Milestone 1:

Preprocessing techniques:

1-first we check if any column contains NAN value

```
train.isnull().sum()
```

```
X1        0
X2     1006
X3        0
X4        0
X5        0
X6        0
X7        0
X8        0
X9     1711
X10       0
X11       0
Y         0
dtype: int64
```

So we now know that we have two column that contain NAN value

We need to replace empty cell too so we replace empty cell with NAN value to clean it in one step

```
train = train.replace(' ', np.nan)
```

Then we fill NAN value

```
train["X9"].fillna( method ='ffill', inplace = True)
train["X2"].fillna( method ='ffill', inplace = True)
train.isnull().sum()
```

```
X1     0
X2     0
X3     0
X4     0
X5     0
X6     0
X7     0
X8     0
X9     0
X10    0
X11    0
Y      0
dtype: int64
```

2-check for unique value of column

```
train["X3"].unique()
```

```
array(['Low Fat', 'Regular', 'low fat', 'LF', 'reg'], dtype=object)
```

When we see X3 column value we notice that there 2 unique value but it Written in different ways so we need to fix it and check for unique value again

```
train["X3"]=train["X3"].replace(to_replace=["LF","low fat"],value="Low Fat")
train["X3"].unique()
```

```
array(['Low Fat', 'Regular', 'reg'], dtype=object)
```

```
train["X3"]=train["X3"].replace(to_replace=["reg"],value="Regular")
train["X3"].unique()
```

```
array(['Low Fat', 'Regular'], dtype=object)
```

3-check for zero value

```
train.head(25)
```

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	Y
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	Medium	Tier 3	Grocery Store	732.3800
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052
5	FDP36	10.395	Regular	0.000000	Baking Goods	51.4008	OUT018	2009	Medium	Tier 3	Supermarket Type2	556.6088
6	FDO10	13.650	Regular	0.012741	Snack Foods	57.6588	OUT013	1987	High	Tier 3	Supermarket Type1	343.5528

Then we need to count number of zero in each column

```
In [105]: train[train["X2"]==0].count()
Out[105]: X2     0
          X3     0
          X4     0
          X5     0
          X6     0
          X8     0
          X9     0
          X10    0
          X11    0
          Y      0
          dtype: int64
```

```
In [106]: train[train["X4"]==0].count()
Out[105]: X2     368
          X3     368
          X4     368
          X5     368
          X6     368
          X8     368
          X9     368
          X10    368
          X11    368
          Y      368
          dtype: int64
```

The replace zero to mean of column contain zeros but first we check for distance between min and max to make sure that the mean is good choice

```
train.nsmallest(361, ['X4'])
```

	X2	X3	X4	X5	X6	X8	X9	X10	X11	Y
3	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	1998	Medium	Tier 3	Grocery Store	732.3800
4	8.930	Low Fat	0.000000	Household	53.8614	1987	High	Tier 3	Supermarket Type1	994.7052
5	10.395	Regular	0.000000	Baking Goods	51.4008	2009	Medium	Tier 3	Supermarket Type2	556.6088
10	11.800	Low Fat	0.000000	Fruits and Vegetables	45.5402	1999	Medium	Tier 1	Supermarket Type1	1516.0266
32	18.700	Low Fat	0.000000	Snack Foods	256.6672	2009	Medium	Tier 3	Supermarket Type2	3068.0064
...	...	...	...	...	...	...	...	...	...	...
5936	17.100	Low Fat	0.000000	Household	167.0842	2009	Medium	Tier 3	Supermarket Type2	1326.2736
5942	6.780	Regular	0.000000	Baking Goods	94.0120	1987	High	Tier 3	Supermarket Type1	1211.7560
5945	20.700	Low Fat	0.000000	Dairy	78.4670	1997	Small	Tier 1	Supermarket Type1	1607.9070
5953	15.300	Low Fat	0.000000	Household	103.5332	2004	Small	Tier 2	Supermarket Type1	3383.5956
3862	5.880	Low Fat	0.003589	Hard Drinks	155.5998	1987	High	Tier 3	Supermarket Type1	1691.7978

361 rows × 10 columns

```
train["X4"].max()#CHECK FOR OUTLIER
```

0.328390948

```
m=train["X4"].mean()
train["X4"]=train["X4"].replace(to_replace=0,value=m)
train[train["X4"]==0].count()
```

4-check for range between number and outlier

First check for min and max value in each column

```
print(train["X2"].min(),train["X2"].max())
```

4.555 21.35

```
print(train["X4"].min(),train["X4"].max())
```

0.003589104 0.328390948

```
print(train["X6"].min(),train["X6"].max())
```

31.29 266.8884

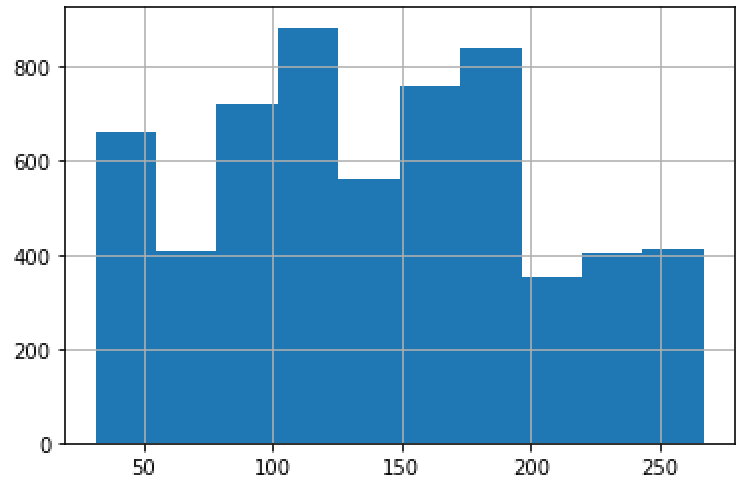
```
print(train["X8"].min(),train["X8"].max())
```

1985 2009

Second check for frequency of value in each column by draw histogram to make sure we don’t have outlier

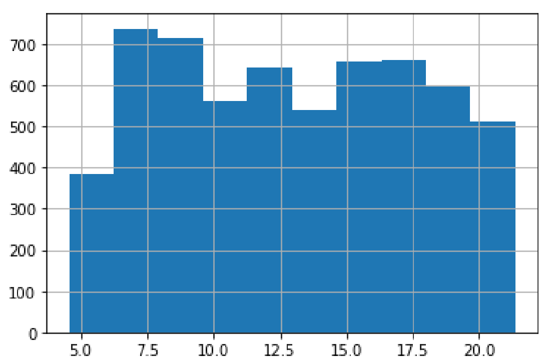
```
train["X6"].hist()
```

<AxesSubplot:>



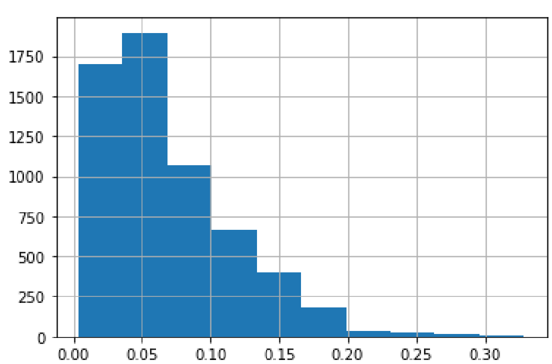
```
train["X2"].hist()
```

<AxesSubplot:>



```
train["X4"].hist()
```

<AxesSubplot:>



Then we use minmax scaler to solve it

```
In [121]: # i will use minmax scaler to do that
          from sklearn.preprocessing import MinMaxScaler
          scaler = MinMaxScaler()
          train["X6"] = scaler.fit_transform(train["X6"].values.reshape(-1,1))
```

```
In [122]: print(train["X6"].min(),train["X6"].max())

          0.0 1.0
```

```
In [123]: #now number is within range 0,1

          train["X2"] = scaler.fit_transform(train["X2"].values.reshape(-1,1))
```

```
In [124]: print(train["X2"].min(),train["X2"].max())

          0.0 1.0
```

5-now we need to change name of column to be more clean and easy to use and understand it

```
#now i want to change name of column to be clean and easy to deal with it
train.rename(columns={'X2': 'Weight of Item', 'X3': 'Amount of Fats in Item','X4': 'area allocated for item in store ','X5': 'Ite
train.head()
```

	Weight of Item	Amount of Fats in Item	area allocated for item in store	Item Category	Item Price	Store Establishment Year	Store Size	Store Location Type	Store Type	label
0	0.282525	Low Fat	0.016047	Dairy	0.927507	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	0.081274	Regular	0.019278	Soft Drinks	0.072068	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	0.770765	Low Fat	0.016760	Meat	0.468288	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	0.871986	Regular	0.066333	Fruits and Vegetables	0.640093	1998	Medium	Tier 3	Grocery Store	732.3800
4	0.260494	Low Fat	0.066333	Household	0.095805	1987	High	Tier 3	Supermarket Type1	994.7052

2-Analysis On Dataset

1-apply correlation matrix to show correlation between feature and label

```
train.corr()
```

	Weight of Item	area allocated for item in store	Item Price	Store Establishment Year	label
Weight of Item	1.000000	-0.015229	0.020948	-0.010053	0.007496
area allocated for item in store	-0.015229	1.000000	-0.010472	-0.073955	-0.138669
Item Price	0.020948	-0.010472	1.000000	0.011276	0.575582
Store Establishment Year	-0.010053	-0.073955	0.011276	1.000000	-0.052275
label	0.007496	-0.138669	0.575582	-0.052275	1.000000

Second drop column that have small relation with my label

```
#weight of item is no related to my label(not strong relation)
#store establisment year is no related to my label (no strong relation)
train.drop(["Weight of Item","Store Establishment Year"],inplace=True,axis=1)
train.head()
```

	Amount of Fats in Item	area allocated for item in store	Item Category	Item Price	Store Size	Store Location Type	Store Type	label
0	Low Fat	0.016047	Dairy	0.927507	Medium	Tier 1	Supermarket Type1	3735.1380
1	Regular	0.019278	Soft Drinks	0.072068	Medium	Tier 3	Supermarket Type2	443.4228
2	Low Fat	0.016760	Meat	0.468288	Medium	Tier 1	Supermarket Type1	2097.2700
3	Regular	0.066333	Fruits and Vegetables	0.640093	Medium	Tier 3	Grocery Store	732.3800
4	Low Fat	0.066333	Household	0.095805	High	Tier 3	Supermarket Type1	994.7052

```
train.drop(["X1","X7"],inplace=True,axis=1)
```

```
train.head()
```

	X2	X3	X4	X5	X6	X8	X9	X10	X11	Y
0	9.30	Low Fat	0.016047	Dairy	249.8092	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	5.92	Regular	0.019278	Soft Drinks	48.2692	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	17.50	Low Fat	0.016760	Meat	141.6180	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	1998	Medium	Tier 3	Grocery Store	732.3800
4	8.93	Low Fat	0.000000	Household	53.8614	1987	High	Tier 3	Supermarket Type1	994.7052

We drop X1, X7, weight of item, store establishment year columns

2-convert all object (string) to number

We use one hot encoder and label encoder

1-one hot encoder to column that have more 3 unique value (to avoid large number and gives priority)

2- Label encoder in column that has 2 or 3 unique value (to avoid large number of feature)

```
# now i want to encode string to number to train model
train["Amount of Fats in Item"]=train["Amount of Fats in Item"].replace(to_replace="Low Fat",value=1)
train["Amount of Fats in Item"]=train["Amount of Fats in Item"].replace(to_replace="Regular",value=0)
train["Amount of Fats in Item"]
```

```
0       1
1       0
2       1
3       0
4       1
       ..
5995    1
5996    1
5997    1
5998    1
5999    1
Name: Amount of Fats in Item, Length: 6000, dtype: int64
```

```
train["Item Category"].unique()
#16
```

```
array(['Dairy', 'Soft Drinks', 'Meat', 'Fruits and Vegetables',
       'Household', 'Baking Goods', 'Snack Foods', 'Frozen Foods',
       'Breakfast', 'Health and Hygiene', 'Hard Drinks', 'Canned',
       'Breads', 'Starchy Foods', 'Others', 'Seafood'], dtype=object)
```

```
from sklearn.preprocessing import OneHotEncoder
onehotencoder = OneHotEncoder()
x=onehotencoder.fit_transform(train["Item Category"].values.reshape(-1,1)).toarray()
dfo=pd.DataFrame(x,columns=["Category_"+str(int(i))for i in range(16)])
train=pd.concat([train,dfo],axis=1)
train.drop("Item Category",inplace=True,axis=1)
train.head()
```


3-Split dataset into train and test

Put all feature in x dataset and label in y dataset then make test 2000 record and train 4000 record

```
from sklearn.model_selection import train_test_split
x=train.iloc[ :,0:27]
y=train['label']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=2000)
```

4-Regression techniques

1-linear model ridge

This model solves a regression model where the loss function is the linear least squares function and regularization is given by the l2-norm. Also known as Ridge Regression or Tikhonov regularization

```
from sklearn.linear_model import Ridge
clf = Ridge(alpha=0.002)
clf.fit(x_train, y_train)
yu=clf.predict(x_test)
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, yu)
```

```
2.1573552795750723e-19
```

2-Neural Network MLPRegressor

MLPRegressor trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters.

It can also have a regularization term added to the loss function that shrinks model parameters to prevent overfitting

```
from sklearn.neural_network import MLPRegressor
regr = MLPRegressor(random_state=1, max_iter=500).fit(x_train, y_train)
```

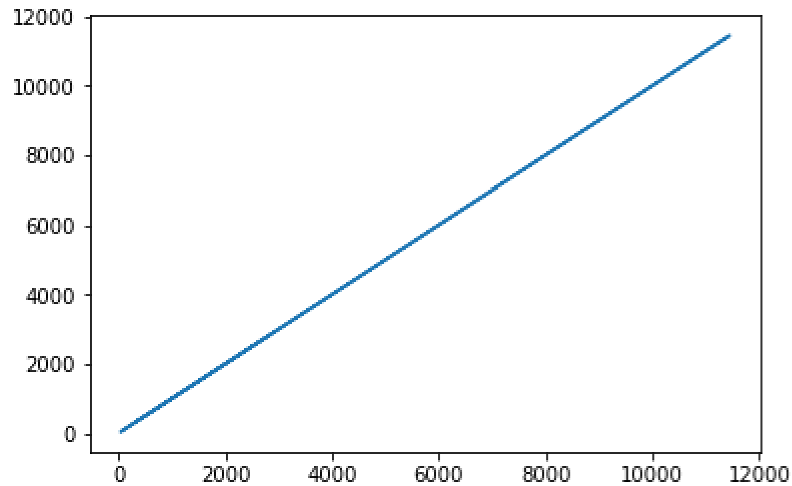
```
yu=regr.predict(x_test)
```

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, yu)
```

```
0.0026279390686992413
```

```
import matplotlib.pyplot as plt
plt.plot(y_test,yu)
```

```
[<matplotlib.lines.Line2D at 0x2879d7bff70>]
```



4-Conclusion

In this phase of project we learn more about how to clean dataset and prepare it to run in model and when we choose the model we must be careful to avoid overfiting or making large error try and try to choose best model first step you need to do in machine learning project is see your dataset and learn more about it and what column may be much related to our label