# Dataset 1 : Apple stock prices prediction

By analysing this dataset I intend to identify the trends and patterns the followed by the apple stock price over the years and predict the future prices. The main goal of this study is to forecast future stock prices by using historical data and a dataset that includes financial indicators like moving averages, open, high, low, and close prices. To find the most accurate and effective way to estimate stock prices, we apply and compare two complex machine learning models—Random Forest and Multiple Linear Regression—in both Python and R environments. In order to manage financial risk, find investment opportunities based on anticipated market movements, and make educated trading decisions, traders, financial analysts, and trading algorithms need to be able to forecast the future.

# Dataset 2 : House Price prediction

In order to develop a predictive model that accurately predicts real estate prices based on various property characteristics, this dataset seeks to use an extensive collection of over 21,000 house sales. This will give buyers, sellers, and real estate professionals important insights for understanding important factors that influence market values and improved investment strategies.

# Dataset 3 : Metro Interstate Traffic Volume Prediction

Through the analysis and forecasting of traffic flow changes due to external factors including weather, time of day, and holidays, this study attempts to improve traffic management by utilising traffic volume data from the I-94 Interstate highway.

|  |  |  |  |
| --- | --- | --- | --- |
| Characteristics | Dataset 1 | Dataset 2 | Dataset 3 |
| Number of dependent and independent variables | Dependent – 1 Independent - 5 | Dependent – 1 Independent - 17 | Dependent – 1 Independent - 8 |
| Number Of Records | 10853 | 21613 | 48204 |
| Data Types of Combination | Float and int are the initial datatypes of variables | Float and int are the initial datatypes of variables | Float and int |
| Summary | I calculated the summary statistics of all the numerical variables that are present. | Find the min, max, mean, median and quantile values. | Find the min, max, mean, median and quantile values. |
| Data Cleaning | There is no missing and duplicate values. | There was no duplicate and Null values | There was missimg colums in ‘holiday’ column which are handled and there was 17 duplicates which are also removed. |
| Data Normalization | I didn’t have to perform normalization as it was already normalized | I have performed normalization using MinMaxScalar | I used MinMaxScalar to perform normalization. |
| Data Balancing and splitting | I split the training and test data in 70:30 ratio. | I split the training and test data in 70:30 ratio. | I split the training and test data in 70:30 ratio. |

Q1: Do the two implementations of identically named techniques perform differently or the same?

I saw minor differences in performance measures like Mean Squared Error, Root Mean Squared Error, and R-squared values in my study using Random Forest and Multiple Linear Regression implemented in both Python and R. Differences in execution time, model correctness, and output interpretation suggest that, despite sharing the same goal of fulfilling statistical functions, the two implementations are not performing in exactly the same way.

**Q2**: If they are performing differently, then what could be the reason? For example, one possible reason maybe they are internally using different algorithms, or implicitly employing some data processing (confirm using the documentation) or maybe some other reason.

The performance variances between the similarly named techniques used in R and Python in my project can be explained by a number of reasons. The way Random Forest and Multiple Linear Regression techniques are implemented varies between Python and R. Differences exist in default settings that influence model behaviour, including how many trees are included in a forest or how statistical outliers are handled.

Furthermore, Python libraries such as Scikit-learn are performance-optimized by C extensions, potentially resulting in speedier execution times than R implementations, which frequently place a higher priority on usability and comprehensive statistical output than on speed. Furthermore, implicit data preparation methods like feature scaling and normalisation may be applied differently in each language, and the effects of these operations on model output might be large. These variances highlight how crucial it is to comprehend the underlying principles of each implementation and how preprocessing and model tuning procedures must be adapted based on the unique features of the environment and tools being utilised.