PROJECT

Machine Learning in Stock Market And Insurance Prediction

GUIDE: PROF TANMAY BASAK

NAME: V. ADITYA RAM
ROLL NO: CH20B114

ML IN STOCK MARKET PREDICTION

1. INTRODUCTION:

What is the Stock Market:

The stock market is the collection of markets where stocks and other securities are bought and sold by investors. Publicly traded companies offer shares of ownership to the public, and those shares can be bought and sold on the stock market. Investors can make money by buying shares of a company at a low price and selling them at a higher price. The stock market is a key component of the global economy, providing businesses with funding for growth and expansion. It is also a popular way for individuals to invest and grow their wealth over time.

Importance of the Stock Market:

Importance	Description
Capital Formation	It provides a source of capital for companies to raise funds for growth and expansion.
Investment Opportunities	Investors can potentially grow their wealth over time by investing in the stock market.
Economic Indicators	The stock market can indicate the overall health of the economy.
Job Creation	Publicly traded companies often create jobs and contribute to the economy's growth.
Corporate Governance	Shareholders can hold companies accountable for their actions and decision-making processes.
Risk Management	Investors can use the stock market to manage their investment risk by diversifying their portfolio.
Market Efficiency	The stock market helps allocate resources efficiently by directing investments to companies with promising prospects.

As the civilization in this world keeps evolving rapidly, the amount of people who trade or invest in the stock market is increasing significantly. This is shown by the observation held by OCBC (Oversea-Chinese Banking Corporation) Securities which denotes that the number of new accounts is increasing significantly (more than doubled), The investor or trader is doing a stock transaction in the stock market. In doing the transaction, people are trying to buy stocks at low prices and selling stocks at high prices in order to get considerable profit. However, stock investment is very high-risk.

Below figure shows the stocks transaction volumes per quarter year in the United States throughout 2016-2020. From the figure, we can see that the transaction volumes are increasing significantly from around 9 hundred million transactions to around 1.3 billion transactions. The trend will continue to go up in the future. Thus, we believe more research is required for this topic.

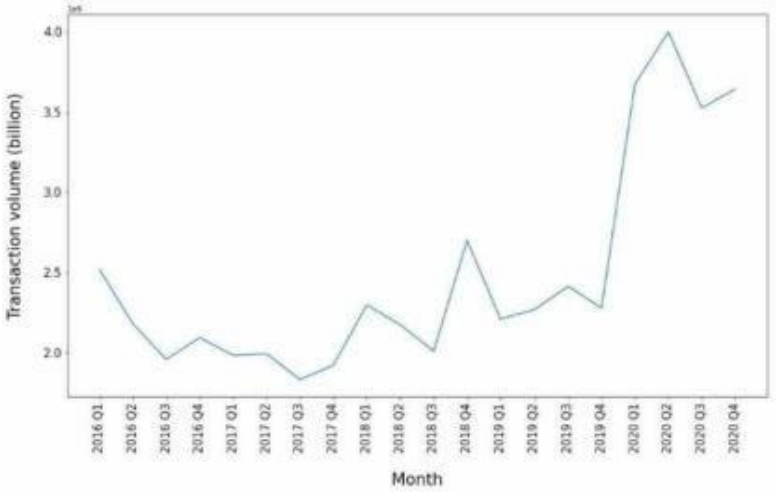


Figure 1.1 Transaction volumes per month

Many people consider investing or trading in the Stock Market is the same as gambling, while actually they are very different. Doing a stock transaction requires judgments and analysis on many aspects, such as the value of the company, the growing trends, etc. However, doing those kinds of things is not an easy task, especially for novice traders or investors.

Responding to that trend, many studies have tried to develop AI models which can predict the future stock price with the intention to lower the risk for people to invest in stock. One AI field which can help us in predicting the future is Machine Learning (ML) in which we train the data to produce a model and then analyse the future based on the model. Therefore, with the help of machine learning we can hopefully determine those aspects which will affect our decision in buying or selling stocks.

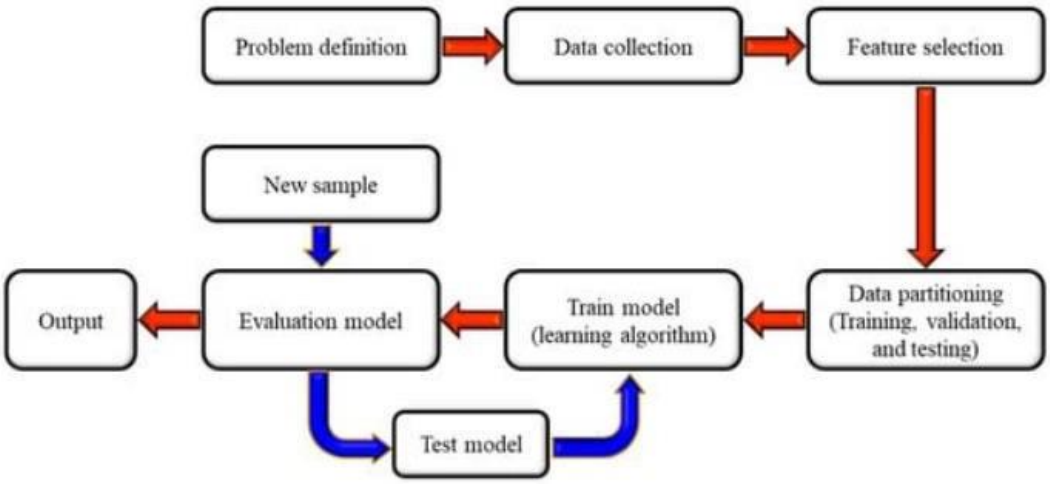
2. METHODOLOGY:

Broadly, stock market analysis is divided into two parts – Fundamental Analysis and Technical Analysis.

- Fundamental Analysis involves analysing the company's future profitability on the basis of its current business environment and financial performance.
- Technical Analysis, on the other hand, includes reading the charts and using statistical figures to identify the trends in the stock market.

My focus will be on the technical analysis part. I will be using a dataset from Quandl (you can find historical data for various stocks here) and for this particular project, I have used the data for Tata Global Beverages.

Various machine learning techniques such as moving averages, linear regression, K-Nearest Neighbours, Auto ARIMA, Prophet, and LSTM models are employed to forecast the daily closing price of stocks.

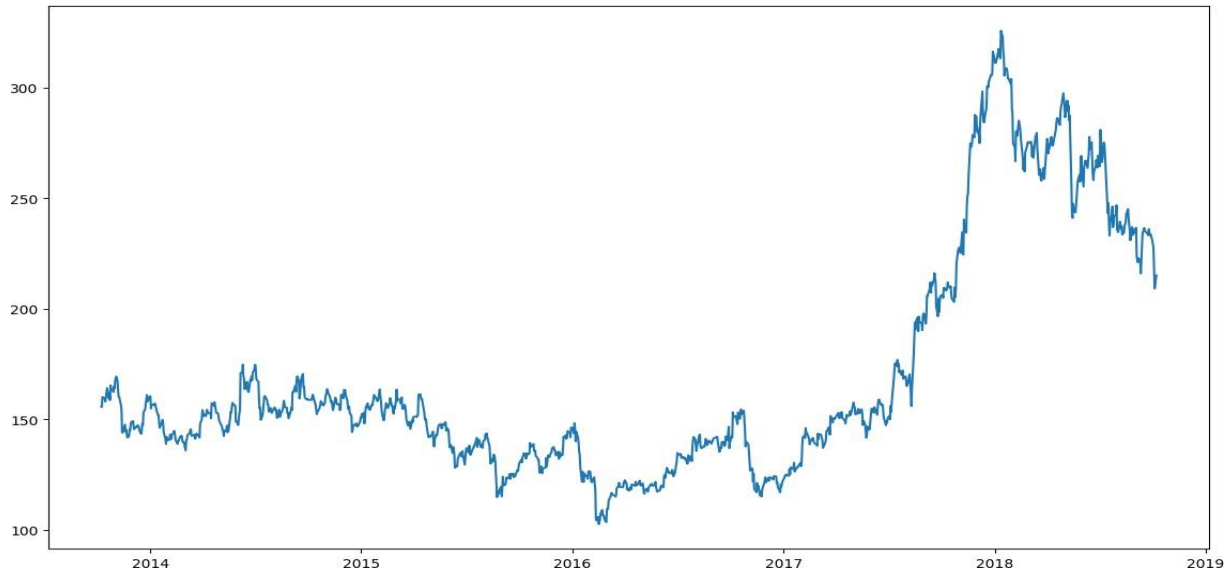


2.1 Data Collection & Analysis:

There are multiple variables in the dataset – date, open, high, low, last, close, total_trade_quantity, and turnover.

- The columns *Open* and *Close* represent the starting and final price at which the stock is traded on a particular day.
- *High*, *Low* and *Last* represent the maximum, minimum, and last price of the share for the day.
- *Total Trade Quantity* is the number of shares bought or sold in the day and *Turnover (Lacs)* is the turnover of the particular company on a given date.

The profit or loss calculation is usually determined by the closing price of a stock for the day; hence we will consider the closing price as the target variable. Let's plot the target variable to understand how it's shaping up in our data:



2.2 Train-Validation Split:

In machine learning, the 85-15 split holds significance by allocating 85% of the data for training and reserving 15% for validation. This division ensures a substantial amount of data for the model to learn from while creating a controlled environment for evaluating its performance.

The larger training subset, encompassing 85% of the dataset, provides a wealth of information for the model to comprehend intricate patterns within the data. Conversely, the 15% validation set serves as a platform to fine-tune model parameters and identify indications of overfitting.

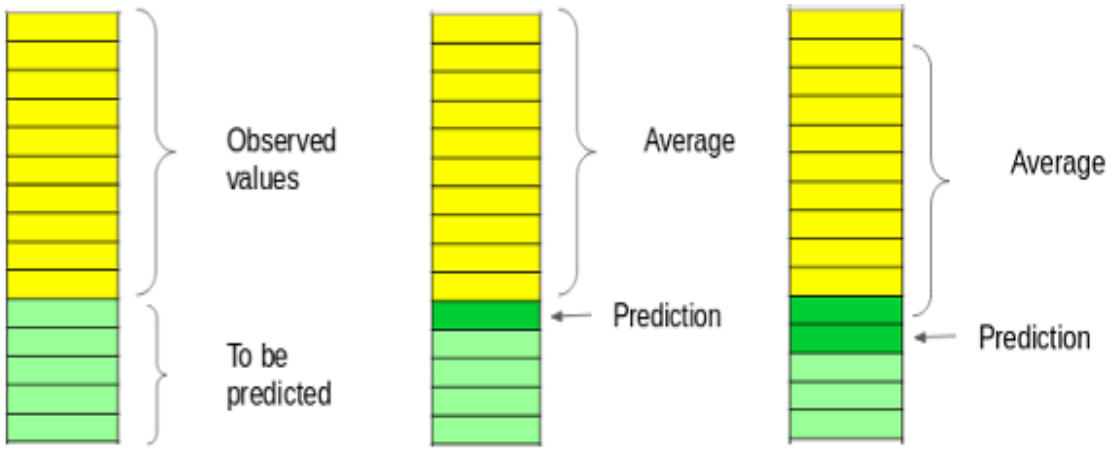
Overfitting, a common challenge in model building, occurs when the model excessively tailors itself to the training data. All models are trained using the train data set and are used to predict the future stock price closing prices, which are basically dates present in validation data sets.

2.3 Model Training & Evaluation:

Model 1: Moving Average

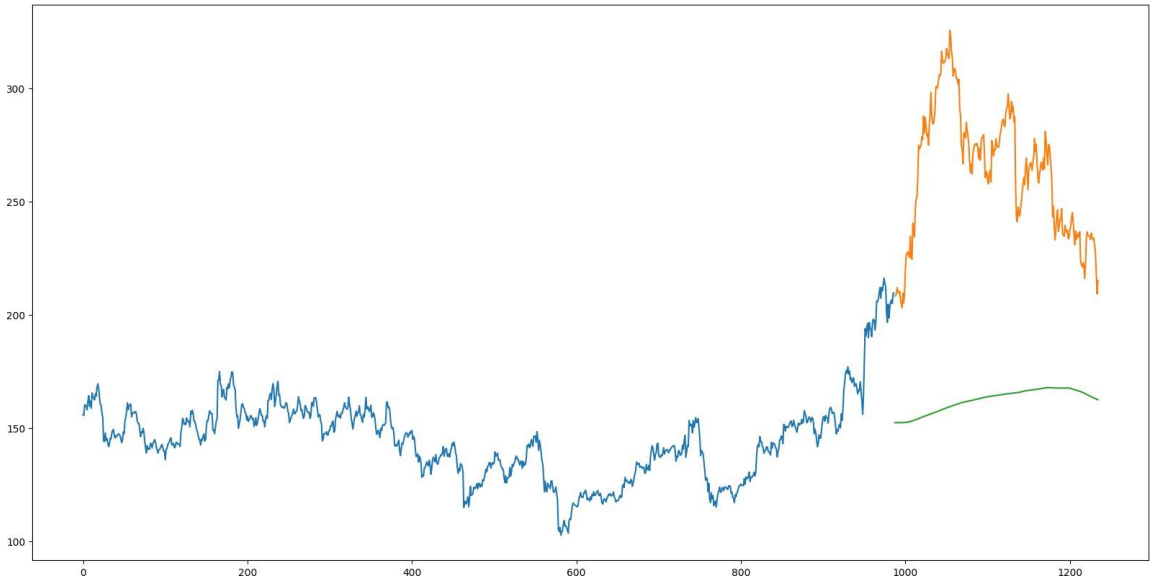
'Average' is easily one of the most common things we use in our day-to-day lives. For instance, calculating the average marks to determine overall performance, or finding the average temperature of the past few days to get an idea about today's temperature – these all are routine tasks we do on a regular basis. So, this is a good starting point to use on our dataset for making predictions.

The predicted closing price for each day will be the average of a set of previously observed values. Instead of using the simple average, we will be using the moving average technique which uses the latest set of values for each prediction. In other words, for each subsequent step, the predicted values are taken into consideration while removing the oldest observed value from the set. Here is a simple figure that will help you understand this with more clarity.



After creating a new dataset comprising of the date and the closing price of the stock, I have made future predictions of the stock closing price by calculating moving averages of past 248-day values. After making predictions, the predicted validation values are compared with original values and I got a Root Mean Squared Error (RMSE) value of **104.51415465984348**

Plot of the predicted values along with the actual values:



Inference:

The RMSE value is close to 105 but the results (green plot) are not very promising (as you can observe from the plot). The predicted values are of the same range as the observed values in the train set (there is an increasing trend initially and then a slow decrease).

Model 2: Linear Regression

The most basic machine learning algorithm that can be implemented on this data is linear regression. The linear regression model returns an equation that determines the relationship between the independent variables and the dependent variable.

The equation for linear regression can be written as:

$$Y = \theta_1 X_1 + \theta_2 X_2 + \dots \theta_n X_n$$

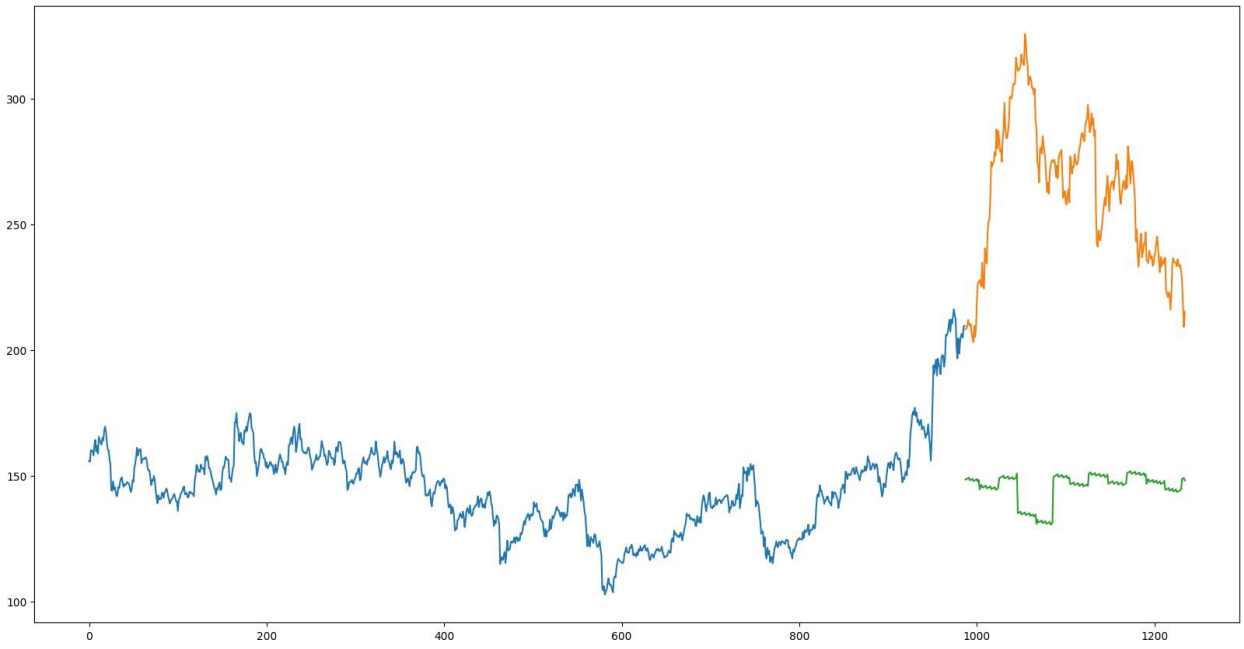
Here, x_1, x_2,…. xn represent the independent variables while the coefficients θ_1, θ_2, …. θ_n represent the weights.

For my problem statement, I do not have a set of independent variables. I have only the dates instead. So, I have used the date column to extract features like – day, month, year, mon/fri etc. and then fit a linear regression model.

I have added different features into the data set like 'Year', 'Month', 'Week', 'Day', 'Dayofweek', 'Dayofyear', 'Is_month_end', 'Is_month_start', 'Is_quarter_end', 'Is_quarter_start', 'Is_year_end', and 'Is_year_start', using fastai python module. On top of that, the first and last days of the week could potentially affect the closing price of the stock far more than the other days. So, I have created a feature that identifies whether a given day is Monday/Friday or Tuesday/Wednesday/Thursday. If the day of week is equal to 0 or 4, the column value will be 1, otherwise 0.

After training the model, on the train data set date points, I have predicted into the future and evaluated the predictions with the validation data set. I obtained an RMSE value of `121.16291596522818`

Plot of the predicted values along with the actual values:



Inference:

Linear regression is a simple technique and quite easy to interpret, but there are afew obvious disadvantages. One problem in using regression algorithms is that the modeloverfits to the date and month column. Instead of taking into account the previous valuesfrom the point of prediction, the model will consider the value from the same *date* a month ago, or the same *date/month* a year ago.

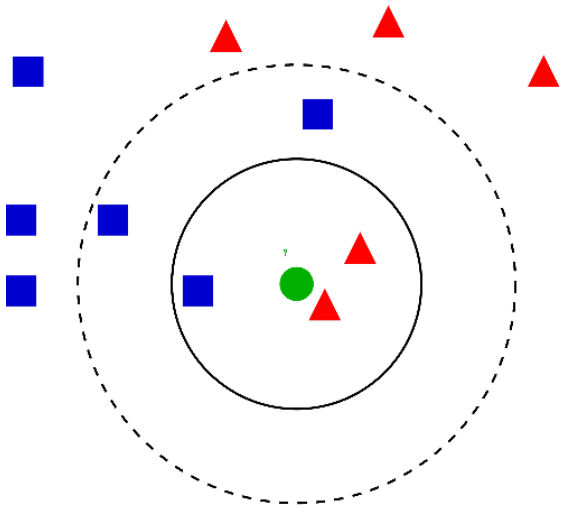
As seen from the plot above, for January 2016 and January 2017, there was a dropin the stock price. The model has predicted the same for January 2018.

Model 3: K-Nearest Neighbours

The K-Nearest Neighbour (KNN) algorithm is a popular machine learning technique used for classification and regression tasks. It relies on the idea that similar data points tend to have similar labels or values.

During the training phase, the KNN algorithm stores the entire training dataset as a reference. When making predictions, it calculates the distance between the input data point and all the training examples, using a chosen distance metric such as Euclidean distance.

Next, the algorithm identifies the K nearest neighbours to the input data point based on their distances. In the case of classification, the algorithm assigns the most common class label among the K neighbours as the predicted label for the input data point. For regression, it calculates the average or weighted average of the target values of the K neighbours to predict the value for the input data point.

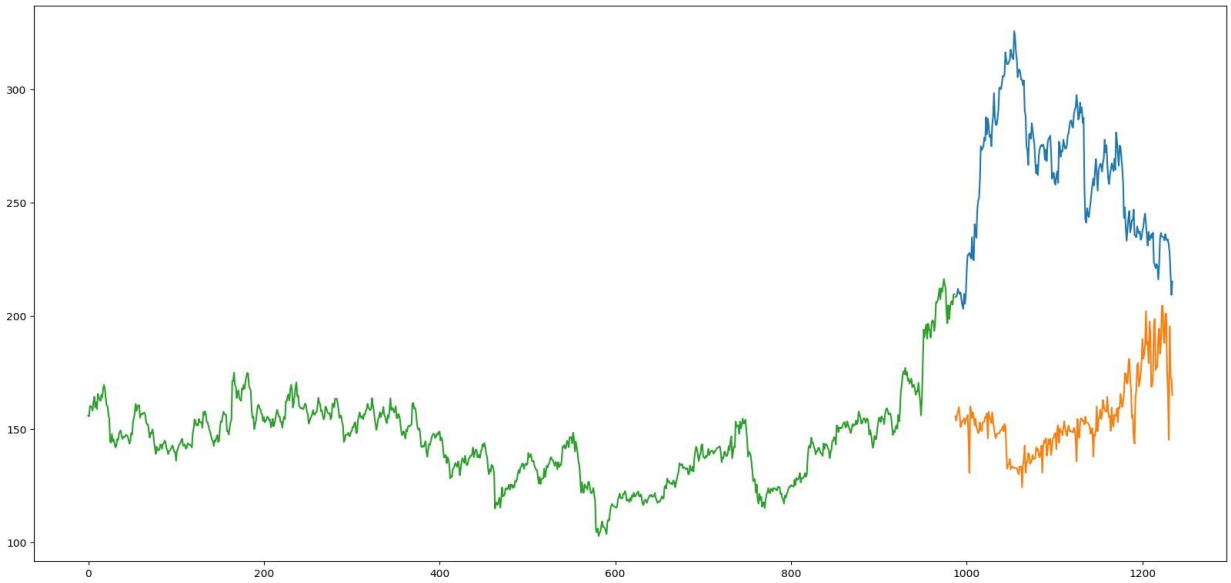


Model's performance can be affected by the choice of K and the distance metric, so careful parameter tuning is necessary for optimal results.

In my model training, I have used GridSearchCV to calculate the optimum parameter value of K in the K-Nearest Neighbour Algorithm. It was validated using GridSearchCV on K parameter values 2,3,4,5,6,7,8,9 and the best parameter value obtained is k=5. When made predictions using the model with k=5, the calculated RMSE values on the predicted closing stock price market value is `115.17086550026721`

```
#using gridsearch to find the best parameter
params = {'n_neighbors':[2,3,4,5,6,7,8,9]}
knn = neighbors.KNeighborsRegressor()
model2 = GridSearchCV(knn, params, cv=5)

#fit the model2 and make predictions
model2.fit(x_train_sca,y_train_1)
preds3 = model2.predict(x_valid_sca)
```

Plot of the predicted values along with the actual values:



Inference:

The RMSE value is almost similar to the linear regression model and the plot shows the same pattern. Like linear regression, KNN also identified a drop in January 2018 since that has been the pattern for the past years. We can safely say that regression algorithms have not performed well on this dataset.

Model 4: Auto Arima

AutoARIMA, or Automatic AutoRegressive Integrated Moving Average, is a time series forecasting method that automatically selects the optimal parameters for an ARIMA model. ARIMA models are widely used for time series analysis, capturing trends, seasonality, and temporal dependencies.

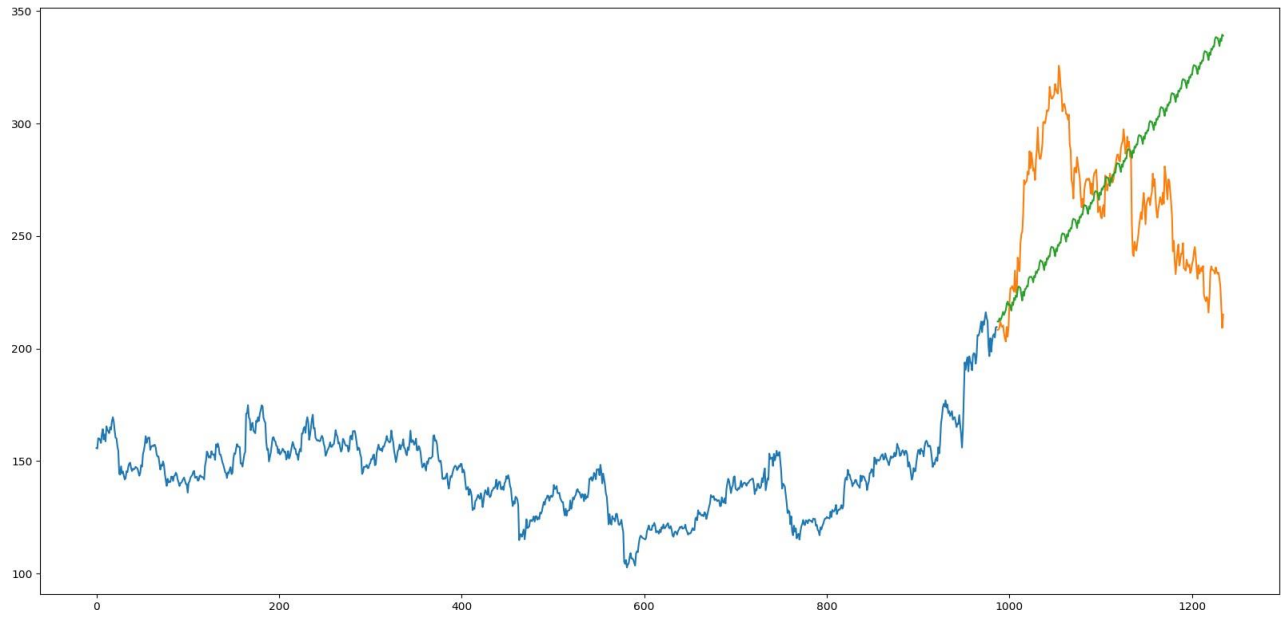
AutoARIMA employs a systematic approach to find the best combination of parameters (p, d, q) for the ARIMA model. It conducts an exhaustive search over multiple potential configurations, evaluating each based on a chosen criterion (often AIC or BIC) to determine the model's goodness of fit.

By automating the selection process, AutoARIMA reduces the need for manual tuning, making time series forecasting more accessible to users without extensive expertise in time series modelling. This tool is particularly useful when dealing with diverse datasets and varying patterns in time series data.

Training the model using by setting the hyper parameter values start_p=1, start_q=1, max_p=3, max_q=3, m=12, start_P=0, seasonal=True,d=1, Dtrace=True,error_action='ignore',suppress_warnings=True

The obtained RMSE value on the validation data set date points is: **54.543229574071674**

Plot of the predicted values along with the actual values:



Inference:

As its evident from the plot, the model has captured a trend in the series, but does not focus on the seasonal part. An auto ARIMA model uses past data to understand the pattern in the time series. Using these values, the model captured an increasing trend in the series. Although the predictions using this technique are far better than that of the previously implemented machine learning models, these predictions are still not close to the real values.

Model 5: Prophet (By Facebook)

There are a number of time series techniques that can be implemented on the stock prediction dataset, but most of these techniques require a lot of data preprocessing before fitting the model. Prophet, designed and pioneered by Facebook, is a time series forecasting library that requires no data preprocessing and is extremely simple to implement.

The input for Prophet is a data frame with two columns: date and target (ds and y). Prophet tries to capture the seasonality in the past data and works well when the dataset is large.

After training the model, on the train data set date points, I have predicted into the future and evaluated the predictions with the validation data set. I obtained an RMSE value of `57.320798699081635`

Plot of the predicted values along with the actual values:

Inference:

Prophet (like most time series forecasting techniques) tries to capture the trend and seasonality from past data. This model usually performs well on time series datasets, but fails to live up to its reputation in this case. As it turns out, stock prices do not have a particular trend or seasonality. It highly depends on what is currently going on in the market and thus the prices rise and fall. Hence forecasting techniques like ARIMA, SARIMA and Prophet would not show good results for this particular problem.

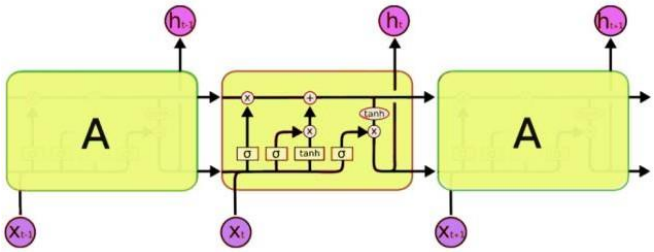
Model 6: Long-Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) designed to address the vanishing gradient problem in traditional RNNs. LSTMs are adept at capturing long-range dependencies and sequential patterns in time series data. With specialized memory cells and gating mechanisms, LSTMs can selectively retain and forget information over extended sequences, making them effective for processing and predicting time-dependent data.

The architecture's ability to handle context information for prolonged periods facilitates tasks such as speech recognition, language modelling, and time series forecasting. LSTMs have become a cornerstone in deep learning applications for sequential data, demonstrating superior performance in capturing intricate temporal relationships.

LSTM has three gates:

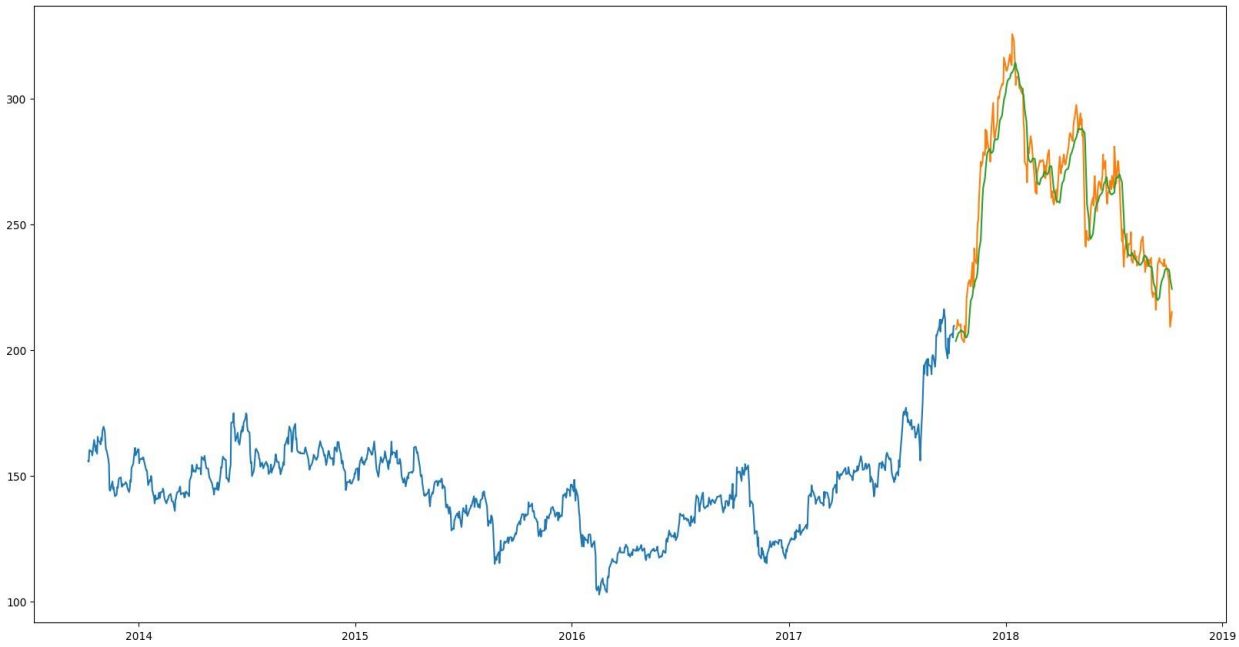
- **The input gate:** The input gate adds information to the cell state
- **The forget gate:** It removes the information that is no longer required by the model
- **The output gate:** Output Gate at LSTM selects the information to be shown as output



The data-preprocessing essentially structures the data to enable the LSTM model to learn from sequences of historical data (x_train)-60 past points and predict the subsequent single data point (y_train). The model learns patterns and dependencies within these sequences to make predictions for unseen data (validation data).

Model is built using the Keras Sequential, comprising two LSTM layers with 50 units each followed by a Dense layer with one unit, which predicts a single value. The LSTM layers are structured to receive input sequences shaped by the data's time steps and features. The model is compiled using the mean squared error as the loss function and the Adam optimizer. The training data, x_train (sequences of historical data), and y_train (corresponding target values) are fitted to the model for one epoch with a batch size of 1, updating the model's weights based on the observed loss, as indicated by the 'verbose=2'. The RMSE values on the predicted validation set using this model is `9.816142776457795`

Plot of the predicted values along with the actual values:



CONCLUSION:

In this study, we explored the application of machine learning algorithms for predicting stock prices. Our analysis encompassed a range of models, trained on historical market data to forecast future stock movements. The models showed varying degrees of predictive accuracy, demonstrating promising results in forecasting stock prices. Feature engineering played a crucial role in enhancing model performance, indicating the significance of selecting and engineering relevant features for predictive tasks.

Market volatility and unexpected events posed challenges, impacting the models' ability to accurately predict stock movements during periods of instability. The models exhibited limitations in accurately predicting extreme market fluctuations and abrupt shifts caused by unforeseen events or news. Incorporating sentiment analysis from news and social media could enhance the models' ability to capture market sentiment and react to breaking news. Exploring advanced deep learning architectures or ensemble techniques may further improve predictive accuracy. Continual refinement and adaptation of the models to evolving market conditions would be essential for sustained performance.

Among the various models constructed and assessed, the **LSTM** Recurrent Neural Network exhibited the lowest RMSE (Root Mean Square Error) value and proved highly proficient in accurately predicting future stock market Closing Prices. In conclusion, while our machine learning models showed promise in predicting stock prices, there remains ample room for refinement and expansion. The dynamic nature of financial markets and the influence of external factors underscore the ongoing challenge in achieving consistently accurate predictions. Nevertheless, this study contributes to the ongoing dialogue on leveraging machine learning in financial forecasting and lays the groundwork for future advancements in this domain.

ML In Insurance Prediction

1. Introduction:

Machine Learning (ML) is all about programming the unprogrammable. For example, if you want to predict an insurance price, ML helps to predict the price. An insurance price depends on various features such as age, type of coverage, amount of coverage needed, gender, body mass index (BMI), region, and other special factors like smoking to determine the price of the insurance. Insurance is a contract whereby an individual obtains financial protection against losses from an insurance company against the risks of financial losses as mentioned in the insurance. Many types of insurance exist today and there are so many companies that offer insurance services. These companies always need to predict whether or not a person will buy insurance so that they can save time and money for the most profitable customers.

Traditionally most insurance companies employ actuaries to calculate the insurance premiums. Actuaries are business professionals who use mathematics and statistics to assess the risk of financial loss and predict the likelihood of an insurance premium and claim, based on the factors/features like age and gender, etc. They typically produce something called an actuarial table provided to an insurance company's underwriting department, which uses the input to set insurance premiums. The insurance company calculates and writes all the programs, but it becomes much simpler by using Machine Learning.

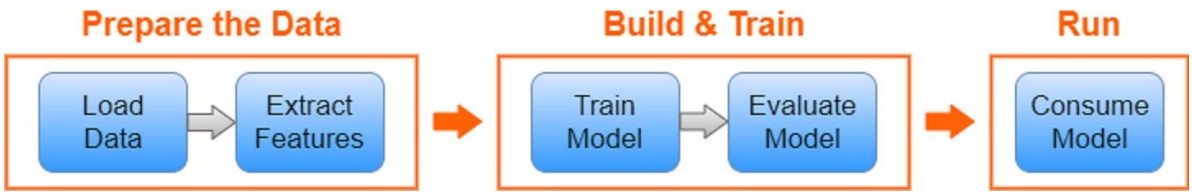
Machine Learning allows a program to learn from a set of data to figure out particular problem characteristics. The ML program looks at different factors like gender, smoking, the number of children, and region to find the overall highest medical charges and determine the price by using specific algorithms based upon the requirement. Smokers and customers with more children tend to have higher medical costs. Hence premiums will be more for those groups. As ML trains more and more data, the ML program becomes more intelligent and smarter in predicting the exact price. In the end, you have a function/program to call to get the insurance premium for a particular person based upon the input factors provided. You don't need to write all of these constructs yourself. ML program looks at all the sets of data provided and trains/learns, and it gives a function, and this function is a machine learning model that you can use in your application.

The task of insurance prediction is something that adds value to every insurance company. They use data from their database about everyone they have contacted to promote their insurance services and try to find the most potential people who can buy insurance. This helps a company to target the most profitable customers and saves time and money for the Insurance Company.

In the section below, I will take you through the task of Insurance Prediction with Machine Learning using Python. For the task of Insurance prediction with machine learning, I have collected a dataset from Kaggle about the previous customers of a travel insurance company. Here our task is to train a machine learning model to predict whether an individual will purchase the insurance policy from the company or not.

2. Methodology:

The key goal is to craft precise machine learning models for insurance prediction, leveraging customer and policy data. This aims to streamline decision-making in the insurance sector, enabling proactive risk management and efficient resource allocation. By accurately forecasting claim likelihoods, these tools empower insurers to optimize processes, identify high- risk scenarios, and enhance overall operational effectiveness. Ultimately, the objective is to fostera data-driven approach that minimizes uncertainties, elevates risk assessment practices, and contributes to a more agile and responsive insurance industry.



2.1 Data Collection:

In this insurance prediction project, the primary goal is to harness machine learning models for accurately forecasting insurance claim likelihoods. The datasets, comprising features such as age, occupation, marital status, and financial balance, are structured into training and testing sets.

	id	age	job	marital	education	balance	housing	loan	contact	month	duration	campaign	pdays	previous	poutcome	y
0	98749	32	admin.	single	secondary	64	yes	no	unknown	may	202	2	-1	0	unknown	0
1	19550	45	blue-collar	married	secondary	534	no	no	cellular	aug	104	6	-1	0	unknown	0
2	75084	45	technician	married	secondary	1477	yes	no	cellular	nov	75	1	132	1	failure	0
3	65715	39	technician	married	tertiary	14	no	no	cellular	jan	114	2	-1	0	unknown	0
4	41412	49	blue-collar	single	unknown	2222	no	no	unknown	jun	114	2	-1	0	unknown	0

The training set, utilized for model training, includes known outcomes, enabling the algorithms to discern patterns. The test set, without target labels, serves to evaluate the models' predictive efficacy. The features, spanning customer and policy details, are meticulously curated to facilitate robust risk assessment, streamlined decision-making, and proactive resource allocation within the insurance industry. The project's success hinges on developing efficient models that enhance operational effectiveness and foster a data-driven approach to insurance claim predictions.

2.2 Data Analysis:

The dataset, comprising 16 variables and 40,689 observations, exhibits a diverse mix of numeric, categorical, and boolean features. Data statistics reveal insightful details, such as the average age of 40.94, a broad range of job categories, and varying education levels. Notably, the balance feature displays a substantial number of zeros (7.8%) and negative values (8.3%), potentially indicating specific financial behaviors. Categorical variables like marital status and contact method offer a glimpse into the demographic and communication preferences of the dataset. Furthermore, the presence of high correlation and imbalance in the 'previous' and 'poutcome' variables suggests potential challenges in the modeling process. The correlation heatmap unveils relationships between features, highlighting noteworthy connections, including the influence of 'duration' on the target variable 'y.'

Careful consideration of these insights is vital for informed data preprocessing and subsequent machine learning model development.

The dataset under examination comprises 15 features and 1 output variable, encompassing a mix of numerical, categorical, and boolean types. Unique 'id' values are uniformly distributed, indicating individuality for each record. Interestingly, 'id' displays negligible correlation with the output 'y,' suggesting limited predictive value. Notably, 'pdays' and 'previous' exhibit a high correlation, hinting at a strong connection between days since the last contact and previous contact frequency. Key predictive features for 'y' include 'duration,' 'poutcome,' and 'month,' with their strong correlation underscoring their impact on subscription likelihoods. Age analysis suggests individuals between 30 and 60 are more prone to subscribe, while job types and housing status play discernible roles in shaping outcomes. Seasonal patterns reveal increased subscriptions in fall and winter. However, the dataset's class imbalance, with 81.7% zeros in 'y,' demands attention during model training. Overall, this data analysis provides nuanced insights crucial for informed preprocessing and subsequent machine learning model development, aiming to enhance predictive accuracy in the insurance domain.

2.3 Data Pre-processing:

Data preprocessing is a critical phase in machine learning that involves cleaning and transforming raw data into a suitable format for model training. This process includes handling missing values, addressing outliers, and encoding categorical variables. Additionally, feature engineering may be applied to create new relevant features, while handling class imbalances ensures fair representation of output categories. This meticulous preparation optimizes the dataset for accurate model predictions.

- **Handling Missing Values:**
  Missing values can introduce bias and hinder model performance. By identifying and addressing them, we ensure a completer and more accurate dataset. The chosen strategy of imputing missing values with the mode helps retain valuable information without sacrificing entire features. This method, implemented through Scikit-learn's SimpleImputer, proves effective in maintaining data integrity while preparing it for further analysis and modeling.

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values='unknown',strategy='most_frequent')
imputer.fit_transform(X_full_train)
imputer.fit_transform(X_test_full)
```

```
array([[5149, 42, 'admin.', ..., 317, 3, 'failure'],
       [6179, 36, 'admin.', ..., -1, 0, 'failure'],
       [4846, 29, 'blue-collar', ..., -1, 0, 'failure'],
       ...,
       [7780, 39, 'blue-collar', ..., -1, 0, 'failure'],
       [5259, 32, 'self-employed', ..., -1, 0, 'failure'],
       [4973, 54, 'retired', ..., -1, 0, 'failure']], dtype=object)
```

- **Handling Categorical Values:**
  Categorical features, while informative, need transformation into numeric formats for machine learning models. One-hot encoding stands out as a preferred method since it

captures the essence of categorical variables without imposing ordinal relationships. This transformation ensures that the model can effectively interpret and learn from these features, contributing to more accurate predictions.

```
low_cardinality_cols = [cname for cname in X_train_full.columns if X_train_full[cname].nunique() < 15 and
                        X_train_full[cname].dtype == "object"]
numeric_cols = [cname for cname in X_train_full.columns if X_train_full[cname].dtype in ['int64', 'float64']]
my_cols = low_cardinality_cols + numeric_cols
X_train = X_train_full[my_cols].copy()
X_valid = X_valid_full[my_cols].copy()
X_test = X_test_full[my_cols].copy()
X_train = pd.get_dummies(X_train)
X_valid = pd.get_dummies(X_valid)
X_test = pd.get_dummies(X_test)

X_train, X_valid = X_train.align(X_valid, join='left', axis=1)
X_train, X_test = X_train.align(X_test, join='left', axis=1)
X_test_copy = X_test.copy()
X_train.head()
```

	id	age	balance	duration	campaign	pdays	previous	job_admin.	job_blue-collar	job_entrepreneur	...	month_jun	month_mar	month_may	month_nov	month_oct	month_sep	poutcome_failure	poutcome_othe
5677	48044	40	783	171	2	-1	0	1	0	0	...	0	0	1	0	0	0	0	
30526	31356	48	1403	76	2	-1	0	1	0	0	...	0	0	0	0	0	0	0	
24659	18713	45	2671	94	4	-1	0	0	0	0	...	0	0	0	0	0	0	0	
24730	15806	56	-246	461	2	204	2	0	0	0	...	0	0	0	0	0	0	1	
20011	63545	49	157	131	2	-1	0	0	0	0	...	0	0	0	0	0	0	0	

5 rows × 49 columns

2.4 Feature Engineering:

Feature engineering involves crafting new features or transforming existing ones to extract more information. The addition of 'new_feature' and the logarithmic transformation of 'previous' enhance the dataset's ability to capture patterns and relationships. The creation of an interaction feature ('previous*pdays') acknowledges the synergy between two existing features, potentially boosting the model's predictive power.

- **Handling Class Imbalance:**
  Class imbalance, where one class dominates the other, can skew model predictions. Addressing this issue through Random Oversampling ensures a more equitable representation of both classes. Achieving a balanced distribution (50-50) enhances the model's ability to discern patterns in the minority class, leading to more reliable predictions. By doing oversampling we got this result.

```
from imblearn.over_sampling import RandomOverSampler
OverS = RandomOverSampler()
X_Over, Y_Over = OverS.fit_resample(X_train, y_train)
X_train = X_Over
y_train = Y_Over
print(sum(y_train)/len(y_train))

##creating copies
X_trains = X_train
X_valids = X_valid
X_tests = X_test
```

```
0.5
```

- **Removing Outliers:**
  Outliers, while sometimes indicative of anomalies, can also introduce noise. Removing the 'id' column, deemed as an outlier due to its unique and non-influential nature, streamlines the dataset. Additionally, addressing potential outliers in 'previous' after feature engineering optimizes the dataset for improved model performance, prioritizing meaningful information over anomalies.

This is the data after removing outliers.

	id	age	balance	duration	campaign	pdays	job_admin.	job_blue-collar	job_entrepreneur	job_housemaid	...	month_may	month_nov	month_oct	month_sep	poutcome_failure	poutcome_other	poutcome_succes
0	5149	42	734	332	2	317	1	0	0	0	...	0	0	0	0	1	0	
1	6179	36	22	77	5	-1	1	0	0	0	...	0	0	0	0	0	0	
2	4846	29	0	215	1	-1	0	1	0	0	...	1	0	0	0	0	0	
3	3676	46	1114	87	8	-1	0	0	0	0	...	0	0	0	0	0	0	
4	4256	39	378	127	2	-1	0	0	0	0	...	1	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4517	7880	46	174	157	1	-1	0	0	0	0	...	1	0	0	0	0	0	
4518	2635	29	751	121	2	-1	0	0	0	0	...	1	0	0	0	0	0	
4519	7780	39	-145	1372	4	-1	0	1	0	0	...	0	0	0	0	0	0	
4520	5259	32	286	175	1	-1	0	0	0	0	...	0	0	0	0	0	0	
4521	4973	54	6449	487	1	-1	0	0	0	0	...	0	0	0	0	0	0	

4522 rows × 50 columns

2.5 Train-Validation Split:

In the realm of machine learning, the 80-20 split holds significance, allocating 80% for training and 20% for validation. This balance ensures ample data for model learning while providing a controlled environment for performance assessment.

The training subset, encompassing 80% of the dataset, serves as a rich source of information, aiding the model in discerning complex patterns. On the other hand, the validation set, constituting 20%, acts as a testbed for fine-tuning model parameters and detecting signs of overfitting.

Overfitting, a common challenge in model development, occurs when a model becomes excessively tailored to the training data. The validation set serves as a vigilant check, ensuring the model generalizes well beyond its training data.

```
from sklearn.model_selection import train_test_split
X_full_train.dropna(axis=0,subset=['y'],inplace=True)
y = X_full_train.y
X_full_train.drop(['y'],axis=1,inplace=True)
print(X_full_train)
X_train_full, X_valid_full, y_train, y_valid = train_test_split(X_full_train, y, train_size=0.8, test_size=0.2)
```

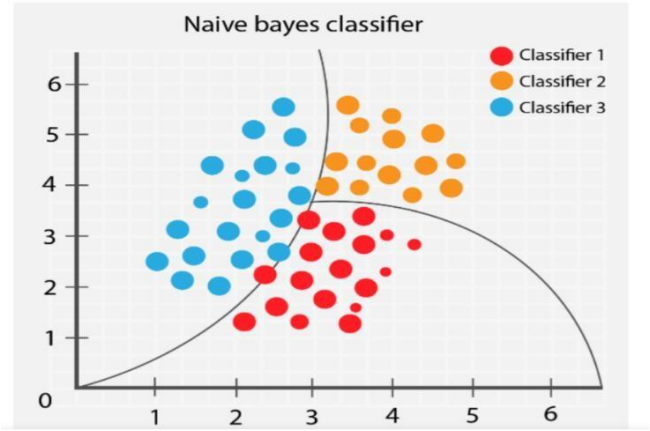
```
          id  age          job  marital  education  balance housing loan  \
0      98749   32       admin.   single  secondary       64     yes   no
1      19550   45  blue-collar  married  secondary      534      no   no
2      75084   45   technician  married  secondary     1477     yes   no
3      65715   39   technician  married   tertiary       14      no   no
4      41412   49  blue-collar   single    unknown     2222      no   no
...      ...  ...          ...      ...        ...      ...     ...  ...
40684  24953   29   technician   single  secondary     3313     yes   no
40685  34002   59       admin.  married  secondary       92     yes   no
40686  76035   54  blue-collar  married  secondary      548      no   no
40687  61279   46  blue-collar  married    primary      258     yes   no
40688  60013   41     services  married  secondary        0     yes  yes

         contact month  duration  campaign  pdays  previous poutcome
0        unknown   may       202         2     -1         0  unknown
1       cellular   aug       104         6     -1         0  unknown
2       cellular   nov        75         1    132         1  failure
3       cellular   jan       114         2     -1         0  unknown
4        unknown   jun       114         2     -1         0  unknown
...          ...   ...       ...       ...    ...       ...      ...
40684    unknown   jun        18         3     -1         0  unknown
40685   cellular   may       139         2    350         1  failure
40686   cellular   aug       520         7     -1         0  unknown
40687    unknown   may       217         1     -1         0  unknown
40688  telephone   may       125         4     -1         0  unknown

[40689 rows x 15 columns]
```

Ultimately, this strategic dataset split not only facilitates robust model development but also gauges a model's readiness for real-world scenarios, making it a pivotal practice in machine learning workflows.

2.6 Model Training & Evaluation:

Model 1: Bayes Classifier

` Bayes Theorem is a foundational concept in probability theory, crucial for the Naive Bayes Classifier. This theorem allows us to calculate the probability of a hypothesis based on prior knowledge. In the Naive Bayes context, it is used to estimate the probability of a particular outcome given observed features.



Model Training:

The Gaussian Naive Bayes classifier is employed, instantiated as `classifier_NB`. During training, the model learns from both the training and validation sets, capturing underlying patterns and relationships in the data.

Prediction:

After training, the model is applied to the test data, generating predictions. The predictions are stored and organized into an output DataFrame, associating them with the corresponding IDs.

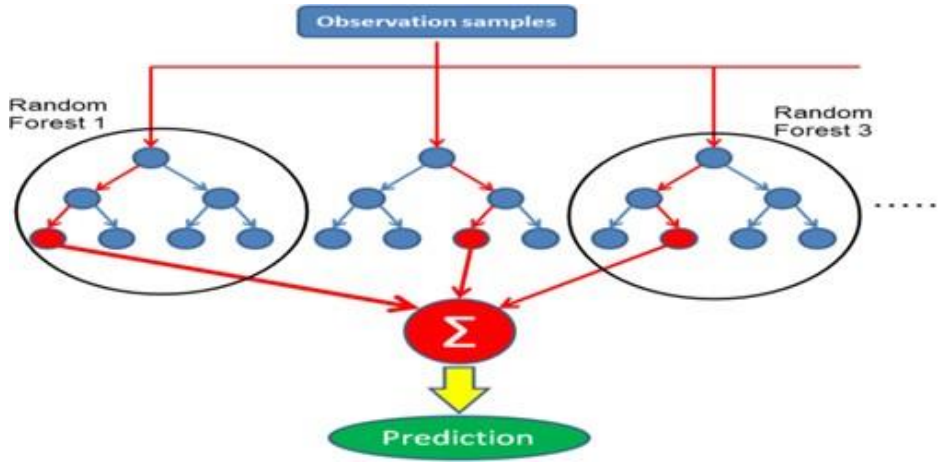
Accuracy and Confusion Matrix:

The model's performance is evaluated using metrics such as accuracy and the confusion matrix. Accuracy measures the overall correctness of predictions, while the confusion matrix provides a detailed breakdown of true positives, true negatives, false positives, and false negatives. These metrics offer insights into the model's ability to make accurate predictions and identify potential areas for improvement.

By using, Naive Bayes Classifier achieved an accuracy of approximately 83.33%. The confusion matrix further illustrates the model's performance in terms of correct and incorrect predictions for each class, aiding in the interpretation of results.

Model 2: Random Forest Classifier

Random Forest is an ensemble learning technique that combines multiple decision trees to create a more robust and accurate model. It's particularly effective for both classification and regression tasks. The "forest" comprises a collection of decision trees, each trained on a different subset of the dataset and using a random selection of features. The predictions from individual trees are then aggregated to produce a final result.



Hyperparameter Tuning:

Utilizing the Random Forest Classifier, hyperparameter tuning aims to enhance model performance. The Grid Search method is employed to iterate over various parameter combinations, evaluating each set through cross-validation. Parameters such as 'max_depth,' 'min_samples_leaf,' and 'n_estimators' are explored to identify the combination that maximizes model accuracy. After execution, the best parameters obtained are 'max_depth' = 5 and 'n_estimators' = 100.

Training the Model:

Once optimal hyperparameters are determined, the Random Forest Classifier is trained on the combined training and validation data. The classifier is instantiated with the identified parameters, and the model learns from the features and target variable, capturing complex relationships within the data.

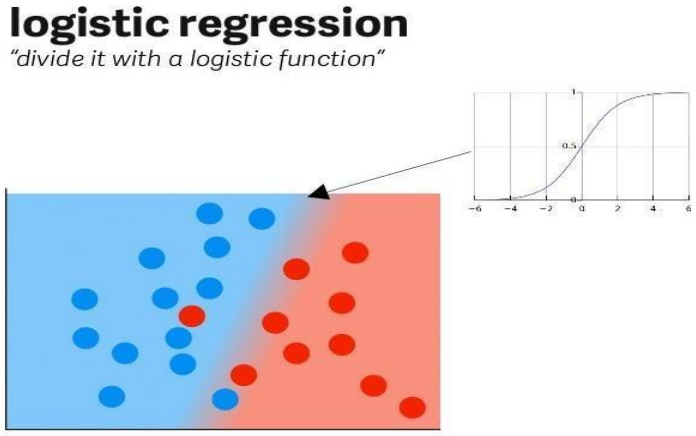
Accuracy and Confusion Matrix:

Model performance is assessed using accuracy metrics and the confusion matrix. The accuracy score provides an overall measure of the model's correctness, while the confusion matrix breaks down the true positives, true negatives, false positives, and false negatives. By using Random Forest Classifier, we achieved an accuracy of approximately 85.11%. The confusion matrix offers detailed insights into the model's ability to make accurate predictions for each class.

The confusion matrix for this model reveals a balanced performance, with notable accuracy in both true positive and true negative predictions. This comprehensive evaluation aids in understanding the model's strengths and areas for potential improvement.

Model 3: Logistic Regression

Logistic Regression is a statistical method used for binary classification problems. Unlike linear regression, it estimates the probability of a categorical dependent variable, employing a sigmoid function that predicts the likelihood of a binary outcome. It's a versatile algorithm often used for its simplicity and interpretability.



Training the Model:

In the Logistic Regression model, training involves fitting the algorithm to the combined training and validation data. The model learns from the provided features and their relationship with the target variable. Logistic Regression estimates the coefficients for each feature to make predictions based on the logistic function.

Prediction:

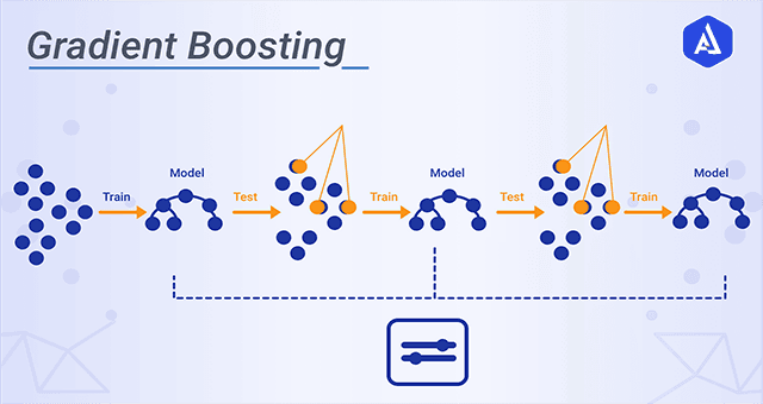
Following training, the trained Logistic Regression model is applied to the test dataset to generate predictions. These predictions are structured into an output DataFrame that associates the predicted values with their respective IDs.

Accuracy and Confusion Matrix:

Model evaluation entails computing the accuracy score and generating a confusion matrix. The accuracy score provides an overall performance measure, while the confusion matrix offers a detailed breakdown of the model's true positive, true negative, false positive, and false negative predictions. In this instance, the Logistic Regression model achieved an accuracy of approximately 79.28%. The confusion matrix depicts the model's ability to correctly predict both positive and negative classes, indicating areas where it excels and where it might need improvements.

Model 4: Gradient Boosting Classifier

Gradient Boosting is an ensemble learning technique that builds a predictive model in a stage-wise fashion. It combines the predictions from multiple weak learners, often decision trees, to create a strong predictive model. Gradient Boosting aims to minimize errors by training new models to correct the errors of previous ones, making it particularly effective for classification and regression tasks.



Hyperparameter Tuning:

Hypermeter tuning for the Gradient Boosting model using Grid Search. Grid Search explores different combinations of hyperparameter values, such as the number of estimators (trees), maximum depth of the trees, and the learning rate. The best combination of hyperparameters is selected based on the model's performance on the training data.

Training the Model:

After determining the optimal hyperparameters, the Gradient Boosting model is trained using the combined training and validation datasets. The selected hyperparameters influence how individual decision trees are constructed and how the model adapts to the training data.

Prediction:

The trained Gradient Boosting model is applied to the test dataset to make predictions. The resulting predictions are organized into an output DataFrame, which is saved as a CSV file. This file can be further analyzed or submitted for evaluation.

Accuracy and Confusion Matrix:

The model's accuracy is evaluated by calculating the accuracy score and generating a confusion matrix. The accuracy score provides an overall measure of the model's correctness, while the confusion matrix gives a detailed breakdown of true positive, true negative, false positive, and false negative predictions. The presented Gradient Boosting model achieved an accuracy of approximately 85.93%, as shown by the confusion matrix. This indicates a strong performance in correctly classifying both positive and negative instances.

3 Results:

After training and evaluating four different classifiers on the validation data, we observed varying levels of performance across the models.

1. Bayes Classifier:
   - Accuracy: 83.33%
   - Confusion Matrix: [[6281 951] [406 500]]
   - F1 Score: 0.68141

2. Random Forest Classifier:
   - Accuracy: 85.11%

- Confusion Matrix: [[6200 1032] [180 726]]
- F1 Score: 0.72283

3. Logistic Regression Classifier:
- Accuracy: 79.28%
- Confusion Matrix: [[5828 1404] [282 624]]
- F1 Score: 0.6099

4. Gradient Boosting Classifier:
- Accuracy: 85.93%
- Confusion Matrix: [[6211 1021] [124 782]]
- F1 Score: 0.7724

Predictions F1 Score:

- Bayes Classifier: 0.68141
- Random Forest Classifier: 0.72283
- Logistic Regression Classifier: 0.6099
- Gradient Boosting Classifier: 0.7724

Upon careful evaluation of the four classifiers on the validation dataset, the Gradient Boosting Classifier emerged as the clear frontrunner, outperforming its counterparts. With the highest accuracy of 85.93% and an impressive F1 score of 0.7724, the Gradient Boosting model demonstrated superior predictive capabilities.

The strength of the Gradient Boosting Classifier lies in its ability to sequentially train weak learners, each focusing on the mistakes of its predecessor. This ensemble technique results in a powerful and accurate model. The inherent capacity of gradient boosting to adapt and improve with each iteration makes it well-suited for complex datasets with intricate patterns and dependencies.

Contrastingly, the Bayes Classifier, Random Forest Classifier, and Logistic Regression Classifier exhibited slightly lower performance metrics. While the Random Forest Classifier showed notable accuracy (85.11%) and F1 score (0.72283), it fell short of surpassing the Gradient Boosting model.

4 Conclusions:

In conclusion, the robust predictive nature of the Gradient Boosting Classifier, coupled with its ability to effectively capture intricate relationships within the data, positions it as the optimal choice for the given task. The decision to favor the Gradient Boosting model is substantiated by its superior performance and the nuanced understanding it provides of the underlying dataset, making it the ideal candidate for deployment in practical scenarios.

Links:

1. ML in Stock Price Prediction: Link
2. ML in Insurance Prediction: Link

REFERENCES:

1.FREISLEBEN, B. STOCK-MARKET PREDICTION WITH BACKPROPAGATION NETWORKS, in LECTURE NOTES IN ARTIFICIAL INTELLIGENCE Volume:604, Pages: 451-460, 1992.

2. S. Gosh, E.A. Collins, and C.L. Scofield. Prediction of mortgage loan performance with a multiple neural network learning system. In *Abstracts of the First Annual INNS Meeting*, volume 1, pages 439–440, 1988.

3. G. Leja. Prediction of stock prices with neural networks (in German), Master's Thesis, Dept. of Computer Science, University of Darmstadt, 1991.

4. Saad, EW (Saad, EW); Prokhorov, DV (Prokhorov, DV); Wunsch, DC (Wunsch, DC). Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. In IEEE TRANSACTIONS ON NEURAL NETWORKS, volume 9, pages 1456-1470

THANK YOU