STOCK PRICE PREDICTION (Phase 3)

Madras Institute of Technology

2021506088 – Santhi Priya D N
2021506096 - Shiek Sajnathul Faizana
2021506103 – Sowmiya J
2021506323 - Revanth P

1.INTRODUCTION:

This phase aims to clean, transform, and engineer features in a way that maximizes the model's ability to capture patterns and make accurate predictions. Through careful data preparation and feature engineering, we enhance the quality of input fed into the models. Effective preprocessing lays the foundation for improved predictive models. **The given dataset has been pre-processed and the outputs are attached with snap shots.**

2.IMPORTING LIBRARIES AND LOADING DATA:

For Pre-Processing the given dataset, the pandas library is used. The given csv file is uploaded to pandas as follows:

```
>>> import pandas as pd
>>> df = pd.read_csv(r"C:\Users\DELL\Downloads\MSFT.csv")
>>> print(df)
            Date        Open        High         Low       Close   Adj Close      Volume
0     1986-03-13    0.088542    0.101563    0.088542    0.097222    0.062549  1031788800
1     1986-03-14    0.097222    0.102431    0.097222    0.100694    0.064783   308160000
2     1986-03-17    0.100694    0.103299    0.100694    0.102431    0.065899   133171200
3     1986-03-18    0.102431    0.103299    0.098958    0.099826    0.064224    67766400
4     1986-03-19    0.099826    0.100694    0.097222    0.098090    0.063107    47894400
...          ...         ...         ...         ...         ...         ...         ...
8520  2019-12-31  156.770004  157.770004  156.449997  157.699997  157.699997    18369400
8521  2020-01-02  158.779999  160.729996  158.330002  160.619995  160.619995    22622100
8522  2020-01-03  158.320007  159.949997  158.059998  158.619995  158.619995    21116200
8523  2020-01-06  157.080002  159.100006  156.509995  159.029999  159.029999    20813700
8524  2020-01-07  159.320007  159.669998  157.330002  157.580002  157.580002    18017762

[8525 rows x 7 columns]
>>>
```

3.UNDERSTANDING THE DATASET:

df.head:

```
>>> print(df.head())
         Date      Open      High       Low     Close  Adj Close      Volume
0  1986-03-13  0.088542  0.101563  0.088542  0.097222   0.062549  1031788800
1  1986-03-14  0.097222  0.102431  0.097222  0.100694   0.064783   308160000
2  1986-03-17  0.100694  0.103299  0.100694  0.102431   0.065899   133171200
3  1986-03-18  0.102431  0.103299  0.098958  0.099826   0.064224    67766400
4  1986-03-19  0.099826  0.100694  0.097222  0.098090   0.063107    47894400
>>>
```

df.describe:

```
>>> print(df.describe())
              Open         High          Low        Close    Adj Close        Volume
count  8525.000000  8525.000000  8525.000000  8525.000000  8525.000000  8.525000e+03
mean     28.220247    28.514473    27.918967    28.224480    23.417934  6.045692e+07
std      28.626752    28.848988    28.370344    28.626571    28.195330  3.891225e+07
min       0.088542     0.092014     0.088542     0.090278     0.058081  2.304000e+06
25%       3.414063     3.460938     3.382813     3.414063     2.196463  3.667960e+07
50%      26.174999    26.500000    25.889999    26.160000    18.441576  5.370240e+07
75%      34.230000    34.669998    33.750000    34.230000    25.392508  7.412350e+07
max     159.449997   160.729996   158.330002   160.619995   160.619995  1.031789e+09
>>>
```

Isnull:

This function is used to identify missing values in the dataset. Since there in no null value, there is no need for handling the missing data.

```
>>> print(df.isnull().sum())
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
>>>
```

4.REMOVING DUPLICATES:

If any row is duplicated in the given dataset, the following code will identify it and remove it. The given dataset does not contain any duplicates and hence the dataset is the same as before.

```
>>> bf = df
>>> bf = bf.drop_duplicates()
>>> print(df.describe())
              Open         High          Low        Close    Adj Close        Volume
count  8525.000000  8525.000000  8525.000000  8525.000000  8525.000000  8.525000e+03
mean     28.220247    28.514473    27.918967    28.224480    23.417934  6.045692e+07
std      28.626752    28.848988    28.370344    28.626571    28.195330  3.891225e+07
min       0.088542     0.092014     0.088542     0.090278     0.058081  2.304000e+06
25%       3.414063     3.460938     3.382813     3.414063     2.196463  3.667960e+07
50%      26.174999    26.500000    25.889999    26.160000    18.441576  5.370240e+07
75%      34.230000    34.669998    33.750000    34.230000    25.392508  7.412350e+07
max     159.449997   160.729996   158.330002   160.619995   160.619995  1.031789e+09
>>> print(bf.isnull().sum())
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
>>>
```

5.DATA TRANSFROMATION:

Normalizing Data:

```
C:\Users\DELL>py
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> from sklearn.preprocessing import MinMaxScaler, LabelEncoder
>>> df=pd.read_csv(r"C:\Users\DELL\Downloads\MSFT.csv")
>>> scaler = MinMaxScaler()
>>> df['Normalized_Open'] = scaler.fit_transform(df[['Open']])
  File "<stdin>", line 1
    df['Normalized_Open'] = scaler.fit_transform(df[['Open']])
                                                 ^
SyntaxError: unterminated string literal (detected at line 1)
>>> df['Normalized_Open'] = scaler.fit_transform(df[['Open']])
>>> label_encoder=LabelEncoder()
>>> df['Encoded_Date'] = label_encoder.fit_transform(df['Date'])
>>> print(df)
           Date        Open        High         Low       Close   Adj Close      Volume  Normalized_Open  Encoded_Date
0     1986-03-13    0.088542    0.101563    0.088542    0.097222    0.062549  1031788800         0.000000             0
1     1986-03-14    0.097222    0.102431    0.097222    0.100694    0.064783   308160000         0.000054             1
2     1986-03-17    0.100694    0.103299    0.100694    0.102431    0.065899   133171200         0.000076             2
3     1986-03-18    0.102431    0.103299    0.098958    0.099826    0.064224    67766400         0.000087             3
4     1986-03-19    0.099826    0.100694    0.097222    0.098090    0.063107    47894400         0.000071             4
...          ...         ...         ...         ...         ...         ...         ...              ...           ...
8520  2019-12-31  156.770004  157.770004  156.449997  157.699997  157.699997    18369400         0.983183          8520
8521  2020-01-02  158.779999  160.729996  158.330002  160.619995  160.619995    22622100         0.995796          8521
8522  2020-01-03  158.320007  159.949997  158.059998  158.619995  158.619995    21116200         0.992909          8522
8523  2020-01-06  157.080002  159.100006  156.509995  159.029999  159.029999    20813700         0.985128          8523
8524  2020-01-07  159.320007  159.669998  157.330002  157.580002  157.580002    18017762         0.999184          8524

[8525 rows x 9 columns]
>>>
```

Z-Score Standardization (for column – high):

Z-score standardization, also known as "z-score normalization" or "z-score scaling," is a statistical method used to standardize or normalize features in a dataset. It's a process that transforms the features by scaling them to have a mean of 0 and a standard deviation of 1. This makes it easier to compare and analyze variables with different units or scales.

The formula to calculate the z-score for a given data point

X in a feature is: **$z=X- \mu/ \sigma$**

where:

X is an individual data point.

μ is the mean of the feature.

σ is the standard deviation of the feature.

The z-score measures how many standard deviations a data point is from the mean. A positive z-score indicates that the data point is above the mean, while a negative z-score indicates it's below the mean.

```
>>> df['High'] = (df['High'] - df['High'].mean())/df['High'].std()
>>> print(df)
           Date        Open      High         Low       Close   Adj Close      Volume  Normalized_Open  Encoded_Date
0     1986-03-13    0.088542 -0.984884    0.088542    0.097222    0.062549  1031788800         0.000000             0
1     1986-03-14    0.097222 -0.984854    0.097222    0.100694    0.064783   308160000         0.000054             1
2     1986-03-17    0.100694 -0.984824    0.100694    0.102431    0.065899   133171200         0.000076             2
3     1986-03-18    0.102431 -0.984824    0.098958    0.099826    0.064224    67766400         0.000087             3
4     1986-03-19    0.099826 -0.984914    0.097222    0.098090    0.063107    47894400         0.000071             4
...          ...         ...       ...         ...         ...         ...         ...              ...           ...
8520  2019-12-31  156.770004  4.480418  156.449997  157.699997  157.699997    18369400         0.983183          8520
8521  2020-01-02  158.779999  4.583021  158.330002  160.619995  160.619995    22622100         0.995796          8521
8522  2020-01-03  158.320007  4.555984  158.059998  158.619995  158.619995    21116200         0.992909          8522
8523  2020-01-06  157.080002  4.526520  156.509995  159.029999  159.029999    20813700         0.985128          8523
8524  2020-01-07  159.320007  4.546278  157.330002  157.580002  157.580002    18017762         0.999184          8524

[8525 rows x 9 columns]
>>>
```

6.FEATURE ENGINEERING:

Feature engineering is the process of creating new features from the existing raw data or modifying the existing features in a way that enhances the performance of machine learning models. The goal is to provide more informative, representative, and discriminative features to improve the model's ability to learn and generalize.

In the given dataset, a new column(feature) High minus Low is obtained by subtracting low from high, which can be used for further analysis.

High Minus Low:

```
>>> df['High_Minus_Low'] = df['High'] - df['Low']
>>> print(df)
            Date        Open        High         Low       Close   Adj Close      Volume  High_Minus_Low
0     1986-03-13    0.088542    0.101563    0.088542    0.097222    0.062549  1031788800        0.013021
1     1986-03-14    0.097222    0.102431    0.097222    0.100694    0.064783   308160000        0.005209
2     1986-03-17    0.100694    0.103299    0.100694    0.102431    0.065899   133171200        0.002605
3     1986-03-18    0.102431    0.103299    0.098958    0.099826    0.064224    67766400        0.004341
4     1986-03-19    0.099826    0.100694    0.097222    0.098090    0.063107    47894400        0.003472
...          ...         ...         ...         ...         ...         ...         ...             ...
8520  2019-12-31  156.770004  157.770004  156.449997  157.699997  157.699997    18369400        1.320007
8521  2020-01-02  158.779999  160.729996  158.330002  160.619995  160.619995    22622100        2.399994
8522  2020-01-03  158.320007  159.949997  158.059998  158.619995  158.619995    21116200        1.889999
8523  2020-01-06  157.080002  159.100006  156.509995  159.029999  159.029999    20813700        2.590011
8524  2020-01-07  159.320007  159.669998  157.330002  157.580002  157.580002    18017762        2.339996

[8525 rows x 8 columns]
>>>
```

7.HANDLING OUTLIERS:

Outliers are data points that significantly differ from other observations in a dataset, deviating markedly from the overall pattern or distribution. They can be unusually high or low values that don't conform to the typical behaviour of the dataset.

The threshold fixed are the end points or outliers, all the values above and below are range are excluded and this process is called handling outliers.

Date fixed as threshold:

```
>>> thresholds={'Date': ("1986-03-19","2020-01-02")}
>>> for col, (lower,upper) in thresholds.items():
...     df = df[(df[col] >= lower) & (df[col] <= upper)]
...
>>> print(df)
            Date        Open        High         Low       Close   Adj Close    Volume  High_Minus_Low
4     1986-03-19    0.099826    0.100694    0.097222    0.098090    0.063107  47894400        0.003472
5     1986-03-20    0.098090    0.098090    0.094618    0.095486    0.061432  58435200        0.003472
6     1986-03-21    0.095486    0.097222    0.091146    0.092882    0.059756  59990400        0.006076
7     1986-03-24    0.092882    0.092882    0.089410    0.090278    0.058081  65289600        0.003472
8     1986-03-25    0.090278    0.092014    0.089410    0.092014    0.059198  32083200        0.002604
...          ...         ...         ...         ...         ...         ...       ...             ...
8517  2019-12-26  157.559998  158.729996  157.399994  158.669998  158.669998  14520600        1.330002
8518  2019-12-27  159.449997  159.550003  158.220001  158.960007  158.960007  18412800        1.330002
8519  2019-12-30  158.990005  159.020004  156.729996  157.589996  157.589996  16348400        2.290008
8520  2019-12-31  156.770004  157.770004  156.449997  157.699997  157.699997  18369400        1.320007
8521  2020-01-02  158.779999  160.729996  158.330002  160.619995  160.619995  22622100        2.399994

[8518 rows x 8 columns]
>>>
```

8.DATA SPLITTING:

Outliers are data points that significantly differ from other observations in a dataset, deviating markedly from the overall pattern or distribution. They can be unusually high or low values that don't conform to the typical behavior of the dataset.

```
>>> import pandas as pd
>>> from sklearn.model_selection import train_test_split
>>> X=df.drop('Close', axis=1)
>>> y=df['Close']
>>> X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
>>> X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```

TRAINING SET:

Purpose: Used to train the model, allowing it to learn patterns and relationships in the data.
Size: Largest portion of the dataset (e.g., 70-80%).
Importance: Fundamental for model training, ensuring the model learns from a variety of examples

```
>>> print("Training set:")
Training set:
>>> print(X_train)
           Date       Open       High        Low  Adj Close     Volume  High_Minus_Low
659   1988-10-19   0.371528   0.373264   0.362847   0.237908   80681600        0.010417
2233  1995-01-11   3.820313   3.859375   3.796875   2.467880   31512000        0.062500
2541  1996-03-29   6.398438   6.507813   6.382813   4.146640   53508800        0.125000
6526  2012-01-27  29.450001  29.530001  29.170000  24.110039   44187700        0.360001
6166  2010-08-24  24.090000  24.350000  24.000000  19.198000   66522500        0.350000
...          ...        ...        ...        ...        ...        ...             ...
5738  2008-12-10  20.820000  20.959999  20.299999  15.850388   61499000        0.660000
5195  2006-10-13  28.340000  28.690001  28.309999  21.063829  129751900        0.380002
5394  2007-08-01  28.950001  29.549999  28.820000  21.975340   80006300        0.729999
864   1989-08-11   0.409722   0.413194   0.401042   0.258013   61472000        0.012152
7274  2015-01-20  46.299999  46.650002  45.570000  41.647015   36161900        1.080002

[5962 rows x 7 columns]
```

VALIDATION SET:

Purpose: Used to fine-tune the model's hyperparameters, aiding in model selection and preventing overfitting.
Size: Smaller portion of the dataset (e.g., 10-15%).
Importance: Helps optimize the model's performance and generalization.

```
>>> print("Validation set:")
Validation set:
>>> print(X_val)
           Date       Open       High        Low  Adj Close    Volume  High_Minus_Low
4273  2003-02-18  24.620001  24.990000  24.400000  16.058205  57415500        0.590000
2775  1997-03-04  12.406250  12.515625  12.328125   7.971599  63920800        0.187500
7203  2014-10-07  45.860001  45.930000  45.419998  40.618759  25723700        0.510002
1083  1990-06-25   1.050347   1.062500   1.031250   0.665696  68979200        0.031250
3189  1998-10-22  26.500000  27.531250  26.312500  17.692333  81910800        1.218750
...          ...        ...        ...        ...        ...       ...             ...
2306  1995-04-26   4.984375   5.015625   4.945313   3.201709  44305600        0.070312
3292  1999-03-23  43.171875  43.531250  41.562500  26.789812  69581200        1.968750
5856  2009-06-02  21.360001  21.980000  21.200001  16.676033  48935700        0.779999
304   1987-05-27   0.380208   0.388889   0.378472   0.244610  48758400        0.010417
1203  1990-12-13   1.013889   1.020833   0.996528   0.650059  50707200        0.024305

[1278 rows x 7 columns]
```

TESTING SET:

Purpose: Used to evaluate the model's performance on unseen data after training and validation.
Size: Smaller portion of the dataset (e.g., 10-15%).
Importance: Provides an unbiased evaluation of the model's performance and generalization to new data.

```
Testing set:
>>> print(X_test)
           Date       Open       High        Low  Adj Close     Volume  High_Minus_Low
1909  1993-09-29   2.617188   2.625000   2.523438   1.668708   87529600        0.101562
2878  1997-07-30  17.593750  17.750000  17.453125  11.349231   78324000        0.296875
7346  2015-05-04  48.369999  48.869999  48.180000  43.616077   34039500        0.689999
7540  2016-02-09  49.020000  50.240002  48.669998  45.445953   46740500        1.570004
401   1987-10-13   0.501736   0.513889   0.493056   0.327263   96809600        0.020833
...          ...        ...        ...        ...        ...        ...             ...
4314  2003-04-16  25.600000  25.740000  24.600000  16.084015   86178700        1.140000
4047  2002-03-26  29.549999  30.459999  29.155001  19.004787   69357200        1.304998
1604  1992-07-16   2.218750   2.257813   2.210938   1.452581   42291200        0.046875
7198  2014-09-30  46.369999  46.480000  46.009998  41.359222   33033100        0.470002
509   1988-03-17   0.451389   0.454861   0.442708   0.285378  125308800        0.012153

[1278 rows x 7 columns]
>>>
```

9.SAVING:

```
>>> df.to_csv('preprocessed_MSFT.csv', index=False)
>>>
```

10.CONCLUSION:

In the third phase, the dataset has been preprocessed, which is fundamental to building accurate and reliable predictive models. This involved handling missing values, scaling, encoding categorical features, and possibly applying other transformations like feature engineering or selection. The preprocessed dataset is now ready for the subsequent phases, where it will be utilized to train and validate models