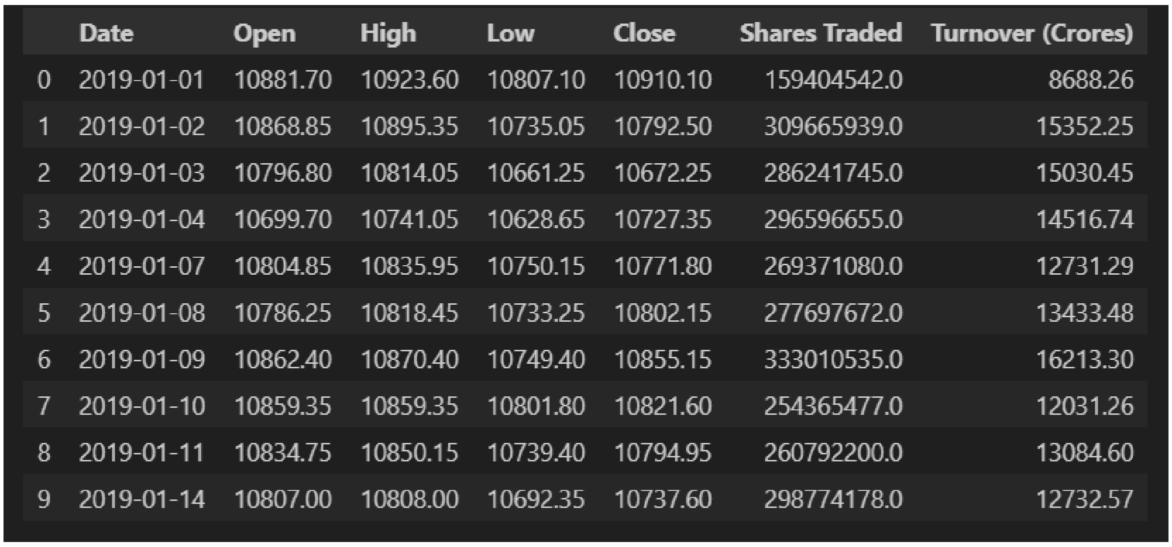
Understanding the dataset:

Context:

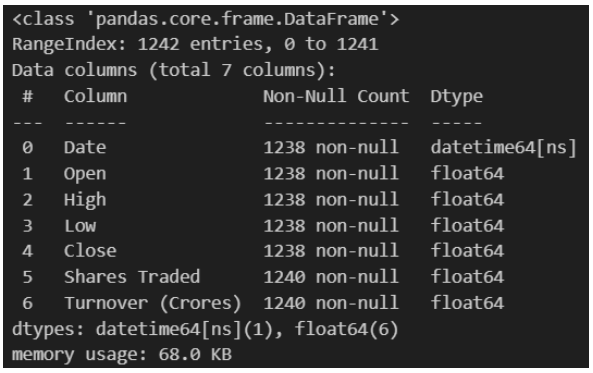
The dataset was provided as various CSV files, each containing the records of a year from 2015 to 2023. They were combined into 3 new CSV files: one from 2015-2019, one from 2019-2023 and one from 2015-2023. The overall structure of the dataset was not altered in this process. The columns were renamed for the sake of convenience, like so



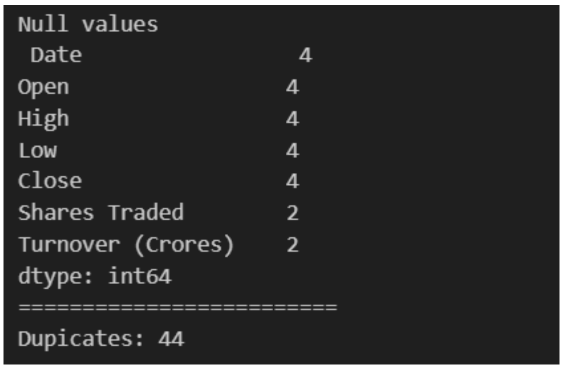
```
df = pd.read_csv("StockMarketData2019-2023.csv")
df.rename(columns={"Date ":"Date"}, inplace=True) #For later convenience
df.rename(columns={"Open ":"Open"}, inplace=True)
df.rename(columns={"High ":"High"}, inplace=True)
df.rename(columns={"Low ":"Low"}, inplace=True)
df.rename(columns={"Close ":"Close"}, inplace=True)
df.rename(columns={"Shares Traded ":"Shares Traded"}, inplace=True)
df.rename(columns={"Turnover (₹ Cr)":"Turnover (Crores)"}, inplace=True)
df['Date'] = pd.to_datetime(df['Date'], format='%d-%b-%Y', errors='coerce')
```



	Date	Open	High	Low	Close	Shares Traded	Turnover (Crores)
0	2019-01-01	10881.70	10923.60	10807.10	10910.10	159404542.0	8688.26
1	2019-01-02	10868.85	10895.35	10735.05	10792.50	309665939.0	15352.25
2	2019-01-03	10796.80	10814.05	10661.25	10672.25	286241745.0	15030.45
3	2019-01-04	10699.70	10741.05	10628.65	10727.35	296596655.0	14516.74
4	2019-01-07	10804.85	10835.95	10750.15	10771.80	269371080.0	12731.29
5	2019-01-08	10786.25	10818.45	10733.25	10802.15	277697672.0	13433.48
6	2019-01-09	10862.40	10870.40	10749.40	10855.15	333010535.0	16213.30
7	2019-01-10	10859.35	10859.35	10801.80	10821.60	254365477.0	12031.26
8	2019-01-11	10834.75	10850.15	10739.40	10794.95	260792200.0	13084.60
9	2019-01-14	10807.00	10808.00	10692.35	10737.60	298774178.0	12732.57



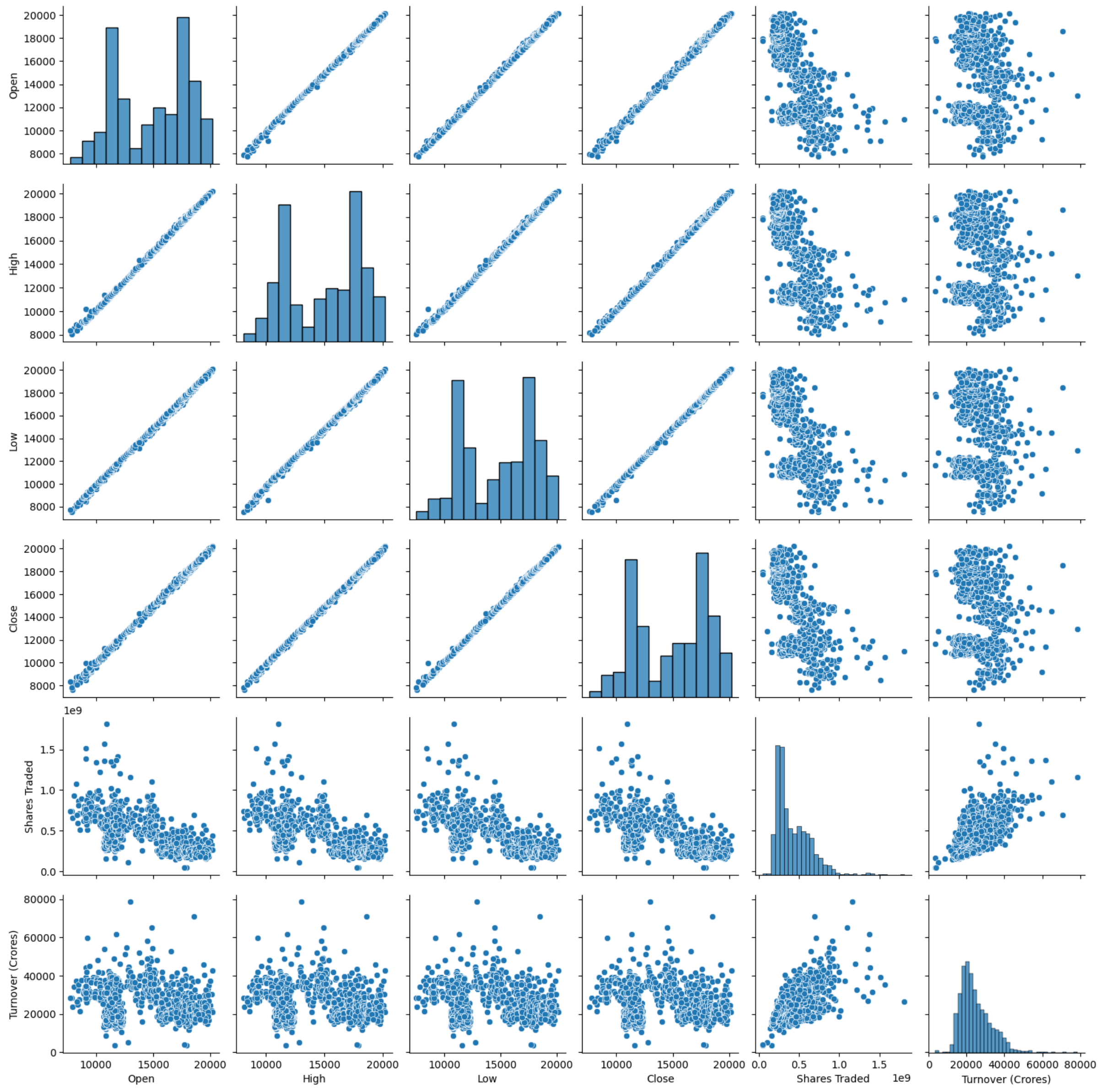
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1242 entries, 0 to 1241
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Date               1238 non-null   datetime64[ns]
 1   Open               1238 non-null   float64
 2   High               1238 non-null   float64
 3   Low                1238 non-null   float64
 4   Close              1238 non-null   float64
 5   Shares Traded      1240 non-null   float64
 6   Turnover (Crores)  1240 non-null   float64
dtypes: datetime64[ns](1), float64(6)
memory usage: 68.0 KB
```



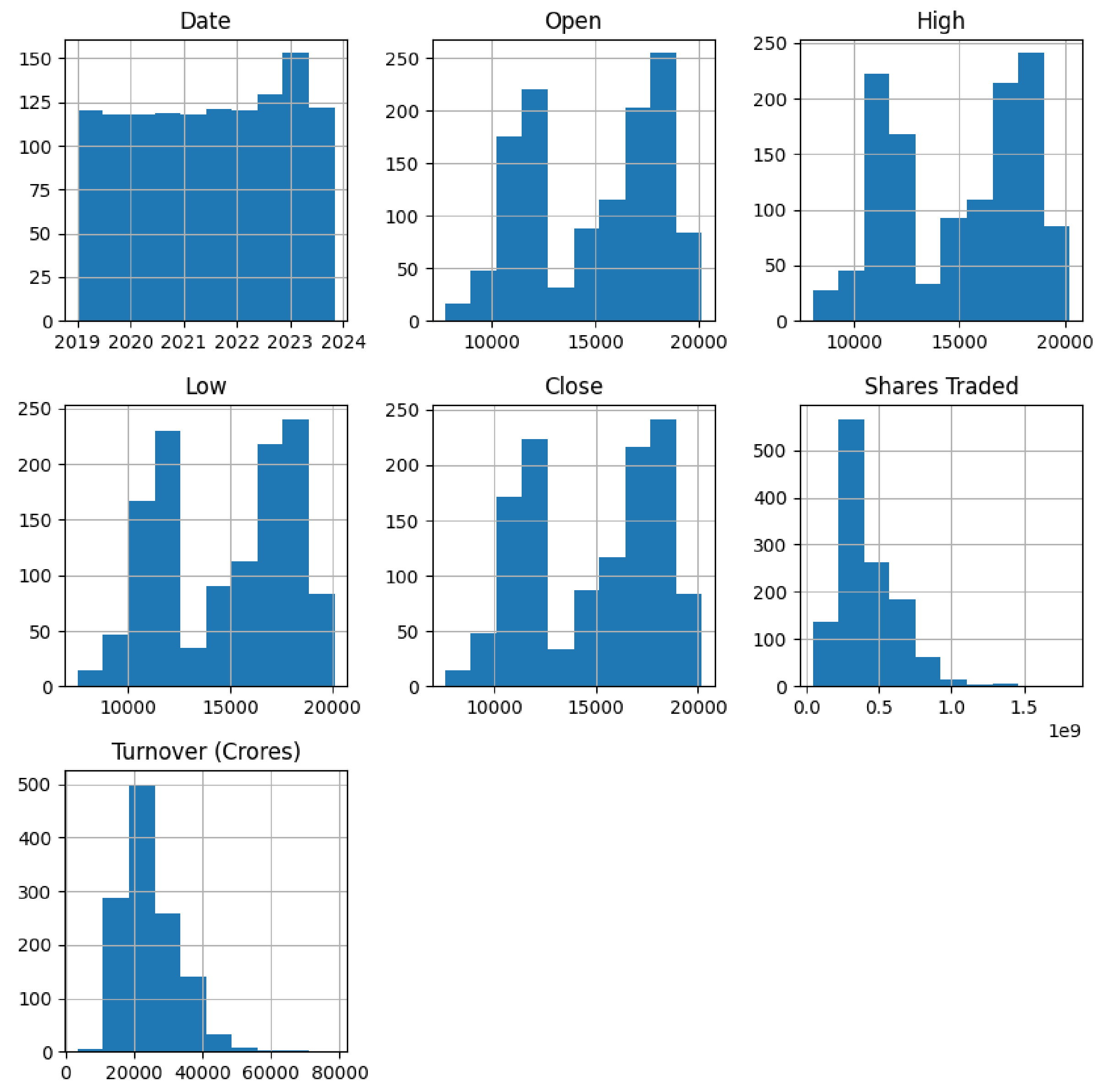
```
Null values
 Date                4
Open                4
High                4
Low                 4
Close               4
Shares Traded       2
Turnover (Crores)   2
dtype: int64
=========================
Dupicates: 44
```

The dataset has 7 columns, namely 'Date', 'Open', 'High', 'Low', 'Close', 'Shares Traded', 'Turnover (Crores)'. All columns have 12422 entries and are of 'float64' datatype, except for the 'Date' feature.

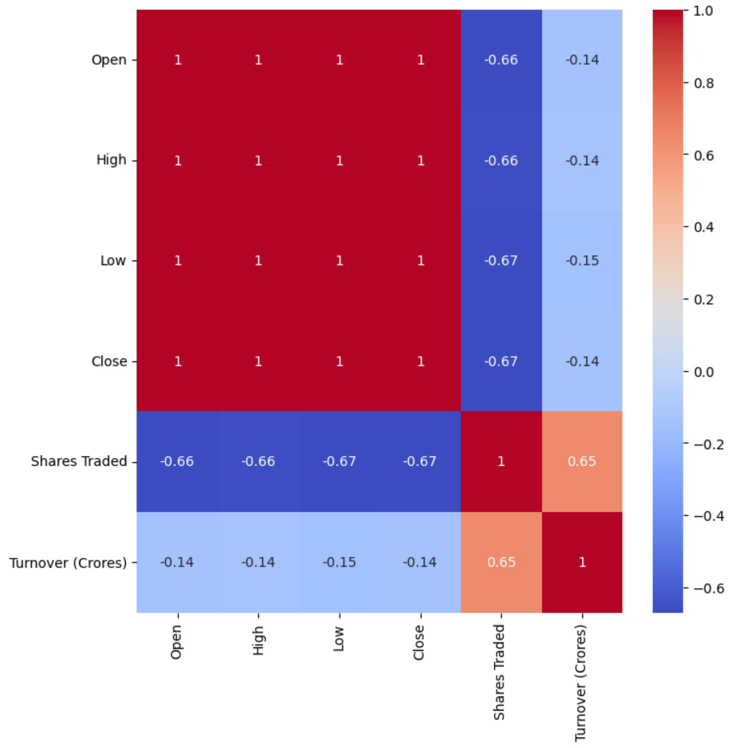
Exploratory Data Analysis:



The pair plot reveals the pairwise relationships between numerical features in the dataset. Diagonal plots depict the distribution of each feature, while off-diagonal plots show scatter plots between feature pairs. Observations indicate strong positive correlations between ‘Open’, ‘High’, ‘Low’, and ‘Close’, indicating a consistent upward trend. A moderate positive correlation exists between ‘Shares Traded’ and ‘Turnover (₹ Cr)’, implying a rise in trading volume alongside an increase in total traded value. The weak positive correlation between Date and other features suggests an overall upward trend in stock prices over the past 5 years.

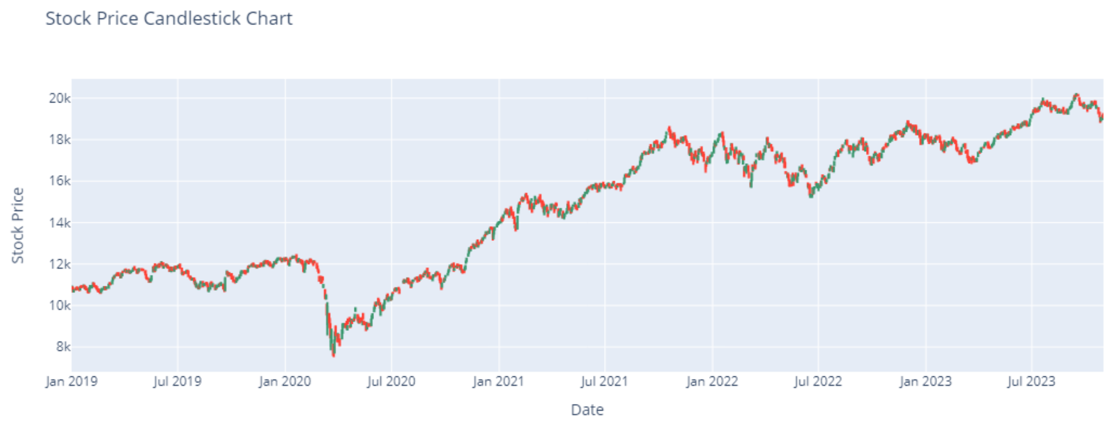


The histograms provide insights into the distribution of data across each column. The ‘Date’ column exhibits a uniform distribution across the entire range, indicating that the stock market data covers the entire 5-year period. The ‘Open’, ‘High’, ‘Low’, and ‘Close’ columns show similar distributions, with a peak in the middle and decreasing values towards the edges, suggesting a generally stable stock price with occasional fluctuations. The ‘Shares Traded’ and ‘Turnover (Crores)’ columns exhibit right-skewed distributions, with a few data points corresponding to very high trading volumes and turnover values, a typical characteristic of stock market data.



	Open	High	Low	Close	Shares Traded	Turnover (Crores)
Open	1	1	1	1	-0.66	-0.14
High	1	1	1	1	-0.66	-0.14
Low	1	1	1	1	-0.67	-0.15
Close	1	1	1	1	-0.67	-0.14
Shares Traded	-0.66	-0.66	-0.67	-0.67	1	0.65
Turnover (Crores)	-0.14	-0.14	-0.15	-0.14	0.65	1

Observations reveal strong positive correlations between ‘Open’, ‘High’, ‘Low’, and ‘Close’, unsurprising given their relationship to the stock price. A moderate positive correlation exists between ‘Shares Traded’ and ‘Turnover (₹ Cr)’, suggesting an increase in trading volume alongside a rise in total traded value. A weak positive correlation between Date and other features indicates an overall upward trend in stock prices over the past 5 years.



This candlestick chart suggests that there has been an overall growth of the stock across the observed time period, with a significant dip around March 2020, which is explained by the start of the Covid-19 pandemic, after that initial fall, recovery has been immense, reaching an all-time high as early as late 2020, after which growth continued to accelerate, with occasional, but temporary drops.

Preprocessing:

These are the tasks done in the preprocessing phase:

1. Cleanup: The function first calls the 'cleanup' function to remove any duplicate rows, null values, and reset the index of the dataframe.
2. Column Renaming: The function checks if certain column names exist in the dataframe ('Date ', 'Open ', 'High ', 'Low ', 'Close ', 'Shares Traded ', 'Turnover (₹ Cr)') and renames them to remove any trailing spaces.
3. Date Conversion: The function converts the 'Date' column to a datetime format using the specified format '%d-%b-%Y'. This ensures that the dates are correctly interpreted as dates rather than strings.
4. Feature Engineering: The function creates additional features based on the 'Date' column. It extracts the year, month, day, day of the week, and day of the year from the date. It also calculates the differences between consecutive values of the 'Open', 'Shares Traded', and 'Turnover (Crores)' columns, and stores them in new columns ('Open-Delta', 'Volume-Delta', 'Turnover-Delta'). These differences represent the changes in these variables from one day to the next.
5. Percentage Change: The function calculates the percentage change of the 'Open', 'Shares Traded', and 'Turnover (Crores)' columns compared to their previous values, and stores them in new columns ('Open-Percent-Change', 'Volume-Percent-Change', 'Turnover-Percent-Change'). These percentage changes represent the relative changes in these variables.
6. Moving Averages: The function calculates the moving averages of the 'Open', 'Shares Traded', and 'Turnover (Crores)' columns using a window size of 5. The moving averages represent the average values of these variables over a specified window of time.
7. Data Cleaning: The function drops the 'Date' column from the dataframe and fills any remaining missing values with 0. It then calls the 'cleanup' function again to perform a final cleanup, removing any duplicate rows and null values.
8. Feature Selection: We plot a correlation matrix to select the most correlated features (i.e., correlation coefficient > 0.3). We can see that 'Volume-Delta', 'Low', 'High', 'Turnover (Crores)', 'Day', 'Open', 'Close', 'Year', 'Shares Traded', 'Month', 'Turnover-Delta', and 'DayofYear' are most correlated. These features are selected to use for model training.

Model Training, Testing and Validation:

- Setup:
  - Utilizes MinMaxScaler from sklearn for data normalization.
  - Splits the dataset into features (X) and target variables (y).

  - Applies MinMax scaling to both X and y.
- Data Splitting:
  - Splits the data into training and testing sets using train_test_split from sklearn.
  - X: all features except 'High', 'Low', 'Close'; Y: 'High', 'Low', 'Close'
  - Prints the shape of X_train, y_train, X_test, and y_test.
- Model Evaluation - Direct Training:
  - Utilizes various regression models (Random Forest, Gradient Boosting, K Nearest neighbours, Linear Regression, XGB, AdaBoost, MultiTaskElasticNet).
  - Trains each model on the training set and evaluates performance on the testing set.
  - Prints metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared for each model.
  - Displays the original and predicted values for the first five samples.
  - Selects the best-performing model based on RMSE.
- Results:
  - Random Forest Regressor - MAE: 0.01, RMSE: 0.01, R-squared: 1.00
  - Gradient Boosting Regressor - MAE: 0.01, RMSE: 0.01, R-squared: 1.00
  - K Nearest Neighbours Regressor - MAE: 0.01, RMSE: 0.02, R-squared: 0.99
  - Linear Regression - MAE: 0.01, RMSE: 0.01, R-squared: 1.00
  - XGB Regressor - MAE: 0.01, RMSE: 0.01, R-squared: 1.00
  - AdaBoost Regressor - MAE: 0.01, RMSE: 0.02, R-squared: 1.00
  - MultiTaskElasticNet - MAE: 0.23, RMSE: 0.26, R-squared: -0.00
- Hyperparameter Tuning:
  - Utilizes GridSearchCV to search for the best hyperparameters for each model.
  - Re-evaluates model performance on the testing set using the tuned hyperparameters.
  - Prints metrics and displays the original and predicted values for the first five samples.
  - Selects the best-performing model based on RMSE after hyperparameter tuning.
- Results:
  - Random Forest Regressor - MAE: 0.01, RMSE: 0.01, R-squared: 1.00
  - Gradient Boosting Regressor - MAE: 0.01, RMSE: 0.01, R-squared: 1.00
  - K neighbours Regressor - MAE: 0.01, RMSE: 0.02, R-squared: 0.99
  - Linear Regression - MAE: 0.01, RMSE: 0.01, R-squared: 1.00
  - XGB Regressor - MAE: 0.01, RMSE: 0.01, R-squared: 1.00
  - AdaBoost Regressor - MAE: 0.02, RMSE: 0.02, R-squared: 0.99
  - MultiTaskElasticNet - MAE: 0.09, RMSE: 0.11, R-squared: 0.82
- Cross Validation:
  - For each model, cross-validation is performed with 2 folds (‘cv=2’).
  - Evaluation metrics include:
  - Root Mean Squared Error (RMSE): A measure of the average deviation of predicted values from actual values.

- Mean Absolute Error (MAE): The average absolute differences between predicted and actual values.
- R-squared (R2): A measure of the proportion of the variance in the dependent variable that is predictable from the independent variables.

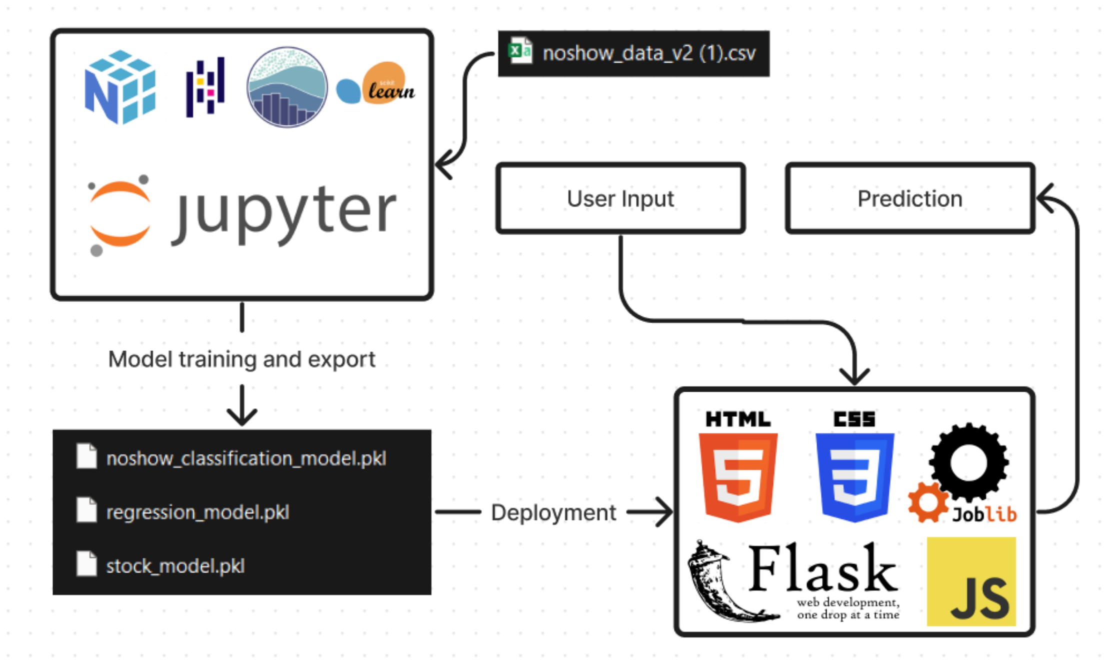
- Results:
  - Linear Regression: RMSE: 0.29, MAE: -0.25, R-squared: 0.99
  - Random Forest Regressor: RMSE: 0.30, MAE: -0.26, R-squared: 0.99
  - XGBoost Regressor: RMSE: 0.30, MAE: -0.26, R-squared: 0.99
  - AdaBoost Regressor: RMSE: 0.51, MAE: -0.44, R-squared: 0.98
  - Gradient Boosting Regressor: RMSE: 0.29, MAE: -0.25, R-squared: 0.99
  - Decision Tree Regressor: RMSE: 0.42, MAE: -0.34, R-squared: 0.99

The best model overall is Linear Regression with RMSE: 0.01

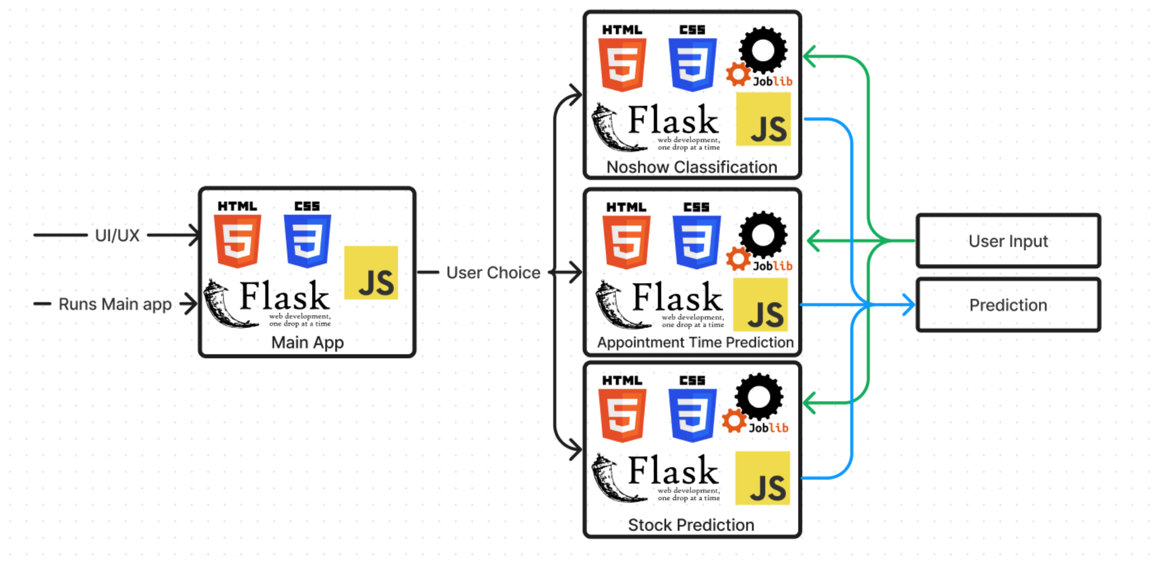
Model Deployment

Overview:

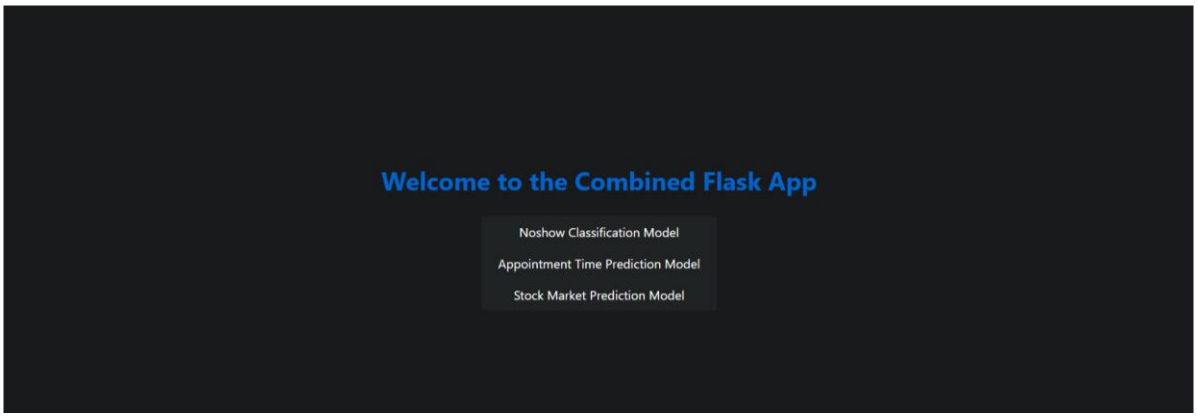
All three models were deployed under a single user interface using flask, HTML and CSS. The models were exported using Joblib.



The project is initialized by running the main_app.py file, which uses the index.html file as a template to render the model selection screen. Upon selecting the model required, it loads the associated flask file, with its relevant HTML, CSS and PKL files. The user prediction is taken by the HTML file and sent to the flask file, where the prediction is calculated and sent back to the HTML file to be rendered to the user.



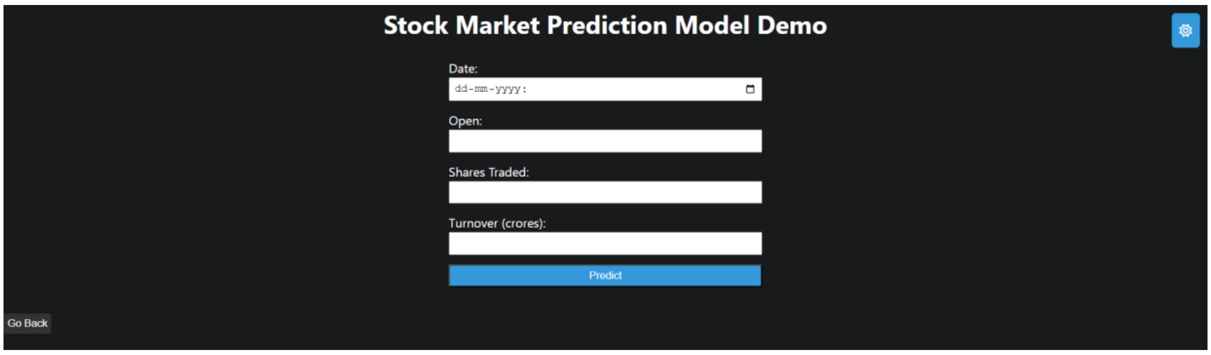
Screenshots:


Welcome to the Combined Flask App

Noshow Classification Model

Appointment Time Prediction Model

Stock Market Prediction Model

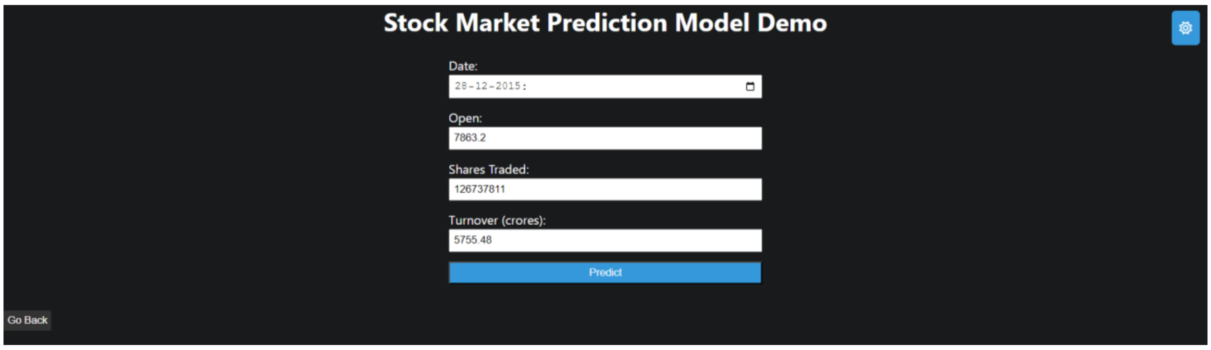

Stock Market Prediction Model Demo

Date: dd-mm-yyyy

Open:

Shares Traded:

Turnover (crores):

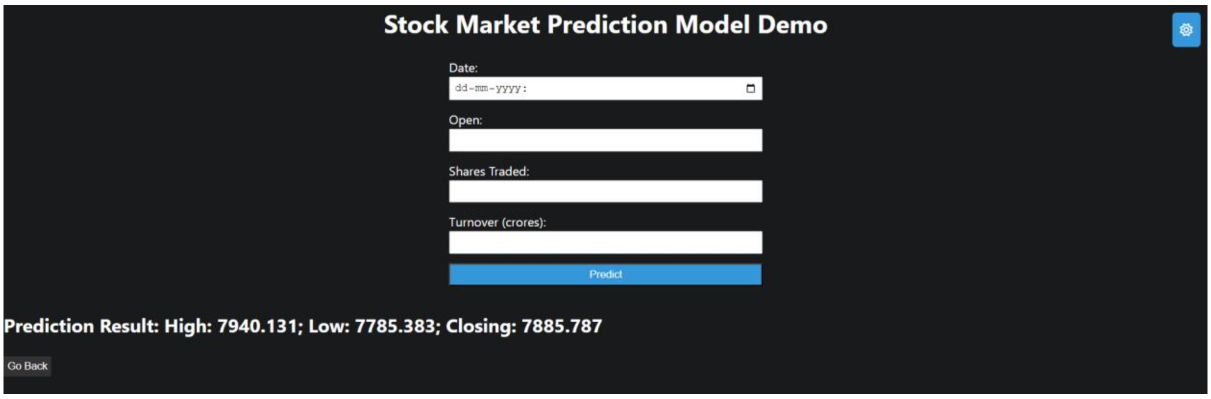

Stock Market Prediction Model Demo

Date: 28-12-2015

Open: 7863.2

Shares Traded: 126737811

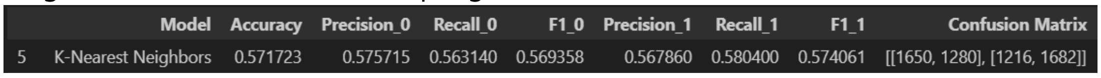
Turnover (crores): 5755.48



Prediction Result: High: 7940.131; Low: 7785.383; Closing: 7885.787

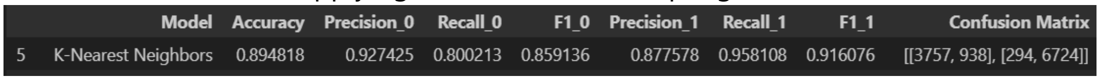
Challenges and Limitations:

- **No Improvements past a threshold:** In the regression tasks, accuracy do not improve past a specific threshold (e.g., 10 points for the stock prediction model). This is likely since the dataset is limited to the trends within the given date range and thus the patterns do not carry over outside of it. For example, the Covid-19 Pandemic drastically increased clinic visits and greatly varied the stock market.
- **Similarity in accuracy among models still led to varying predictions when practically tested.** This is probably due to the inherent differences in the assumptions the algorithms take to predicting the answer, since the difference is not very significant.
- Class imbalances in the dataset (specifically the noshow dataset) led to very inaccurate results in the beginning. However, using resampling techniques greatly helped with the accuracy of the prediction. Here is the result of the K-Nearest neighbours model before resampling



	Model	Accuracy	Precision_0	Recall_0	F1_0	Precision_1	Recall_1	F1_1	Confusion Matrix
5	K-Nearest Neighbors	0.571723	0.575715	0.563140	0.569358	0.567860	0.580400	0.574061	[[1650, 1280], [1216, 1682]]

Here is the result after applying SMOTE-ENN resampling



	Model	Accuracy	Precision_0	Recall_0	F1_0	Precision_1	Recall_1	F1_1	Confusion Matrix
5	K-Nearest Neighbors	0.894818	0.927425	0.800213	0.859136	0.877578	0.958108	0.916076	[[3757, 938], [294, 6724]]

- In the Stock prediction model, unscaled predictions led to drastic levels of error in the predictions. However, after adding a MinMaxScaler, the predictions became much more accurate.
- Because of the drastic changes to the stock market during the Covid 19 pandemic, using data from 2019-2023 drastically varies predicted values. If the models are trained and tested on pre-pandemic data, the error range is halved.

Scope for Improvement:

- Using Deep learning models to predict values can lead to better results
- The user interface is barebones and can use more quality-of-life improvements
- Trying time series prediction to improve validation, however that decreases recall values

- Using SHAP values for tree-based models to interpret the feature importance.

Learnings and Takeaways:

- Machine Learning Workflow:
  - Classification and regression workflows
  - Exploratory Data Analysis (EDA)
  - Data preprocessing and resampling
- Model Deployment and UI/UX:
  - Deployment with Flask
  - Basic UI/UX design
  - Integration of HTML, CSS, and JS
- Data Handling:
  - Data extraction and standardization
  - Identifying crucial features
  - Analysing feature importance
  - Adjusting data for imperfections
- Model Selection:
  - Choosing the right model for the task
- Results Presentation:
  - Presenting results in a user-readable format
- Problem Solving Process:
  - Formulating a structured problem-solving process
  - Understanding and analysing problem-related information
- Inference and Communication:
  - Deriving inferences from information
  - Communicating answers comprehensibly
- Problem Resolution:
  - Approaching difficulties and resolving them
- Information Management:
  - Retaining useful findings
  - Discarding irrelevant information
- Effective Communication:
  - Communicating answers comprehensibly