

Project Report On  
**Using Classification algorithms to  
find the best stocks based on the  
financial parameters**

Carried out at



CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING,  
ADVANCED COMPUTING TRAINING SCHOOL, HARDWARE  
PARK, HYDERABAD.

Under The Supervision of

Mrs. Priya Joshi

Submitted By

Mukkala Akhil Babu (220950325029)

Nandini Maurya (220950325031)

Narbariya Juhi Yogesh (220950325032)

Nishi Siddhu (220950325033)

Palvekar Payal Bhimrao (220950325034)

PG DIPLOMA IN BIG DATA ANALYTICS  
C-DAC, HYDERABAD

## Declaration

We hereby certify that the work being presented in the report entitled using classification algorithms to find the best stocks, in partial fulfilment of the requirements for the award of PG Diploma Certificate and submitted in the department of PG-DBDA of the C-DAC Hyderabad, is an authentic record of our work carried out during the period, 15<sup>th</sup> February 2023 to 15<sup>th</sup> March 2023 under the supervision of Mrs. Priya Joshi, C-DAC Hyderabad. The matter presented in the report has not been submitted by us for the award of any degree of this or any other Institute/University.

**Mr. Mukkala Akhil Babu**

**PRN: 220950325029**

**Ms. Nandini Maurya**

**PRN: 220950325031**

**Ms. Narbariya Juhi Yogesh**

**PRN: 220950325032**

**Ms. Nishi Siddhu**

**PRN: 220950325033**

**Ms. Palvekar Payal Bhimrao**

**PRN: 220950325034**

# Acknowledgement

**“Outstanding achievements are not possible in vacuum. It needs lots of help and assistance besides a healthy environment, luckily, we had one.”**

We take this opportunity to express our profound sense of gratitude and respect to **C-DAC ACTS (Hyderabad) and all the faculty** for providing us the environment that helped us learn and implement our learning in the project.

Working on this project **Using Classification algorithms to find the best stock** was a great learning experience for us. We would like to thank our project guide **Mrs. Priya Joshi** for her valuable guidance that encouraged us to select such a project to work on. Her support helped us overcome various obstacles and intricacies encountered during the course of this project work.

Mukkala Akhil Babu: 220950325029

Nandini Maurya: 220950325031

Narbariya Juhi Yogesh: 220950325032

Nishi Siddhu: 220950325033

Palvekar Payal Bhimrao: 220950325034

# **Chapter 1**

## **Introduction and Overview of Project**

Stock market analysis has always been a hot area for researchers and investors. People have come up with a lot of theoretical foundation in mathematics, and developed a variety of methods to analyze the stock market with the help of modern computer technology. Among them, the Feature Selection method is a very important research field. It evaluates a lot of factors that are considered important to the stock market, and selects out the most significant ones for people to depict the market trend. The function of Feature Selection method is to discard the dross and select the essence. The computational time could be dramatically reduced, since the significant factors are pointed out for the investors. In order to figure out the prominent features in the stock market, researches on effective Feature Selection methods are extremely needed.

This Project contains all the steps required to build a dataset of financial data for a lot of stocks and to analyze it with different machine learning models. You will see throughout the project that this modus-operandi does not leverage historic ticker price data, but financial indicators found in the 10-K filings that each publicly traded company releases yearly. In particular, for the sake of this project, we will use the financial data from 2018 in order to make some fictitious predictions regarding the performance of stocks during 2019 (meaning from the first trading day of Jan '19 to the last trading day of Dec '19).

### **1.1 Objectives of the project**

- Feature selection is a major step in classification problems and here we use Logistic Regression to find out important paraments from over 100+ columns.
- Build a thorough ML model algorithm that can make predictions about whether a stock is likely to go up or down in value and advise investors about whether they should buy or sell the stock.
- Perform hyperparameter tuning to improve the accuracy of the models.
- Generate the feature importance using the model with best accuracy.
- Use the Features to predict whether to buy/hold or sell the stock

## **1.2 Technology and framework used**

- Database: Mongo dB  
Version: 6.0
- Stream lit: Deployment
- Anaconda Environment  
Version: Anaconda Navigator 2.3.2

## **1.3 Software Required**

- Jupyter Notebook
- PyCharm 2022.2.3 (Community Edition)
- Mongo DB Compass
- Windows 11 Home Operating System

## 1.4 Machine Learning

In a world where nearly, all manual tasks are being automated, the definition of “manual” is changing. We are living in an era of constant technological progress, and one such advancement is in the field of machine learning.

Machine learning is a data analytics technique that teaches computers to do what comes naturally to humans and animals: learn from experience. Machine learning algorithms use computational methods to “learn” information directly from data without relying on a predetermined equation as a model. The algorithms adaptively improve their performance as the number of samples available for learning increases. Machine learning algorithms find natural patterns in data that generate insight and help you make better decisions and predictions.

Machine learning uses two types of techniques: Supervised learning, which trains a model on known input and output data so that it can predict future outputs, and Unsupervised learning, which finds hidden patterns or intrinsic structures in input data.

Supervised machine learning builds a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data.

Supervised learning uses classification and regression techniques to develop machine learning models. Classification techniques predict discrete responses. we use classification if our data can be tagged, categorized, or separated into specific groups or classes. Here in our problem, we have two discrete responses as “buy/hold” or “sell”. So, we used classification technique.

Regression techniques predict continuous responses. Unsupervised learning finds hidden patterns or intrinsic structures in data. It is used to draw inferences from datasets consisting of input data without labelled responses.

### 1.3.1 Advantages of Machine Learning

- Machine learning algorithms are continuously improving in accuracy and efficiency, which makes better decisions.
- Machine learning is responsible for cutting the workload and time. By automating things, we let the algorithm do the hard work for us.

- Machine learning algorithms are good at handling data that is multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments
- Machine learning has a wide variety of applications. This means that we can apply Machine learning to any of the major fields. Machine learning has a role everywhere, from medical, business, and banking to science and tech.

## **Chapter 2**

### **Evaluation Metrics**

#### **2.1 Accuracy**

Accuracy is one metric for evaluating classification models. Informally, Accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$Accuracy = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \quad (2.1)$$

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.2)$$

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

#### **2.2 Precision and Recall**

Precision is a good measure to determine, when the costs of False Positive is high. For instance, email spam detection. In email spam detection, a false positive means that an email that is non-spam (actual negative) has been identified as spam (predicted spam). The email user might lose important emails if the precision is not high for the spam detection model.



In the field of information retrieval precision is the fraction of retrieved documents that are relevant to the query:

$$\begin{aligned} \textit{Precision} &= \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Positive}} \\ &= \frac{\textit{True Positive}}{\textit{Total Predicted Positive}} \end{aligned} \tag{2.3}$$

Recall calculates how many of the Actual Positives our model capture through labelling it as Positive (True Positive). Applying the same understanding, we know that Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.

$$\begin{aligned} \textit{Recall} &= \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Negative}} \\ &= \frac{\textit{True Positive}}{\textit{Total Actual Positive}} \end{aligned} \tag{2.4}$$

## 2.3 F1 Score

F1 is a function of Precision and Recall.

$$F1 = 2 * \frac{\textit{Precision} * \textit{Recall}}{\textit{Precision} + \textit{Recall}} \tag{2.5}$$

F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives).

## 2.4 Confusion matrix

A Confusion matrix is an  $N \times N$  matrix used for evaluating the performance of a classification model, where  $N$  is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a  $2 \times 2$  matrix as shown below with 4 values:

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP	FP
	NEGATIVE	FN	TN

Figure 2.1: Confusion Matrix

Let's decipher the matrix:

- The target variable has two values: Positive or Negative
- The columns represent the actual values of the target variable
- The rows represent the predicted values of the target variable

## **Chapter 3**

# **Data Pre-processing & EDA**

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

Before going into pre-processing and data exploration we will explain some of the concepts that allowed us to select our features.

### **3.1 Pre-processing**

#### **3.1.1 Loading data**

This dataset contains 171 financial indicators, that are commonly found in the 10-K filings each publicly traded company releases yearly, for a plethora of around 4k US stocks. They have built this dataset by using Financial Modeling Prep API(<https://site.financialmodelingprep.com/developer/docs/>) we got the data from the above site and then we stored that pre-processed dataset into MongoDB for further use here below is the screenshot

#### **Imports**

Here we are importing the all of the basic libraries for data pre-processing.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_auc_score
```

## Loading dataset

### 1) Explore the data set Us\_Stock\_data\_2018

This dataset contains 200+ financial indicators, that are commonly found in the 10-K filings each publicly traded company releases yearly, for a plethora of US stocks (on average, 4k stocks are listed in each dataset)

We loaded the csv dataset into the Mongo dB database using the Mongo Compass and then formed a data frame and check its rows and columns i.e., shape of data frame.

```
In [4]: import pymongo
# build a new client instance for MongoClient
client = pymongo.MongoClient("mongodb://localhost:27017")

# create new database and client collection
db = client["Project"]
db

Out[4]: Database(MongoClient(host='localhost:27017', document_class=dict, tz_aware=False, connect=True), 'Project')

In [5]: mycollection=db["Stock"]
record=mycollection.find()
print(record)

<pymongo.cursor.Cursor object at 0x0000020E14332D60>

In [8]: list1=list(record)

In [7]: import pandas as pd
stock_data=pd.DataFrame(list1)
stock_data

Out[7]:
```

Unnamed: 0	Revenue	Revenue Growth	Cost of Revenue	Gross Profit	R&D Expenses	SG&A Expense	Operating Expenses	Operating Income	...	Receivables growth	Inventory Growth	Asset Growth	Book Value per Share Growth
CMCSA	94507000000	0.1115	0	94507000000	0	64822000000	75498000000	19009000000	...	0.257	0	0.3426	0.0722
KMI	14144000000	0.032	7288000000	6856000000	0	601000000	3062000000	3794000000	...	0.0345	-0.092	-0.0024	0.0076

```
stock_data = pd.read_csv('2018_US_Stock_Data.csv')
stock_data.shape

(4291, 171)
```

After loading dataset, we have to explore the data to ensure that the data is normally distributed. The second-to-last column, PRICE VAR [%], lists the percent price variation of each stock for the year. For example, if we consider the dataset 2018\_Financial\_Data.csv, we will have:

200+ financial indicators for the year 2018;

percent price variation for the year 2018 (meaning from the first trading day on Jan 2018 to the last trading day on Dec 2018).

The last column, class, lists a binary classification for each stock, where

for each stock, if the PRICE VAR [%] value is positive, class = 1. From a trading perspective, the 1 identifies those stocks that a hypothetical trader should BUY at the start of the year and sell at the end of the year for a profit.

for each stock, if the PRICE VAR [%] value is negative, class = 0. From a trading perspective, the 0 identifies those stocks that a hypothetical trader should NOT BUY, since their value will decrease, meaning a loss of capital.

The columns PRICE VAR [%] and class make possible to use the datasets for both classification and regression tasks:

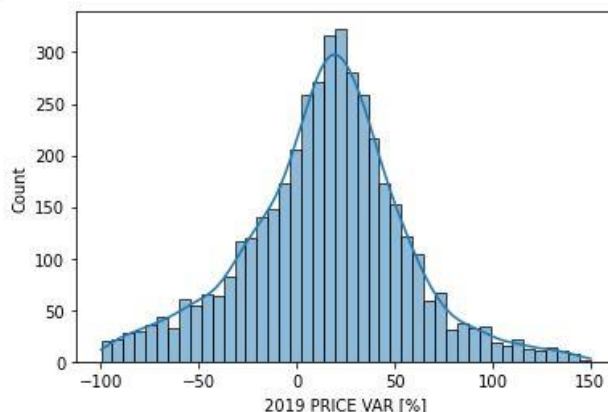
If the user wishes to train a machine learning model so that it learns to classify those stocks that in buy-worthy and not buy-worthy, it is possible to get the targets from the class column;

If the user wishes to train a machine learning model so that it learns to predict the future value of a stock, it is possible to get the targets from the PRICE VAR [%] column.

So, Using PRICE VAR [%] column we check whether data is normally distributed or not. From below image we can see that data is normally distributed i.e.; the company's data in the given data is varied normally all together.

```
sns.histplot(data=stock_data, x="2019 PRICE VAR [%]", kde=True);
```

```
## Target variable is normaly distributed
```



## 2) Data Pre-Processing

Data pre-processing work is done first which included handling null values, removing unnecessary columns and converting categorical to numerical columns using one hot encoding.

Some financial indicators are not necessary like the columns with zero variance and PRICE VAR [%] as we derived the class column to buy or sell stock using it. There are values are (nan cells) in some columns, so the user can select the best technique to clean each dataset (dropna, fillna, etc.).

```
unwanted_cols = []
# Finding the columns where variance is zero
for val in stock_data.var().iteritems():
    if val[1]==0:
        unwanted_cols.append(val[0])
print("Cols with variance zero" ,unwanted_cols)
## 'Unnamed: 0' and '2019 PRICE VAR [%]' are unwanted columns
unwanted_cols.extend(['2019 PRICE VAR [%]'])
stock_data.drop(columns =unwanted_cols, inplace=True, axis=1)
print("Total Removed columns: ", unwanted_cols)
```

```
Cols with variance zero ['operatingProfitMargin']
Total Removed columns: ['operatingProfitMargin', '2019 PRICE VAR [%]']
```

```
stock_data.isnull().sum()
```

```
Unnamed: 0      0
Revenue         46
Revenue Growth  139
Cost of Revenue 185
Gross Profit    64
...
Debt Growth     263
R&D Expense Growth 258
SG&A Expenses Growth 248
Sector          0
Class           0
Length: 169, dtype: int64
```

```
for col in stock_data.columns:
    if stock_data[col].isnull().sum():
        if stock_data[col].dtype in ['float64','int64']:
            stock_data[col].fillna(value=stock_data[col].median(), inplace=True)
        elif stock_data[col].dtype == 'object':
            stock_data[col].fillna(value=stock_data[col].mode(), inplace=True)
```

```
stock_data[col].isnull().sum()
```

```
0
```

The third-to-last column, Sector, lists the sector of each stock. Indeed, in the US stock market each company is part of a sector that classifies it in a macro-area. We are converting categorical to numerical columns using one hot encoding.

```
cat_cols = []
for col in x.columns:
    if x[col].dtype not in ['float64', 'int64']:
        cat_cols.append(col)
print("The categorical cols in the dataset are: ", cat_cols)

x_1 = pd.get_dummies(x, columns=cat_cols, drop_first=True)
x_1
```

The categorical cols in the dataset are: ['Sector']

...	...	Sector_Communication Services	Sector_Consumer Cyclical	Sector_Consumer Defensive	Sector_Energy	Sector_Financial Services	Sector_Healthcare	Sector_Industrials	Sector_Real Estate	Sector
...	...	0	1	0	0	0	0	0	0	...
...	...	0	0	0	1	0	0	0	0	...
...	...	0	0	0	0	0	0	0	0	...
...	...	0	0	0	0	0	0	0	0	...
...	...	0	0	0	0	0	0	1	0	...

After Completing the one-shot encoding, we describe the info and performed descriptive statistics of the data.

```
stock_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4291 entries, 0 to 4290
Columns: 169 entries, Revenue to ...
memory usage: 5.5+ MB
```

```
stock_data.describe()
```

	Revenue	Revenue Growth	Cost of Revenue	Gross Profit	R&D Expenses	SG&A Expense	Operating Expenses	Operating Income	Interest Expense	Earnings before Tax	...
count	4.291000e+03	4291.000000	4.291000e+03	4.291000e+03	4.291000e+03	4.291000e+03	4.291000e+03	4.291000e+03	4.291000e+03	4.291000e+03	...
mean	5.180052e+09	3.422626	3.083985e+09	2.062112e+09	1.131720e+08	8.878144e+08	1.411290e+09	6.659926e+08	9.817802e+07	5.643574e+08	...
std	2.061774e+10	194.637357	1.494490e+10	7.712701e+09	9.184175e+08	3.634336e+09	5.478425e+09	2.990435e+09	3.745514e+08	2.647542e+09	...
min	-6.894100e+07	-3.461500	-2.669055e+09	-1.818220e+09	-1.042000e+08	-1.401594e+08	-4.280000e+09	-1.455700e+10	-1.408252e+09	-2.177200e+10	...
25%	7.077400e+07	0.000000	6.297000e+06	3.981147e+07	0.000000e+00	2.255300e+07	4.705280e+07	-3.888500e+06	0.000000e+00	-7.976956e+06	...
50%	5.259670e+08	0.075900	1.860960e+08	2.414580e+08	0.000000e+00	9.675400e+07	1.879630e+08	4.565950e+07	6.153022e+06	3.052850e+07	...
75%	2.484587e+09	0.180950	1.216504e+09	9.820030e+08	9.796500e+06	3.924500e+08	6.532550e+08	2.933000e+08	5.522450e+07	2.252125e+08	...
max	5.003430e+11	12739.000000	3.733960e+11	1.269470e+11	2.883700e+10	1.065100e+11	1.065100e+11	7.089800e+10	9.168000e+09	7.290300e+10	...



## Chapter 4

### Feature Selection

Feature selection is a major step in classification problems. We have used logistic regression for feature selection in this analysis. Logistic regression is a popular method for feature selection because it is simple to implement, easy to interpret, and can be regularized to prevent overfitting. In logistic regression, the goal is to predict the probability of a binary outcome using a set of input features. The model estimates the probability that an example belongs to one class (e.g., "positive" class) using a logistic function of the input features.

During training, the model adjusts the values of the coefficients associated with each feature to maximize the likelihood of the observed data. After training, the magnitude of the coefficients can be used to determine the importance of each feature in predicting the outcome.

Before that we split the data to test and train using features and target variable and then perform the logistics Regression.

```
y = stock_data['Class'] #target variable
x_1 = stock_data.drop(['Unnamed: 0', 'Class'], axis=1)
```

#### Logistic Regression

```
# Adding constant
import statsmodels.api as sm
x_2=sm.add_constant(x_1)
x_2.shape

(4291, 177)

from sklearn.linear_model import LogisticRegression

x_train_2,x_test_2,y_train_2,y_test_2=train_test_split(
    x_2,y,test_size=0.2,random_state=10)

x_train_2.shape,x_test_2.shape,y_train_2.shape,y_test_2.shape

((3432, 177), (859, 177), (3432,), (859,))

logR_2=sm.Logit(y_train_2,x_train_2)

#Fit the model
logR_2=logR_2.fit()
```

---

```
LinAlgError                                Traceback (most recent call last)
Input In [18], in <cell line: 4>()
      1 logR_2=sm.Logit(y_train_2,x_train_2)
```



## Understanding the error

This error is letting you know that your independent variables are correlated, which can result in a matrix that is singular also the columns are highly correlated therefore it is showing Singular matrix error. So, find out the highly correlated columns and removed them to perform the feature selection.

## Removing highly correlated columns

```
LinAlgError: Singular matrix
```

### Highly Correlated columns

```
def ger_high_correlated_cols(dataset, threshold):
    col_corr = set() # List of all the names of deleted columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if (corr_matrix.iloc[i, j] >= threshold) and (corr_matrix.columns[j] not in col_corr):
                colname = corr_matrix.columns[i] # getting the name of column
                col_corr.add(colname)
    return col_corr

col_to_be_removed = ger_high_correlated_cols(x_2, 0.70)
print("No of cols to be removed: ", len(col_to_be_removed))
col_to_be_removed

No of cols to be removed: 66

{'Average Inventory',
 'Average Receivables',
 'Book Value per Share',
 'Cash and short-term investments',
 'Cash per Share',
 'Consolidated Income',
 'Cost of Revenue',
 'Days Payables Outstanding',
 'Days Sales Outstanding',
 'Days of Inventory on Hand',
 'Debt to Assets',
 'Debt to Equity',
 'Deposit Liabilities',
 'Depreciation & Amortization',
 'EBIT',
 'EBITDA',
 'EPS Diluted',
 'Earnings Before Tax Margin',
 'Earnings before Tax',
 'Free Cash Flow margin',
 'Free Cash Flow per Share',
 'Gross Profit',
 'Gross Profit Growth',
 'Income Tax Expense',
 'Interest Coverage',
 'Inventory Turnover'}
```

```

totalDebtToCapitalization }

In [22]: features_2 = list(set(x_2.columns)-col_to_be_removed)

In [23]: x_3 = x_2[features_2]

In [24]: x_3

```

0	1.198500e+10	2.56	9.4801	12.9254	0	0.1428	1.551136	0.000000e+00	3.81
1	2.119000e+09	0.66	9.6022	16.0840	0	0.0961	1.110304	1.566000e+09	3.28
2	1.425100e+10	4.57	11.4929	15.1845	1	0.3312	1.420326	0.000000e+00	3.01
3	8.521000e+09	12.27	6.5803	7.1329	1	1.1342	1.959680	1.022000e+09	6.50
4	7.090000e+08	-2.62	3.4583	89.2974	0	-0.2961	0.530021	1.211700e+10	3.11
...	...	...	...	...	...	...	...	...	...
4286	-8.160000e+05	0.60	0.0000	0.0000	0	0.2747	1.431051	0.000000e+00	3.25
4287	-6.290432e+06	-0.08	0.0000	0.0000	0	-2.0649	1.431051	6.604801e+06	1.20
4288	-8.796000e+06	-0.92	3.9857	0.0000	0	-0.0673	-208.428571	0.000000e+00	3.35
4289	-2.702133e+06	0.52	1.5678	0.0000	0	-2.9498	-4.784443	2.995960e+05	7.68
4290	-1.176652e+07	-0.18	0.0000	0.0000	0	-0.5716	-18.009018	0.000000e+00	1.94

4291 rows x 111 columns

After deleting the highly correlated columns we can continue to perform Logistic regression to find the important features by selecting the most significant columns in each turn.

```

In [25]: x_train_3,x_test_3,y_train_3,y_test_3=train_test_split(
        x_3,y,test_size=0.2,random_state=10)

        x_train_3.shape,x_test_3.shape,y_train_3.shape,y_test_3.shape

Out[25]: ((3432, 111), (859, 111), (3432,), (859,))

```

```

In [26]: logR_3=sm.Logit(y_train_3,x_train_3)

        # Fit the model
        logR_3=logR_3.fit()

Optimization terminated successfully.
      Current function value: 0.527870
      Iterations 21

```

```

In [27]: logR_3.summary2()

```

```

summart_result = logR_3.summary2().tables[1]
significant_cols = list(summart_result[summart_result['P>|z|'] <0.05].index.values)
print(len(significant_cols))
print(significant_cols)
## Cols with p value lesser than 0.05 are selected as significant cols

17
['Sector_Healthcare', 'Dividend Yield', 'priceEarningsRatio', 'Dividend per Share', '3Y Dividend per Share Growth (per Share)',
'Sector_Energy', 'Net Debt', 'Net Current Asset Value', 'Capital Expenditure', 'longtermDebtToCapitalization', 'interestCoverag
e', 'Sector_Financial Services', 'Operating Income', 'Total current assets', 'SG&A Expenses Growth', 'Sector_Real Estate', 'Sec
tor_Utillities']

In [28]: x_4 = x_3[significant_cols]

```

```

: x_train_4,x_test_4,y_train_4,y_test_4=train_test_split(
    x_4,y,test_size=0.2,random_state=10)

x_train_4.shape,x_test_4.shape,y_train_4.shape,y_test_4.shape

: ((3432, 17), (859, 17), (3432,), (859,))

: logR_4=sm.Logit(y_train_4,x_train_4)

# Fit the model
logR_4=logR_4.fit()

Optimization terminated successfully.
    Current function value: 0.559342
    Iterations 8

: summart_result = logR_4.summary2().tables[1]
significant_cols = list(summart_result[summart_result['P>|z|'] <0.05].index.values)
print(len(significant_cols))
print(significant_cols)
## Cols with p value lesser than 0.05 are selected as significant cols

13
['Sector_Healthcare', 'Dividend Yield', 'priceEarningsRatio', 'Dividend per Share', 'Sector_Energy', 'Net Current Asset Value',
'Capital Expenditure', 'longtermDebtToCapitalization', 'interestCoverage', 'Sector_Financial Services', 'Operating Income', 'Se
ctor_Real Estate', 'Sector_Utillities']

: x_5 = x_4[significant_cols]

: x_train_5,x_test_5,y_train_5,y_test_5=train_test_split(
    x_5,y,test_size=0.2,random_state=10)

x_train_5.shape,x_test_5.shape,y_train_5.shape,y_test_5.shape

: ((3432, 13), (859, 13), (3432,), (859,))

: logR_5=sm.Logit(y_train_5,x_train_5)

# Fit the model
logR_5=logR_5.fit()

Optimization terminated successfully.
    Current function value: 0.560416
    Iterations 8

: summart_result = logR_5.summary2().tables[1]
significant_cols = list(summart_result[summart_result['P>|z|'] <0.05].index.values)
print(len(significant_cols))
print(significant_cols)
## Cols with p value lesser than 0.05 are selected as significant cols

12
['Sector_Healthcare', 'Dividend Yield', 'priceEarningsRatio', 'Dividend per Share', 'Sector_Energy', 'Capital Expenditure', 'lo
ngtermDebtToCapitalization', 'interestCoverage', 'Sector_Financial Services', 'Operating Income', 'Sector_Real Estate', 'Sector
_Utillities']

: x_6 = x_5[significant_cols]
x_6

```

```
x_train_6,x_test_6,y_train_6,y_test_6=train_test_split(
    x_6,y,test_size=0.2,random_state=10)

x_train_6.shape,x_test_6.shape,y_train_6.shape,y_test_6.shape

((3432, 12), (859, 12), (3432,), (859,))
```

```
logR_6=sm.Logit(y_train_6,x_train_6)
```

```
# Fit the model
logR_6=logR_6.fit()
```

```
Optimization terminated successfully.
    Current function value: 0.560631
    Iterations 7
```

```
logR_6.params
```

```
Sector_Healthcare      -3.030494e-01
Dividend Yield         -3.861386e+00
priceEarningsRatio      3.223571e-03
Dividend per Share      6.042921e-01
Sector_Energy          -6.720330e-01
Capital Expenditure     1.002984e-10
longtermDebtToCapitalization 6.735127e-01
interestCoverage        -2.314675e-05
Sector_Financial Services 1.359011e+00
Operating Income        1.682921e-10
Sector_Real Estate      8.856723e-01
Sector_Uilities         8.379988e-01
dtype: float64
```

Insight: These are the 12 most important parameters in determination of target variables

## Important features

x_6											
	Sector_Healthcare	Dividend Yield	priceEarningsRatio	Dividend per Share	Sector_Energy	Capital Expenditure	longtermDebtToCapitalization	interestCoverage	Sec		
0	0	0.0269	13.3008	0.917	0	-1.231200e+10	0.599833	5.2662			
1	0	0.0471	23.3030	0.725	1	-2.924000e+09	0.501908	2.1455			
2	0	0.0256	10.2691	1.200	0	-1.518100e+10	0.251834	-184.0556			
3	0	0.0000	4.2999	0.000	0	-8.879000e+09	0.104785	42.8216			
4	0	0.0508	0.0000	0.370	0	-3.537000e+09	0.758131	-3.3036			
...	...	...	...	...	...	...	...	...			
4286	1	0.0000	2.8833	0.000	0	0.000000e+00	0.002398	0.0000			
4287	0	0.0000	0.0000	0.000	0	0.064000e+03	0.368359	-0.3406			
4288	0	0.0000	0.0000	0.000	0	-4.200000e+04	0.496599	0.0000			
4289	0	0.0000	6.1538	0.000	0	-4.671380e+05	0.000000	7.7919			
4290	0	0.0000	0.0000	0.000	0	-6.189970e+05	0.000000	0.0000			
4291 rows x 12 columns											
4											
insignificant_col={'interestCoverage','Operating Income','Capital Expenditure'}											
signi_col=list(set(x_6.columns)-insignificant_col)											
x_7=x_6[signi_col]											

In [66]: x\_7

Out[66]:

	Sector_Financial Services	longtermDebtToCapitalization	Sector_Energy	Sector_Healthcare	Dividend per Share	Sector_Real Estate	priceEarningsRatio	Sector_Utillities	Dividend Yield
0	0	0.599833	0	0	0.917	0	13.3008	0	0.0269
1	0	0.501908	1	0	0.725	0	23.3030	0	0.0471
2	0	0.251834	0	0	1.200	0	10.2891	0	0.0256
3	0	0.104785	0	0	0.000	0	4.2999	0	0.0000
4	0	0.758131	0	0	0.370	0	0.0000	0	0.0508
...	...	...	...	...	...	...	...	...	...
4286	0	0.002398	0	1	0.000	0	2.8833	0	0.0000
4287	0	0.368359	0	0	0.000	1	0.0000	0	0.0000
4288	0	0.496599	0	0	0.000	0	0.0000	0	0.0000
4289	0	0.000000	0	0	0.000	0	6.1538	0	0.0000
4290	0	0.000000	0	0	0.000	0	0.0000	0	0.0000

4291 rows x 9 columns



# Chapter 5

## Data balancing & Standardization

After feature selection, we will balance the dataset. Balancing a dataset for machine learning is important because most machine learning algorithms are sensitive to imbalanced datasets. Imbalanced datasets are those in which the data points' classes (or categories) are not evenly distributed. For example, a dataset may have many more data points belonging to one class than to another. This can cause problems for machine learning algorithms because they may be biased towards the majority class, and may not be able to accurately predict the minority class.

To balance a dataset, the data points are typically re-sampled or re-weighted so that the classes are more evenly distributed. This can be done using a variety of techniques, such as under-sampling, oversampling, or using class weights. Balancing a dataset can improve the performance of a machine learning algorithm by making it more sensitive to the minority class, and by reducing the bias towards the majority class. This can help the algorithm to make more accurate predictions. We have done under sampling of the majority class and oversampling of the minority class to balance the data.

### Dealing with imbalanced dataset

```
In [73]: stock_new['Class'].value_counts()
Out[73]: 1    2945
         0    1346
         Name: Class, dtype: int64
```

```
In [74]: 1346/(1346+2945)
Out[74]: 0.3136797949195992
```

Dataset is quite unbalanced as dataset contains 69% of 1 class and 39% of 0 class

### Resolving the Imbalance

```
stock_yes=stock_new[stock_new['Class']==1]
```

```
stock_yes.shape
(2945, 10)
```

```
stock_no=stock_new[stock_new['Class']==0]
stock_no.shape
(1346, 10)
```

### Upsampling

```
from sklearn.utils import resample
stock_no_up=resample(stock_no, replace=True,random_state=100,n_samples=2000)
stock_no_up.shape
(2000, 10)
```

### Downsampling

```
from sklearn.utils import resample
stock_yes_down=resample(stock_yes, replace=False,random_state=100,n_samples=2500)
stock_yes_down.shape
(2500, 10)
```

**Creating the dataset by combining**

```
[80]: stock_new=pd.concat([stock_no_up,stock_yes_down])
      stock_new.shape
```

```
[80]: (4500, 10)
```

```
[81]: stock_new
```

```
[81]:
```

	Sector_Financial Services	longtermDebtToCapitalization	Sector_Energy	Sector_Healthcare	Dividend per Share	Sector_Real Estate	priceEarningsRatio	Sector_Utiliti
2887	0	0.140011	0	1	0.000	0	0.0000	
403	0	0.284316	0	0	0.920	0	14.1481	
1552	0	0.899275	0	1	0.000	0	0.0000	
242	0	0.742698	0	0	0.000	0	0.0000	
2915	0	0.000000	0	1	0.000	0	0.0000	
...	...	...	...	...	...	...	...	
690	0	1.130008	0	0	0.000	0	2.8554	
1941	1	0.000247	0	0	1.490	0	13.5141	
530	0	0.467364	0	0	0.172	0	11.6582	
3688	1	0.165749	0	0	0.000	0	21.4548	
915	1	0.119748	0	0	0.000	0	10.3499	

4500 rows x 10 columns

**Shuffling**

```
[82]: from sklearn.utils import shuffle
      stock_new=shuffle(stock_new)
      stock_new
```

```
[82]:
```

	Sector_Financial Services	longtermDebtToCapitalization	Sector_Energy	Sector_Healthcare	Dividend per Share	Sector_Real Estate	priceEarningsRatio	Sector_Utiliti
278	0	0.330251	0	0	1.352	0	33.3597	
2103	0	0.133028	0	0	0.000	0	0.0000	
2957	0	0.437369	0	0	0.000	0	20.8219	

New data frame with important features along with company name and class variables is formed after balancing and shuffling the dataset.

After balancing the data, we need to standardization of the feature values. Standardizing the data is important for classification machine learning algorithms because these algorithms make assumptions about the data they are trained on. For example, many classification algorithms assume that the data is normally distributed, which means that the data is centered around a mean value and that the data is evenly distributed around that mean value. Standardizing the data helps to ensure that these assumptions are met, which can improve the performance of the algorithm. Standardizing the data can also help reduce the impact of outliers on the model, which can further improve its performance.

## Standardisation of features ¶

```

]: from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()

x_scaled=scaler.fit_transform(x)
x_scaled

]: array([[ -4.57905470e-01,  -3.02513314e-02,  -2.71059555e-01,  ...,
           1.62514238e-01,  -1.44468486e-01,   1.78749892e-02],
        [ -4.57905470e-01,  -3.11322964e-01,  -2.71059555e-01,  ...,
          -2.76493450e-01,  -1.44468486e-01,  -1.36615031e-01],
        [ -4.57905470e-01,   1.22408252e-01,  -2.71059555e-01,  ...,
          -2.48096816e-03,  -1.44468486e-01,  -1.36615031e-01],
        ...,
        [ -4.57905470e-01,  -5.00908130e-01,   3.68922615e+00,  ...,
          -1.18093946e-01,  -1.44468486e-01,   1.15224865e-01],
        [  2.18385686e+00,  -1.09238838e-01,  -2.71059555e-01,  ...,
          -1.69922585e-01,  -1.44468486e-01,  -1.36615031e-01],
        [ -4.57905470e-01,   1.19855599e+00,  -2.71059555e-01,  ...,
          -2.76493450e-01,  -1.44468486e-01,  -1.36615031e-01]])

]: x_train_6,x_test_6,y_train_6,y_test_6=train_test_split(
    x_scaled,y,test_size=0.2,random_state=10)

x_train_6.shape,x_test_6.shape,y_train_6.shape,y_test_6.shape

]: ((3600, 9), (900, 9), (3600,), (900,))

```



# **Chapter 6**

## **Model Building**

The modelling process was divided into two main parts: Building different model of machine learning to find out best algorithm and hyper parameter optimization of selected model.

### **6.1 Building Different Model**

In Machine Learning, we have different classification algorithm. Few of them were basic algorithms such as Decision Tree, and Logistic Regression. Also based on this basic algorithm we have different ensemble algorithms namely Random Forest, Boost, Ad boost, Support Vector Machine, Bagging Classifier, Gradient Boosting.

#### **Base line model**

Models without sampling, scaling and outlier treatment. Models were not able to accurately find the best stocks due to heavy unbalanced target attribute.

Below are the results.

Model	ROC	Accuracy
Logistic	65.09	69
Decision tree	64.93	65
Random Forest	69.45	70
SVM	62.13	64
KNN	59.39	67
K-Means	28.03	51
G-Boosting	69.03	69

## Model building with sampling techniques and standardization

Then we build ML models with various sampling techniques. There was major improvement in results.

Model	Precision	Recall	f1-score	ROC	Accuracy
Logistic	0.71	0.74	0.71	63.04	74
Decision tree	0.79	0.77	0.77	78.03	77
Random Forest	0.82	0.81	0.81	81.77	81
SVM	0.69	0.69	0.69	69.12	70
KNN	0.72	0.72	0.72	72.03	72
K-Means	0.41	0.28	0.27	51.84	28
G-Boosting	0.75	0.74	0.74	74.45	74

Random Forest under sampler gave better result as compared to other sampling techniques. To improve model performance, we performed hyper parameter optimization on machine learning models.

## 6.2 Hyper parameter Optimization

A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters.

However, there is another kind of parameters, known as hyper parameters, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.

Machine learning models are not intelligent enough to know what hyperparameters would lead to the highest possible accuracy on the given dataset. However, hyperparameter values when set right can build highly accurate models, and thus we allow our models to try different combinations of hyperparameters during the training process and make predictions with the best combination of hyperparameter values. Some of the hyperparameters in Random Forest Classifier are n estimators (total number of trees in a forest), max depth (the depth of each tree in the forest), and criterion (the method to make splits in each tree). n estimators set

to 1 or 2 doesn't make sense as a forest must have a higher number of trees.

## 6.2.1 Random Search and Grid Search

Random Search replaces the exhaustive enumeration of all combinations by selecting them randomly. This can be simply applied to the discrete setting described above, but also generalizes to continuous and mixed spaces. It can outperform Grid search, especially when only a small number of hyper parameters affects the final performance of the machine learning algorithm. In this case, the optimization problem is said to have a low intrinsic dimensionality. Random Search is also embarrassingly parallel, and additionally allows the inclusion of prior knowledge by specifying the distribution from which to sample.

Grid Search uses a different combination of all the specified hyperparameters and their values and calculates the performance for each combination and selects the best value for the hyperparameters.

This makes the processing time-consuming and expensive based on the number of hyperparameters involved. In GridSearchCV, along with Grid Search, cross-validation is also performed. Cross-Validation is used while training the model. As we know that before training the model with data, we divide the data into two parts – train data and test data. In cross-validation, the process divides the train data further into two parts – the train data and the validation data.

### Extensive Hyperparameter tuning ¶

```

: from sklearn.model_selection import GridSearchCV
: rfc_gs=GridSearchCV(rfc,{'n_estimators':range(75,125),
:                       'criterion':['gini','entropy','log_loss']})
:
: rfc_gs.fit(x_train_6,y_train_6)
:
: GridSearchCV(estimator=RandomForestClassifier(random_state=100),
:               param_grid={'criterion': ['gini', 'entropy', 'log_loss'],
:                             'n_estimators': range(75, 125)})
:
: rfc_gs.best_params_
: {'criterion': 'gini', 'n_estimators': 83}
:
: rfc_new=RandomForestClassifier(n_estimators=83,
:                               criterion='gini',random_state=10)
:
: rfc_new.fit(x_train_6,y_train_6)
: RandomForestClassifier(n_estimators=83, random_state=10)

```

Fig: Implementing the Grid search

```

: # Performance

cm=confusion_matrix(y_test_6,rfc_new.predict(x_test_6))
report=classification_report(y_test_6,rfc_new.predict(x_test_6))
score=roc_auc_score(y_test_6,rfc_new.predict(x_test_6))
print('CM:\n',cm)
print('Report:\n',report)
print('ROC-AUC Curve',score)

CM:
[[340  61]
 [106 393]]
Report:

```

	precision	recall	f1-score	support
0	0.76	0.85	0.80	401
1	0.87	0.79	0.82	499
accuracy			0.81	900
macro avg	0.81	0.82	0.81	900
weighted avg	0.82	0.81	0.81	900

```

ROC-AUC Curve 0.8177277247762358

```

The traditional way of performing hyperparameter optimization has been grid search, or a parameter sweep, which is simply an exhaustive searching through a manually specified subset of the hyperparameter space of a learning algorithm. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set or evaluation on a hold-out validation set. Since the parameter space of a machine learner may include real-valued or unbounded value spaces for certain parameters, manually set bounds and discretization may be necessary before applying grid search.

After we have implemented all this algorithm to find out the best features and make a data frame with feature importance to check the weightage of each feature.

```

rfc_new.feature_importances_

array([0.02613187, 0.3785973 , 0.01877922, 0.01343753, 0.12984544,
       0.0129432 , 0.28852913, 0.00535603, 0.12638027])

# Creating a DF

df=pd.DataFrame({'Feature':x.columns,'Feature Imp':rfc_new.feature_importances_})
df

```

	Feature	Feature Imp
0	Sector_Financial Services	0.026132
1	longtermDebtToCapitalization	0.378597
2	Sector_Energy	0.018779
3	Sector_Healthcare	0.013438
4	Dividend per Share	0.129845
5	Sector_Real Estate	0.012943
6	priceEarningsRatio	0.288529
7	Sector_Utilities	0.005356
8	Dividend Yield	0.126380

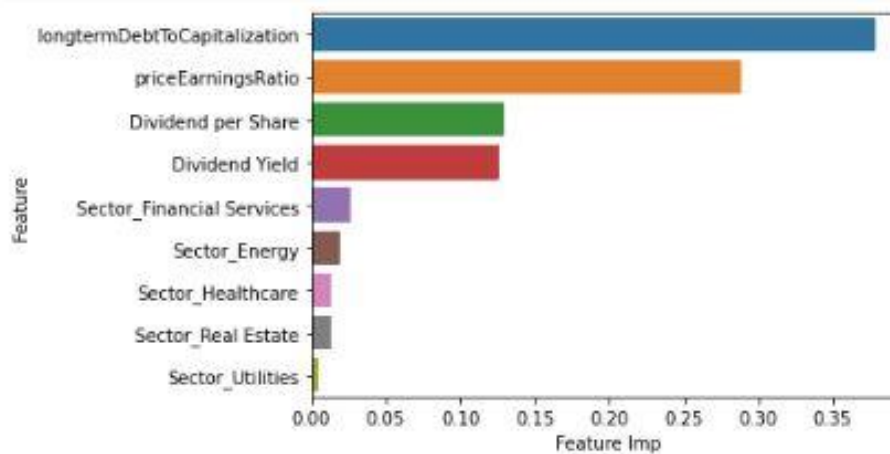
In [134]: # Adding one column

```
df['Feature Imp Cum']=df['Feature Imp'].cumsum()  
df
```

Out[134]:

	Feature	Feature Imp	Feature Imp Cum
1	longtermDebtToCapitalization	0.378597	0.378597
6	priceEarningsRatio	0.288529	0.667126
4	Dividend per Share	0.129845	0.796972
8	Dividend Yield	0.126380	0.923352
0	Sector_Financial Services	0.026132	0.949484
2	Sector_Energy	0.018779	0.968263
3	Sector_Healthcare	0.013438	0.981701
5	Sector_Real Estate	0.012943	0.994644
7	Sector_Utilities	0.005356	1.000000

```
: sns.barplot(x=df['Feature Imp'],y=df['Feature'],data=df);
```



From above chart we can see the feature and their importance. After this we perform Model deployment on Stream lit platform using obtained important features to check whether to sell or buy/hold stocks after analyzing the year data.

# **Chapter 7**

## **Deployment**

### **Streamlit**

A Streamlit is an open-source app framework in python language. It helps us create beautiful web apps for data science and machine learning in a little time. It is compatible with major python libraries such as scikit-learn, keras, PyTorch, latex, NumPy, pandas, matplotlib, etc.

Flask and Django are somewhat heavy so it takes more than one article and time to understand them better (we would be focusing on this one also), but for now, we would be discussing Streamlit. So, let's start with a question.

### **Why Streamlit?**

- Streamlit lets you create apps for your Machine Learning project using simple code.
- It also supports hot-reloading that lets your app update live as you edit and save your file.
- Using streamlit creating an app is very easy, adding a widget is as simple as declaring a variable.
- No need to write a backend, no need to define different routes or handle HTTP requests.

To start deployment, install stream lit in your anaconda environment and also in PyCharm.



```
(base) C:\Users\Akhil>pip install streamlit
Collecting streamlit
  Downloading streamlit-1.20.0-py2.py3-none-any.whl (9.6 MB)
----- 9.6/9.6 MB 44.4 kB/s eta 0:00:00
Collecting protobuf<4,>=3.12
  Downloading protobuf-3.20.3-cp39-cp39-win_amd64.whl (904 kB)
----- 904.2/904.2 kB 51.3 kB/s eta 0:00:00
Requirement already satisfied: packaging>=14.1 in c:\users\akhil\anaconda3\lib\site-packages (from streamlit) (21.3)
Collecting blinker>=1.0.0
  Downloading blinker-1.5-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: tornado>=6.0.3 in c:\users\akhil\anaconda3\lib\site-packages (from streamlit) (6.1)
Requirement already satisfied: numpy in c:\users\akhil\anaconda3\lib\site-packages (from streamlit) (1.21.5)
Collecting pyarrow>=4.0
  Downloading pyarrow-11.0.0-cp39-cp39-win_amd64.whl (20.6 MB)
----- 20.6/20.6 MB 41.2 kB/s eta 0:00:00
Requirement already satisfied: python-dateutil in c:\users\akhil\anaconda3\lib\site-packages (from streamlit) (2.8.2)
Collecting rich>=10.11.0
  Downloading rich-13.3.2-py3-none-any.whl (238 kB)
----- 238.7/238.7 kB 100.1 kB/s eta 0:00:00
Requirement already satisfied: typing-extensions>=3.10.0.0 in c:\users\akhil\anaconda3\lib\site-packages (from streamlit) (4.3.0)
Collecting gitpython<=3.1.19
  Downloading GitPython-3.1.31-py3-none-any.whl (184 kB)
----- 184.3/184.3 kB 40.8 kB/s eta 0:00:00
Requirement already satisfied: pillow>=6.2.0 in c:\users\akhil\anaconda3\lib\site-packages (from streamlit) (9.2.0)
Requirement already satisfied: click>=7.0 in c:\users\akhil\anaconda3\lib\site-packages (from streamlit) (8.0.4)
Collecting semver
  Downloading semver-2.13.0-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: toml in c:\users\akhil\anaconda3\lib\site-packages (from streamlit) (0.10.2)
Requirement already satisfied: pandas<2,>=0.25 in c:\users\akhil\anaconda3\lib\site-packages (from streamlit) (1.4.4)
Requirement already satisfied: watchdog in c:\users\akhil\anaconda3\lib\site-packages (from streamlit) (2.1.6)
Requirement already satisfied: importlib-metadata>=1.4 in c:\users\akhil\anaconda3\lib\site-packages (from streamlit) (4.11.3)
Collecting validators>=0.2
  Downloading validators-0.20.0.tar.gz (30 kB)
  Preparing metadata (setup.py) ... done
Collecting altair<5,>=3.2.0
  Downloading altair-4.2.2-py3-none-any.whl (813 kB)
----- 813.6/813.6 kB 50.2 kB/s eta 0:00:00
Collecting cachetools>=4.0
  Downloading cachetools-4.2.1-py3-none-any.whl (10.3 kB)
----- 10.3/10.3 kB 10.3 kB/s eta 0:00:00
```

Create a folder to store csv data file and the python code file of the project in one single folder.

This PC > Desktop > Stocks

Name	Date modified	Type	Size
.venv	13-03-2023 14:12	File folder	
2018_US_Stock_Data	24-02-2023 15:37	Microsoft Excel C...	6,240 KB
app5	12-03-2023 23:56	JetBrains PyChar...	18 KB
Exp3	14-03-2023 00:00	JetBrains PyChar...	8 KB
Exp4	14-03-2023 00:16	JetBrains PyChar...	8 KB
Exp5	14-03-2023 09:02	JetBrains PyChar...	8 KB
Exp6	14-03-2023 11:39	JetBrains PyChar...	8 KB
Exp7	14-03-2023 11:39	JetBrains PyChar...	8 KB

Now install necessary libraries related to streamlit and python code.

```
# Import necessary libraries
import pandas as pd
import streamlit as st
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_auc_score
# Load the dataset
```

Now, using the previous code till model evaluation and hyper parameter tuning write the code related to app in order to check the status of stocks using the feature importance that we got at the end of model evaluation.

## App:1]

```
# Define the Streamlit app
def app():
    st.title("Stock Predictor")
    st.write("Use this app to predict whether to buy, sell, or hold a stock based on its financial indicators.")

    # Display the feature importances
    st.subheader("Feature Importances")
    imp_df = pd.DataFrame({"Feature": features, "Importance": rfc_new.feature_importances_})
    imp_df = imp_df.sort_values("Importance", ascending=False)
    st.dataframe(imp_df)

    st.subheader("Top 10 Stocks to Buy or Hold")
    buy_stocks = []
    for index, row in stock_new.iterrows():
        Stock_data = row[features].to_frame().T
        prediction = rfc_new.predict(Stock_data)
        if prediction == 1:
            buy_stocks.append({"Name": row['Name']})
    buy_stocks = sorted(buy_stocks, key=lambda x: x["Name"])[-10:]
    for stock in buy_stocks:
        st.write(f"{stock['Name']} - Buy/Hold")
```

```
st.subheader("Top 10 Stocks to Sell")
sell_stocks = []
for index, row in stock_new.iterrows():
    sStock_data = row[features].to_frame().T
    prediction = rfc_new.predict(sStock_data)
    if prediction != 1:
        sell_stocks.append({"Name": row['Name']})
sell_stocks = sorted(sell_stocks, key=lambda x: x["Name"])[-10:]
for stock in sell_stocks:
    st.write(f"{stock['Name']} - Sell")

# Run the app
if __name__ == '__main__':
    app()
```



Run the code in anaconda environment by using “streamlit run file path”

```
(base) C:\Users\Akhil>streamlit run C:\Users\Akhil\Desktop\Stocks\Exp6.py
```

You can now view your Streamlit app in your browser.

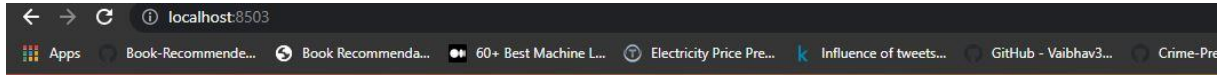
Local URL: <http://localhost:8503>  
 Network URL: <http://192.168.0.107:8503>

Optimization terminated successfully.  
 Current function value: 0.527870  
 Iterations 21

Optimization terminated successfully.  
 Current function value: 0.559342  
 Iterations 8

Optimization terminated successfully.  
 Current function value: 0.560416  
 Iterations 8

Optimization terminated successfully.  
 Current function value: 0.560631  
 Iterations 7



## Stock Predictor

Use this app to predict whether to buy, sell, or hold a stock based on its financial indicators.

### Feature Importances

	Feature	Importance
6	longtermDebtToCapitalization	0.3728
1	priceEarningsRatio	0.2937
0	Dividend per Share	0.1352
5	Dividend Yield	0.1245
8	Sector_Financial Services	0.0245
2	Sector_Energy	0.0192
4	Sector_Healthcare	0.0133
7	Sector_Real Estate	0.0127
3	Sector_Utilities	0.0041

## Top 10 Stocks to Buy or Hold

AA - Buy/Hold

AABA - Buy/Hold

AAL - Buy/Hold

AAMC - Buy/Hold

AAME - Buy/Hold

AAN - Buy/Hold

AAOI - Buy/Hold

AAON - Buy/Hold

AAP - Buy/Hold

AAWW - Buy/Hold

## Top 10 Stocks to Sell

AAU - Sell

ABIL - Sell

ACTG - Sell

ADOM - Sell

AEY - Sell

AGI - Sell

AGYS - Sell

AIRG - Sell

AKCA - Sell

AKG - Sell

## App:2]

Here you are allowed to provide the values of the parameters and after calculating the app will provide the result whether to sell or buy/hold

```
# Define the Streamlit app
def app():
    st.title("Stock Predictor")
    st.write("Use this app to predict whether to buy, sell, or hold a stock based on its financial indicators.")

    # Display the feature importances
    st.subheader("Feature Importances")
    imp_df = pd.DataFrame({"Feature": features, "Importance": importances})
    imp_df = imp_df.sort_values("Importance", ascending=False)
    st.dataframe(imp_df)

    # Allow the user to input the stock data
    st.subheader("Enter Stock Data")
    data_input = {}
    for feature in features:
        data_input[feature] = st.number_input(f"Enter {feature}")
    data_input = pd.DataFrame(data_input, index=[0])

    # Predict the action and display the result
    prediction = rfc_new.predict(data_input)
    if prediction[0] == 0:
        st.subheader("Prediction")
        st.write("Hold")
    else:
        st.subheader("Prediction")
        st.write("Sell")

# Run the app
if __name__ == '__main__':
    app()
```

**App2 Interface:**

## Stock Predictor

Use this app to predict whether to buy, sell, or hold a stock based on its financial indicators.

### Feature Importances

	Feature	Importance
3	longtermDebtToCapitalization	0.3797
5	priceEarningsRatio	0.2965
4	Dividend Yield	0.1267
8	Dividend per Share	0.1255
6	Sector_Financial Services	0.0249
2	Sector_Energy	0.0167
0	Sector_Healthcare	0.0127
1	Sector_Real Estate	0.0123
7	Sector_Utillities	0.005

## Enter Stock Data

Enter Sector\_Healthcare

0.11

- +

Enter Sector\_Real Estate

0.71

- +

Enter Sector\_Energy

0.36

- +

Enter longtermDebtToCapitalization

0.30

- +

Enter Dividend Yield

6.30

- +

Enter priceEarningsRatio

4.30

- +

Enter Sector\_Financial Services

1.36

- +

Enter Sector\_Utilities

0.33

- +

Enter Dividend per Share

0.09

- +

## Recommendation

This stock is a hold.

# **Chapter 8**

## **Conclusion**

- Most important features are obtained using Logistic Regression.
- There are 9 important parameters in total.
- We were able to predict stock is whether to buy/hold or sell based on the feature importance.
- Out of 7 Models Performed, best resulting model is Random Forest with Accuracy 81.09 % and recall 81%. The motivation for the study was to find the most efficient ML algorithm to find the important parameters so that they can be used to check which stock to buy/hold or sell. This study compares the different evaluation metrics of ensemble algorithms such as Random Forest, GradientBoosting, XG Boost, Decision tree, SVM algorithms for predicting. The result of this study indicates that the Random Forest algorithm is the most efficient algorithm for the prediction of stock status.
- Finally, we built an app to deploy the result using streamlit.

# References

- [1] E. Hjalmarsson, "On the Predictability of Global Stock Returns," School of Business, Economics and Law. Goteborg University, Gothenburg, 2005” A Tour of Machine Learning Algorithms” by Jason Brownlee from <https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/>
- [2] Chih-Fong Tsai, Yuah-Chiao Lin, David C. Yen, Yan-Min Chen, "Predicting stock returns by classifier ensembles", Applied Soft Computing, Volume 11, Issue 2, 201
- [3] K. S. Loke, "Impact of financial ratios and technical analysis on stock price prediction using random forests," 2017 International Conference on Computer and Drone Applications (IConDA), 2017, pp. 38-42, DOI: 10.1109/ICONDA.2017.8270396
- [4] A. Fan and M. Palaniswami, "Stock selection using support vector machines," IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222), 2001, pp. 1793-1798 vol.3, doi: 10.1109/IJCNN.2001.938434.
- [5] S. Ravikumar and P. Saraf, "Prediction of Stock Prices using Machine Learning (Regression, Classification) Algorithms," 2020 International Conference for Emerging Technology (INCET), 2020, pp. 1-5, doi: 10.1109/INCET49848.2020.9154061.
- [6] Deepali Vora, N. S. (Volume 163 – No 5, April 2017). Stock Prediction using Machine Learning a Review. International Journal of Computer Applications (0975 – 8887), 36-43
- [7] A. Phongmekin and P. Jarumaneeroj, "Classification Models for Stock's Performance Prediction: A Case Study of Finance Sector in the Stock Exchange of Thailand," 2018 International Conference on Engineering, Applied Sciences, and Technology (ICEAST), 2018, pp. 1-4, doi: 10.1109/ICEAST.2018.843439