Name: RISHU VERMA
Department: MBA in BABD
University: Thapar University
University

Name: SIDDHARTH DOGRA
Department: MBA IN BABD
University: Thapar

TEXTILE STOCK PRICE PREDICTION MODEL

APRIL 29,2023

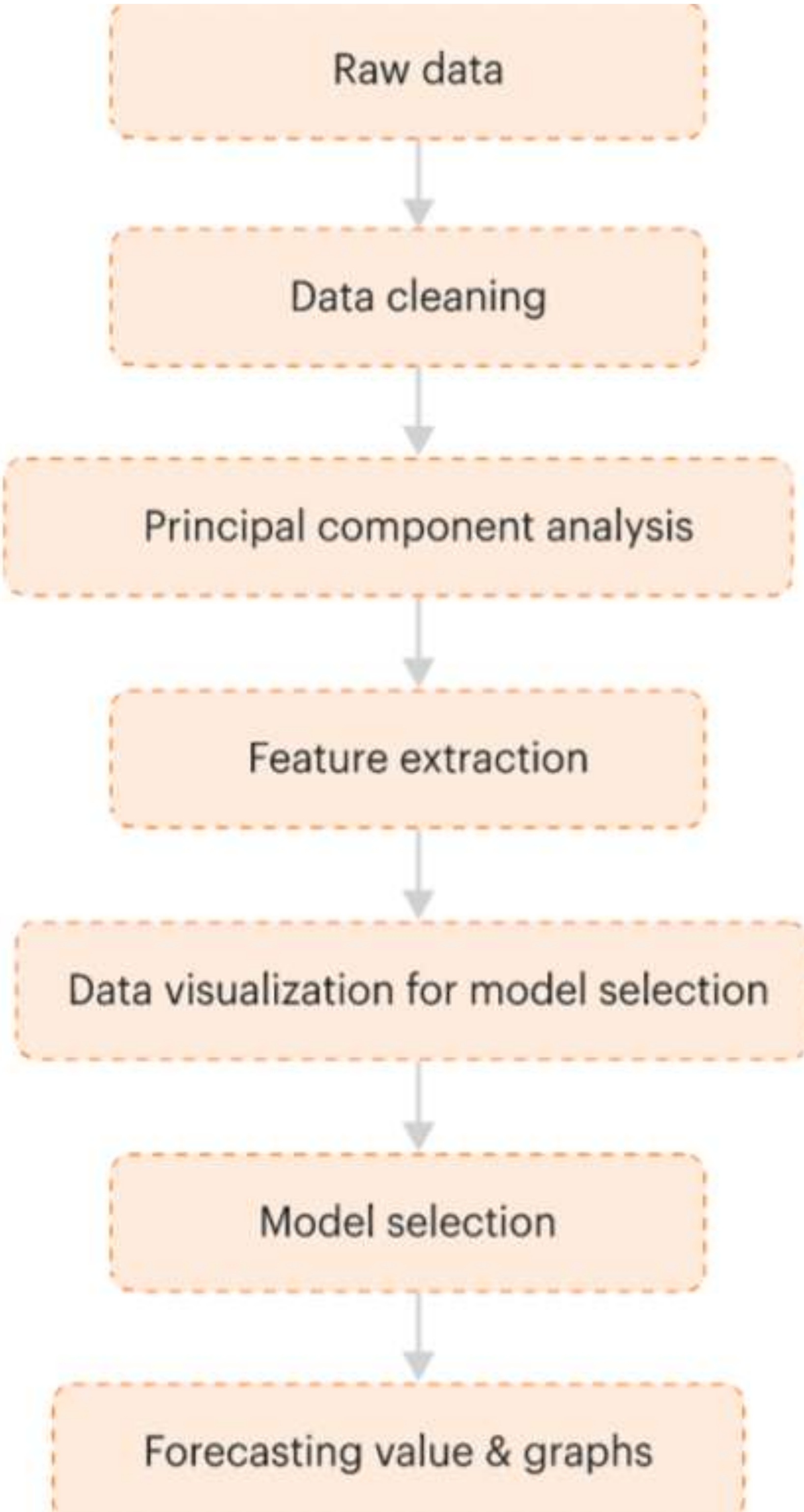
Overview

The dataset we are using here to train a textile stock price prediction model was downloaded from yahoo finance. It contains data about all the main features that contribute to the price of a stock . So let’s start this task by importing the necessary Python libraries and the dataset:

```
import pandas as pd
import numpy as np
# Load the data into a Pandas dataframe
df = pd.read_csv('Textile- STOCK_MARKET_DATA.csv')
```

	Company Name	Last Price	% Change	52 wk High	52 wk Low	Market Cap (Rs. cr)
1	KPR Mill	574.8	-2.95	704	479.55	19,647.47
2	Trident	30.35	1.68	58	29.35	15,466.23
3	Raymond	1,270.20	1.16	1,644.00	645	8,456.20
4	Swan Energy	253.2	-0.10	379	155	6,682.38
5	Welspun India	66.4	NaN	112.05	62.3	6,560.71
...	...	...	...	...	...	...
95	Amarjothi Spin	180.4	0.56	209.95	144	121.77
96	KG Petrochem	199	5.29	NaN	178.05	115.72
97	Virat Ind	234.9	2.24	282.6	125.3	115.65
98	Vippy Spinpro	188.25	-4.13	217.7	91.1	110.5
99	Premco Global	331.1	-2.10	492.95	290.55	109.42

99 rows × 6 columns


Raw data
Data cleaning
Principal component analysis
Feature extraction
Data visualization for model selection
Model selection
Forecasting value & graphs

There are 6 columns and 99 rows in this dataset, so it is very important to check whether or not this dataset contains null values before going any further:

```
df.isnull().sum()
```

```
Last Price              0
% Change                4
52 wk High              8
52 wk Low               3
Market Cap (Rs. cr)     0
dtype: int64
```

So this dataset does have a few null values, lets drop rows containing missing values and handle it:

```
df = df.dropna()
df.iloc[1:14]
```

	Company Name	Last Price	% Change	52 wk High	52 wk Low	Market Cap (Rs. cr)
1	KPR Mill	574.8	-2.95	704	479.55	19,647.47
2	Trident	30.35	1.68	58	29.35	15,466.23
3	Raymond	1,270.20	1.16	1,644.00	645	8,456.20
4	Swan Energy	253.2	-0.10	379	155	6,682.38
6	Alok Industries	12.47	-2.96	29.8	10.07	6,191.65
8	Garware Technic	2,953.00	0.27	3,752.55	2,611.20	6,017.67
10	PDS	325.25	-1.02	414.84	282	4,257.36
12	Indo Count	128.95	0.27	189.4	119.7	2,553.91
13	Kewal Kiran	394	4.65	592.35	178.5	2,428.03
15	Arvind	83	1.34	138.5	77.7	2,170.43
17	Siyaram Silk	452.55	-0.21	698	409.3	2,121.11
18	Ganesha Ecosph	891.25	1.58	985.05	543.25	1,945.55
19	Dollar Ind	316.7	0.24	639	310.25	1,796.20

SORTING ON BASIS OF LAST PRICE

```
df.sort_values('Last Price')
```

	Company Name	Last Price	% Change	52 wk High	52 wk Low	Market Cap (Rs. cr)
38	Cheviot Company	1,088.00	-0.26	1,485.00	1,051.00	654.64
3	Raymond	1,270.20	1.16	1,644.00	645	8,456.20
30	Ambika Cotton	1,441.00	-0.59	2,664.90	1,324.45	824.97
92	Indian Acrylics	10.03	-3.74	17.65	10	135.73
68	Digjam	103	1.98	219	87	206
...	...	...	...	...	...	...
65	MK Exim	84.15	0.18	119.7	66.3	226.46
49	Nahar Ent	84.8	-1.05	238	84.05	337.8
18	Ganesha Ecosph	891.25	1.58	985.05	543.25	1,945.55
63	Zodiac Clothing	90.3	-2.38	129.9	78.55	234.72
87	Weizmann	93.25	-0.59	132.15	42.3	147.89

86 rows × 6 columns

now let’s look at some of the other important insights to get an idea of what kind of data we’re dealing with:

- ```
  df.shape
  # df[1].value_counts()
  ```

  ```
  (86, 6)
  ```

- ```
  df.describe()
  ```

	% Change
count	86.000000
mean	0.283372
std	2.576751
min	-4.970000
25%	-1.155000
50%	0.225000
75%	1.827500
max	10.560000

Now lets try to Replace Missing Values With Median:

```python
for col in df.columns[1:]:
    # Check if the column contains string values
    if df[col].dtype == 'object':
        # Remove commas from the string values
        df[col] = df[col].str.replace(',', '').astype(float)

    # Calculate the median of the column
    fill_value = df[col].median()

    # Replace missing values with the median
    df[col].fillna(fill_value, inplace=True)

print("After replacing with median")
df
```

```
After replacing with median
```

```
After replacing with median
```

	Last Price	% Change	52 wk High	52 wk Low	Market Cap (Rs. cr)
0	38,014.85	0.64	54262.300	37138.95	42401.28
1	574.8	-2.95	704.000	479.55	19647.47
2	30.35	1.68	58.000	29.35	15466.23
3	1,270.20	1.16	1644.000	645.00	8456.20
4	253.2	-0.10	379.000	155.00	6682.38
...	...	...	...	...	...
95	180.4	0.56	209.950	144.00	121.77
96	199	5.29	213.825	178.05	115.72
97	234.9	2.24	282.600	125.30	115.65
98	188.25	-4.13	217.700	91.10	110.50
99	331.1	-2.10	492.950	290.55	109.42

Now to see if there is any missing value left:

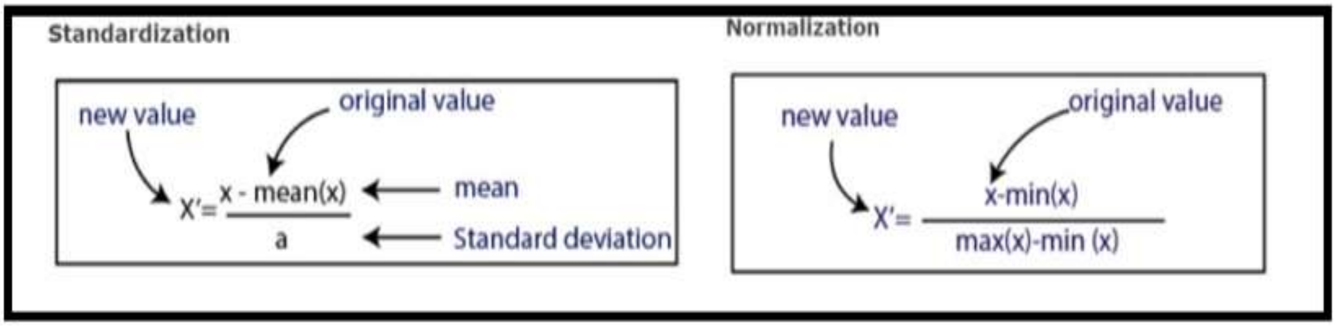
```
# Check for missing values
print(df.isnull().sum())
df.shape
```

```
Company Name           0
Last Price             0
% Change               0
52 wk High             0
52 wk Low              0
Market Cap (Rs. cr)    0
dtype: int64

   (86, 6)
```

We can see no missing values left,all have been taken care off.

Standard Scale or Normalization: The variables in the data set maybe in different scale or units, so it becomes difficult for the machine algorithm to do regression and give optimized and accurate results. In order to overcome this

problem, a need arises to scale such data within a specific range. Hence, we either use standard scalar or normalization technique to bring such data within a specific range. This helps to ensure uniformity within a data.



```python
from sklearn.preprocessing import MinMaxScaler

# Create a MinMaxScaler object
scaler = MinMaxScaler()

# Scale the data
df[df.columns[1:]] = scaler.fit_transform(df[df.columns[1:]])

print("After feature scaling")
df
```

```
After feature scaling
```

	Company Name	Last Price	% Change	52 wk High	52 wk Low	Market Cap (Rs. cr)
0	Page Industries	1.000000	0.361236	1.000000	1.000000	1.000000
1	KPR Mill	0.015061	0.130071	0.012878	0.012867	0.461981
2	Trident	0.000738	0.428203	0.000971	0.000745	0.363115
3	Raymond	0.033355	0.394720	0.030203	0.017322	0.197361
4	Swan Energy	0.006601	0.313587	0.006888	0.004128	0.155419
...	...	...	...	...	...	...
94	Salona Cotspin	0.006093	0.613651	0.006039	0.004855	0.000323
95	Amarjothi Spin	0.004686	0.356085	0.003772	0.003832	0.000292
97	Virat Ind	0.006119	0.464263	0.005111	0.003328	0.000147
98	Vippy Spinpro	0.004892	0.054089	0.003915	0.002408	0.000026
99	Premco Global	0.008650	0.184804	0.008988	0.007778	0.000000

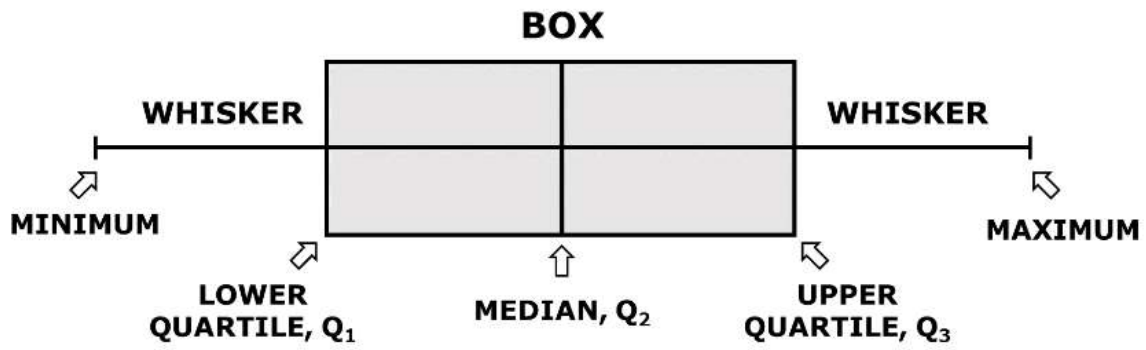
```
86 rows × 6 columns
```

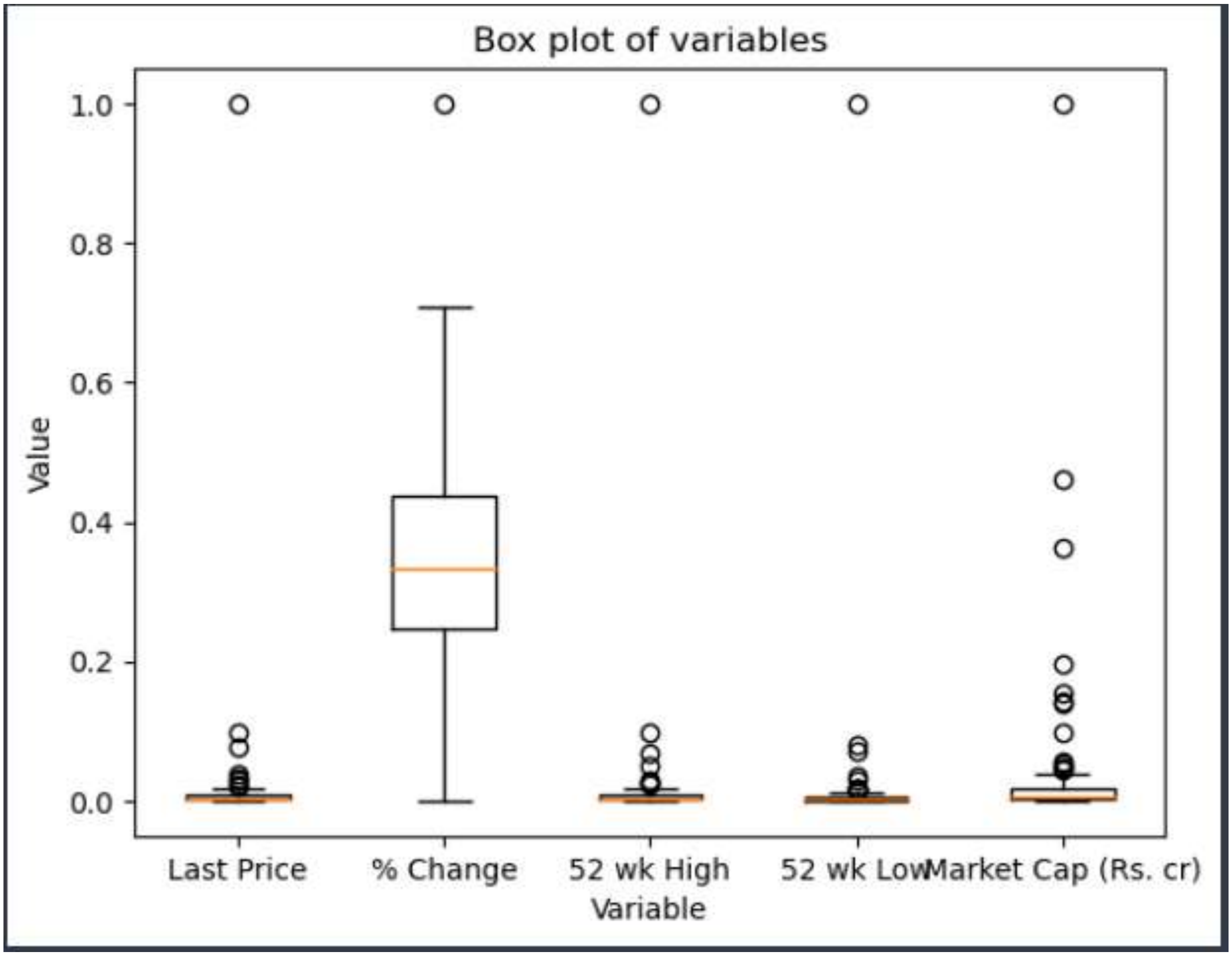
```python
# Create a MinMaxScaler object
scaler = MinMaxScaler()

# Normalize the data
df[df.columns[1:]] = scaler.fit_transform(df[df.columns[1:]])

print("After normalization")
df
```

DATA VISUALIZATION

For visualization we had used matplotlib library.In this we take help of box plot as it is useful for indicating whether a distribution is skewed and whether there are potential unusual observations (outliers) in the dataset.

Figure 4.5.2.1 Building a box and whisker plot





Standardization

The result of **standardization** is that the features will be rescaled to ensure the mean and the standard deviation to be 0 and 1, respectively.The concept of standardization comes into the picture when continuous independent variables are measured at different scales.

```
from sklearn.preprocessing import StandardScaler

# Create a StandardScaler object
scaler = StandardScaler()

# Standardize the data
df[df.columns[1:]] = scaler.fit_transform(df[df.columns[1:]])

print("After standardization")
df
```

The various machine learning algorithms used for the prediction stock price are Logestic regression, Multiple linear regression, KNN Regression, Decision tree regression, SVR (Support vector regression), Random Forest regression. Out of these models, the model which gives the most accurate prediction of the stock price is selected.

1. Logestic Regression: Logistic regression is another technique borrowed by machine learning from the field of statistics.

It is the go-to method for binary classification problems (problems with two class values). In this post you will discover the logistic regression algorithm for machine learning.

2. Decision Tree Regression: A decision tree is used to develop regression models and has a tree-like structure. It gradually divides a dataset into smaller and smaller subgroups depending on the information gain value for each unique feature while also developing an associated decision tree. Finally, we have a tree containing decision nodes and leaf node

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
LRC=LogisticRegression()
LRC.fit(X_train,Y_train)
```

```
C:\Users\user\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: Dat
aConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)


   LogisticRegression()
```

```
Y_pred=LRC.predict(X_test)
Y_pred
```

```
array(['PDS', 'Lakshmi Mills', 'Integra Essenti', 'Ruby Mills',
       'Nandan Denim', 'Integra Essenti', 'Arvind', 'Loyal Textiles',
       'Ruby Mills', 'Ruby Mills', 'Reliance Chemo', 'Integra Essenti',
       'RRIL', 'Mallcom (India)', 'Rupa and Comp', 'Jasch Ind',
       'Pasupati Acrylo', 'Filatex Fashion', 'Trident', 'Ganesha Ecosph',
       'JCT', 'Loyal Textiles', 'Sangam India', 'Filatex Fashion',
       'Mallcom (India)', 'Kewal Kiran'], dtype=object)
```

After doing this let's make a [heatmap using seaborn library](#).

A heatmap (or heat map) is a graphical representation of data where values are depicted by color.The variation in color may be by hue or intensity, giving obvious visual cues to the reader about how the phenomenon is clustered or varies over space. There are two fundamentally different categories of heat maps: the cluster heat map and the spatial heat map.

```
import seaborn as sns
```

```
#correlation among all the features of this dataset:
print(df.corr())
plt.figure(figsize=(20, 15))
correlations = df.corr()
sns.heatmap(correlations, cmap="coolwarm", annot=True)
plt.show()
```

```
#correlation among all the features of this dataset:
print(df.corr())
plt.figure(figsize=(20, 15))
correlations = df.corr()
sns.heatmap(correlations, cmap="coolwarm", annot=True)
plt.show()
```