# STOCK PRICE PREDICTION

## INTRODUCTION:

- Stock Price Prediction using machine learning helps you discover the future value of company stock and other financial assets traded on an exchange. The entire idea of predicting stock prices is to gain significant profits. Predicting how the stock market will perform is a hard task to do. There are other factors involved in the prediction, such as physical and psychological factors, rational and irrational behavior, and so on. All these factors combine to make share prices dynamic and volatile. This makes it very difficult to predict stock prices with high accuracy.

- Stock market prediction has been a significant area of research in Machine Learning. Machine learning algorithms such as regression, classifier, and support vector machine (SVM) help predict the stock market.

- Obtaining an exact value is difficult since it is directly dependent on external factors such as the economic, social, psychological, and political areas, all of which have a substantial impact.

- Stock market prediction is the process of using historical data, statistical models, and various factors to make educated guesses about the future movements of stock prices.

## DETAIL ABOUT DATASET AND COLUMNS USED:

The dataset used is the "Microsoft Lifetime Stocks Dataset" available on Kaggle. This dataset includes a detailed record of the Microsoft Corporation's stock prices from its inception until the date the dataset was last updated.

Here is a breakdown of the columns in the dataset:

1. **Date:** This column stores the date of the stock market data. Each row represents a single day's data. The format is usually YYYY-MM-DD.

2. **Open:** The 'Open' column represents the price of the stock at the beginning of the trading day.

3. **High:** The 'High' column indicates the highest price at which the stock traded during the day.

4. **Low:** The 'Low' column shows the lowest price at which the stock traded during the day.

5. **Close:** The 'Close' column represents the final price at which the stock traded at the end of the trading day.

6. **Adj Close:** The 'Adj Close' or Adjusted Closing Price takes into account factors such as dividends, stock splits, and new stock offerings to give a more accurate picture of a stock's value.

7. **Volume:** The 'Volume' column indicates the number of shares that were traded during the day.

This dataset is a valuable resource for financial analysts and data scientists who are interested in analyzing trends in Microsoft's stock performance over time. It can be used for various tasks such as time-series forecasting, trend analysis, and more.

It's important to note that financial data like this is highly volatile and influenced by a multitude of factors. Therefore, while it can provide useful insights, predictions based on this data should be taken with a grain of caution.

## DATASET LINK:

**https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset**

## LIBRARIES USED AND WAYS TO DOWNLOAD:

There are several Python libraries that can be used for stock price prediction using the "Microsoft Lifetime Stocks Dataset". These libraries provide tools for data processing, feature engineering, model building, and evaluation:

**Pandas:** This library is used for data manipulation and analysis. It provides data structures and functions needed to manipulate structured data .

**Download command:** !pip install pandas

**NumPy:** This library supports large, multi-dimensional arrays and matrices and includes a collection of mathematical functions to operate on these arrays .

**Download command:** !pip install numpy

**Matplotlib and Seaborn**: These libraries are used for data visualization. They provide functions to create a variety of plots for exploratory data analysis .

**Download command:** !pip install matplotlib seaborn

**scikit-learn:** This library provides a range of supervised and unsupervised learning algorithms in Python. It includes various regression, classification and clustering algorithms, and it's also used for splitting data into training and test sets, feature selection, and tuning model parameters .

**Download command:** !pip install scikit-learn

**statsmodels:** This library is used to build statistical models (like ARIMA) and conduct statistical tests and data exploration.

**Download command:** !pip install statsmodels

**Keras and TensorFlow:** These libraries are used for building deep learning models. Keras is a high-level neural networks API, capable of running on top of TensorFlow. They can be used to build sequence models like LSTM (Long Short Term Memory) for time series prediction .

**Download command:** !pip install keras tensorflow

**Prophet:** Developed by Facebook, this library is designed for making forecasts for univariate time series datasets. It's especially good at handling the seasonality in time series data .

**Download command:** !pip install prophet

## TRAIN AND TEST:

To train and test a stock price prediction model using the "Microsoft Lifetime Stocks Dataset" and machine learning libraries,

**Data Preparation:** Load the dataset into a pandas DataFrame. A DataFrame is a two-dimensional data structure that can hold data of different types. It's like a spreadsheet or SQL table, or a dictionary of Series objects

**Data Preprocessing:** This step involves cleaning and transforming the raw data to make it suitable for machine learning. This can include converting the 'Date' column to a datetime format for easier manipulation, creating new features that could be predictive (such as 'open_close_ratio'), and shifting the data "forward" one trading day to set the target variable (the next day's closing price). This step is crucial as it prepares the data for the machine learning model to learn from .

**Feature Selection and Data Splitting:** In this step, you identify the relevant features (independent variables) and the target variable (dependent variable) for the prediction task. Features are the input variables that the model uses to make the prediction. The target variable is what you want to predict. After identifying the features and the target,split the data into a training set and a testing set. The training set is used to train the model, and the testing set is used to evaluate the model's performance on unseen data. This is a common practice in machine learning to prevent overfitting and to get an unbiased evaluation of the model's performance .

**Model Training:** In this step, train a machine learning model on the training data. Training a model involves feeding it the training data and allowing it to learn the relationship between the features and the target variable. This could involve adjusting internal parameters of the model to minimize the difference between its predictions and the actual values. The goal is to make the model's predictions as accurate as possible .

**Model Testing:** Once the model is trained, it's used to make predictions on the testing data. The model's performance is then evaluated by comparing its predictions to the actual values. This provides an indication of how well the model is likely to perform on new, unseen data. . Common metrics for evaluating model performance include accuracy for classification problems, and mean squared error or root mean squared error for regression problems .

## EXPLANATION OF HOW TO PREDICT STOCK PRICES USING RANDOM FOREST ALGORITHM:

**Data Preparation:** Start by loading the dataset and performing any necessary data preprocessing steps, such as handling missing values, encoding categorical variables, and splitting the data into training and testing sets.

**Feature Selection:** Identify the relevant features that can help in predicting stock prices. This can include historical stock prices, trading volume, technical indicators, news sentiment, and other factors that may influence stock prices.

**Train-Test Split:** Split the dataset into a training set and a testing set. The training set will be used to train the random forest model, while the testing set will be used to evaluate its performance.

**Random Forest Algorithm:** The random forest algorithm is an ensemble learning method that combines multiple decision trees to make predictions. Each decision tree is trained on a random subset of the training data and uses a random subset of features. The final prediction is obtained by averaging the predictions of all the individual trees.

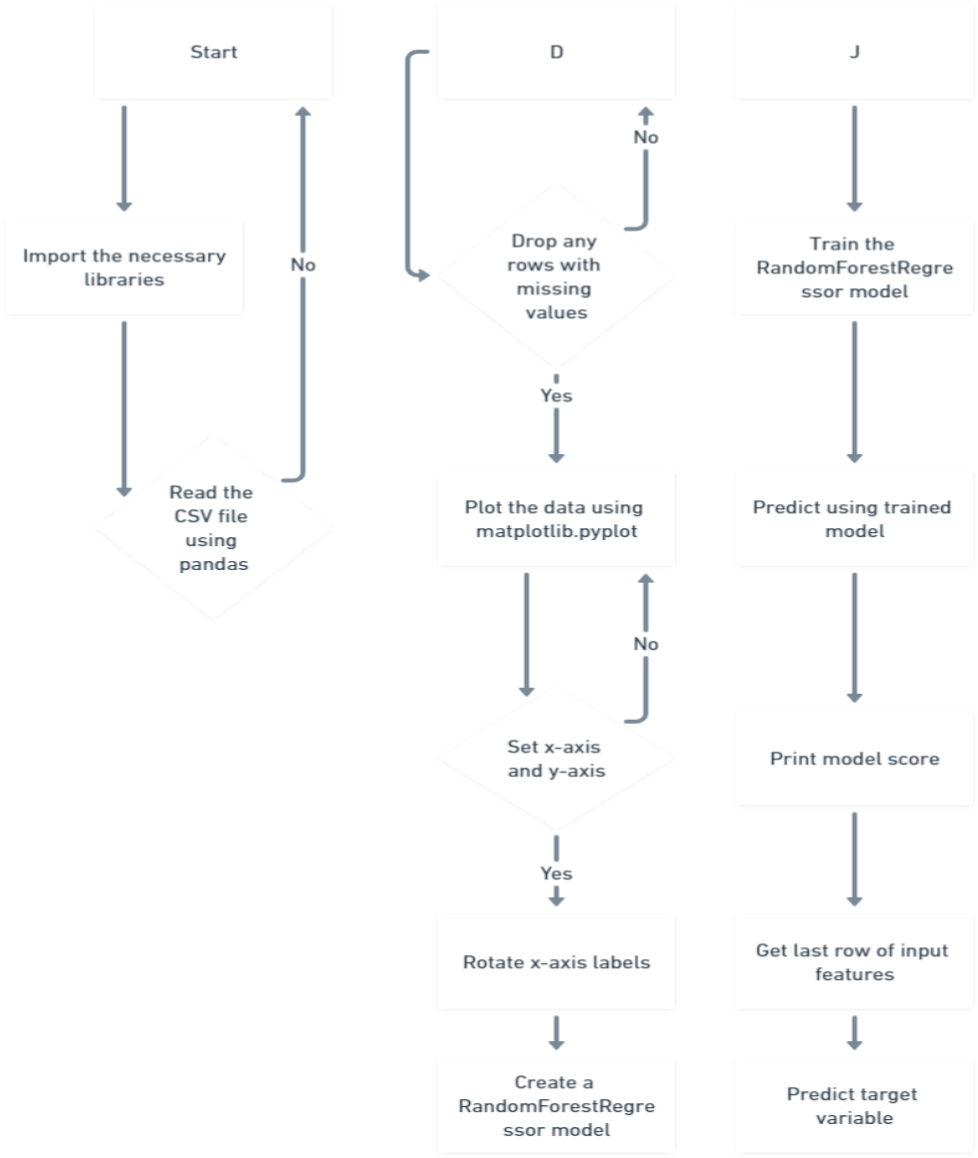
**Training the Random Forest Model:** Use the training set to train the random forest model. The model will learn the patterns and relationships between the input features and the target variable (stock prices) based on the training data.

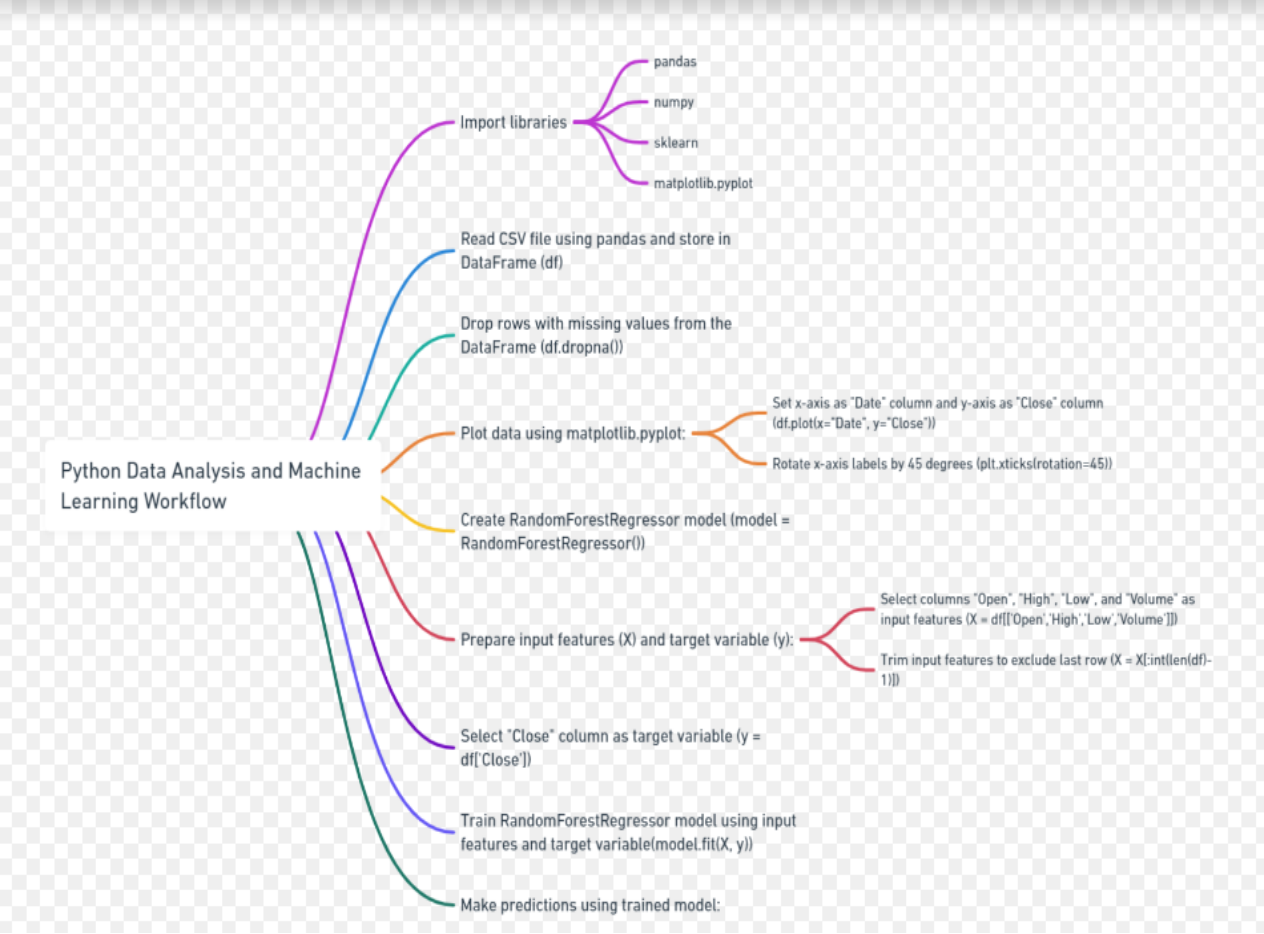
**Prediction:** Once the model is trained, use it to make predictions on the testing set. The model will use the input features from the testing set to predict the corresponding stock prices.

**Evaluation:** Evaluate the performance of the model by comparing the predicted stock prices with the actual stock prices from the testing set. Common evaluation metrics for regression tasks include mean squared error (MSE), root mean squared error (RMSE), and mean absolute error (MAE).

**Fine-tuning and Optimization**: Experiment with different parameters of the random forest algorithm, such as the number of trees in the forest and the maximum depth of each tree, to optimize the model's performance. This can be done using techniques like cross-validation and grid search.

**Deployment and Monitoring:** Once the model is trained and optimized, it can be deployed to make predictions on new, unseen data. It's important to monitor the model's performance over time and retrain or update it as needed to ensure accurate predictions.




Python Data Analysis and Machine Learning Workflow
Import libraries
pandas
numpy
sklearn
matplotlib.pyplot
Read CSV file using pandas and store in DataFrame (df)
Drop rows with missing values from the DataFrame (df.dropna())
Plot data using matplotlib.pyplot:
Set x-axis as "Date" column and y-axis as "Close" column (df.plot(x="Date", y="Close"))
Rotate x-axis labels by 45 degrees (plt.xticks(rotation=45))
Create RandomForestRegressor model (model = RandomForestRegressor())
Prepare input features (X) and target variable (y):
Select columns "Open", "High", "Low", and "Volume" as input features (X = df[['Open','High','Low','Volume']])
Trim input features to exclude last row (X = X[:int(len(df)-1)])
Select "Close" column as target variable (y = df['Close'])
Train RandomForestRegressor model using input features and target variable(model.fit(X, y))
Make predictions using trained model:

## ACCURACY METRICS USED:

Some common metrics used for stock price prediction to test the accuracy are:

**Mean Absolute Error (MAE):**

MAE measures the average absolute difference between predicted stock prices and actual stock prices. A lower MAE indicates a better model fit.

**Mean Squared Error (MSE):**

MSE measures the average squared difference between predicted and actual stock prices. It gives higher weight to large errors. Smaller MSE values indicate a better model fit.

**Root Mean Squared Error (RMSE):**

RMSE is the square root of MSE and provides a measure of the average magnitude of errors. Like MSE, lower RMSE values indicate better model performance.

**Mean Absolute Percentage Error (MAPE):**

MAPE calculates the percentage difference between predicted and actual stock prices. It's useful when you want to understand the prediction accuracy relative to the actual stock price values. It's expressed as a percentage.

**Directional Accuracy (Accuracy or Hit Ratio):**

In classification tasks (e.g., predicting whether the stock will go up or down), directional accuracy measures how often the model's prediction matches the actual direction of the stock movement. It's particularly relevant for binary classification tasks.

**Sharpe Ratio:**

In finance, the Sharpe Ratio evaluates the risk-adjusted return of a portfolio. It's not a direct metric for stock price prediction but can be used in the context of portfolio optimization or risk management.

**Information Coefficient (IC):**

IC is often used in quantitative finance. It measures the correlation between predicted and actual returns. A higher IC indicates better predictions.

**Profit and Loss (P&L):**

In trading and investment applications, P&L metrics measure the cumulative profit or loss generated by a trading strategy based on model predictions.

**F1 Score, Precision, and Recall:**

These classification metrics are relevant when you're predicting stock price movement as a binary classification task. They help you assess the trade-off between precision and recall in your predictions.

**R-squared ($R^2$) or Coefficient of Determination:**

R-squared measures the proportion of the variance in the dependent variable (stock prices) that's predictable from the independent variables (features in your model). A higher $R^2$ indicates a better model fit.

**PROJECT COLAB LINK:**

**https://colab.research.google.com/drive/1gBPPmXh1vNTSTTHNyzMmRSsIZY4NbGbT?usp=sharing**