Analysis of stock exchange using python

P. Vishnupriya[1], S. Hariprasad[2], T. Sreenivas [3], P. Udaya[4], K. Dhana Lakshmi[5], D. Giri [6], B. Sarojamma[7*]

[1]*Department of Statistics, S V University, Tirupati.*

[2] *Department of Statistics, PVKN Govt. Degree College(A), Chittoor*

[3]*Department of Statistics, GOVT. Degree College for Women, Kurnool*

[4] *Department of Mathematics, SGS Arts college(A), TTD, Tirupati*

[5] *Department of Mathematics, Sri Padmavathi Women's Degree & PG College(A), TTD. Tirupati*

[6] *Department of Statistics, D.K Govt. college for Women(A), Nellore*

[7] *Department of Statistics, S V University, Tirupati*

Abstract- Stock market is important for business people. We are collected Microsoft, Google, Apple, and Amazon companies highest stock closing price from stock market. We analyzed data using Python software. Programming for ARIMA and SARIMAX was written in Python. Appropriate graphs also drawn using Phython for data.

Key words: Python, ARIMA, SARIMAX, Stock market

I. INTRODUCTION

Stock market forecasts are very important for businesspeople. In this paper were forcing using python software. Stijn Heldens et.al, [1], says that, it is frequently required of researchers to investigate novel scientific fields outside of their primary expertise. Examples include early career researchers (such as undergraduates) who wish to comprehend the scientific domain of the subject they will be working on, or seasoned academics who wish to investigate novel research avenues. However, with so many pertinent articles now available, it might be challenging to acquire a solid, wide perspective of a field of research. For example, searching for "deep learning" on Elsevier's Scopus returns 166,583 results, while searching for "energy-efficient computer architectures" yields 17,085 results and searching for "parallel programming model" yields 6,306 results.1. It would take an immense amount of work to manually go through such a number of articles. A remedy to this issue is provided by literature reviews, although they are not always readily available, may have a scope that is too wide or too limited, and may become out of date too rapidly for fields of study that move swiftly.

Steven D. Meyers et.al,[2], explains, the marine sector is evolving for several decades, the number and size of commercially operated marine boats on the world's oceans and inland seas have been increasing. This has led to greater economies of scale, but it has also brought with it new logistical and competitive issues. The study and application of advanced analytics and artificial intelligence (AI) to big maritime data, including to vessel navigation records, have been sparked by these advancements along with improvements in affordable computational memory and speed. The main objectives of these developments are increased operational safety and efficiency. Like many other sectors, funding for this kind of fundamental marine research is frequently dependent on national agencies that favour domestic Principle Investigators. This is because, in part, basic research fosters innovation and technical advancement, which increases long-term economic competitiveness. New government goals, the introduction of new technology, or the difficulties faced by governments or scientific communities abroad can all have an impact on the resources allotted to researchers in the public and private sectors. Funding agencies and researchers looking for support are interested in tracking support levels for a particular field globally.

Tom De Smedt et.al,[3], explained that, the World Wide Web, often known as "Web as Corpus," is a vast collection of linguistic data that has gained popularity over the past ten years as a useful tool for activities like opinion mining, machine translation, and trend identification. Because the Web lacks metadata and is dotted with code, using it for this purpose presents a problem. With an emphasis on

user-friendliness, "Pattern" is a Python package for web mining, natural language processing, machine learning, and network analysis. It provides a combination of often used tools for using the Web as a corpus, which typically calls for a number of separate toolkits connected in a useful way. Both scientists and non-scientists should find the bundle beneficial. The syntax is easy to understand. The instructions are self-explanatory due to the careful selection of function names and arguments. No prior knowledge is assumed in the documentation. We think that PATTERN is useful for web developers as a quick development framework, for students as a learning environment.

Ceyhun Ozgur et.al,[4], introduces the usefulness of MatLab, Python, and R in a teaching setting is compared in this work. In this essay, we have attempted to determine which programming language is most appropriate for instructing college students in statistics and operations research. We have also made an effort to ascertain which talent is most useful to know in the job. One kind of programming language is Python. The C implementation of this programming language is the most widely used. Python is a programming language with a sizable standard library in addition to that. This library has modules for networking, databases, OS-specific programming, threading, and general programming with a focus on general programming. The two most well-known uses of Matlab are as a commercial numerical computing environment and as a programming language. Similar to Matlab, it offers a standard library, but its applications also include matrix algebra and a sizable network for graphing and data processing. Additionally, it includes toolkits for ardent learners, albeit the user will have to pay extra for these. R is an open-source, free statistics programme. The programme was developed in 1993 by Robert Gentleman and Ross Ihaka, two colleagues at the University of Auckland in New Zealand, who recognised the need for a better software environment for their lectures. R has undoubtedly exceeded its beginnings; according to a R Community website, it currently has more than two million members.

Noel M O'Boyle et.al,[5], discusses, the Cheminformaticians frequently have to construct one-time scripts to do basic statistics, prepare data for analysis, or extract data from text files. For these routine jobs, scripting languages like Perl, Python, and Ruby are perfect; unfortunately, they are orders of magnitude slower than compiled languages like C++. Since many cheminformatics techniques are computationally costly, cheminformatics toolkits are usually implemented in compiled languages for performance. This is because cheminformaticians frequently work with molecular files that include thousands of molecules. A C++ toolkit called Open Babel has a wide range of functions for reading and writing molecular file formats and working with molecular data. Included are several common chemical algorithms, such as those for finding the smallest set of rings, recognising bond order, adding hydrogens, and allocating Gasteiger charges. Regarding cheminformatics, OpenBabel offers group contribution descriptors for LogP, polar surface area (PSA), and molar refractivity (MR), as well as support for SMARTS searches and molecular fingerprints. In this article, we go over how to use and construct Pybel, a Python module that gives Python programmers access to the OpenBabel C++ library. Pybel enhances the standard Python bindings to facilitate routine cheminformatics activities.

William F. Holmgren et.al,[6], introduces a well-known MATLAB package for solar modelling and analysis is called PVLIB Toolbox. Members of the Photovoltaic Performance and Modelling Collaboration (PVPMC) have contributed to its expansion, which began with development at Sandia National Laboratories. Although many public and private laboratories still choose to use MATLAB, Python's popularity has skyrocketed in the past ten years. Many prestigious colleges now provide beginning programming courses in Python. It has a sizable scientific computing community and is free and open source, beautiful, and simple to read and write on several platforms. When the necessary scientific packages (NumPy, SciPy, Matplotlib, statisticsmodels, pandas) are installed, Python offers a strong substitute for R and MATLAB. A single language may be used for all stages of the data collecting, processing, and analysis workflow thanks to the scientific Python stack, which can lead to quicker development and fewer defects. Overall, the purpose of this package is not to replace other well-known PV modelling programmes that are available for free or for business, such Helioscope, PVSyst, SAM, and PVWatts. Alternatively, PVLIB-Python provides an extensible, easily accessed, and cooperatively created analytic package that may be used to extract profound insights into the operation

of PV systems and the modelling tools. Users may model and analyse every link in the PV system performance chain by using a code-level and modular approach to system modelling. They can also leverage Python's powerful data analytic features to analyse large data sets.

Alan W Sousa da Silva et.al,[7], explains, a wrapper script for the ANTECHAMBER tool, known as ACPYPE (or AnteChamber PYthon Parser interfacE), makes it easier to generate small molecule topologies and parameters for a range of molecular dynamics systems, including GROMACS, CHARMM, and CNS. The Python programming language was used to write it. created as a tool for interacting with other Python-based programmes, such ARIA (which calculates structures from NMR data) and the CCPN software package (which analyses NMR data). ACPYPE is an object-oriented tool that makes it easier to generate topology and parameters automatically in various formats for various molecular mechanics programmes, including partial charge calculations. It may also be integrated with other applications.

Nicholas K. Sauter et.al,[8], says that, the software development may be accelerated by using modular and reusable code, as is often known. The availability of libraries that offer standard file formats for describing atomic structures, structural factors, and electron-density maps has considerably aided the collaborative efforts in crystallography, which are encapsulated in the CCP4 software package. Besides object-oriented programming, which allows one to approach computational issues at a high degree of abstraction, and scripting, which facilitates quick testing of novel concepts, two other aspects of software architecture have also been significant. The flexibility to modify scripted tools for usage in novel settings and the interoperability of different software toolboxes are essential components in creating new programmes that facilitate the evolution of the crystallographic experiment. This article examines the ways in which a toolbox like cctbx may support both the development of better algorithms to handle marginal data in general, such as diffraction patterns displaying two or more lattices, and the immediate data-processing events surrounding data acquisition. We first give a quick overview of the cctbx design and how it might help with software collaboration. Then, we give some instances of how the architecture is being used in the beamline computing environment today and in the future.

Skipper Seabold et.al,[9], introduces a Python package for economic and statistical analysis is called Statsmodels. The target audience comprises of econometricians, theoretical and practical statisticians, and Python developers from many fields that utilise statistical models. Statsmodels can be a helpful addition to your toolkit if you work in financial econometrics, R, Stata, SAS, SPSS, NLOGIT, GAUSS, or MATLAB for statistics, but you would like to work in Python because of all its advantages. This work is intended for researchers with some prior expertise with NumPy/SciPy and Python, and it introduces statsmodels. The primary statsmodel developers at the moment have econometrics training and are qualified economists. As a result, econometrics has dominated a large portion of development during the past year applications. Statsmodels' design, on the other hand, adheres to a standard pattern that makes it simple to use and expand for developers across all fields. The code is becoming increasingly helpful for typical statistical modelling requirements with new contributions and continuous effort. With any luck, our continuous work will provide a software that can be applied to any kind of statistical modelling.

II. METHODOLOGY

Auto-correlation: is a statistical technique used to measure the degree of similarity between observations of a time series at different time lags. It helps in understanding the relationship and dependence between the current value of a variable and its past values. Auto-correlation is commonly assessed using a correlogram or autocorrelation function (ACF) plot, which measures the correlation coefficient between each observation and its lagged values.

The **autocorrelation** function (ACF) measures the correlation between values of a time series at different lags. The formula for ACF is:

$$r_k = \frac{\sum_{t=k+1}^{n}(Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{\sum_{t=1}^{n}(Y_t - \bar{Y})^2}$$

where r_k is the autocorrelation at lag k,

y_t is the value of the time series at time t,

$\bar{y}$ is the mean of the time series

Partial correlation, on the other hand, measures the degree of association between two variables after removing the effects of other variables. In time series analysis, partial correlation can help determine the relationship between the current value of a variable and its past values, while controlling for the effects of other variables. It is commonly assessed using a partial autocorrelation function (PACF) plot.

Partial correlation measures the strength and direction of a linear relationship between two variables, while controlling for the effects of other variables. The formula for the partial correlation coefficient between variables

$$\rho_{XY \cdot Z} = \frac{\rho_{XY} - \rho_{XZ}\rho_{ZY}}{\sqrt{1-\rho_{XZ}^2}\sqrt{1-\rho_{ZY}^2}}.$$

$\rho_{XY.Z}$, is the partial correlation coefficient between X and Y controlling for Z.

ρ_{XY}, is the correlation coefficient between X and Y.

ρ_{XZ}, is the correlation coefficient between X and Z

ρ_{YZ}, is the correlation coefficient between Y and Z

ARIMA (AUTO REGRESSIVE INTEGRATED MOVING AVERAGE):

ARIMA (Autoregressive Integrated Moving Average) is a statistical model used for time series forecasting as

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} + \epsilon_1$$

SARIMAX (SEASONAL AUTOREGRESSIVE INTEGRATED MOVING AVERAGE EXOGENIOUS)

$$Y\_t = c + \varphi_1 Y(t-1) + ... + \varphi\_pY\_(t-p) + \theta_1 e\_(t-1) + ... + \theta\_qe\_(t-q) + \Phi_1 Y\_(t-s) + ... + \Phi\_PY\_(t-Ps) + \Theta_1 e\_(t-s) + ... + \Theta\_Qe\_(t-Qs) + e\_t$$

Coding language: PYTHON

PYTHON LIBRARIES (SEABORN, MATPLOTLIB)

III.PYTHON LIBRARIES

Python libraries are pre-written collections of code that provide various functions, tools, and modules to simplify and expedite specific tasks. For data analysis and visualization, Python offers several powerful libraries that cater to different aspects of the process. Let's explore some of these libraries: **Pandas, NumPy**, **Matplotlib, Seaborn,** Statsmodels, **pandas data reader:** use to remove information from large number of web sources and **datetime**

These libraries collectively provide a comprehensive toolkit for data analysts to import, clean, analyze, and visualize data. By leveraging the capabilities of these libraries, analysts can gain insights, communicate findings effectively, and make informed decisions based on data.

Empirical investigations:

The above graph shows which company stock prices are moving highly.

Based on the above graph

Yellow- Microsoft

Green- Amazon

Red – Google

Blue-Apple

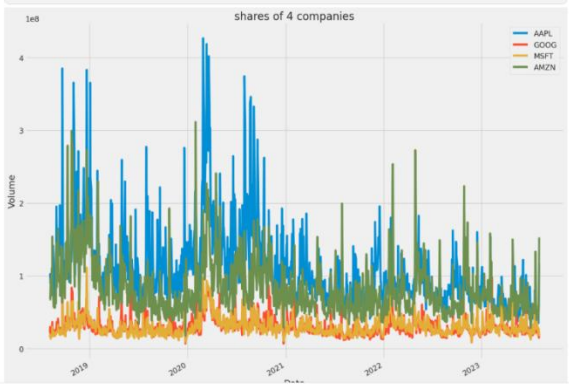
Microsoft closing prices going high then other stock closing prices.

COMPARISION OF SHARES:

```
plt.figure(figsize=(15, 10))
plt.subplots_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(company_list, 1):
    plt.subplot(1,1,1)
    company['Volume'].plot(label=f"{tech_list[i-1]}")
    plt.ylabel('Volume')
    plt.xlabel("Date")
    plt.title("shares of 4 companies")
    plt.legend()

plt.tight_layout()
```



MOVING AVERAGES FOR STOCK EXCHANGE DATA:

Moving averages are a statistical technique used in data analysis to smooth out fluctuations in a dataset and identify trends over time. This creates a series of average values that can help reveal patterns and trends by reducing short-term noise and highlighting longer-term changes.

```
ma_day = [10, 20, 50]

for ma in ma_day:
    for company in company_list:

        column_name = f'MA for {ma} days'
        company[column_name] = company['Adj Close'].rolling(ma).mean()


fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_figheight(10)
fig.set_figwidth(15)

AAPL[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,0])
axes[0,0].set_title('APPLE')

GOOG[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,1])
axes[0,1].set_title('GOOGLE')

MSFT[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,0])
axes[1,0].set_title('MICROSOFT')

AMZN[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,1])
axes[1,1].set_title('AMAZON')
plt.legend()

fig.tight_layout()
```

In the above graph 10 days moving average is moving smoothly without more fluctuations than 20, 50 days moving averages

IV.RISK ANALYISIS

```
closing_df = pdr.get_data_yahoo(tech_list, start=start, end=end)['Adj Close']

# Make a new tech returns DataFrame
risk_data = closing_df.pct_change()
risk_data.head()
```

```
[**********100%**********]  4 of 4 completed
```

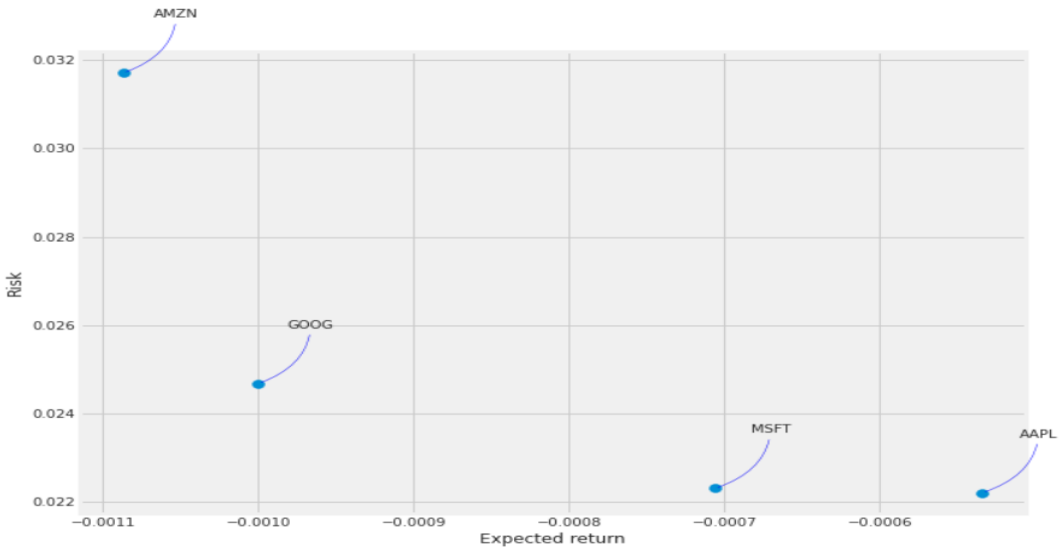
Date	AAPL	AMZN	GOOG	MSFT
2018-08-06	NaN	NaN	NaN	NaN
2018-08-07	-0.009375	0.007972	0.014248	0.006936
2018-08-08	0.000676	0.012907	0.002729	0.005602
2018-08-09	0.007865	0.006361	0.002802	0.001644
2018-08-10	-0.002979	-0.006437	-0.009199	-0.006109

```
risk = risk_data.dropna()

area = np.pi * 20

plt.figure(figsize=(10, 8))
plt.scatter(risk.mean(), risk.std(), s=area)
plt.xlabel('Expected return')
plt.ylabel('Risk')

for label, x, y in zip(risk.columns, risk.mean(), risk.std()):
    plt.annotate(label, xy=(x, y), xytext=(50, 50), textcoords='offset points', ha='right', va='botto
                 arrowprops=dict(arrowstyle='-', color='blue', connectionstyle='arc3,rad=-0.3'))
```

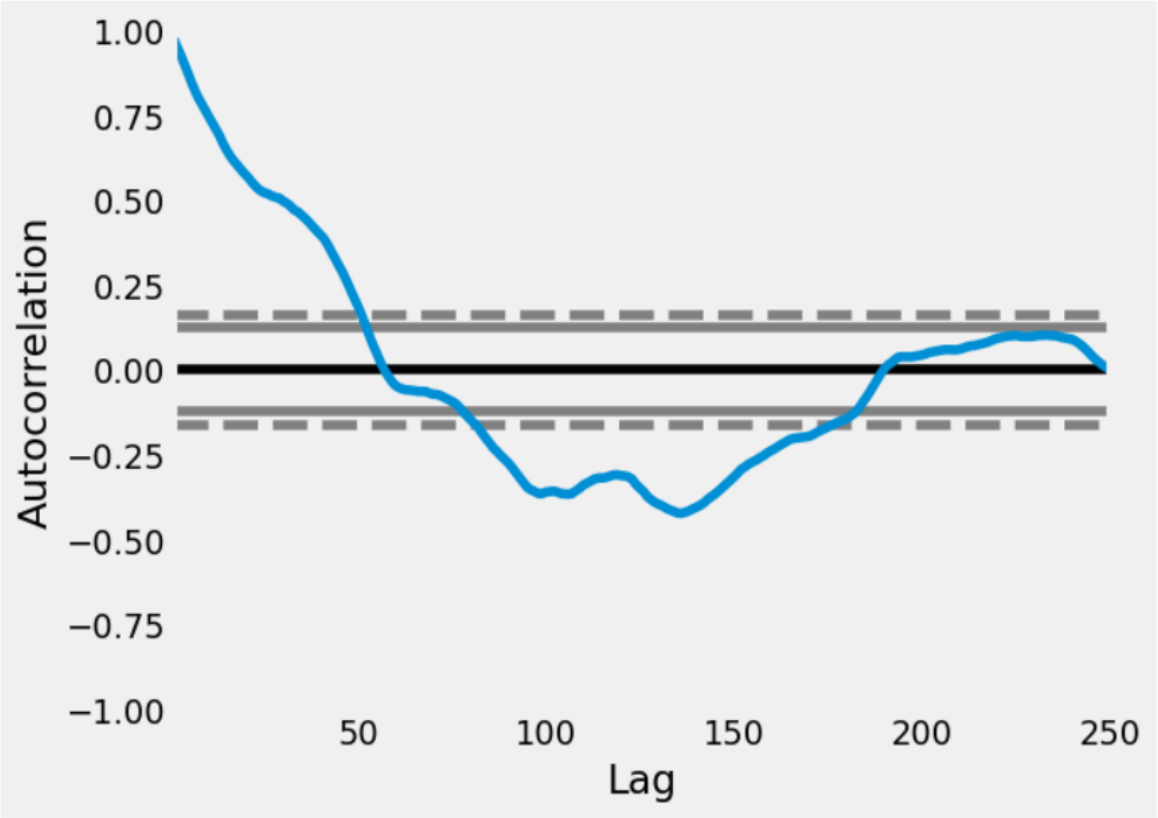


In the above graph Amazon have high risk and high gain

Google -avg risk avg gain, Microsoft and Apple-low risk and low gain

AUTOCORRELATION:

```
pd.plotting.autocorrelation_plot(data["Close"])
```

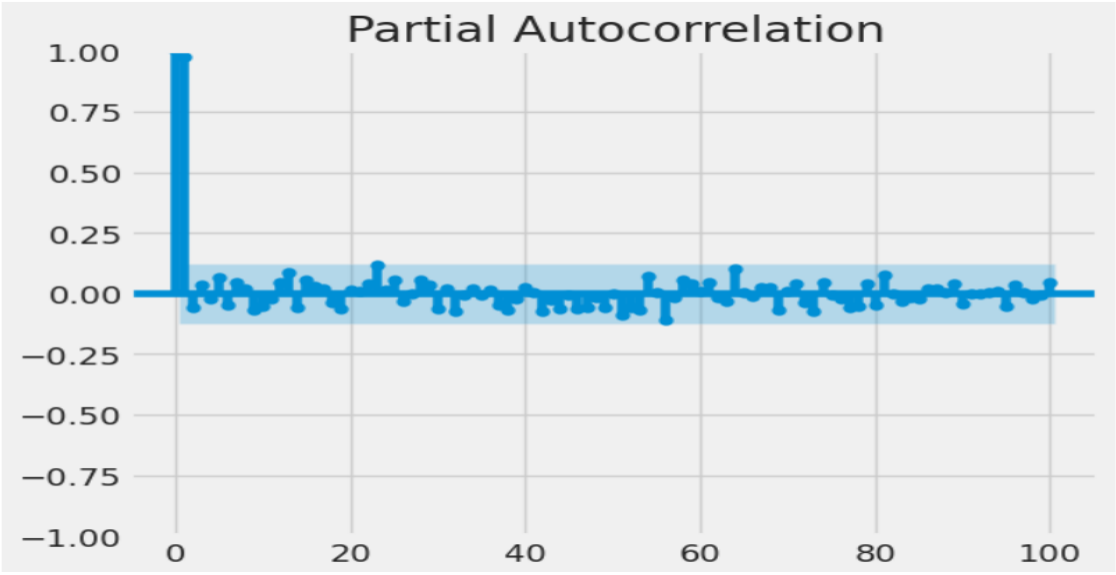


After observing the graph we decide the p-value is 5

PARTIAL AUTOCORRELATION:

```
from statsmodels.graphics.tsaplots import plot_pacf

plot_pacf(data["Close"], lags = 100)
```



From above graph we decide that q value is 2.

PREDICTION OF STOCK PRICES USING ARIMA:

```
from statsmodels.tsa.arima.model import ARIMA
p,d,q=5,1,2
model=ARIMA(data['Close'],order=(p,d,q))
fitted=model.fit()
forecast=fitted.forecast(steps=12)
print("predicted stock prices for next 12 days:")
print(forecast)
```

The above ARIMA code gives constant future price values for next 12 days

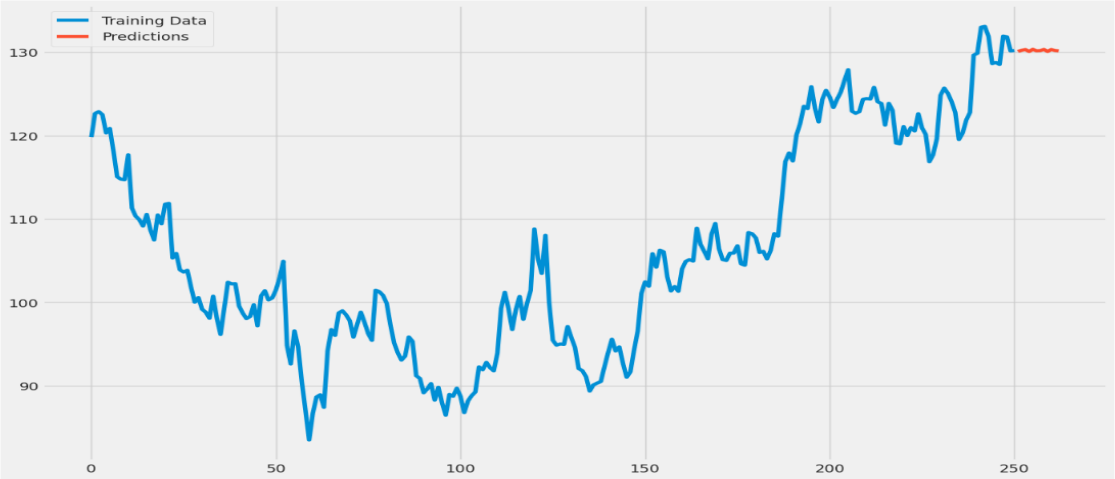
predicted stock prices for next 12 days:

251 130.085243
252 130.240976
253 130.330640
254 130.086732
255 130.357378
256 130.171043
257 130.192240
258 130.343147
259 130.091020
260 130.330381
261 130.209495
262 130.160929

Name: predicted mean, dtype: float64

```
data["Close"].plot(legend=True, label="Training Data", figsize=(15, 10))

forecast.plot(legend=True, label="Predictions")
```



The above plot gives constant results so, this ARIMA method gives the best results for only Stationary data so this method is unfit for seasonal dataset

PREDICTION OF STOCK PRICESUSING SARIMAX:

```
import statsmodels.api as sm

import warnings

model=sm.tsa.statespace.SARIMAX(data['Close'],order=(p, d, q),seasonal_order=(p, d, q, 12))

model=model.fit()

print(model.summary())
predictions = model.predict(len(data),len(data)+20)
print(predictions)
```

```
251    130.126064
252    130.622152
253    130.216849
254    128.525851
255    128.629836
256    129.005361
257    129.739444
258    129.892461
259    131.559059
260    133.424462
261    132.654673
262    132.588544
263    131.542933
264    132.343357
265    132.203204
266    131.640401
267    132.523342
268    132.734920
269    132.579195
270    133.073097
271    133.432702
Name: predicted_mean, dtype: float64
```

The above code gives the future stock prices of 21 days, they are:

130.126064130.622152130.216849128.525851

128.629836129.005361129.739444 129.892461

131.559059133.424462132.654673 132.588544

131.542933132.343357132.203204131.640401

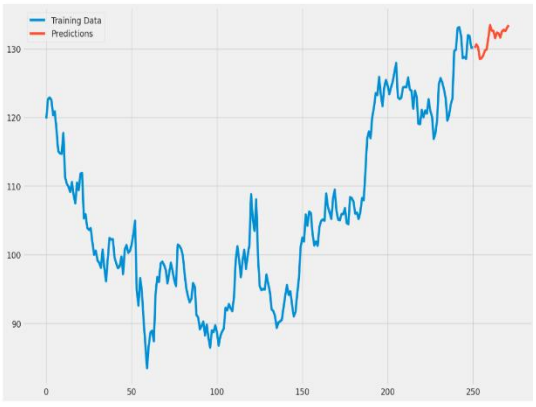
132.523342132.734920132.579195133.073097

133.432702

```
data["Close"].plot(legend=True, label="Training Data", figsize=(15, 10))

predictions.plot(legend=True, label="Predictions")
```

```
[36]: <Axes: >
```



The above plot shows the future stock prices of google based on 5 years closing prices

The dataset is Seasonal so SARIMAX gives the best results for future stock prices

V.SUMMARY AND CONCLUSIONS

We compared the stock closing prices of four companies: Microsoft (MSFT) had the highest stock closing price compared with Google, Apple, and Amazon. And also compared four companies stock shares, in that Apple having highest shares than Amazon, Google and Microsoft companies 10-days moving average have smooth and low fluctuations than 20- and 50-day MA. Microsoft is best daily stock returns company than Apple, Google, Amazon. We predicted the stock closing prices of google using ARIMA and SARIMAX models in time series. SARIMAX is best model for seasonal data.

REFERENCES

[1] Stijn Heldens, Alessio Sclocco, Henk Dreuning, Ben van Werkhoven, Pieter Hijma, Jason Maassen, Rob V. van Nieuwpoort, “litstudy: A Python package for literature reviews”, Published by Elsevier Ltd, 2352-7110/© 2022.

[2] Steven D. Meyers, Laura Azevedo, Mark E. Luther, “A Scopus-based bibliometric study of maritime research involving the Automatic Identification System”, Transportation Research Interdisciplinary Perspectives, Published by Elsevier Ltd, 2590-1982/© 2021.

[3] Tom De Smedt and Walter Daelemans, “Pattern for Python”, Journal of Machine Learning Research 13 (2012) 2063-2067.

[4] Ceyhun Ozgur, Taylor Colliau, Grace Rogers, Zachariah Hughes4, Elyse “Bennie” Myer-Tyson, “MatLab vs. Python vs. R”, Journal of data science: JDS · July 2017, 355-372.

[5] Noel M O'Boyle, Chris Morley and Geoffrey R Hutchison, “Pybel: a Python wrapper for the Open Babel cheminformatics toolkit”, Chemistry Central Journal 2008, 2:5.

[6] William F. Holmgren, Robert W. Andrews, Antonio T. Lorenzo, Joshua S. Stein, “PVLIB Python 2015”, SAND2015-6181C.

[7] Alan W Sousa da Silva and Wim F Vranken, “ACPYPE - AnteChamber PYthon Parser interface”, BMC Research Notes 2012, 5:367.

[8] Nicholas K. Sauter, Johan Hattne, Ralf W. GrosseKunstleve and Nathaniel Echols, “New Python-based methods for data processing”, Acta Cryst. (2013). D69, 1274–1282, Acta Crystallographica, Section D, Biological Crystallography, ISSN 0907-4449.

[9] Skipper Seabold and Josef Perktold, "Statsmodels: Econometric and Statistical Modeling with Python", PROC. OF THE 9th PYTHON IN SCIENCE CONF. (SCIPY 2010).
[10] https://finance.yahoo.com/