**Stock Market Prediction: Approach, Methodologies, and Insights**

**1. Introduction**

The goal of this project is to predict stock price movements using historical stock data. We use a machine learning approach to create a model that can forecast whether the stock price will increase or decrease the following day. The dataset contains various stock market metrics, and we focus on specific columns for prediction.

**2. Data Preparation**

**Data Loading:**

python

Copy code

df\_stock = pd.read\_csv('/kaggle/input/stock-market-prediction/infolimpioavanzadoTarget.csv')

**Initial Data Inspection:**

python

Copy code

df\_stock.head() df\_stock.shape df\_stock.info() df\_stock.describe() df\_stock.isnull().sum()

From the initial inspection, we observed that the dataset contains 7781 rows and 1285 columns with various data types and some missing values.

**Column Selection and Data Cleaning:** We selected the essential columns (**date**, **open**, **high**, **low**, **close**) for our analysis. We created a new column **tomorrow** to store the stock's closing price for the next day and a **target** column to indicate if the next day's closing price is higher than the current day's closing price.

python

Copy code

data\_new = df\_stock[['date', 'open', 'high', 'low', 'close']].copy() data\_new['tomorrow'] = data\_new['close'].shift(-1) data\_new['target'] = data\_new['tomorrow'] > data\_new['close'] data\_new.dropna(inplace=True)

**3. Exploratory Data Analysis (EDA)**

During the EDA phase, we checked for any anomalies, trends, or patterns in the dataset. We used summary statistics and visualizations to understand the data distribution and relationships between variables.

**Visualizations:**

python

Copy code

import seaborn as sns import matplotlib.pyplot as plt # Plotting the stock prices plt.figure(figsize=(12,6)) sns.lineplot(data=data\_new, x='date', y='close') plt.title('Stock Closing Prices Over Time') plt.xlabel('Date') plt.ylabel('Closing Price') plt.show()

**4. Model Selection**

**Random Forest Classifier:** We chose the Random Forest Classifier for its robustness and ability to handle non-linear patterns in the data. It also reduces overfitting by averaging multiple decision trees.

**Hyperparameters:**

* **n\_estimators=200**: Number of trees in the forest.
* **min\_samples\_split=100**: Minimum number of samples required to split an internal node.

**5. Training and Testing the Model**

**Data Splitting:** We split the data into training and testing sets. The last 100 rows were used as the test set to evaluate the model's performance.

python

Copy code

train = data\_new.iloc[:-100] test = data\_new.iloc[-100:] predictors = ["open", "high", "low", "close"]

**Model Training:**

python

Copy code

from sklearn.ensemble import RandomForestClassifier model = RandomForestClassifier(n\_estimators=200, min\_samples\_split=100, random\_state=1) model.fit(train[predictors], train['target'])

**Model Prediction and Evaluation:**

python

Copy code

from sklearn.metrics import precision\_score, accuracy\_score, recall\_score, f1\_score pred = model.predict(test[predictors]) precision = precision\_score(test['target'], pred) accuracy = accuracy\_score(test['target'], pred) recall = recall\_score(test['target'], pred) f1 = f1\_score(test['target'], pred) print("Precision Score:", precision) print("Accuracy:", accuracy) print("Recall:", recall) print("F1 Score:", f1)

**6. Insights Gained**

* **Data Quality:** The dataset had a significant number of columns and some missing values. Proper cleaning and selection of relevant columns were crucial.
* **Model Performance:** The Random Forest Classifier provided a balance between bias and variance, handling the non-linearity in the stock market data effectively.
* **Evaluation Metrics:** Precision, accuracy, recall, and F1 score offered a comprehensive evaluation of the model's performance. Precision is particularly important in stock market predictions to minimize false positives.

**7. Conclusion**

This project demonstrated the use of machine learning for stock market prediction, focusing on data cleaning, preparation, model selection, and evaluation. The Random Forest Classifier proved to be an effective model, but there is room for improvement through hyperparameter tuning and feature engineering.

**8. Future Work**

* **Hyperparameter Tuning:** Use GridSearchCV or RandomizedSearchCV to find the best hyperparameters.
* **Feature Engineering:** Incorporate additional features such as moving averages, volume changes, and other technical indicators.
* **Ensemble Methods:** Explore other ensemble methods and compare their performance.

This approach provides a solid foundation for predicting stock market movements and can be expanded with more advanced techniques and data sources.

Top of Form