Indian Institute of Technology Patna
भारतीय प्रौद्योगिकी संस्थान पटना

MUTUAL FUNDS

DATA SCIENCE PROJECT 2022

THE TEAM:

1 AMAN JHA

2 FERSHA JJ

3 GARGI CHANDRAKER

4 GARV JAIN

5 KANISHK GIRI

6 PRADIPTI MONDAL

Introduction to mutual funds

Mutual Funds refer to a pool of money collected from many investors to invest in securities like stocks, bonds, money market, instrument etc on the behalf of investors by the mutual fund companies'.

Companies give access to investor of the portfolio.when you buy a unit or share of a mutual fund, you are buying the performance of its portfolio or, more precisely, a part of the portfolio's value. Investing in a share of a mutual fund is different from investing in shares of stock. Unlike stock, mutual fund shares do not give its holders any voting rights. A share of a mutual fund represents investments in many different stocks (or other securities) instead of just one holding.

Terms used:

NAV(Net Asset Value):
Just like profits(Revenue - Cost), Net assets is calculated as Total value of entity's asset minus the total value of liabilities, it is used to identify potential investment opportunities with in mutual funds.
Repurchase price(Redemption Price) :
The Repurchase/Redemption Price is the price per Unit at which a Mutual Fund would 'repurchase' the units (i.e., buys back units from the investor) upon redemption of units or switch-outs of units to other schemes/plans of the Mutual Fund by the investors
Sale price:
Sale Price is the price payable per unit by an investor for purchase of units (subscription) and/or switch-in from other schemes of a mutual fund.

ISIN Div Reinvestment:

First of all, what is ISIN? ISIN stands for International Securities Identification Number.It is a 12-digit alphanumeric number. The first two letters of the ISIN code refer to the country in which the issuing company is based. The next nine digits identify specific security and act as a unique identifier.A dividend reinvestment plan (DRIP) is a program that allows investors to reinvest their cash dividends into additional shares or fractional shares of the underlying stock on the dividend payment date that the dividend that a person get on mutual fund or stock automatically is invested back in same fund/ stock.

ISIN Div Payout/ISIN Growth:

ISIN Growth simply refers to Growth Stocks, so what is a Growth Stock?A growth stock is any share in a company that is anticipated to grow at a rate significantly above the average growth for the market. These stocks generally do not pay dividends. This is because the issuers of growth stocks are usually companies that want to reinvest any earnings they accrue in order to accelerate growth in the short term. When investors invest in growth stocks, they anticipate that they will earn money through capital gains when they eventually sell their shares in the future.

HOW DID WE GET HERE?

DATA UNDERSTANDING

OUT OF 5711 DIFFERENT DATASETS , 31 DATASETS WERE CHOSEN IN A SYSTEMATIC RANDOM SAMPLING WITH A GAP OF 6 MONTH i.e. 1ST JANUARY, 1ST JULY OF EVERY YEAR FROM THE AVAILABLE DATA.

THE IMPORTANT COLUMNS IN THE DATA SETS AND THEIR DEFINITIONS ARE:

1> Net asset value(NAV)= (ASSETS-LIABILITIES)/TOTAL SHARES
2> Repurchase Price = Applicable NAV *(1 – Exit Load, if any)
3> Sale price (SP) = Cost Price + Profit Margin.

DATA PREPROCESSING

Data cleaning:

- Removed NaN values from ISIN Div Payout/ISIN Growth and ISIN Div Reinvestment

- Last 6 data sets for analysis is not being considered, i.e. 26 data files are used out of 31 files, due to the NaN values found in Repurchase Price and Sale Price.

Why is the last 6 data not being considered?

- In the last 6 data sets, the repurchase value and sale value columns, several rows of data have NaN values,which boils down to only one data column with one meaningful data column, i.e, net asset value.

- Due to this the analysis of data of last 6 dataset is not taken into consideration.

Walking through the code

Importing libraries:

- **Pandas**
- **Numpy**
- **Matplotlib**

PANDAS:

- **read_csv**: Pandas function which is “read_csv()“, reads the file from the provided location.
- pandas **“head()” function** to display the top 5 rows from our data set.
- **“tail()” function** will show the last 5 rows from our data set.
- **“shape” function** will show the dimensions in (No. of rows, No. of columns) format.
- **“columns”** get the name of all the features/columns in our data frame.
- **"info() function"** plays a major role for information
- about our data such as the column names, no. of non-null values in each column(feature), type of each column, memory usage, etc.
- Using **“isnull()”** and **“sum()”** functions, we can find the number of null values in a DataFrame for every feature.
- When column is completely empty,(after seeing the sum of null values(from isnull(), sum()) so it will not provide any information which is beneficial to us. Hence, we will drop that particular column using the pandas **“drop()”** function.
- Now, using the **“describe()” function**, we can get various information about the numerical columns in our DataFrame, such as total count of rows, mean, the minimum value, the maximum value, and the spread of the values in the particular feature.

DATA COLLECTION

```
#install Kaggle
!pip install -q Kaggle
```

Installing Kaggle in Google Colab with the help of pip.

```
#uploading our kaggle API token from local files
from google.colab import files
files.upload()
```

Here we first import files from Google colab and then upload our Kaggle API token.

```
#create a kaggle folder
!mkdir ~/.kaggle
```

Creating a kaggle folder.

```
#permission for the json to act
!cp kaggle.json ~/.kaggle/
```

Asking for permission for the json to act.

```
datasets download -d hk7797/mutual-fund-i
```

Downloading the dataset directly from kaggle.

```
!unzip mutual-fund-india.zip
```

Unzipping the downloaded files

Data Pre-processing and Cleaning

Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format.

Steps involved in data pre-processing :

1. **Data Cleaning**
2. **Data Transformation**
3. **Data Reduction**

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled.

Steps involved in data cleaning :

1. **Removal of unwanted observations**
2. **Fixing Structural errors**
3. **Managing Unwanted outliers**
4. **Handling missing data**

WALKING THROUGH THE STEPS OF DATA CLEANING AND PREPROCESSING.

```
data7 = pd.read_csv(r"/content/DailyNAV/data_2009_07_01.csv")
```

Reading the data file by the command pd.read_csv using pandas.

```
data7.head()
```

	Scheme Code	Scheme Name	ISIN Div Payout/ISIN Growth	ISIN Div Reinvestment	Net Asset Value	Repurchase Price	Sale Price	Date
0	110147	Canara Robeco Interval Scheme Series2 (Quarter...	INF760K01BG4	NaN	10.0326	10.0326	0.0000	2009-07-01
1	110146	Canara Robeco Interval Scheme Series2 (Quarter...	INF760K01BH2	NaN	10.2865	10.2865	0.0000	2009-07-01
2	110148	Canara Robeco Interval Scheme Series2 (Quarter...	INF760K01BI0	NaN	10.0269	10.0269	0.0000	2009-07-01
3	110145	Canara Robeco Interval Scheme Series2 (Quarter...	INF760K01BJ8	NaN	10.5730	10.5730	0.0000	2009-07-01
4	111879	Canara Robeco Short Term Fund- Institutional P...	INF760K01688	NaN	10.2699	10.2699	10.2699	2009-07-01

data.head() to show the starting 5 entries.

```
.tail()
```

Scheme Code	Scheme Name	ISIN Div Payout/ISIN Growth	ISIN Div Reinvestment	Net Asset Value	Repurchase Price	Sale Price	Date
105497	UTI Long Term Advantage Fund - Dividend Option	NaN	NaN	9.72	0	0	2009-07-01
105498	UTI Long Term Advantage Fund - Growth Option	NaN	NaN	9.72	0	0	2009-07-01
108231	UTI-Long Term Advantage Fund Series -II - Grow...	NaN	NaN	10.33	0	0	2009-07-01
108232	UTI-Long Term Advantage Fund Series-II - Divid...	NaN	NaN	10.33	0	0	2009-07-01
104129	Kotak Flexi Fund of Funds Series II - Dividend	NaN	NaN	11.339	11.339	11.339	2009-07-01

data.tail() to show the last 5 entries.

```
data7.isnull().sum()
```

```
Scheme Code                    0
Scheme Name                    0
ISIN Div Payout/ISIN Growth    516
ISIN Div Reinvestment          977
Net Asset Value                0
Repurchase Price               0
Sale Price                     0
Date                           0
dtype: int64
```

data.isnull().sum() to get the sum of number of null values in each column of the dataset.

```
89] data7.shape
```

```
(1590, 8)
```

data7.shape() to get the no of columns and rows

```
[ ] data8.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2071 entries, 0 to 2070
Data columns (total 8 columns):
 #   Column                          Non-Null Count  Dtype 
---  ------                          --------------  ----- 
 0   Scheme Code                     2071 non-null   int64 
 1   Scheme Name                     2071 non-null   object
 2   ISIN Div Payout/ISIN Growth     1242 non-null   object
 3   ISIN Div Reinvestment           814 non-null    object
 4   Net Asset Value                 2071 non-null   object
 5   Repurchase Price                2071 non-null   object
 6   Sale Price                      2071 non-null   object
 7   Date                            2071 non-null   object
dtypes: int64(1), object(7)
memory usage: 129.6+ KB
```

```
[ ] data8["Scheme Code"]=data8["Scheme Code"].to_string()
    data8["Net Asset Value"]=pd.to_numeric(data8["Net Asset Value"],errors='coerce')
    data8["Repurchase Price"]=pd.to_numeric(data8["Repurchase Price"],errors='coerce')
    data8["Sale Price"]=pd.to_numeric(data8["Sale Price"],errors='coerce')
    data8['Repurchase Price'] = data8['Repurchase Price'].fillna(0)
    data8['Sale Price'] = data8['Sale Price'].fillna(0)
    data8['Net Asset Value'] = data8['Net Asset Value'].fillna(0)

    data8.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 2071 entries, 0 to 2070
    Data columns (total 8 columns):
     #   Column                          Non-Null Count  Dtype 
    ---  ------                          --------------  ----- 
     0   Scheme Code                     2071 non-null   object
     1   Scheme Name                     2071 non-null   object
     2   ISIN Div Payout/ISIN Growth     1242 non-null   object
     3   ISIN Div Reinvestment           814 non-null    object
     4   Net Asset Value                 2071 non-null   float64
     5   Repurchase Price                2071 non-null   float64
     6   Sale Price                      2071 non-null   float64
     7   Date                            2071 non-null   object
    dtypes: float64(3), object(5)
    memory usage: 129.6+ KB
```

data7["Scheme Code"]=data7["Scheme Code"].to_string()

This scheme code of the dataset is in int64 and if used directly for final statistics calculation, using function "describe()", then even the statistics of the scheme code is calculated which isn't required and is meaningless.

Hence, it is being converted to object.

---

data7["Net Asset Value"]=pd.to_numeric(data7["Net Asset Value"],errors='coerce')
data7["Repurchase Price"]=pd.to_numeric(data7["Repurchase Price"],errors='coerce')
data7["Sale Price"]=pd.to_numeric(data7["Sale Price"],errors='coerce')

The Net Asset, Repurchase and Sales are in Object data type, and for statistics calculation it is being converted into numeric type , so they are being converted into float64.
The cells with 0 value when converted into float64 become null values, these values are difficult to work with during data analysis(like hypothesis testing). So, null values are being converted to 0 using the function fillna().

```
dataclean_7 = data7.drop(['Scheme Name', 'Date','ISIN Div Payout/ISIN Growth','ISIN Div Reinvestment'], axis=1)
dataclean_7.head()
```

	Scheme Code	Net Asset Value	Repurchase Price	Sale Price
0	0 110147\n1 110146\n2 110148...	10.0326	10.0326	0.0000
1	0 110147\n1 110146\n2 110148...	10.2865	10.2865	0.0000
2	0 110147\n1 110146\n2 110148...	10.0269	10.0269	0.0000
3	0 110147\n1 110146\n2 110148...	10.5730	10.5730	0.0000
4	0 110147\n1 110146\n2 110148...	10.2699	10.2699	10.2699

Finally moving on to the cleaning of data after preprocessing it.

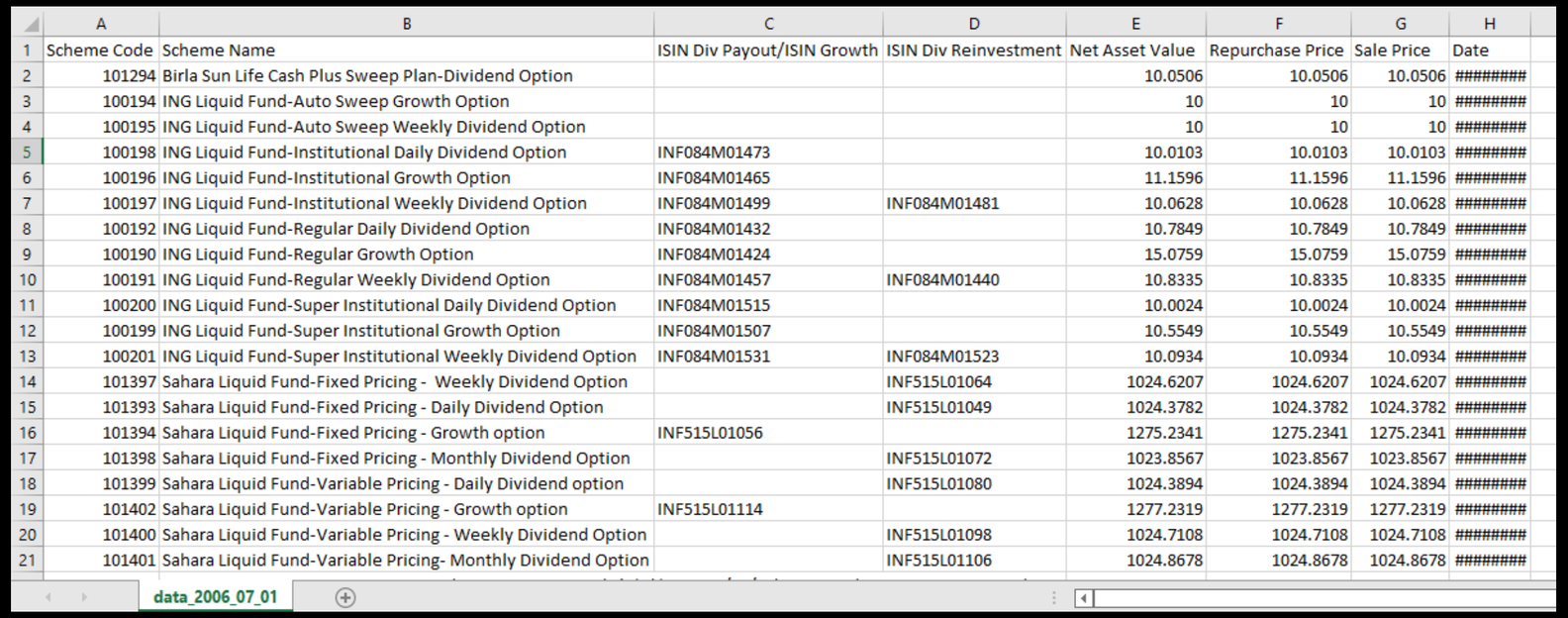
We are using "**.drop (['Name1', 'Name2', 'Name3', 'Name4'], axis=1)**"

Date is the same for every entry and is the date of the dataset. Hence we are dropping it.

'Scheme Name', 'ISIN Div Payout/ISIN Growth' and 'ISIN Div Reinvestment' will not be used in our further steps so we are dropping it. Then showing the initial part of the cleaned data by .head().

VISUALIZATION: using matplotlib

- Data visualization is the graphical representation of information and data in a pictorial or graphical format
- plotly.express: Plotly Express provides functions to visualize a variety of types of data. Most functions such as px. bar or px.
- box: In a box plot, we draw a box from the first quartile to the third quartile. A vertical line goes through the box at the median. The whiskers go from each quartile to the minimum or maximum.
- scatter:The scatter() function plots one dot for each observation.
- "xlabel", "ylabel" labels the x and y axis respectively. "title" is used to label the whole graph.
- "show" is to show the graph.



	A	B	C	D	E	F	G	H
1	Scheme Code	Scheme Name	ISIN Div Payout/ISIN Growth	ISIN Div Reinvestment	Net Asset Value	Repurchase Price	Sale Price	Date
2	101294	Birla Sun Life Cash Plus Sweep Plan-Dividend Option			10.0506	10.0506	10.0506	########
3	100194	ING Liquid Fund-Auto Sweep Growth Option			10	10	10	########
4	100195	ING Liquid Fund-Auto Sweep Weekly Dividend Option			10	10	10	########
5	100198	ING Liquid Fund-Institutional Daily Dividend Option	INF084M01473		10.0103	10.0103	10.0103	########
6	100196	ING Liquid Fund-Institutional Growth Option	INF084M01465		11.1596	11.1596	11.1596	########
7	100197	ING Liquid Fund-Institutional Weekly Dividend Option	INF084M01499	INF084M01481	10.0628	10.0628	10.0628	########
8	100192	ING Liquid Fund-Regular Daily Dividend Option	INF084M01432		10.7849	10.7849	10.7849	########
9	100190	ING Liquid Fund-Regular Growth Option	INF084M01424		15.0759	15.0759	15.0759	########
10	100191	ING Liquid Fund-Regular Weekly Dividend Option	INF084M01457	INF084M01440	10.8335	10.8335	10.8335	########
11	100200	ING Liquid Fund-Super Institutional Daily Dividend Option	INF084M01515		10.0024	10.0024	10.0024	########
12	100199	ING Liquid Fund-Super Institutional Growth Option	INF084M01507		10.5549	10.5549	10.5549	########
13	100201	ING Liquid Fund-Super Institutional Weekly Dividend Option	INF084M01531	INF084M01523	10.0934	10.0934	10.0934	########
14	101397	Sahara Liquid Fund-Fixed Pricing - Weekly Dividend Option		INF515L01064	1024.6207	1024.6207	1024.6207	########
15	101393	Sahara Liquid Fund-Fixed Pricing - Daily Dividend Option		INF515L01049	1024.3782	1024.3782	1024.3782	########
16	101394	Sahara Liquid Fund-Fixed Pricing - Growth option	INF515L01056		1275.2341	1275.2341	1275.2341	########
17	101398	Sahara Liquid Fund-Fixed Pricing - Monthly Dividend Option		INF515L01072	1023.8567	1023.8567	1023.8567	########
18	101399	Sahara Liquid Fund-Variable Pricing - Daily Dividend option		INF515L01080	1024.3894	1024.3894	1024.3894	########
19	101402	Sahara Liquid Fund-Variable Pricing - Growth option	INF515L01114		1277.2319	1277.2319	1277.2319	########
20	101400	Sahara Liquid Fund-Variable Pricing - Weekly Dividend Option		INF515L01098	1024.7108	1024.7108	1024.7108	########
21	101401	Sahara Liquid Fund-Variable Pricing- Monthly Dividend Option		INF515L01106	1024.8678	1024.8678	1024.8678	########

data_2006_07_01

The data in the excel sheet looks like this!

```
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
##import os
##print(os.listdir())
import matplotlib.pyplot as plt
```

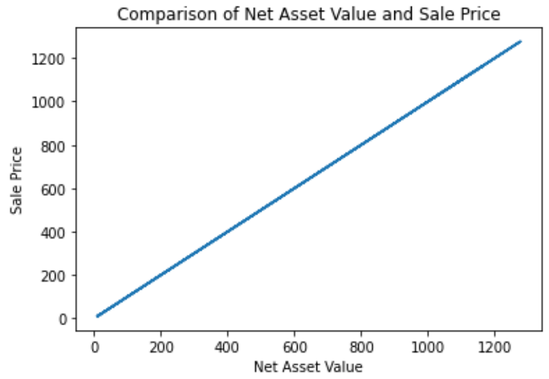
Starting by importing the libraries which we will use in our code.

VISUALIZATION FOR 1ST DATASET

LINE PLOTS:

```
plt.plot(dataclean_1['Net Asset Value'], dataclean_1['Sale Price'] )
Fitting_Log = np.polyfit(np.array(dataclean_1['Net Asset Value']), np.array(dataclean_1['Sale Price']), 1)
Slope_Log_Fitted = Fitting_Log[0]
print(Slope_Log_Fitted)
plt.xlabel('Net Asset Value')
plt.ylabel('Sale Price')
plt.title('Comparison of Net Asset Value and Sale Price')
plt.show
```

```
0.9999999999999998
<function matplotlib.pyplot.show>
```

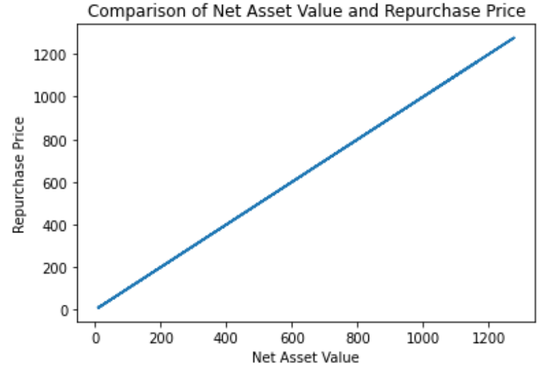


Graph of Net Asset Value Vs Sale Price.

Conclusion: Sale price varies linearly Net Asset value with slope 0.9999999999999998

```
plt.plot(dataclean_1['Net Asset Value'], dataclean_1['Repurchase Price'] )
Fitting_Log = np.polyfit(np.array(dataclean_1['Net Asset Value']), np.array(dataclean_1['Repurchase Price']), 1)
Slope_Log_Fitted = Fitting_Log[0]
print(Slope_Log_Fitted)
plt.xlabel('Net Asset Value')
plt.ylabel('Repurchase Price')
plt.title('Comparison of Net Asset Value and Repurchase Price')
plt.show
```

```
0.9999999999999998
<function matplotlib.pyplot.show>
```



Graph of Net Asset Value Vs Repurchase Price

Conclusion: Repurchase price varies linearly N Asset value with slope 0.9999999999999998.

Sale Price = Applicable NAV *(1 + Sales Load, if any), Repurchase Price = Applicable NAV *(1 - Exit Load, if any)

NET ASSET VALUE,REPURCHASE VALUE,SALES PRICE OR VARIOUS GRAPHS

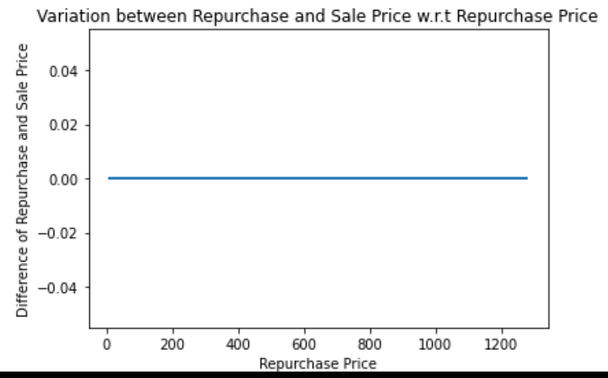
```
plt.plot(dataclean_1['Sale Price'], dataclean_1['Repurchase Price'] )
Fitting_Log = np.polyfit(np.array(dataclean_1['Sale Price']), np.array(dataclean_1['Repurchase Price']), 1)
Slope_Log_Fitted = Fitting_Log[0]
print(Slope_Log_Fitted)
plt.ylabel('Repurchase Price')
plt.xlabel('Sale Price')
plt.title('Comparison of Sale Price and Repurchase Price')
plt.show
```

```
0.9999999999999998
<function matplotlib.pyplot.show>
```

- **Graph of Sale Price Vs Repurchase Price.**
- **Conclusion:** Repurchase price varies linearly with Sale Price with slope 0.9999999999999998.

```
column_5= dataclean_1['Sale Price']- dataclean_1['Repurchase Price']
dataclean_1["Column_5"] = column_5
plt.plot(dataclean_1['Repurchase Price'], dataclean_1['Column_5'])
Fitting_Log = np.polyfit(np.array(dataclean_1['Repurchase Price']),np.array(dataclean_1['Column_5']), 1)
Slope_Log_Fitted = Fitting_Log[0]
print(Slope_Log_Fitted)
plt.xlabel('Repurchase Price')
plt.ylabel('Difference of Repurchase and Sale Price')
plt.title('Variation between Repurchase and Sale Price w.r.t Repurchase Price')
plt.show
```

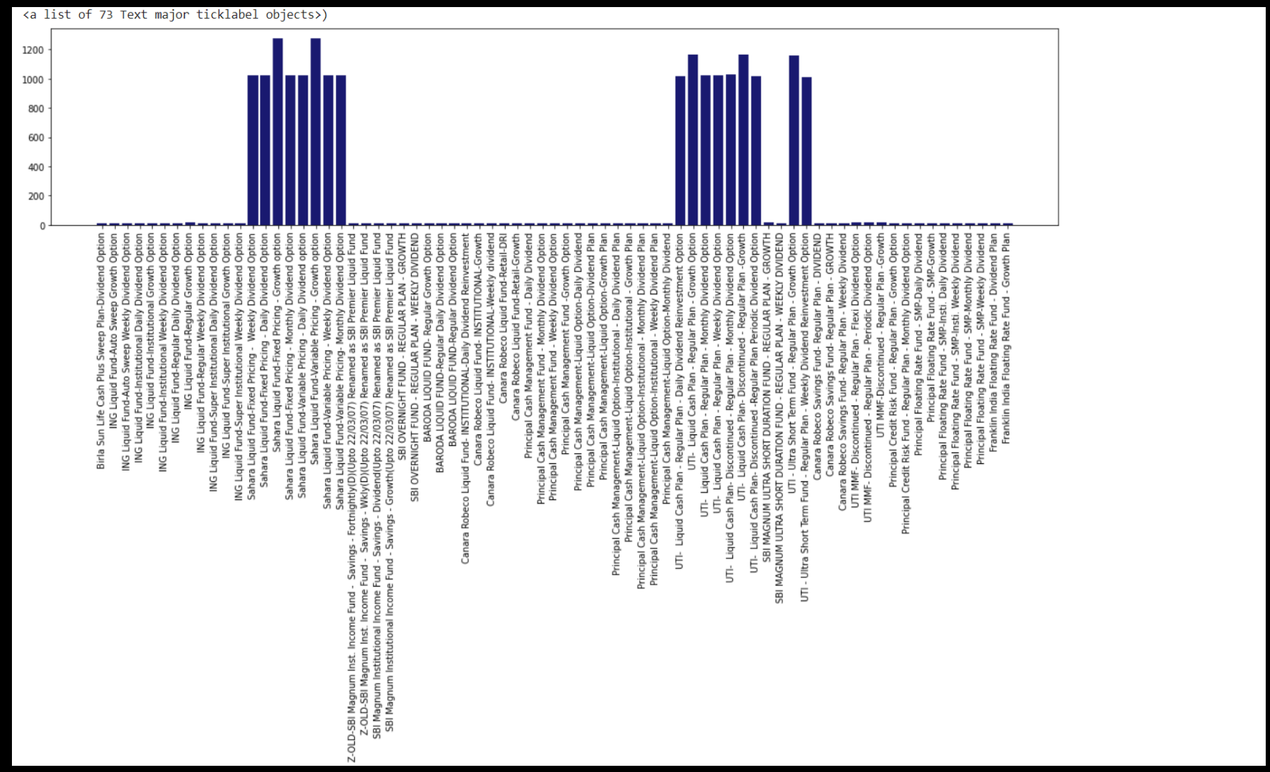
```
0.0
<function matplotlib.pyplot.show>
```



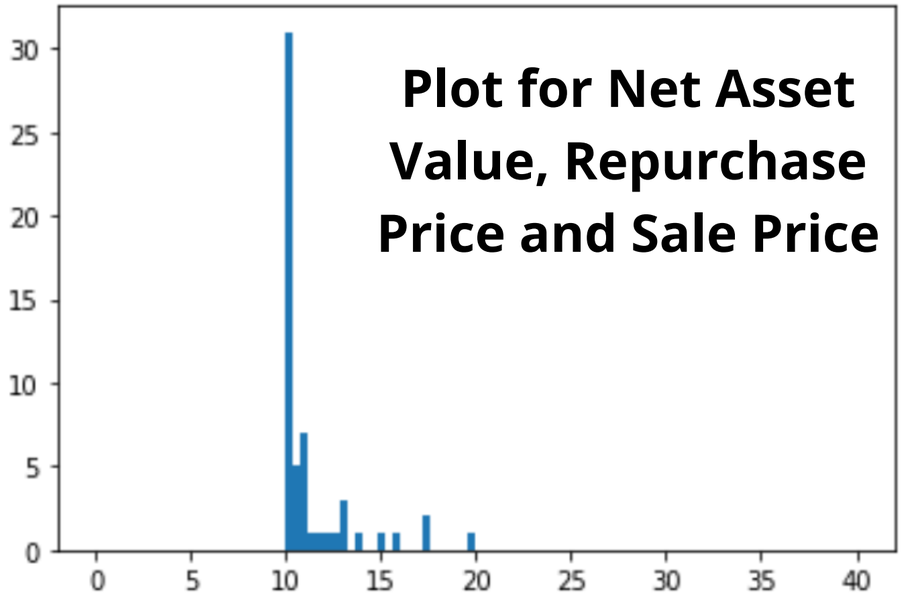
- **Graph of difference of Sale and Repurchase Price Vs Repurchase Price.**
- **Conclusion:** Repurchase price varies linearly with Sale Price with slope 0 i.e. Repurchase and sale price is same.

Bargraphs for 1st dataset:

```
plt.figure(figsize=(20,4))

plt.bar(x=data1['Scheme Name'],

        height=data1['Net Asset Value'],

        color='midnightblue')

plt.xticks(rotation=90)
```



<function matplotlib.pyplot.show>



Plot for Net Asset Value, Repurchase Price and Sale Price

```
plt.hist(dataclean_1['Net Asset Value'], range = (0,40), bins = 100)
plt.show
```

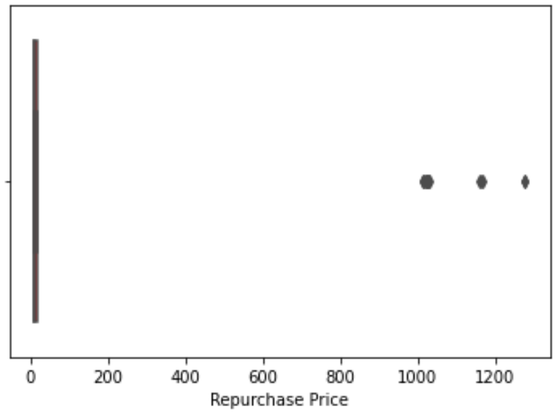
```
[379] plt.hist(dataclean_1['Sale Price'], range = (0,40), bins = 100)
      plt.show
```

```
[380] plt.hist(dataclean_1['Repurchase Price'], range = (0,40), bins = 100)
      plt.show
```

BOX PLOT FOR 1ST DATASET:

```
sns.boxplot(dataclean_1['Repurchase Price'], color='red')
```

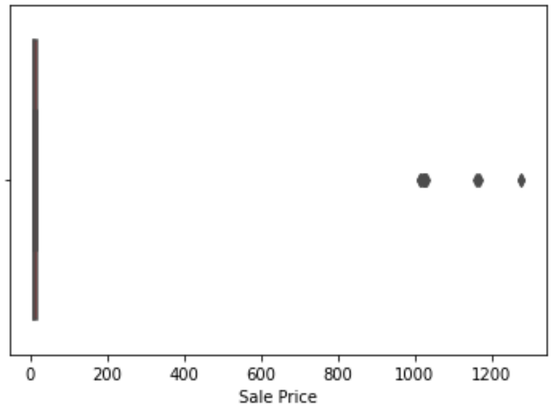
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4c221ffc50>
```



Box plot for Repurchase Price

```
sns.boxplot(dataclean_1['Sale Price'], color='red')
```

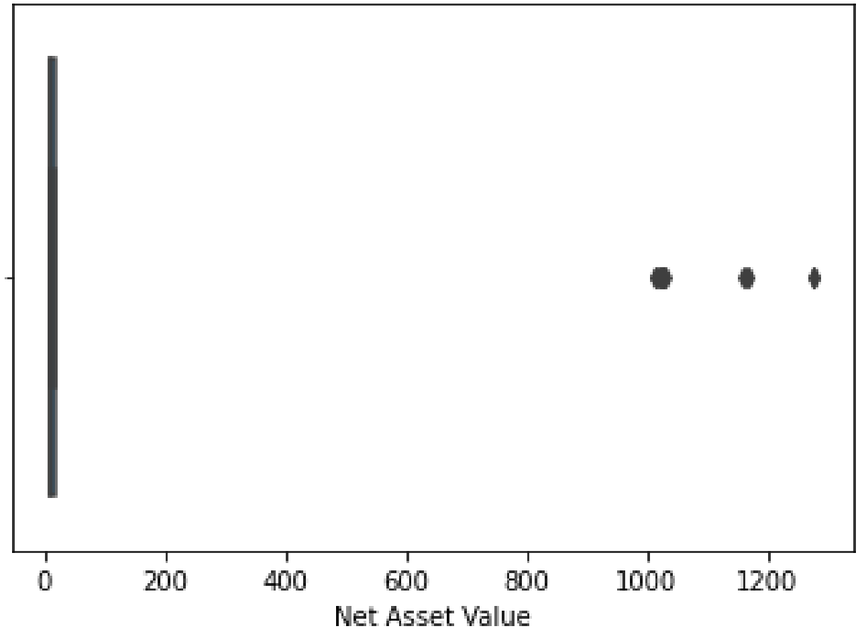
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4c2275dbd0>
```



Box plot for Sale Price

```
sns.boxplot(dataclean_1['Net Asset Value'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4c26840250>
```



Box plot for Net Asset Value

Conclusion:

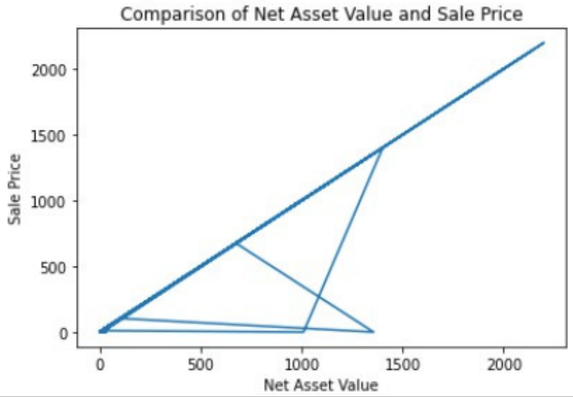
For 1st dataset Net Asset Value, Repurchase Price and Sale Price is almost same. Most of the Values are near 10 and some of them are outliers.

VISUALIZATION FOR 6TH DATASET

LINE PLOTS:

```
plt.plot(dataclean_6['Net Asset Value'], dataclean_6['Sale Price'] )
Fitting_Log = np.polyfit(np.array(dataclean_6['Net Asset Value']),np.array(dataclean_6['Sale Price']),1)
Slope_Log_Fitted=Fitting_Log[0]
print(Slope_Log_Fitted)
plt.xlabel('Net Asset Value')
plt.ylabel('Sale Price')
plt.title('Comparison of Net Asset Value and Sale Price')
plt.show
```

```
0.9886106468359787
<function matplotlib.pyplot.show>
```

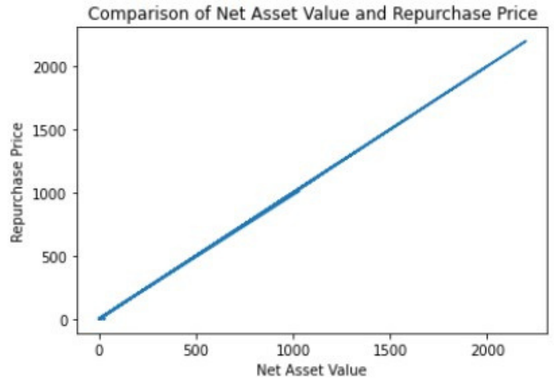


Graph of Net Asset Value Vs Sale Price.

Conclusion: Sale price varies linearly Net Asset value with slope 0.9886106468359787.

```
plt.plot(dataclean_6['Net Asset Value'], dataclean_6['Repurchase Price'] )
Fitting_Log = np.polyfit(np.array(dataclean_6['Net Asset Value']),np.array(dataclean_6['Repurchase Price']),1)
Slope_Log_Fitted=Fitting_Log[0]
print(Slope_Log_Fitted)
plt.xlabel('Net Asset Value')
plt.ylabel('Repurchase Price')
plt.title('Comparison of Net Asset Value and Repurchase Price')
plt.show
```

```
0.9995278589156632
<function matplotlib.pyplot.show>
```



Graph of Net Asset Value Vs Repurchase Price.

Conclusion: Repurchase price varies linearly Net Asset value with slope 0.9995278589156632.

Sale Price = Applicable NAV *(1 + Sales Load, if any), Repurchase Price = Applicable NAV *(1 - Exit Load, if any)

NET ASSET VALUE,REPURCHASE VALUE,SALES PRICE OR VARIOUS GRAPHS

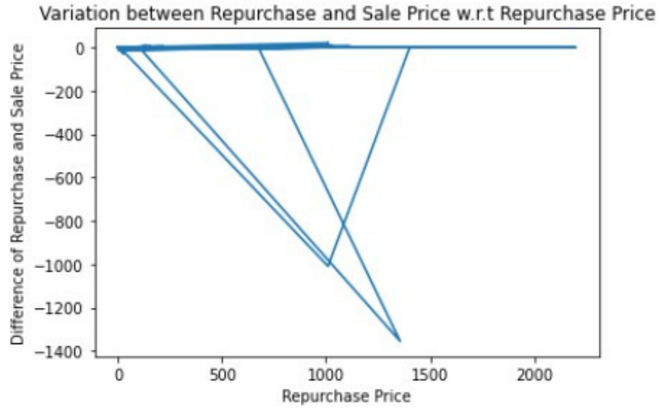
```
plt.plot(dataclean_6['Sale Price'], dataclean_6['Repurchase Price'] )
Fitting_Log = np.polyfit(np.array(dataclean_6['Sale Price']),np.array(dataclean_6['Repurchase Price']),1)
Slope_Log_Fitted=Fitting_Log[0]
print(Slope_Log_Fitted)
plt.ylabel('Repurchase Price')
plt.xlabel('Sale Price')
plt.title('Comparison of Sale Price and Repurchase Price')
plt.show
```

```
0.9964728481966566
<function matplotlib.pyplot.show>
```

- **Graph of Sale Price Vs Repurchase Price.**
- **Conclusion:** Repurchase price varies linearly with Sale Price with slope 0.9964728481966566.

```
plt.plot(dataclean_6['Repurchase Price'], dataclean_6['Column_5'])
Fitting_Log = np.polyfit(np.array(dataclean_6['Repurchase Price']),np.array(dataclean_6['Col
Slope_Log_Fitted=Fitting_Log[0]
print(Slope_Log_Fitted)
plt.xlabel('Repurchase Price')
plt.ylabel('Difference of Repurchase and Sale Price')
plt.title('Variation between Repurchase and Sale Price w.r.t Repurchase Price')
plt.show
```

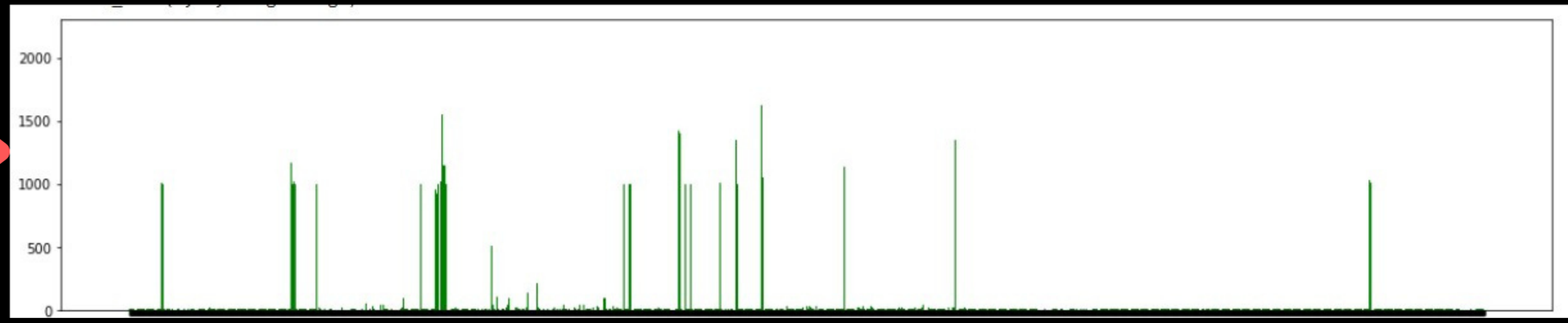
```
-0.010952830734941438
<function matplotlib.pyplot.show>
```

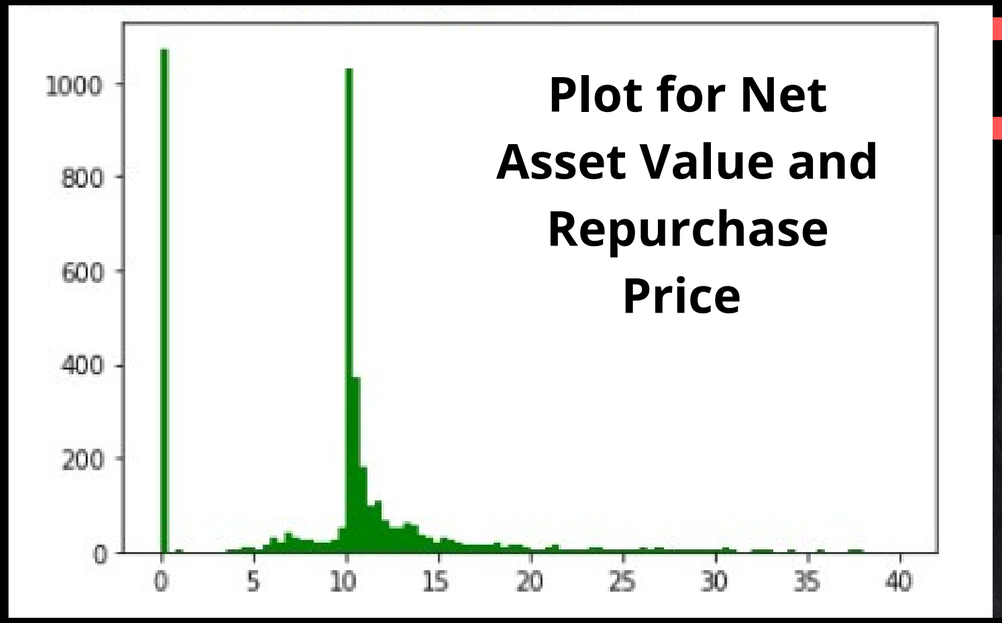


- **Graph of difference of Sale and Repurchase Price Vs Repurchase Price.**
- **Conclusion:** Repurchase price varies linearly with Sale Price with slope -0.010952830734941438 i.e. Repurchase and sale price is almost same.

Bargraphs for 6th dataset:

```
plt.figure(figsize=(20,4))
plt.bar(x=data6['Scheme Name'],
        height=data6['Net Asset Value'],
        color='green')
plt.xticks(rotation=90)
```

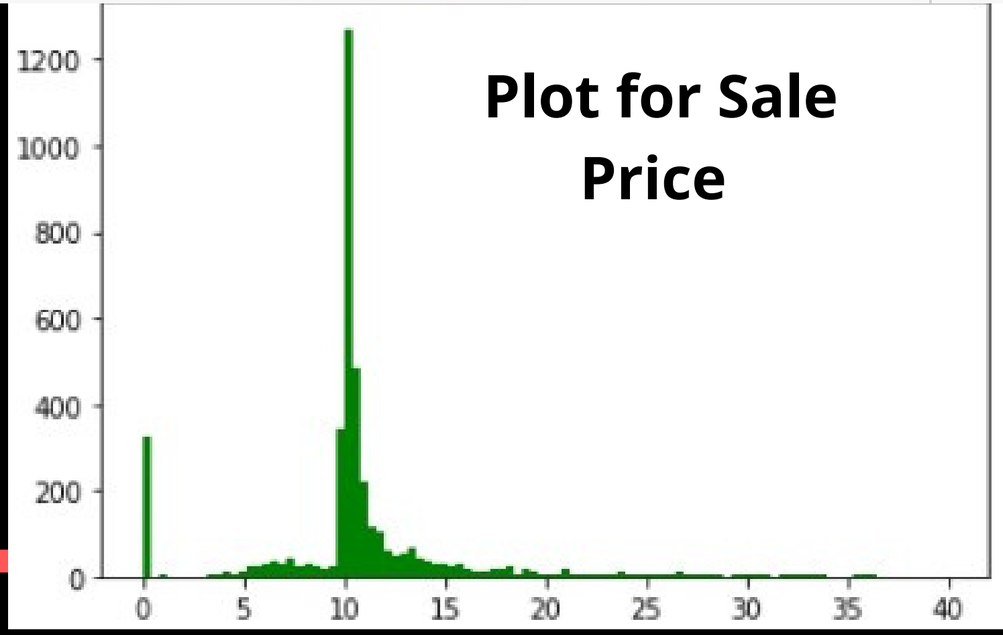




Plot for Net Asset Value and Repurchase Price

```
plt.hist(dataclean_6['Net Asset Value'], range = (0,40), bins = 100,color='green')
```

```
plt.hist(dataclean_6['Repurchase Price'], range = (0,40), bins = 100, color='green')
plt.show
```



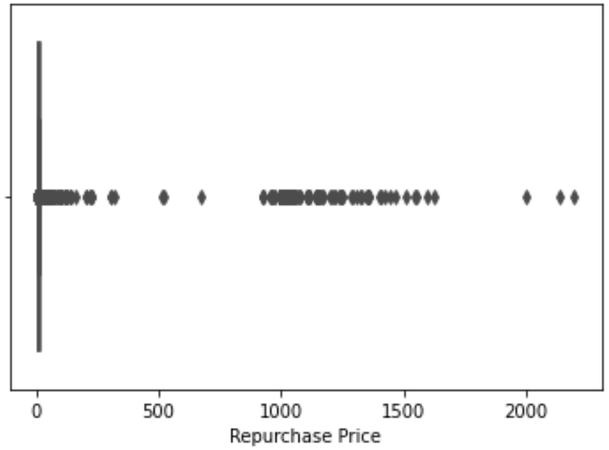
Plot for Sale Price

```
plt.hist(dataclean_6['Sale Price'], range = (0,40), bins = 100, color='green')
plt.show
```

BOX PLOT FOR 6TH DATASET:

```
[468] sns.boxplot(dataclean_6['Repurchase Price'], color='red')
```

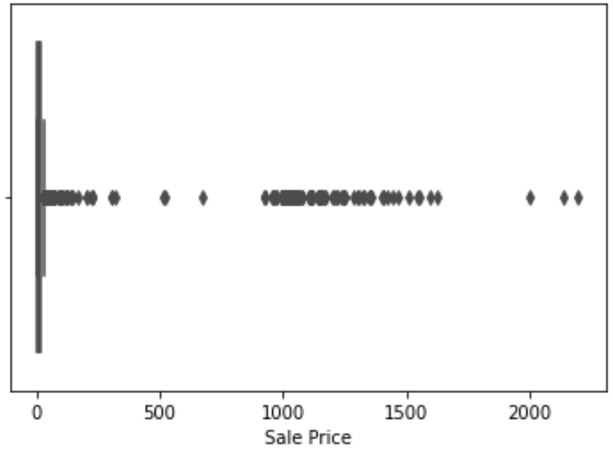
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4c21f707d0>
```



Box plot for Repurchase Price

```
sns.boxplot(dataclean_6['Sale Price'], color='red')
```

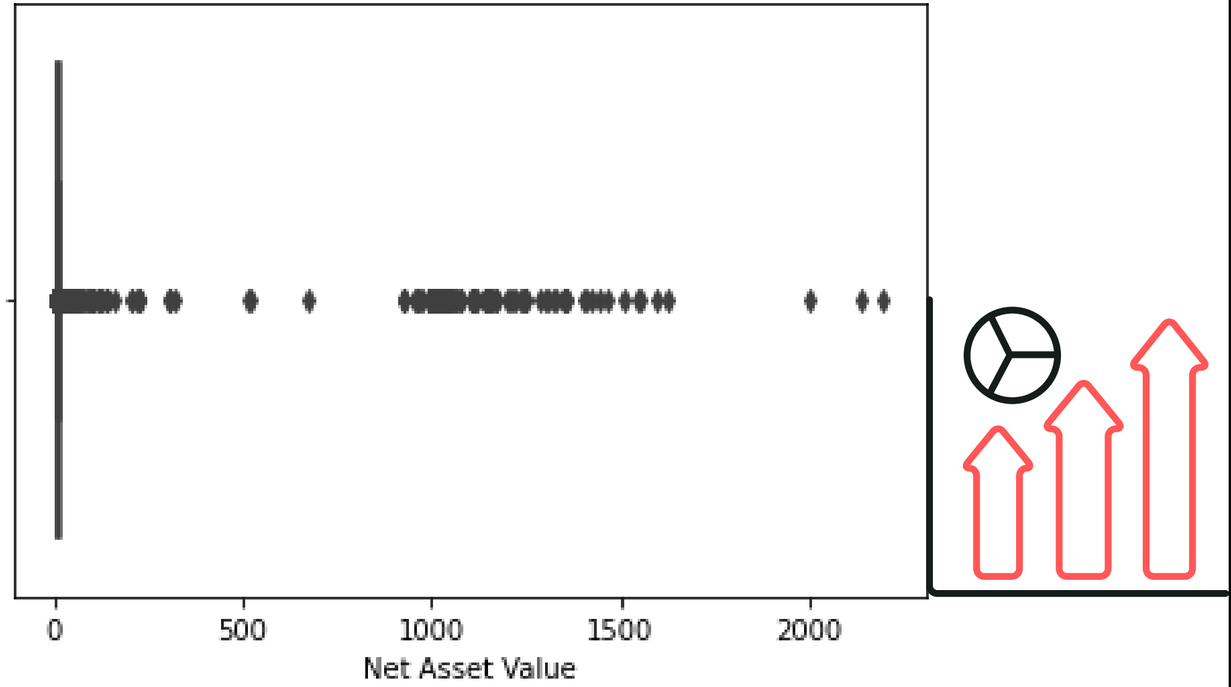
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4c21ed5750>
```



Box plot for Sale Price

```
sns.boxplot(dataclean_6['Net Asset Value'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ce224d850>
```



Box plot for Net Asset Value

Conclusion:

For 6th dataset Net Asset Value, Repurchase Price and Sale Price is almost same. Most of the Values are near 10 and some of them are outliers.

Statistics

- **describe()**:

The describe() method from pandas library is used for calculating some statistical data like percentile, mean and std of the numerical values of the Series or DataFrame. It analyzes both numeric and object series and also the DataFrame column sets of mixed data types.

Syntax

1. **DataFrame.describe(percentiles=None, include=None, exclude=None)**

Parameters

- **percentile**: It is an optional parameter which is a list like data type of numbers that should fall between 0 and 1. Its default value is [.25, .5, .75], which returns the 25th, 50th, and 75th percentiles.
- **include**: It is also an optional parameter that includes the list of the data types while describing the DataFrame. Its default value is None.
- **exclude:** It is also an optional parameter that exclude the list of data types while describing DataFrame. Its default value is None.

Returns

It returns the statistical summary of the Series and DataFrame.

```
dataclean_1.describe()
```

	Net Asset Value	Repurchase Price	Sale Price
count	73.000000	73.000000	73.000000
mean	259.519208	259.519208	259.519208
std	455.884939	455.884939	455.884939
min	10.000000	10.000000	10.000000
25%	10.032500	10.032500	10.032500
50%	10.833500	10.833500	10.833500
75%	17.559500	17.559500	17.559500
max	1277.231900	1277.231900	1277.231900

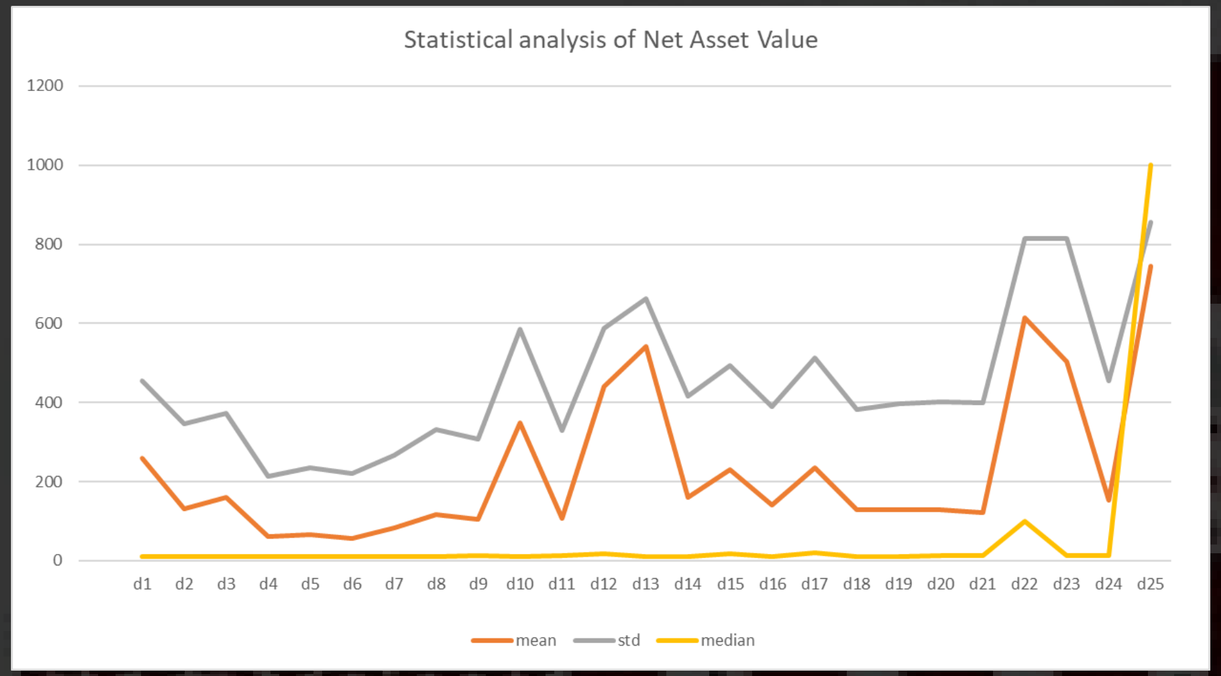
*Percentile meaning: how many of the values are less than the given percentile

EXPLANATION OF TERMS IN THE DESCRIBED CLEAN DATA

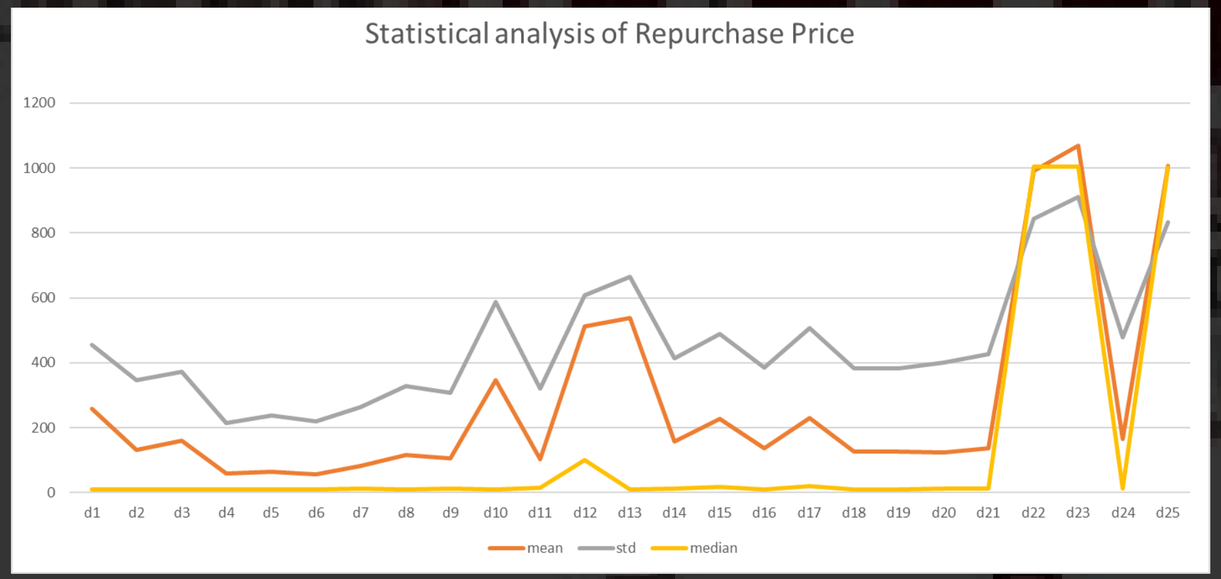
count - The number of non-empty values.
mean - The average (mean) value of the column.
std - The standard deviation of the column.
min - the minimum value of the column.

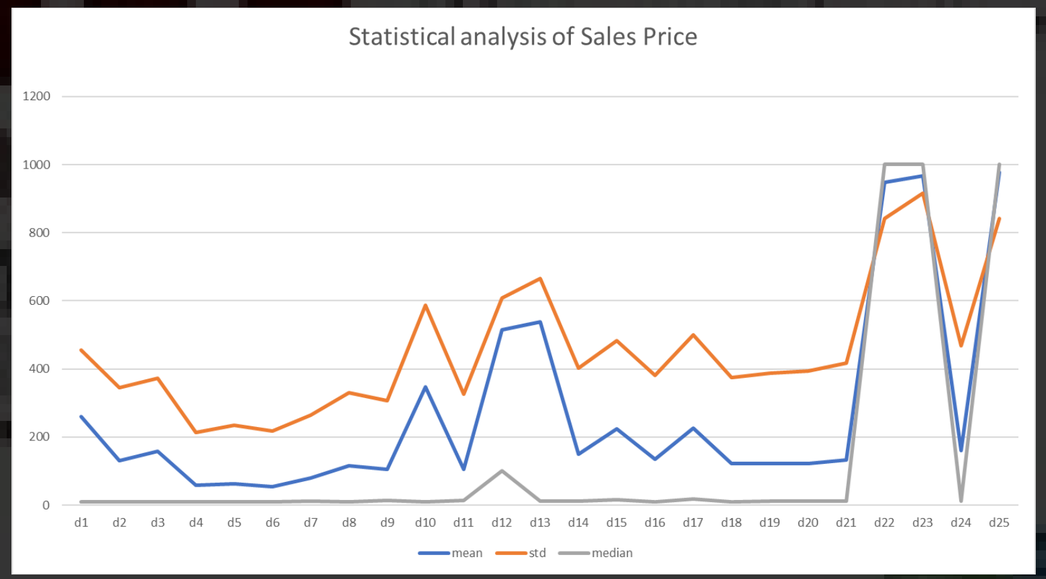
```
[ ]  dataclean_6.describe()
```

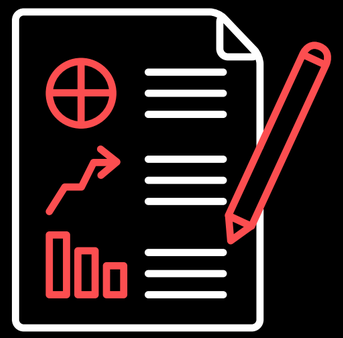
	Net Asset Value	Repurchase Price	Sale Price	Column_5
count	4194.000000	4194.000000	4194.000000	4194.000000
mean	57.001870	56.534043	54.315146	-2.218896
std	219.482123	219.387542	218.568580	26.375546
min	0.000000	0.000000	0.000000	-1356.048200
25%	10.032225	10.000000	0.000000	0.000000
50%	10.420550	10.291100	10.175850	0.000000
75%	11.766650	11.741775	11.758275	0.100775
max	2199.277900	2199.277900	2199.277900	20.641600



This is the plot of mean, median and standard deviation of Net Asset Value,Repurchase price and Sale price for the data sets we have taken for analysis.
Mean and standard deviation of Net Asset value ,Repurchase Price and Sale Price are almost following the same trend for the datasets plotted.
For median, Repurchase price and Sale price follow the same trend through out the plot and Net Asset Value follow the trend till data set 21 but, it highly deviates from dataset 21 to 24 and again follow the same trend after dataset 24.







Hypothesis

- **scipy.stats: scipy. stats module specializes in random variables** and probability distributions. It implements more than 80 continuous distributions and 10 discrete distributions.
- import **scipy.stats** as stats**(used in 2 sample T test)**
- from **statsmodels.stats** import weightstats as stests**(used in 2 sample z test)**

```
#Z test
ztest ,pval1 = stests.ztest(dataclean_16['Repurchase Price'], x2=dataclean_16['Net Asset Value'], value=0,alternative='two-sided')
print(float(pval1))
if pval1<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
#Z test
ztest ,pval1 = stests.ztest(dataclean_16['Sale Price'], x2=dataclean_16['Net Asset Value'], value=0,alternative='two-sided')
print(float(pval1))
if pval1<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
2 Sample T test p Value is nan
Accepting Null Hypothesis
2 Sample T test p Value is nan
Accepting Null Hypothesis
nan
accept null hypothesis
nan
accept null hypothesis
```

Hypothesis Testing

All possible test:
T test, Z Test, Chi-square, ANOVA Test
Test implemented:
T test And Z Test

SIX STEPS FOR HYPOTHESIS TESTING.

- HYPOTHESES.
- ASSUMPTIONS.
- TEST STATISTIC (or Confidence Interval Structure)
- REJECTION REGION (or Probability Statement)
- CALCULATIONS (Annotated Spreadsheet)
- CONCLUSIONS.

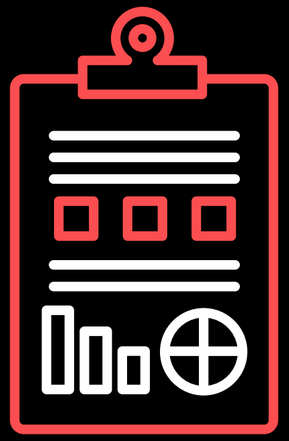
On the basis of visualization we analyzed that the value of dependent variables Repurchase Price and Sale Price linearly depend on Independent variable NAV.

Therefore the variations in Repurchase Purchase Price and Sale Price depends on NAV and its Variation.
Therefore hypothesizing based on this analysis

T-test is implement for sample space less than or equal to 30 , as a result for sample space greater than 30 we are getting same result form T and Z test.

Implementation of T-test is same as Z-Test.

Assumption:
Data points should be independent from each other. In other words, one data point isn't related or doesn't affect another data point.
Your data should be normally distributed. However, for large sample sizes (over 30) this doesn't always matter.
Your data should be randomly selected from a population, where each item has an equal chance of being selected. Assuming that they have equal variance.



Z test

Hypothesis :

Hypothesis 1:

Null hypothesis:

Mean of repurchase Price = mean of Net Asset Value

Alternate hypothesis

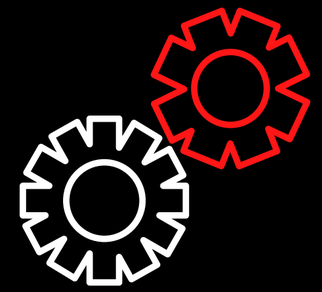
Mean of repurchase Price != mean of Net Asset Value

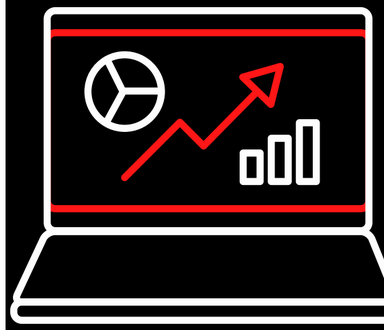
Test Statistic

The CVs for a significance level of 0.05 produce a confidence level of 1 - 0.05 = 0.95 or 95%.
We are using significance of 0.05 and considering for two tailed test that is for critical value :- 1.96 < Z < 1.96
For P value, the value greater than 0.05 lie within critical region while for less than 0.05 it lie outside the critical region

Rejection Region

Our rejection region will be the part of P value less than 0.05 , that is for P value less than 0.05 we will reject our null hypothesis.





Calculation:

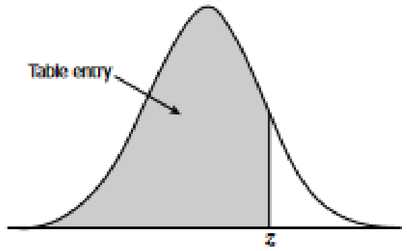


Table entry for z is the area under the standard normal curve to the left of z.

z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
0.0	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
0.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
0.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
0.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
0.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
0.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
0.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7517	.7549
0.7	.7580	.7611	.7642	.7673	.7704	.7734	.7764	.7794	.7823	.7852
0.8	.7881	.7910	.7939	.7967	.7995	.8023	.8051	.8078	.8106	.8133
0.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9279	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9429	.9441
1.6	.9452	.9463	.9474	.9484	.9495	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9649	.9656	.9664	.9671	.9678	.9686	.9693	.9699	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9761	.9767
2.0	.9772	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9864	.9868	.9871	.9875	.9878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9979	.9979	.9980	.9981
2.9	.9981	.9982	.9982	.9983	.9984	.9984	.9985	.9985	.9986	.9986
3.0	.9987	.9987	.9987	.9988	.9988	.9989	.9989	.9989	.9990	.9990

$$z = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

$$z = \frac{(23.1 - 19.2) - (0)}{\sqrt{\frac{3.5^2}{62} + \frac{4.8^2}{46}}} = 4.66$$

From Z table we can then determine the respective P value

Conclusion Of Hypothesis

For hypothesis 1

As we can see the p value is greater than 0.05 (for this dataset it is 1) there we can’t reject our null hypothesis and thus mean for Repurchase Price is equal to that of NAV, from the analysis we conclude that they have strong relation between them.

For hypothesis 2

As we can see the p value is greater than 0.05 (for this dataset it is 1) there we can’t reject our null hypothesis and thus mean for Sale Price is equal to that of NAV, from the analysis we conclude that they have strong relation between them.

For the complete set of 31 files we find that in the case of both hypotheses the null hypothesis can’t be rejected therefore we can conclude that the Repurchase Price and Sale Price both are related to NAV in the given dataset.

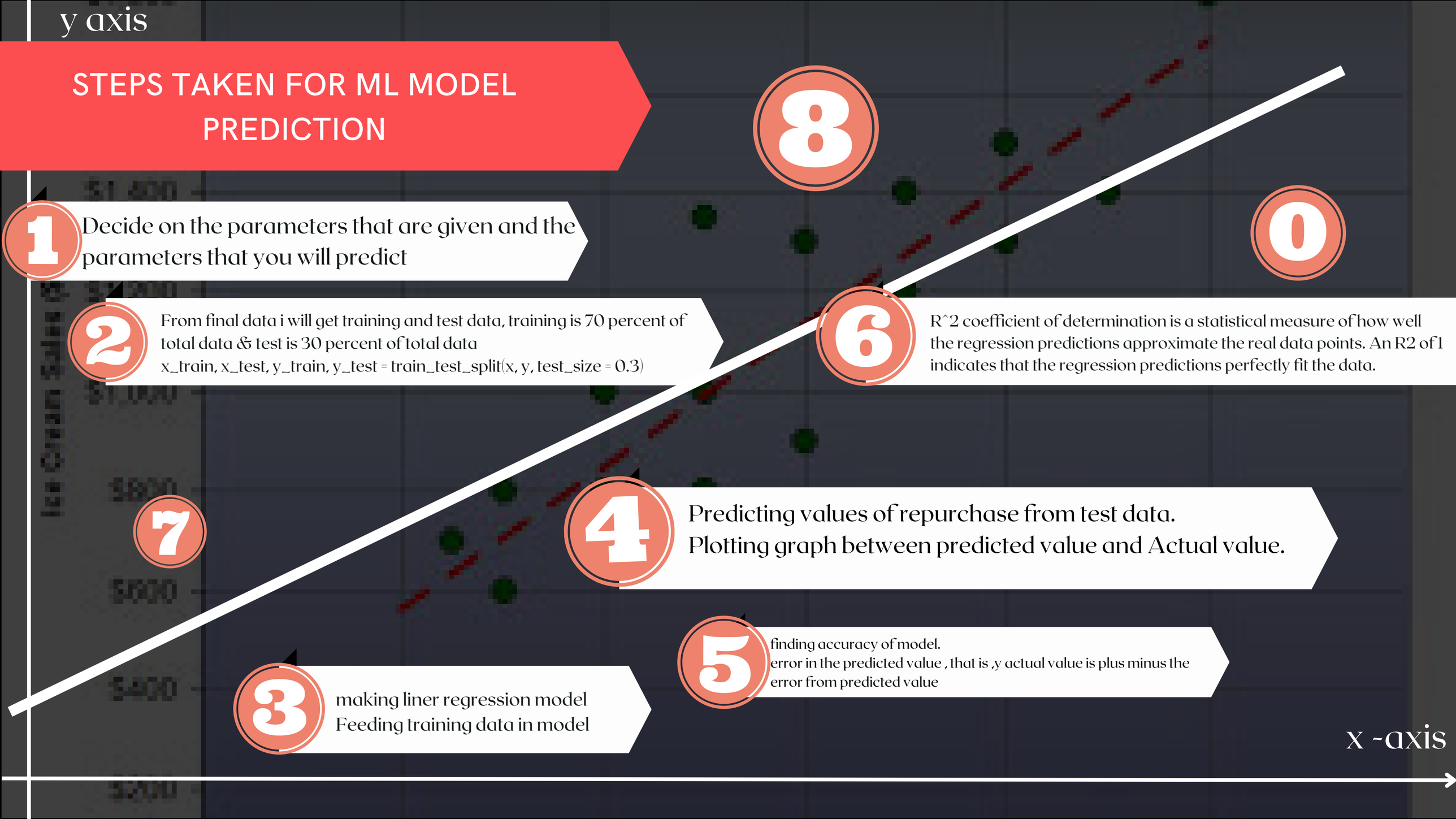
So on the basis of hypothesis testing, visualization and statistics I can say that the Repurchase and Sale prices both are strongly related to NAV and the magnitude(value) of Repurchase and Sale Price are equal to that of NAV.

ML MODEL FOR PREDICTION: LINEAR REGRESSION

Libraries and their attributes=>

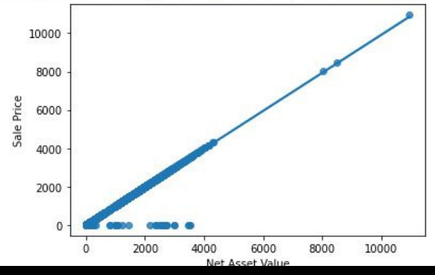
- Seaborn is a library that uses Matplotlib underneath to plot graphs. It will be used to visualize random distributions.
- seaborn.regplot():This method is used to plot data and a linear regression model fit.
- sklearn: sklearn is a dummy project on PyPi that will in turn install scikit-learn .
- scikit-learn: The sklearn library used for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

- train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.
- Linear regression:LinearRegression fits a linear model with coefficients w = (w1, ..., wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. Parameters fit_interceptbool, default=True. Whether to calculate the intercept for this model.
- model.fit(x_train,y_train):model. fit() : fit training data. For supervised learning applications, this accepts two arguments: the data X and the labels y (e.g. model. fit(X, y)). For unsupervised learning applications, this accepts only a single argument, the data X (e.g. model.


y axis
STEPS TAKEN FOR ML MODEL PREDICTION
8
1
Decide on the parameters that are given and the parameters that you will predict
0
2
From final data i will get training and test data, training is 70 percent of total data & test is 30 percent of total data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
6
R^2 coefficient of determination is a statistical measure of how well the regression predictions approximate the real data points. An R2 of 1 indicates that the regression predictions perfectly fit the data.
7
4
Predicting values of repurchase from test data.
Plotting graph between predicted value and Actual value.
5
finding accuracy of model.
error in the predicted value , that is ,y actual value is plus minus the error from predicted value
3
making liner regression model
Feeding training data in model
x -axis

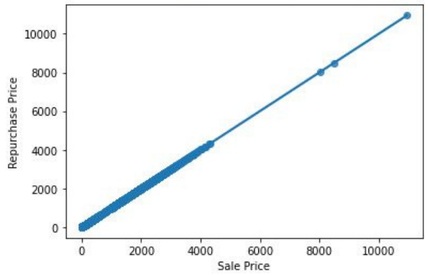
```
sns.regplot(x['Net Asset Value'], x['Sale Price'], ci=None)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, t
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f1ce76ea410>
```




```
sns.regplot(x['Sale Price'], y, ci=None)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f1ce76dd590>
```

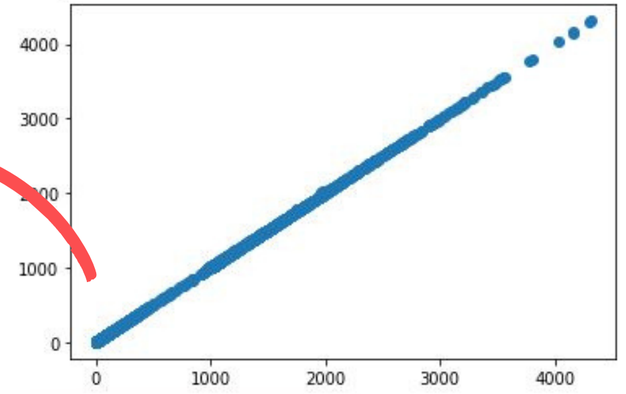



```
[ ] from sklearn import metrics
    from sklearn.metrics import accuracy_score

[ ] #finding accuracy of model.
    # higher R2 means I can predict unknown value more accurately , (max R2 score is 1.0)
    print("Mean absolute error =",(metrics.mean_absolute_error(y_test, predictions)))
    #error in th epredicted value , that is ,y actual value is plus minus the error from predicted value
    print("Mean squared error = ", metrics.mean_squared_error(y_test, predictions))
    print("R2 score =", (metrics.r2_score(y_test, predictions)))

    Mean absolute error = 0.2837041742154559
    Mean squared error =  1.3421269918873409
    R2 score = 0.9999925302577807
```

```
[ ] (y['Repurchase Price'] == 0).sum()

    13851

[ ] from sklearn.model_selection import train_test_split

[ ] #From final data i will get training and test data, training is 70 percent of total data & test is 30 percent of total data
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)

[ ] from sklearn.linear_model import LinearRegression
    #making liner regression model
    model = LinearRegression()
    #Fedding traiing data in model
    model.fit(x_train, y_train)

    LinearRegression()

[ ] #paramters of my model after going through test data
    print(model.coef_)
    print(model.intercept_)

    [[9.99303510e-01 1.54898238e-04]]
```

```
#Predicting vaues of repurchase from test data.
predictions = model.predict(x_test)
#Plotting graph between predicted value and Actual value.
plt.scatter(y_test, predictions)
```

```
<matplotlib.collections.PathCollection at 0x7f1ce77c8bd0>
```



THE WORK IN ACTION!

CONCLUSION: We can predict the repurchase price from the net asset value and sale price of mutual funds with the R^2 value of 0.99999253025.

SUMMARY OF OUR ENTIRE PROJECT!

- DATA COLLECTION
- DATA UNDERSTANDING
  (i) Sampling the data out of the complete data.
- DATA PREPROCESSING AND CLEANING
  (i) Dropping the unnecessary columns.
  (ii) Replacing NAN value with 0.
  (iii) Changing original datatype to appropriate datatypes.
- DATA VISUALIZATION
  (i) Line plots for showing relationship of net asset value, purchase values, sales price.
  (ii) Bar graph for all interested parameters.
  (iii) Boxplot for all interested parameters.

- DATA STATISTICS
  (i) Mean, median,
  (ii) Minimum, Maximum.
  (iii) Standard deviation.
- HYPOTHESIS TESTING
  (i) T-test.
  (ii) Z-test.
- ML MODEL FOR PREDICTION
  (i) Linear regression.
  (ii) R^2 coefficient method to check accuracy
  (iii) mean absolute and mean square error.

THANK YOU!