10/11/2023

|  |
| --- |
| TEAM MEMBER | Students |



|  |  |
| --- | --- |
| [Type the company name] | stock price prediction |

# 

AU732521104019 : JAYASURYA.A

**PHASE 2: SUBMISSION DOCUMENT**

Team member

Project title: stock price prediction



**INTRODUCTION:**

Greetings from the Kaggle bot! This is an automatically-generated kernel with starter code demonstrating how to read in the data and begin exploring. If you're inspired to dig deeper, click the blue "Fork Notebook" button at the top of this kernel to begin editing.

1. **Data Collection:** Stock price prediction relies heavily on historical data. This data typically includes historical stock prices, trading volumes, financial reports, and other relevant information. Data can be collected from various sources such as financial news websites, stock exchanges, and financial databases.
2. **Features:** In machine learning and data analysis, features are variables that help in making predictions. In stock price prediction, features can include historical stock prices, trading volumes, company financial indicators, economic indicators, sentiment analysis of news articles, and more

.

1. **Time Series Analysis:** Stock price data is typically analyzed as a time series, which means that the data points are ordered chronologically. Time series analysis techniques are often used to identify patterns, trends, and seasonality in the data.
2. **Model Selection:** To predict stock prices, various machine learning models can be used, such as linear regression, decision trees, random forests, support vector machines, and deep learning models like recurrent neural networks (RNNs) and long short-term memory networks (LSTMs). The choice of model depends on the complexity of the problem and the quality of the data.
3. **Evaluation Metrics:** To assess the performance of a stock price prediction model, several evaluation metrics can be used. Common metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and accuracy scores.
4. **Feature Engineering:** This is the process of selecting, transforming, or creating new features from the raw data to improve the model's predictive performance. It may involve techniques like smoothing, differencing, or lagging.
5. **Overfitting and Generalization:** Preventing overfitting is crucial in stock price prediction. Overfitting occurs when a model learns to perform well on the training data but doesn't generalize well to new, unseen data. Regularization techniques and careful model selection help mitigate overfitting.
6. **Market Sentiment Analysis:** Public sentiment, as expressed in news articles, social media, and analyst reports, can influence stock prices. Sentiment analysis techniques are often integrated into models to capture this aspect.
7. **Risk Management:** Stock price prediction is inherently risky. It's important to understand that no model can predict stock prices with absolute certainty. Risk management strategies, such as stop-loss orders and portfolio diversification, are essential for minimizing potential losses.
8. **Backtesting:** To assess the effectiveness of a stock price prediction model, it's common to conduct backtesting. This involves applying the model to historical data to see how it would have performed in the past.

**EXPLORATORY ANALYSIS**

To begin this exploratory analysis, first import libraries and define functions for plotting the data using matplotlib. Depending on the data, not all plots will be made. (Hey, I'm just a simple kerneling bot, not a Kaggle Competitions Grandmaster!)

**DATA SOURCE:**

**DATASET LINK**: <https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset/>

**PROGRAM:**

**STOCK PRICE PREDICTION**

from mpl\_toolkits.mplot3d import Axes3D

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt *# plotting*

import numpy as np *# linear algebra*

import os *# accessing directory structure*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

There is 1 csv file in the current version of the dataset:

|  |
| --- |
| In [2]:  for dirname, \_, filenames **in** os.walk('/kaggle/input'):  for filename **in** filenames:  print(os.path.join(dirname, filename))  /kaggle/input/MSFT.csv |

The next hiden code cells define functions for plotting data. Click on the "Code" button in the published kernel to reveal the hidden code.

|  |
| --- |
| # Distribution graphs (histogram/bar graph) of column data  def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):  nunique = df.nunique()  df = df[[col for col in df if nunique[col] > 1 and nunique[col] < 50]] # For displaying purposes, pick columns that have between 1 and 50 unique values  nRow, nCol = df.shape  columnNames = list(df) |
| nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow  plt.figure(num = None, figsize = (6 \* nGraphPerRow, 8 \* nGraphRow), dpi = 80, facecolor = 'w', edgecolor = 'k')  for i in range(min(nCol, nGraphShown)):  plt.subplot(nGraphRow, nGraphPerRow, i + 1)  columnDf = df.iloc[:, i]  if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):  valueCounts = columnDf.value\_counts()  valueCounts.plot.bar()  else:  columnDf.hist()  plt.ylabel('counts')  plt.xticks(rotation = 90)  plt.title(f'{columnNames[i]} (column {i})')  plt.tight\_layout(pad = 1.0, w\_pad = 1.0, h\_pad = 1.0)  plt.show( | |

Now you're ready to read in the data and use the plotting functions to visualize the data.

Let's check 1st file: /kaggle/input/MSFT.csv

|  |  |
| --- | --- |
| |  | | --- | | In [6]:  nRowsRead = 1000 *# specify 'None' if want to read whole file* | |

*# MSFT.csv may have more rows in reality, but we are only loading/previewing the first 1000 ro*

|  |
| --- |
| df1 = pd.read\_csv('/kaggle/input/MSFT.csv', delimiter=',', nrows = nRowsRead)  df1.dataframeName = 'MSFT.csv'  nRow, nCol = df1.shape  print(f'There are **{nRow}** rows and **{nCol}** columns') |

There are 1000 rows and 7 columns

Let's take a quick look at what the data looks like:

|  |
| --- |
| In [7]:  df1.head(5) |

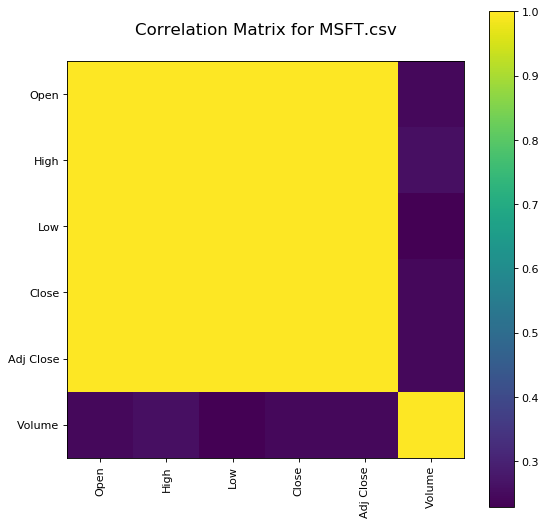
Out[7]:

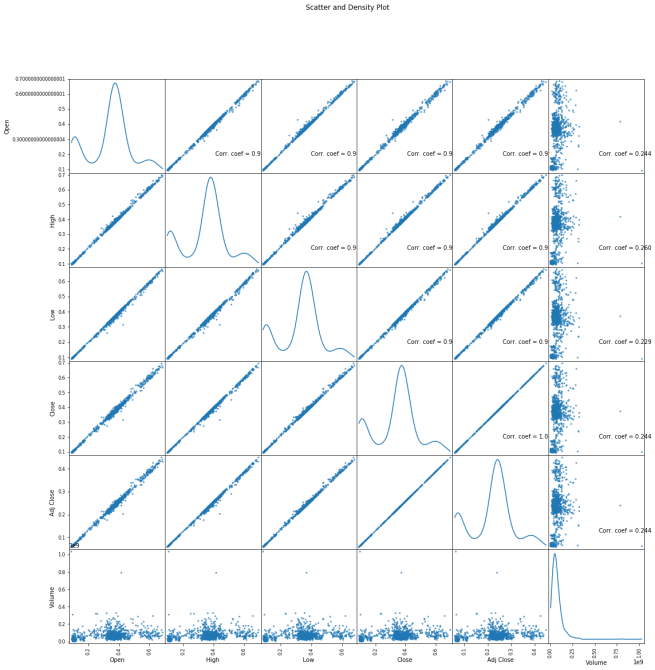
Distribution graphs (histogram/bar graph) of sampled columns:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Date | Open | High | Low | Close | Adj Close | Volume |
| 1986-03-13 | 0.088542 | 0.101563 | 0.088542 | 0.097222 | 0.062549 | 1031788800 |
| 1986-03-14 | 0.097222 | 0.102431 | 0.097222 | 0.100694 | 0.064783 | 308160000 |
| 1986-03-17 | 0.100694 | 0.103299 | 0.100694 | 0.102431 | 0.065899 | 133171200 |
| 1986-03-18 | 0.102431 | 0.103299 | 0.098958 | 0.099826 | 0.064224 | 67766400 |
| 1986-03-19 | 0.099826 | 0.100694 | 0.097222 | 0.098090 | 0.063107 | 47894400 |

|  |
| --- |
| In [8]:  plotPerColumnDistribution(df1, 10, 5)  <Figure size 2400x512 with 0 Axes>  Correlation matrix: |

|  |
| --- |
| In [9]:  plotCorrelationMatrix(df1, 8) |





Scatter and density plots:

**CONCLUSION**

This concludes your starter analysis! To go forward from here, click the blue "Fork Notebook" button at the top of this kernel. This will create a copy of the code and environment for you to edit. Delete, modify, and add code as you please. Happy Kaggling!