MileStone2

On

"Store Sales Prediction"

Submitted To

School Of Data Science And Forecasting DAVV, Indore(M.P)

Submitted By :

- Himanshu Jain
- Tarun Choudhary
- Tushar Sonp

Under The Guidance Of :

Prof. Vandit Hedau

Tech Event Organizer :

- Krati Vyas
- Hitesh Kumawat

Date: 05 March 2022

After Having EDA on Store Sales dataset we find that in the dataset we have some null value and some unwanted feature. So after EDA we start cleaning dataset by filling null values, droping unwanted feature and converting the labels into a numeric values

In Store Sales dataset **Item_Weight** and **Outlet_Size** columns we have some missing values.

Here, in **Outlet_Size** column we see that “Medium” is Median in that column so we fill null values with “Medium” and **Item_Weight** column fill with mean of that column.

```
[5]: df.isnull().sum()
```

```
[5]: Item_Identifier               0
     Item_Weight                1463
     Item_Fat_Content              0
     Item_Visibility               0
     Item_Type                     0
     Item_MRP                      0
     Outlet_Identifier             0
     Outlet_Establishment_Year     0
     Outlet_Size                2410
     Outlet_Location_Type          0
     Outlet_Type                   0
     Item_Outlet_Sales             0
     dtype: int64
```

```
[9]: print(df["Outlet_Size"].unique())
```

```
['Medium' nan 'High' 'Small']
```

```
11]: print(df["Outlet_Size"].value_counts())
```

```
Medium    2793
Small     2388
High       932
Name: Outlet_Size, dtype: int64
```

```
12]: df["Outlet_Size"] = df["Outlet_Size"].fillna('Medium')          #by observation we find that the medium has high occurance than
```

```
df["Item_Weight"] = df["Item_Weight"].fillna(df["Item_Weight"].mean())
```

```
13]: print(df.isnull().sum())
```

```
Item_Identifier            0
Item_Weight                0
Item_Fat_Content            0
Item_Visibility            0
Item_Type                  0
Item_MRP                   0
Outlet_Identifier          0
Outlet_Establishment_Year  0
Outlet_Size                0
Outlet_Location_Type       0
Outlet_Type                0
Item_Outlet_Sales          0
dtype: int64
```

In “**Item_Fat_Content**” column we found that it contain five unique value ['Lo w Fat', 'Regular', 'low fat', 'LF', 'reg'] in which “Low Fat”, “low fat”, “LF” are sa me and “Regular” , “reg” both are same. So we replace “low fat”, “LF” as “Low Fat” and “reg” as “Regular”.

```
In [89]: df["Item_Fat_Content"].unique()

Out[89]: array(['Low Fat', 'Regular', 'low fat', 'LF', 'reg'], dtype=object)

In [92]: df["Item_Fat_Content"] = df["Item_Fat_Content"].replace({"low fat":"Low Fat","LF": "Low Fat", "reg": "Regular"}, regex=True)

In [93]: df["Item_Fat_Content"].unique()

Out[93]: array(['Low Fat', 'Regular'], dtype=object)
```

After dealing with null values we make label encoding to convert feature label into numeric values.

```
In [16]: from sklearn.preprocessing import LabelEncoder

In [48]: encode = LabelEncoder()
         df["Item_Fat_Content"] = encode.fit_transform(df["Item_Fat_Content"])
         df["Item_Type"] = encode.fit_transform(df["Item_Type"])
         df["Outlet_Size"] =encode.fit_transform(df["Outlet_Size"])
         df["Outlet_Type"] = encode.fit_transform(df["Outlet_Type"])
         df["Outlet_Location_Type"] = encode.fit_transform(df["Outlet_Location_Type"])

In [50]: df

Out[50]:
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Locat
0	FDA15	9.300	1	0.016047	4	249.8092	OUT049	1999	1	
1	DRC01	5.920	2	0.019278	14	48.2692	OUT018	2009	1	
2	FDN15	17.500	1	0.016760	10	141.6180	OUT049	1999	1	
3	FDX07	19.200	2	0.000000	6	182.0950	OUT010	1998	1	
4	NCD19	8.930	1	0.000000	9	53.8614	OUT013	1987	0	
...	...	...	...	...	...	...	...	...	...	
8518	FDF22	6.865	1	0.056783	13	214.5218	OUT013	1987	0	
8519	FDS36	8.380	2	0.046982	0	108.1570	OUT045	2002	1	
8520	NCJ29	10.600	1	0.035186	8	85.1224	OUT035	2004	2	

Then we separate the independent and dependent variables as "**x**" and "**y**" which is require for model building. Also we drop unwanted columns in the dataset i.e. **"Item_Identifier"**, **"Outlet_Identifier"** which is nothing but some unique id's for item and outlet.

```
In [47]: x = df.drop(columns = ["Item_Identifier", "Item_Outlet_Sales", "Outlet_Identifier"])
```

```
In [52]: y = df.Item_Outlet_Sales
```

```
In [51]: x
```

Out[51]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
0	9.300	1	0.016047	4	249.8092	1999	1	0	1
1	5.920	2	0.019278	14	48.2692	2009	1	2	2
2	17.500	1	0.016760	10	141.6180	1999	1	0	1
3	19.200	2	0.000000	6	182.0950	1998	1	2	0
4	8.930	1	0.000000	9	53.8614	1987	0	2	1
...	...	...	...	...	...	...	...	...	...
8518	6.865	1	0.056783	13	214.5218	1987	0	2	1
8519	8.380	2	0.046982	0	108.1570	2002	1	1	1
8520	10.600	1	0.035186	8	85.1224	2004	2	1	1
8521	7.210	2	0.145221	13	103.1332	2009	1	2	2
8522	14.800	1	0.044878	14	75.4670	1997	2	0	1

8523 rows x 9 columns

```
In [53]: y
```

```
Out[53]: 0       3735.1380
         1        443.4228
         2       2097.2700
         3        732.3800
         4        994.7052
                   ...
         8518    2778.3834
         8519     549.2850
         8520    1193.1136
         8521    1845.5976
         8522     765.6700
         Name: Item_Outlet_Sales, Length: 8523, dtype: float64
```

To removes the mean and scales each feature/variable to unit variance we apply standardscaler in x dataset.

```
In [20]: #to make mean 0 and standard deviation 1
         from sklearn.preprocessing import StandardScaler

In [21]: scaler = StandardScaler()

In [22]: x_scaler = scaler.fit_transform(x)

In [23]: x_scaler = pd.DataFrame(x_scaler, columns=x.columns)

In [24]: x_scaler
```

Out[24]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
0	-0.841872	-0.572844	-0.970732	-0.766479	1.747454	0.139541	-0.284581	-1.369334	-0.252658
1	-1.641706	0.978092	-0.908111	1.608963	-1.489023	1.334103	-0.284581	1.091569	1.002972
2	1.098554	-0.572844	-0.956917	0.658786	0.010040	0.139541	-0.284581	-1.369334	-0.252658
3	1.500838	0.978092	-1.281758	-0.291391	0.660050	0.020085	-0.284581	1.091569	-1.508289
4	-0.929428	-0.572844	-1.281758	0.421242	-1.399220	-1.293934	-1.950437	1.091569	-0.252658
...	...	...	...	...	...	...	...	...	...
8518	-1.418084	-0.572844	-0.181193	1.371418	1.180783	-1.293934	-1.950437	1.091569	-0.252658
8519	-1.059578	0.978092	-0.371154	-1.716656	-0.527301	0.497909	-0.284581	-0.138882	-0.252658

```
In [27]: x_scaler.describe()
```

Out[27]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_
count	8.523000e+03	8.523000e+03	8.523000e+03	8.523000e+03	8.523000e+03	8.523000e+03	8.523000e+03	8.523000e+03	8.523000
mean	3.127265e-16	1.549988e-16	-8.548444e-17	1.025422e-16	-1.644427e-16	1.135381e-14	3.088251e-16	-6.574323e-16	-4.064826
std	1.000059e+00	1.000059e+00	1.000059e+00	1.000059e+00	1.000059e+00	1.000059e+00	1.000059e+00	1.000059e+00	1.000059
min	-1.964716e+00	-2.123779e+00	-1.281758e+00	-1.716656e+00	-1.761688e+00	-1.532846e+00	-1.950437e+00	-1.369334e+00	-1.508289
25%	-8.395053e-01	-5.728436e-01	-7.586531e-01	-7.664793e-01	-7.574307e-01	-1.293934e+00	-2.845812e-01	-1.369334e+00	-2.526583
50%	4.035383e-14	-5.728436e-01	-2.364792e-01	-2.913909e-01	3.243893e-02	1.395408e-01	-2.845812e-01	-1.388824e-01	-2.526583
75%	7.435985e-01	9.780922e-01	5.514755e-01	6.587859e-01	7.170372e-01	7.368218e-01	1.381274e+00	1.091569e+00	-2.526583
max	2.009608e+00	4.079964e+00	5.083050e+00	1.846507e+00	2.021724e+00	1.334103e+00	1.381274e+00	1.091569e+00	2.258603