**Title: - Stock Market Analysis**

Name: Hemanth Kumar LV

Reg No: 1BO23MC014

Department: MCA (VTU)

Semester: Third

# Objective: -

This study explores the dynamics of stock market behavior, focusing on the identification of key trends, market drivers, and predictive modeling techniques. By examining historical data from major stock indices, the research highlights the role of macroeconomic indicators, investor sentiment, and geopolitical events in influencing stock prices. Statistical tools, such as time series analysis and machine learning algorithms, are employed to predict short-term and long-term market movements. The findings suggest that while traditional financial indicators remain essential, newer machine learning models, incorporating sentiment analysis from social media and news sources, significantly improve prediction accuracy. This research offers insights into effective strategies for investors and emphasizes the growing importance of data-driven decision-making in modern financial markets.

# Problem statement: -

The stock market is influenced by a wide range of factors, including macroeconomic conditions, company performance, investor sentiment, and external events such as geopolitical crises. Despite significant advancements in financial modeling and data analysis, predicting stock price movements remains a complex and highly uncertain task. Traditional analysis techniques, such as fundamental and technical analysis, often fail to account for the volatile and multifaceted nature of modern financial markets. Moreover, the rise of alternative data sources, such as social media, news sentiment, and economic indicators, has added complexity to stock market prediction models.

# Solution: -

The solution to this problem is to build a **Screen Time Analysis** application that tracks and visualizes the time users spend on their devices, across different apps, and even provides insights into usage trends. The app will:

* **Track screen time**: Automatically monitor and log screen time for different devices (smartphones, laptops, tablets).
* **Categorize usage**: Break down screen time by application, categorizing them into work, social media, entertainment, etc.
* **Provide insights**: Offer detailed reports and analytics, such as daily, weekly, or monthly trends.
* **Set goals**: Allow users to set time limits for specific apps or total screen time per day.
* **Notifications and alerts**: Send reminders or warnings when users exceed their set limits.
* **Visualization**: Provide charts and graphs (using tools like Plotly or Matplotlib) to make the data easier to interpret.

# Implementation: -

import yfinance as yf

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Define stock symbols and time period

stocks = ['AAPL', 'GOOGL', 'MSFT', 'AMZN']

start\_date = '2020-01-01'

end\_date = '2024-01-01'

# Fetch historical data

data = yf.download(stocks, start=start\_date, end=end\_date)['Adj Close']

# Calculate daily returns

daily\_returns = data.pct\_change()

# Plot stock price trends

plt.figure(figsize=(12, 6))

data.plot(title='Stock Price Trends', figsize=(12, 6))

plt.xlabel('Date')

plt.ylabel('Adjusted Close Price')

plt.legend(stocks)

plt.show()

# Calculate moving averages (50-day and 200-day)

moving\_avg\_50 = data.rolling(window=50).mean()

moving\_avg\_200 = data.rolling(window=200).mean()

# Plot moving averages

plt.figure(figsize=(12, 6))

plt.plot(data.index, data['AAPL'], label='AAPL Price', alpha=0.6)

plt.plot(data.index, moving\_avg\_50['AAPL'], label='50-day MA', linestyle='dashed')

plt.plot(data.index, moving\_avg\_200['AAPL'], label='200-day MA', linestyle='dashed')

plt.title('AAPL Stock Price with Moving Averages')

plt.xlabel('Date')

plt.ylabel('Price')

plt.legend()

plt.show()

# Calculate stock volatility (Standard Deviation of returns)

volatility = daily\_returns.std() \* np.sqrt(252) # Annualized volatility

print("Stock Volatility:")

print(volatility)

# Correlation matrix of stock returns

correlation\_matrix = daily\_returns.corr()

# Plot correlation heatmap

plt.figure(figsize=(8, 6))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', linewidths=0.5)

plt.title('Stock Correlation Matrix')

plt.show()

Output: -

