**Stock Price Prediction using Support Vector Machine (SVM)**

**Introduction**

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification and regression tasks. In the context of stock price prediction, SVM can be applied to predict the movement or direction of stock prices based on historical data and relevant features.

**Importing Required Libraries**

The initial step involves importing necessary libraries such as sklearn.svm for implementing SVM, pandas and numpy for data manipulation, matplotlib for visualization, and warnings to suppress any unwanted warnings during execution.

**Exploratory Data Analysis**

**Date Column Transformation**

Before delving into the analysis, it's crucial to ensure the dataset is structured appropriately for further investigation and modeling. The code below serves to transform the 'Date' column into the index column, facilitating time-series analysis and alignment of data points based on chronological order.

# Changes The Date column as index columns

df.index = pd.to\_datetime(df['Date'])

df

By setting the 'Date' column as the index, we enable easier manipulation and visualization of time-series data, which is often crucial in stock price analysis.

# drop The original date column

df = df.drop(['Date'], axis='columns')

df

**Dropping the Original Date Column**

After setting the index to the 'Date' column, it becomes redundant to keep the original 'Date' column in the DataFrame. Hence, the code below drops the original 'Date' column from the DataFrame.

Removing the redundant column helps streamline the dataset and avoids potential redundancy in further analysis and modeling processes.

df = df.dropna(axis=0, how='any')

This step helps maintain the integrity of the dataset by removing incomplete or unreliable data points, ensuring a more accurate and robust analysis.

**Creating Predictor Variables**

**Open-Close and High-Low Calculation**

To capture potential patterns in stock price movements, we compute two additional predictor variables: 'Open-Close' and 'High-Low'. These variables represent the difference between opening and closing prices ('Open-Close') and the difference between high and low prices ('High-Low') for each trading day.

# Create predictor variables

df['Open-Close'] = df.Open - df.Close

df['High-Low'] = df.High - df.Low

These calculations provide insights into the intraday price movements and volatility, which can be valuable indicators for predicting future price changes.

Storing Predictor Variables

All predictor variables are stored in a variable named 'X', which includes the 'Open-Close' and 'High-Low' columns from the DataFrame.

# Store all predictor variables in a variable X

X = df[['Open-Close', 'High-Low']]

X.head()

By isolating these predictor variables, we prepare the data for model training and analysis, focusing on the features that are most relevant for predicting stock price movements.

**Target Variable Definition**

**Price Movement Classification**

To facilitate supervised learning, we define the target variable 'y' based on the direction of price movements. Using a binary classification approach, 'y' is assigned a value of 1 if the closing price of the next trading day is higher than the current closing price, and 0 otherwise.

# Target variables

y = np.where(df['Close'].shift(-1) > df['Close'], 1, 0)

y

This classification scheme enables the model to learn patterns associated with upward or downward movements in stock prices, aiding in the prediction of future price trends.

**Data Splitting for Training and Testing**

**Splitting the Dataset**

To evaluate the performance of the predictive model, we divide the dataset into training and testing sets. The code snippet below splits the dataset based on a specified percentage, with 80% of the data allocated for training and the remaining 20% for testing.

split\_percentage = 0.8

split = int(split\_percentage \* len(df))

# Train data set

X\_train = X[:split]

y\_train = y[:split]

# Test data set

X\_test = X[split:]

y\_test = y[split:]

This separation ensures that the model is trained on a subset of the data and then tested on unseen data, allowing us to assess its generalization performance and effectiveness in predicting future price movements.

**Support Vector Classifier**

**Model Training**

A Support Vector Classifier (SVC) is utilized to train the predictive model using the training data. The SVC is fitted to the training features ('X\_train') and corresponding target labels ('y\_train').

# Support vector classifier

cls = SVC().fit(X\_train, y\_train)

**Predicting Stock Price Movements**

Using the trained SVC model, predictions are made for the entire dataset. The predicted signals for each data point ('Predicted\_Signal') are calculated based on the model's predictions.

df['Predicted\_Signal'] = cls.predict(X)

**Strategy Evaluation**

**Calculating Daily Returns**

Daily returns are computed based on the percentage change in closing prices ('Close') from one trading day to the next.

df['Return'] = df.Close.pct\_change()

**Calculating Strategy Returns**

Strategy returns are determined by multiplying the daily returns by the predicted signals, which represent the model's predicted movements.

df['Strategy\_Return'] = df.Return \* df.Predicted\_Signal.shift(1)

**Calculating Cumulative Returns**

Cumulative returns for both the actual ('Cum\_Ret') and strategy ('Cum\_Strategy') are calculated to assess the performance over the entire period.

df['Cum\_Ret'] = df['Return'].cumsum()

df['Cum\_Strategy'] = df['Strategy\_Return'].cumsum()

These steps enable the evaluation of the predictive model's performance in generating trading signals and its effectiveness in generating returns compared to a buy-and-hold strategy.

**Cumulative Returns Comparison Plot**

**Purpose**

The purpose of this plot is to visually compare the cumulative returns of two different strategies: a buy-and-hold strategy and the strategy generated by the Support Vector Classifier (SVC) model.

Visual Representation

The plot displays two lines:

**Buy-and-Hold Strategy:** Represented by the red line, this line shows the cumulative returns of holding the asset for the entire duration without any trading decisions based on predictive models.

**Strategy Generated by SVC Model:** Represented by the blue line, this line depicts the cumulative returns of the strategy generated by the SVC model, which involves making trading decisions based on signals predicted by the model.

**Analysis**

By comparing the two lines, investors and analysts can assess the performance of the SVC model-generated strategy against a passive buy-and-hold approach. This comparison aids in evaluating the effectiveness of the predictive model in generating profitable trading signals and potentially outperforming a simple investment strategy.

**Conclusion**

**Performance Comparison**

The graphical representation illustrates the comparative performance between the strategy generated by the Support Vector Classifier (SVC) model ('Cum\_Strategy') and the benchmark or original returns ('Cum\_Ret'). Notably, the 'Cum\_Strategy' exhibits higher cumulative returns than simply holding the asset, indicating its potential for generating superior returns over the analysis period.

**Strategy Effectiveness**

An analysis of the performance over time reveals fluctuations in the relative performance of 'Cum\_Strategy' compared to 'Cum\_Ret'. Initially, in 2016-2017, 'Cum\_Strategy' demonstrates lower performance, but from 2019 onwards, it exhibits a notable increase, suggesting the effectiveness of the strategy in generating additional returns compared to passive asset holding.

**Volatility and Risk**

The volatility of the 'Cum\_Strategy', represented by the blue line, appears to be comparable to that of the red line (benchmark). This observation indicates that while the strategy may generate higher returns, it does not entail significantly higher risk, suggesting a balanced risk-return profile.

**Drawdown Analysis**

During the period between 2020 and 2021, both 'Cum\_Strategy' and 'Cum\_Ret' experience similar drawdowns. This trend is understandable, considering the market turbulence and economic uncertainties associated with the COVID-19 pandemic and subsequent lockdowns.

**Periods of Outperformance**

A notable observation is the significant outperformance of the blue line (strategy) over the red line (benchmark) between 2019 and 2020. This extended period of outperformance suggests that the strategy derived from the Support Vector Machine model has the potential to consistently outperform the benchmark over longer periods.