Sales Predictions

Project 1

Reasoning process to explore our project

1. Problem statement – BigMart collected 2013 sales data for 1559 products across 10 stores in different cities. The aim is to build a predictive model to forecast sales and find out what features are important to increase sales.

2. Data Exploration – looking at categorical and continuous feature summaries and making observations about the data.

3. Data Cleaning – imputing missing values in the data and checking for outliers

4. Feature Engineering – modifying existing variables and creating new ones for analysis

5. Model Building – making predictive models on the data

Problem Statement

- BigMart collected 2013 sales data for 1559 products across 10 stores in different cities.
- Build a predictive model to forecast sales, this is a regression problem.
- Find out what features are important to sales to understand how sales can be driven up.

```
#check info on df_SalesPredictions
df_SalesPredictions.info()
```

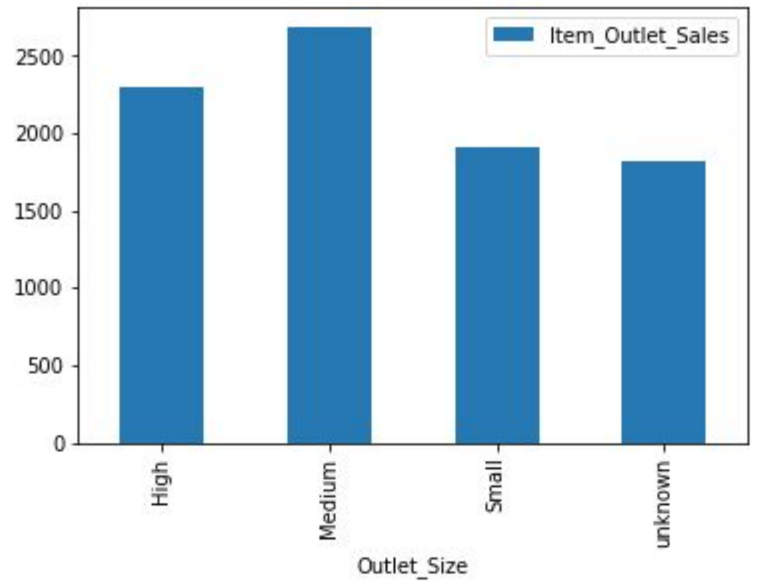
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            8523 non-null   object
 1   Item_Weight                7060 non-null   float64
 2   Item_Fat_Content           8523 non-null   object
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object
 7   Outlet_Establishment_Year  8523 non-null   int64
 8   Outlet_Size                6113 non-null   object
 9   Outlet_Location_Type       8523 non-null   object
 10  Outlet_Type                8523 non-null   object
 11  Item_Outlet_Sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
```

- ➔ Our dataset has 12 columns, 8523 entries
- ➔ Our dataset has categorical variables and numerical variables which define the matrix features
- ➔ Our target vector is Item_Outlet_Sales

Data Exploration

- looking at categorical and continuous feature summaries; making observations about the data.

Categorical variables



➔ Outlet_Size Medium drives the most highest sales amongst different outlet sizes considered.

Data Exploration

- looking at categorical and continuous feature summaries; observations and inferences about the data.

Numerical variables

```
#stats on the dataframe
#DESCRIBE - statistical summary of each column, such as count, column mean value, column standard deviation
#Note: item visibility needs to be multiplied by 100; the best column to look at stats summary is Item_Oulet_Sales
df_SalesPredictions.describe()
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.226124	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	9.310000	0.026989	93.826500	1987.000000	834.247400
50%	12.857645	0.053931	143.012800	1999.000000	1794.331000
75%	16.000000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

- ➔ Item_Visibility has a min value of zero. The visibility cannot be 0.
- ➔ Item_Visibility can be changed into %
- ➔ Outlet_Establishment_Years vary from 1985 to 2009. Values can be converted to how old a particular store is.

Data Cleaning

- imputing mis-coded and missing values in the data and checking for outliers

❖ Mis-coded variables

```
#we need only two categories Low Fat or Regular
#replacing values of cells in particular column
df_SalesPredictions['Item_Fat_Content'].replace(['low fat','LF','reg'],['Low Fat','Low Fat','Regular'],inplace = True)
```

```
Item_Fat_ContentIrr = df_SalesPredictions['Item_Fat_Content'].value_counts()
Item_Fat_ContentIrr
print(Item_Fat_ContentIrr)
```

```
Low Fat     5517
Regular     3006
Name: Item_Fat_Content, dtype: int64
```

➔ Item_Fat_Content had mis-coded values to be changed

Data Cleaning

- imputing mis-coded and missing values in the data and checking for outliers

❖ Missing values

```
# continuous variable Item_Weight
# filling missing values
# with mean column values
# create a variable mean and assign it the mean of the column Item_Weight
mean = df_SalesPredictions['Item_Weight'].mean()
df_SalesPredictions['Item_Weight'].fillna(value = mean, inplace=True)
```

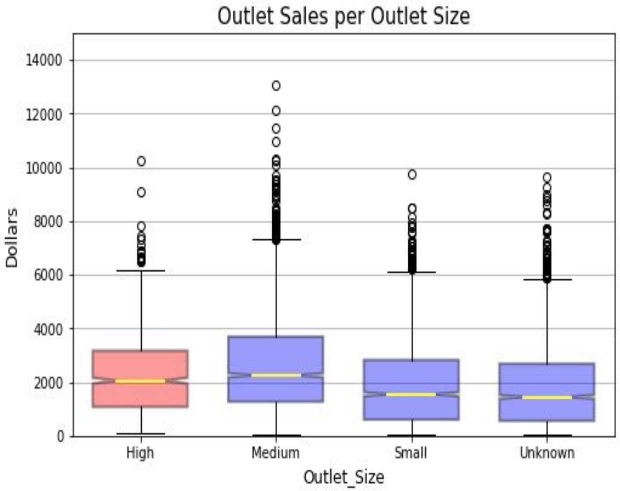
```
#categorical variable Outlet_Size
#Treat missing data as just another category : Unknown
df_SalesPredictions['Outlet_Size'].fillna('unknown', inplace=True)
```

➔ Item_Weight missing values changed into the mean value for this feature
➔ Outlet_Size missing values changed into unknown category

Data Cleaning

- imputing mis-coded and missing values in the data and checking for outliers

❖ Some outliers



The boxplot visualization confirms the barcharts results based on median and mean; Medium size outlets do the most sales.

➔ Item_Sales has some outliers in the higher end of all its different distributions grouped by Outelt_Size

➔ Some outliers may be reasonable while outliers in values exceeding 12000 USD needs to be checked.

➔ Medium Outlet_Size is confirmed to drive the most highest Outlet_Sales.

Feature Engineering

- modifying existing variables and creating new ones for analysis

❖ One Hot Encode

```
# Method 1: Pandas get_dummies
df_SalesPredictions = pd.get_dummies(df_SalesPredictions, columns = ['Item_Fat_Content','Item_Type','Outlet_Identifier','Outlet_Size',
'Outlet_Location_Type', 'Outlet_Type'], drop_first = False)
```

➔ One-Hot-Coding refers to creating dummy variables, to replace categorical variables. Each new variables will have binary numbers – 0 (if the category is not present) and 1(if category is present).
➔ *Item_Fat_Content mis-coded values changed*
➔ *Item_Weight missing values changed into the mean value for this feature*
➔ *Outlet_Size missing values changed into unknown category*

Model Building

- making predictive models on the data

RMSE comparison

```
#RMSE
from sklearn.metrics import mean_squared_error
#RMSE lnreg
print('lnreg Training RMSE:', np.sqrt(mean_squared_error(y_train, lnreg.predict(X_train))))
print('lnreg Testing RMSE:' , np.sqrt(mean_squared_error(y_test, lnreg.predict(X_test))))
#RMSE dec_tree
print('dec_tree_5 Training RMSE:', np.sqrt(mean_squared_error(y_train, dec_tree_5.predict(X_train))))
print('dec_tree_5 Testing RMSE:' , np.sqrt(mean_squared_error(y_test, dec_tree_5.predict(X_test))))
#RMSE bagreg
print('bagreg Training RMSE:', np.sqrt(mean_squared_error(y_train, bagreg.predict(X_train))))
print('bagreg Testing RMSE:' , np.sqrt(mean_squared_error(y_test, bagreg.predict(X_test))))
#RMSE rf
print('rf_5  Training RMSE:', np.sqrt(mean_squared_error(y_train, rf_5.predict(X_train))))
print('rf_5  Testing RMSE:' , np.sqrt(mean_squared_error(y_test, rf_5.predict(X_test))))
```

```
lnreg Training RMSE: 1139.1040937388918
lnreg Testing RMSE: 1092.8630817241494
dec_tree_5 Training RMSE: 1082.6461900869947
dec_tree_5 Testing RMSE: 1057.4431299496734
bagreg Training RMSE: 487.3900381277651
bagreg Testing RMSE: 1137.5430389051805
rf_5  Training RMSE: 1073.5872542554382
rf_5  Testing RMSE: 1047.1044311431237
```

The best low variance (represents error in training data set) and low bias (represents error in testing data set) is the Random Forrest Trees where for a rf depth of 5 training RMSE is 1073.58 and testing RMSE is 1047.10.

Feature Importance

- most important features to consider to improve target vector forcast

```
#all variables' correlation to each other
corr = df_SalesPredictions.corr()
corr
print(corr['Item_Outlet_Sales'].sort_values(ascending=False))
```

```
Item_Outlet_Sales                  1.000000
Item_MRP                           0.567574
Outlet_Type_Supermarket Type3      0.311192
Outlet_Identifier_OUT027           0.311192
Outlet_Size_Medium                 0.204701
Outlet_Type_Supermarket Type1      0.108765
```

➔ Outlet_Sales is influenced by Item_MRP with the most influence which is expected.

➔ Looking into store OUT27 across cities can help us understand how to replicate this store success across the other 9 stores.

➔ Store OUT27 may be a Supermarket Type 3 of a medium size which are other features correlated with Outlet_Sales

Recommendations

Better Hypothesis Generation

- Understanding the problem and making some hypothesis about what could potentially have a good impact on the outcome is a critical step and this should be done before looking at the data (cf here)

Better Data Cleaning and Engineering

- Per observations: age of stores, change visibility into %, create broad category of Type of item

Regression score can be improved

- Looking again in the dataset when it comes to building a regression method to predict our target vector

❖ Our target vector is influenced by features that we can address such as visibility of item by making items more accessible; maximum price retail of item by adding a more expensive range of items in stores; Type/Size of store needs further analysis to understand demographics around these features and OUT27 seems to be the best performing store across cities. Studying it might give us more informations into understanding how to improve sales in all stores.