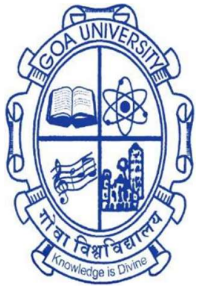


Goa University

Goa Business School

Department of Computer Science & Technology

PROJECT 2022-23

Conducted in field of Machine Leaning

Stock Recommendation System using Machine Learning for Choosing the Best Stocks from a Given Pool "Top Picks"

Submitted By

Shubham Gaonkar - 2262

Shreevesh Naik - 2257

Amberly Silva - 2246

Grishma Chodanker - 2259

Certificate

This is to certify that the project entitled **"Stock Recommendation System"** under my guidance. The project involved the development of a stock recommendation system using machine learning techniques.

Shreevesh Naik	2257
Shubham Gaonkar	2262
Grishma Chodanker	2259
Amberly Silva	2246

They have effectively utilized various tools and technologies to build a robust system that provides accurate stock recommendations based on historical data and technical indicators.

_____

Prof. S. Baskar

(Supervisor)

ABSTRACT

The purpose of this project is to develop a recommendation system that suggests the best four stocks to invest in, from a given set of ten stocks. The system uses various financial indicators such as moving averages, relative strength index, and stochastic oscillator, to analyse the performance of each stock. The system then employs a machine learning algorithm to predict the future stock prices and evaluate their potential for profitable investments. The project aims to assist investors, particularly those with limited financial knowledge, in making informed decisions and maximizing their returns. The system is implemented using Python programming language, and the data is obtained using Yahoo Finance API. The results of the analysis are presented in an interactive. Overall, the project has the potential to provide significant value to investors by reducing the risks associated with investment decisions and increasing their chances of making profitable investments.

ACKNOWLEDGEMENT

Acknowledgments are an important part of any project, as they recognize the contributions and support of those who have helped along the way. I would like to extend my sincere gratitude to our project incharge and professor, Mr. S. Bhaskar, for providing us with this valuable opportunity to enhance our practical knowledge. His guidance and support were invaluable throughout the project.

I would also like to thank my classmates for their constant moral support and confidence. They provided us with all the necessary data formats, full guidance, and other valuable assistance, which helped us to complete this project successfully.

I am open to suggestions for further improvement and would be grateful to receive any feedback. Once again, I would like to express my heartfelt thanks to everyone who has contributed to this project.

TABLE OF CONTENT

1. OBJECTIVE

The objective of this project is to develop a recommendation system that helps investors make informed decisions by suggesting the best 4 stocks from a given set of 10 stocks based on various factors such as historical stock prices, technical indicators, and market trends.

Examine different stock price predictions done using machine learning based on past returns.

The project aims to provide an easy-to-use tool that can assist users in optimizing their investment portfolio and maximizing their returns.

2. LIBRARIES USED

1. yfinance: a library for downloading historical market data from Yahoo Finance. It provides a simple and easy-to-use interface for accessing financial data in Python.

2. pandas: a library for data manipulation and analysis. It provides data structures for efficiently storing and manipulating large datasets, as well as functions for data cleaning, preprocessing, and analysis.

3. mplfinance: a library for plotting financial charts and technical indicators. It provides a simple and easy-to-use interface for creating candlestick charts, line charts, and other types of financial charts.

4. plotly: a library for creating interactive visualizations in Python. It provides a range of chart types, including candlestick charts, and allows for customization of chart elements, such as colors, annotations, and legends.

5. sklearn: a machine learning library that provides tools for data mining and data analysis. It includes a wide range of algorithms for tasks such as classification, regression, clustering, and dimensionality reduction. It also includes tools for data preprocessing, model selection, and evaluation.

6. Math: is used for simple mathematical calculations and basic statistical analysis.

3. DATA PROCESSING

```
#Dataset
dataset = yf.Ticker("PG")
dataset = dataset.history(start="2020-01-01", end="2023-05-10")

del dataset["Dividends"]
del dataset["Stock Splits"]

dataset.isnull().sum()
```

This retrieves the historical stock data of Procter & Gamble (PG) from Yahoo Finance from January 1, 2020, to May 10, 2023. The data is stored in a pandas DataFrame named "dataset". The two columns, "Dividends" and "Stock Splits", are removed from the dataset because they are not relevant to this analysis.

The "isnull().sum()" method is then used to check if there are any missing values in the dataset. This method calculates the total number of missing values for each column, and "sum()" adds them up. If there are any missing values in the dataset, this will tell us how many.

Overall, this code downloads the historical stock data of Procter & Gamble, prepares it by removing unnecessary columns.

4. MODEL USED

A. Random Forest

```
#Random Forest
from sklearn.ensemble import RandomForestRegressor

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train the random forest model on the training set
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(X_train, y_train)

# Predict the testing set
predicted = rf_regressor.predict(X_test)

# add predicted column to dataset
dataset['Predicted'] = rf_regressor.predict(X)

# print the evaluation metrics
print('Mean Absolute Error:' , metrics.mean_absolute_error(y_test, predicted))
print('Mean Squared Error:' , metrics.mean_squared_error(y_test, predicted))
print('Root Mean Squared Error:', math.sqrt(metrics.mean_squared_error(y_test, predicted)))
```

```
Mean Absolute Error: 0.706094738504161
Mean Squared Error: 1.0445277555070276
Root Mean Squared Error: 1.022021406579641
```

This implements a **Random Forest Regression** model for stock price prediction using the scikit-learn library in Python. It uses the RandomForestRegressor class to create a model with 100 decision trees (n_estimators) and a fixed random seed value (random_state) of 42.

The dataset is split into training and testing sets using the train_test_split function from scikit-learn, with a test size of 0.3 and a random state of 42. The model is trained on the training set using the fit method of the Random Forest Regressor.

Once the model is trained, it is used to predict the stock prices on the testing set using the predict method. The predicted values are then added as a new column to the original dataset.

Finally, evaluation metrics such as mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE) are printed using the metrics module of scikit-learn and the math library. These metrics provide information on how well the model is performing on the testing set. The lower the values for these metrics, the better the model's performance.

B. Linear Regression

```
#Linear Regression
# Train the model on the whole dataset
regressor = LinearRegression()
regressor.fit(X, y)

# Predict the testing set
predicted = regressor.predict(X_test)

# add predicted column to dataset
dataset['Predicted'] = regressor.predict(X)

# print the evaluation metrics
print('Mean Absolute Error:' , metrics.mean_absolute_error(y_test, predicted))
print('Mean Squared Error:' , metrics.mean_squared_error(y_test, predicted))
print('Root Mean Squared Error:', math.sqrt(metrics.mean_squared_error(y_test, predicted)))
```

```
Mean Absolute Error: 0.5017855114523866
Mean Squared Error: 0.4796092273389082
Root Mean Squared Error: 0.6925382497298674
```

This code is performing **linear regression analysis** on the dataset. Linear regression is a type of supervised machine learning algorithm used to predict a continuous target variable.

The code first creates a linear regression object by calling LinearRegression() function from the sklearn library. The model is then trained on the entire dataset using the fit() function, where X represents the features and y represents the target variable.

Next, the model is used to make predictions on the test set using the predict() function. The predicted values are stored in the variable predicted.

The code then adds a new column called "Predicted" to the dataset by predicting the target variable values for the entire dataset using the predict() function.

Finally, the it prints the evaluation metrics for the model performance on the test set. The evaluation metrics used here are Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). These metrics measure the difference between the actual target variable values and the predicted values by the model. A lower value of these metrics indicates a better performance of the model.

Compare Random Forest and Linear Regression Model

Based on the parameters, the **Linear Regression model** has a lower MAE, MSE, and RMSE than the Random Forest model. This suggests that the Linear Regression model is slightly more accurate in predicting the stock prices than the Random Forest model.

5. TRAINING

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

# Train the model on the whole dataset
regressor = LinearRegression()
regressor.fit(X, y)
```

The training of dataset starts with splitting the dataset into training and testing sets using the train_test_split function. The training set is used to train the model, while the testing set is used to evaluate the model's performance. Then, a Linear Regression model is initialized, and the fit method is called on the training set to train the model.

The entire dataset is also used for training the model by calling the fit method on the input features and target variable of the entire dataset. The model is trained to predict the next trading day's closing price using the predict method on the Open, High, Low, and Volume of the latest trading day.

6. TESTING AND VALIDATION

```
# Predict the next trading day's closing price
next_day = yf.download(ticker, period="2d")
next_day_pred = regressor.predict(next_day.iloc[-2][['Open', 'High', 'Low', 'Volume']].values.reshape(1, -1))[0]
print(f"\n\nSymbol: {ticker}")
print('Next day predicted price:', next_day_pred)

# add predicted column to dataset
dataset['Predicted'] = regressor.predict(X)

# Predict the testing set
predicted = regressor.predict(X_test)
```

```
# Predict the next trading day's closing price
next_day = yf.download(ticker, period="2d")
next_day_pred = regressor.predict(next_day.iloc[-2][['Open', 'High', 'Low', 'Volume']].values.reshape(1, -1))[0]
print(f"\n\nSymbol: {ticker}")
print('Next day predicted price:', next_day_pred)
```

```
# Calculate annualized returns
start_date = dataset.index.min()
end_date = dataset.index.max()
num_years = (end_date - start_date).days / 252
annualized_return = (((y[-1] / y[0]) ** (1 / num_years)) - 1)*100
formatted_return_percent = '{:.2f}'.format(annualized_return)

# print the evaluation metrics
print('Mean Absolute Error:' , metrics.mean_absolute_error(y_test, predicted))
print('Mean Squared Error:' , metrics.mean_squared_error(y_test, predicted))
print('Root Mean Squared Error:', math.sqrt(metrics.mean_squared_error(y_test, predicted)))
# Print annualized returns
print('Annualized Return:', formatted_return_percent)
```

The target variable is predicted for the testing set using the predict method on the input features of the testing set. The performance of the model is evaluated by calculating the MAE, MSE, and RMSE metrics using the mean_absolute_error, mean_squared_error, and sqrt functions from the metrics and math libraries, respectively. The annualized returns of the stock are calculated using the formula:

((Closing Price at the end of the period / Closing Price at the beginning of the period)^(1/number of years))-1

where the number of years is calculated by dividing the number of days in the dataset by 252, which is the number of trading days in a year.

7. EVALUATION OF MODEL USING PLOTS

```
# plot actual v/s predicted line chart with EMAs
fig = plt.figure(figsize=(12, 8))
fig.set_facecolor('white')
ax = fig.add_subplot(111)
compare[['Predicted', 'Close', 'EMA50', 'EMA200']].plot(kind='line', ax=ax, linewidth=0.75,
color={'Predicted': 'red', 'Close': 'blue', 'EMA50': 'green', 'EMA200': 'red'})
ax.set_title(f'Actual v/s Predicted Stock Prices for {ticker}', fontsize=20, color='black')
ax.set_facecolor('white')
ax.set_xlabel('Date', fontsize=16, color='white')
ax.set_ylabel('Closing Price', fontsize=16, color='white')
ax.tick_params(axis='both', which='major', labelsize=12, color='black')
labelcolor = 'black'
ax.legend(['Close', 'Predicted', 'EMA50', 'EMA200'], fontsize=16)
ax.set_xlim(pd.to_datetime('2021-01-01'), pd.to_datetime('2023-05-06'))
plt.show()
```

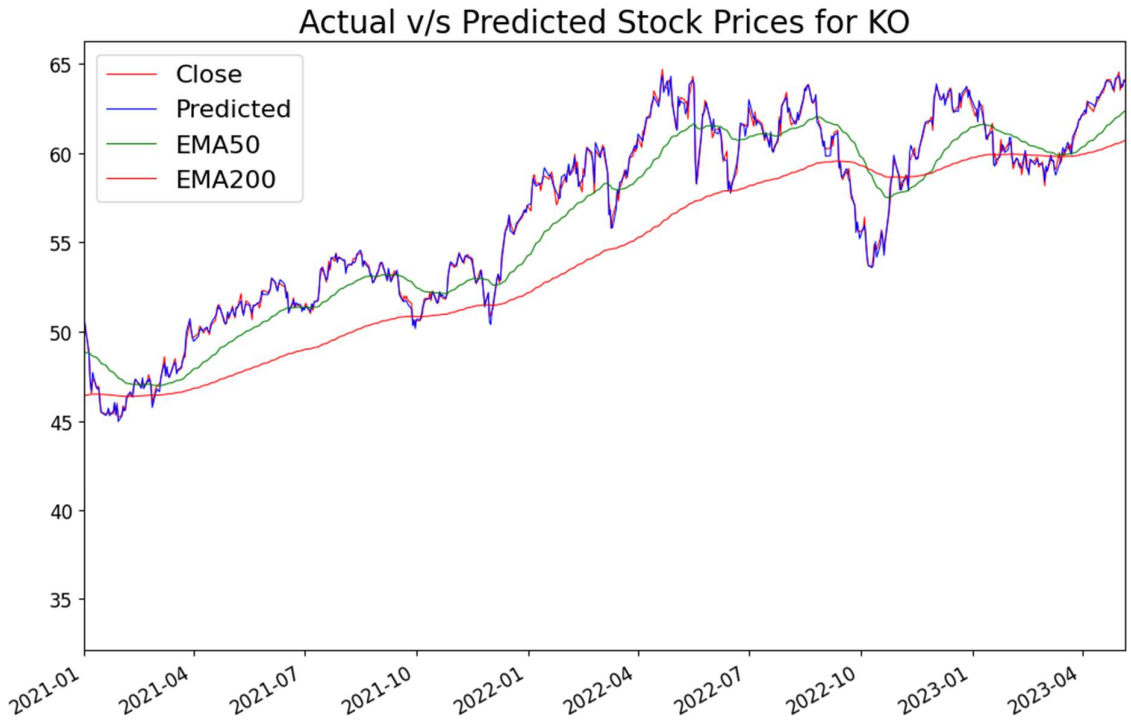
The evaluation of models using plots involves plotting the actual and predicted closing prices of the stock on a line chart with the 50-day and 200-day EMAs. The compare dataframe is created by selecting the Close, Predicted, EMA50, and EMA200 columns from the dataset. The line chart is plotted using the plot method on the compare dataframe, with the kind parameter set to 'line' and the linewidth parameter set to 0.75. The colors of the lines are set to blue for the Close, red for the Predicted, and green for the EMA50 and EMA200. The title, x-axis label, y-axis label, and legend are added using the set_title, set_xlabel, set_ylabel, and legend methods, respectively, on the ax object returned by the add_subplot method of the fig object. The limits of the x-axis are set to show data from January 1, 2021, to May 6, 2023, using the set_xlim method on the ax object. Finally, the line chart is displayed using the show method of the plt module.

Overall, the code trains a Linear Regression model to predict the closing price of a stock based on its open, high, low, and volume values,

I. Coca-Cola Company (KO)

```
Symbol: KO
Next day predicted price: 63.8780573486709
Mean Absolute Error: 0.15806150739089306
Mean Squared Error: 0.049910323210629604
Root Mean Squared Error: 0.2234061843607504
Annualized Return: 6.64
```

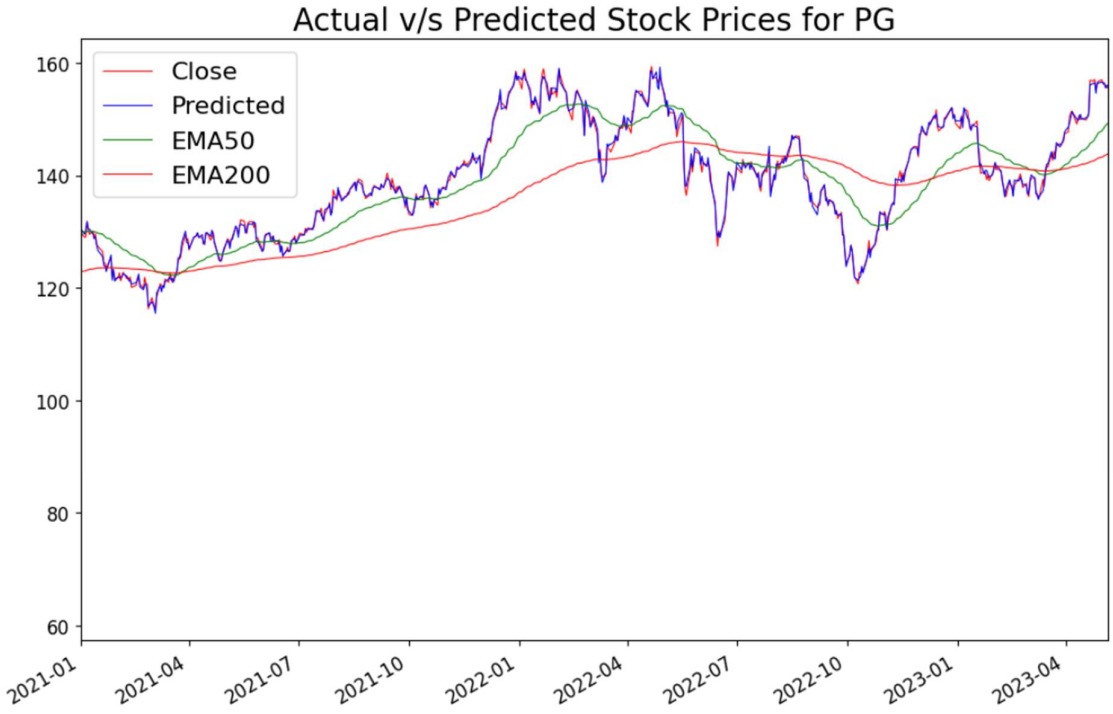
Date	Close	Predicted	EMA50	EMA200
2018-01-02 00:00:00-05:00	38.548206	38.618709	38.548206	38.548206
2018-01-03 00:00:00-05:00	38.463554	38.535645	38.544887	38.547364
2018-01-04 00:00:00-05:00	39.005310	38.913415	38.562942	38.551921
2018-01-05 00:00:00-05:00	38.996830	38.919916	38.579958	38.556348
2018-01-08 00:00:00-05:00	38.937576	38.943743	38.593982	38.560141
...	...	...	...	...
2023-05-03 00:00:00-04:00	63.650002	63.836462	62.177180	60.607028
2023-05-04 00:00:00-04:00	63.720001	63.741890	62.237683	60.638002
2023-05-05 00:00:00-04:00	64.019997	64.096141	62.307578	60.671654
2023-05-08 00:00:00-04:00	63.919998	63.878057	62.370810	60.703976
2023-05-09 00:00:00-04:00	63.450001	63.566505	62.413131	60.731300



II. Procter & Gamble (PG)

```
Symbol: PG
Next day predicted price: 155.34643674790095
Mean Absolute Error: 0.4067819879643376
Mean Squared Error: 0.31776084960608775
Root Mean Squared Error: 0.5637028025529833
Annualized Return: 9.14
```

Date	Close	Predicted	EMA50	EMA200
2018-01-02 00:00:00-05:00	78.130028	78.353437	78.130028	78.130028
2018-01-03 00:00:00-05:00	78.035217	78.364604	78.126310	78.129084
2018-01-04 00:00:00-05:00	78.586830	78.795643	78.144369	78.133639
2018-01-05 00:00:00-05:00	78.638542	78.286057	78.163749	78.138663
2018-01-08 00:00:00-05:00	79.052238	78.986858	78.198591	78.147753
...	...	...	...	...
2023-05-03 00:00:00-04:00	156.229996	156.431819	148.547171	143.480907
2023-05-04 00:00:00-04:00	155.509995	155.754962	148.820223	143.600599
2023-05-05 00:00:00-04:00	156.029999	155.597155	149.102960	143.724275
2023-05-08 00:00:00-04:00	155.300003	155.346437	149.345981	143.839456
2023-05-09 00:00:00-04:00	153.869995	154.238292	149.523393	143.939262



➢ Comparison and Selection of Top Performing Stocks

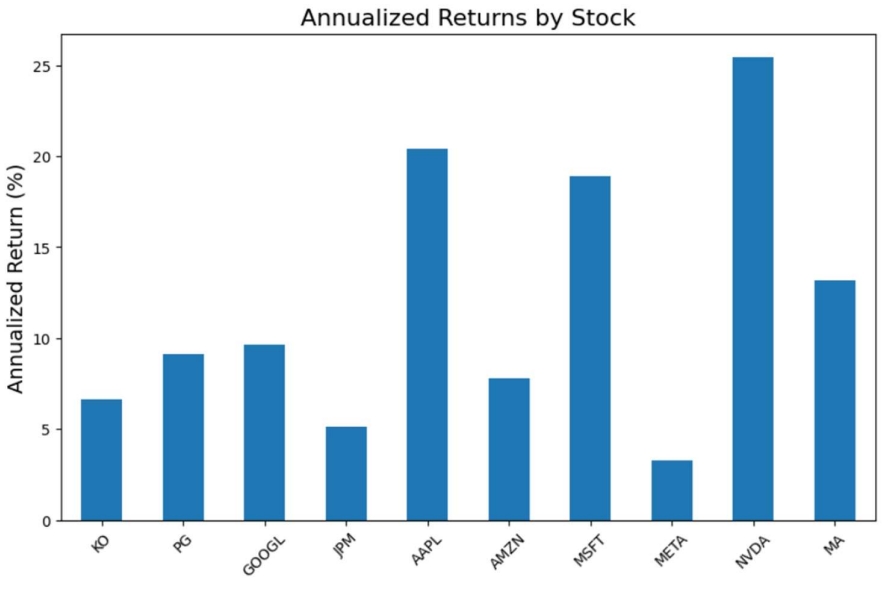
```
Comparison of all stocks according to parameters:
            MAE       MSE      RMSE Annualized Return
KO     0.158061  0.049910  0.223406              6.64
PG     0.406531  0.317589  0.563550              9.15
GOOGL  0.447553  0.378617  0.615318              9.63
JPM    0.496875  0.419576  0.647747              5.16
AAPL   0.497392  0.505722  0.711141             20.41
AMZN   0.656685  0.790528  0.889116              7.81
MSFT   0.912916  1.760892  1.326986             18.90
META   1.350710  3.408664  1.846257              3.29
NVDA   1.114776  3.593590  1.895677             25.45
MA     1.532847  4.384404  2.093897             13.20

Top 4 stocks:
KO
PG
GOOGL
JPM
```

We compares the results of different stocks based on the evaluation parameters, including the root mean square error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), and annualized return.

The results are presented in a Pandas DataFrame that is sorted in ascending order based on the RMSE values. The code also prints out the top 4 stocks based on the RMSE values.

This can be useful in identifying the stocks that perform well according to the specified evaluation metrics and can be considered for further analysis or investment purposes.



➢ Creating Candlestick charts with EMA lines

I. KO

II. PG

III. GOOGL

➢ Allocation and Visualization of Investment in Top 4 Stocks

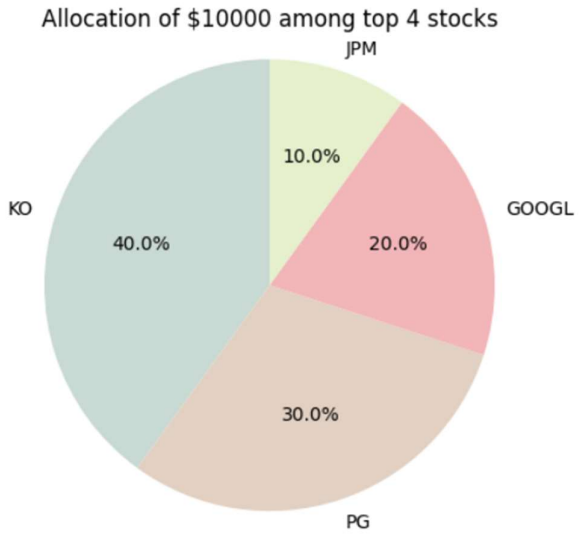
```
Investment in KO: $4000.00, capital after 1 year: $4265.60
Investment in PG: $3000.00, capital after 1 year: $3274.50
Investment in GOOGL: $2000.00, capital after 1 year: $2192.60
Investment in JPM: $1000.00, capital after 1 year: $1051.60

Total capital gain after 1 year: 7.84%
```

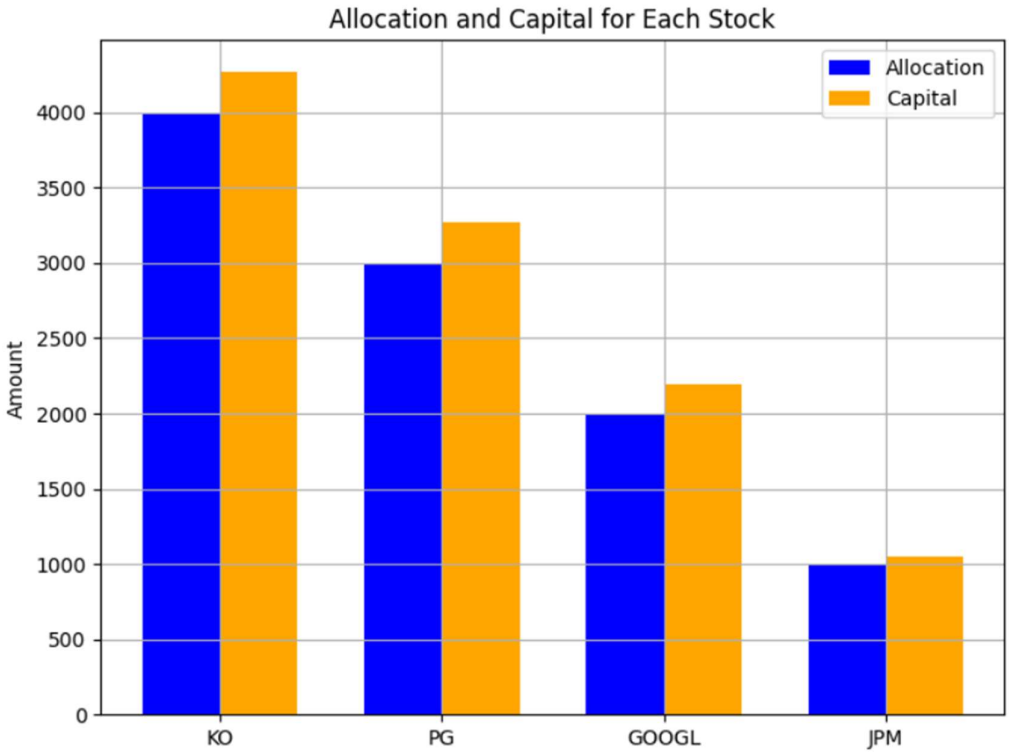
This is an implementation of a portfolio allocation strategy for a set of stocks based on their annualized returns. The code first selects the top 4 stocks based on their annualized returns and allocates a certain percentage of a $10,000 investment to each of them. The percentage allocation for each stock is hardcoded in the code as 0.4, 0.3, 0.2, and 0.1, respectively, for the top 1st, 2nd, 3rd, and 4th stocks.

For each of the top 4 stocks, the code calculates the capital gain after one year based on the allocated investment, the annualized return of the stock, and the compounding effect. The code then prints the investment and capital gain for each stock, and the total capital gain for the portfolio after one year.

Finally, the code creates a pie chart to visualize the allocation of the $10,000 investment among the top 4 stocks.



➢ Creating a bar plot for Allocation and Capital for Each Stock



This creates a bar plot using matplotlib to visualize the allocation and capital for each stock in the portfolio.

The data for the stocks, allocations, and capitals are defined and used to create the bars. The figure and axes for the plot are created, and the bar plot is generated using the bar() method from the axes object.

The bar width is set to 0.35 and the x-axis position of the bars are defined using a list comprehension. The labels for the x-axis are set to the stocks, and the y-axis label and title are also defined.

The legend and grid are added to the plot, and it is displayed using the show() method.

8. SUMMARY

The project aims to create a recommendation system that suggests the best four stocks from a given list of ten stocks. The system uses historical data and machine learning algorithms to identify patterns and trends in the stock market. The project also includes the visualization of stock data through candlestick charts, which allows users to analyse the price movements and identify potential trading opportunities. The candlestick charts are customized to display moving averages of 50 and 200, and color-coded to highlight bullish and bearish trends. The project has the potential to aid investors and traders in making informed decisions and improving their trading strategies. Future work includes integrating additional technical indicators and expanding the dataset to include more stocks and a longer time period.

9. FUTURE WORK

1. **Implement a web application:** Currently, this project is implemented as a Python program. However, a more user-friendly and accessible interface can be created through a web application. Users can input their preferred stocks and receive recommendations in real-time.

2. **Incorporate more data sources:** Currently, this project uses stock data and technical indicators to make recommendations. However, incorporating more data sources, such as news sentiment analysis, social media sentiment analysis, and fundamental analysis, could lead to more accurate recommendations.

3. **Optimize the algorithm:** The current recommendation algorithm could be improved by using more advanced machine learning algorithms or optimization techniques, such as deep learning, ensemble learning, or genetic algorithms. These approaches may lead to more accurate recommendations.

4. **Implement trading strategies:** Instead of only recommending stocks, the project could implement trading strategies based on the recommended stocks. This could include automatic trading, buy and hold strategies, or other trading techniques.

5. **Expand to other markets:** Currently, this project only focuses on the stock market. However, the project could be expanded to include other markets, such as commodities, cryptocurrencies, or foreign exchange. This would provide users with more investment opportunities and diversification options.

10. REFERENCES

1. Yahoo Finance API: https://finance.yahoo.com/
2. Pandas documentation: https://pandas.pydata.org/pandas-docs/stable/index.html
3. Matplotlib documentation: https://matplotlib.org/stable/contents.html
4. mplfinance documentation: https://github.com/matplotlib/mplfinance
5. Plotly documentation: https://plotly.com/python/
6. Technical Analysis Library in Python (TA-Lib): https://github.com/mrjbq7/ta-lib
7. Chat-gpt: https://chat.openai.com/