SUMMER TRAINING REPORT

ON

FUNDAMENTALS OF PYTHON & MACHINE LEARNING

Under the able guidance of:

Himanshu Shankar
Civil Machines Tech Pvt. Ltd.

Divyadeep Bhatnagar
GGS Indraprastha University

Prepared By:

SAURAV VERMA
CSE & BDA
2010767

Submitted To

GraphicEra
(Deemed to be University)
Accredited by NAAC with Grade A

Computer Science and Information Technology Department

CONTENTS

CANDIDATE'S DECLARATION

I hereby declare that the Summer Training Report presented hereby is an authentic record of my own work as allotted during my Training on "Fundamentals of Python" in June-July 2018 and "Machine Learning" in August-September 2018 for the award of certificate for Summer Training at Acadview Software Pvt. Ltd.

I have taken care in all respect to honor the intellectual property right and have acknowledged the contribution of others for using them in academic purpose. Our supervisors should not be held responsible for full or partial violation of copyright or intellectual property right.

(Signature of student)

Date: 23 October 2018

ABOUT THE INSTITUTION

AcadView Software Pvt. Ltd. is an ed-tech Organization founded by Mr. Himanshu Batra (Ex. Google). AcadView offers engineering students courses in front-end, back-end, and full-stack development while allowing engineering professionals to teach online. AcadView connects the freshers with expert mentors from organizations across India who teach the various front-end, back-end, and full-stack development courses offered by the company. Once students finish the projects, automated résumés are generated**.**

The task of onboarding mentors was preceded by a campaign which revealed that working professionals want to earn extra money. AcadView allows them to do that by taking classes after office hours.

The AcadView team conducts sales pitches in front of students where they get to know their level of interest in technologies. They then aggregate and customize the content.

PART 1

PYTHON FUNDAMENTALS

INTRODUCTION

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

At Acadview I was introduced to the given topics of Python followed by the final project submission:

- Data types in Python
- Decision Control System
- Loops
- Functions
- Classes & Modules
- Web APIs
- File Handling
- Regular Expression
- GUI
- Numpy & Pandas library, etc.

PROJECT REPORT

OBJECTIVE:

To build a text editor in Python that is easily customizable and performs basic read and write operations like save, open, change font, redo, undo, cut, copy, paste, etc.

TECHNOLOGIES UTILIZED:

- Tkinter library for GUI

KEY CONCEPTS APPLIED:

- Data Types
- Operators
- Looping
- Functions
- Modules
- Files & I/O

PREREQUISITE:

- Tkinter Library:

  Components of Text editor GUI The application is built using python library Tkinter. Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

WORKFLOW:

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.
- File handling
- To open and save files , the project is using Python file handling system and its basic method
- GUI
- The user interface is developed using Tkinter components.
- Grid
- Toplevel
- Messagebox
- Mainloop

CODE:

Definition of some functions and application of file handling:

```
from tkinter import *
from tkinter import filedialog,scrolledtext,END,messagebox,simpledialog
import os

#-------------------------------------WINDOW-CREATION-------------------------------------------------------------
root = Tk()
root.title("Silk v1.0 Text Editor")

root.iconbitmap('C:/Users/Intel/Desktop/TE images/silk.ico')
root.geometry('800x500')

#-------------------------------------FUNCTIONS-------------------------------------------------------------------
def new_():
    if len(content_text.get('1.0',END+'-1c')) > 0:
        if messagebox.askyesno("Closing file","Do you want to save?"):
            save_()
    else:
        content_text.delete('1.0',END)

def open_():
    file = filedialog.askopenfile(initialdir="/", title='Select a text file',filetypes=[("Text file","*.txt"),("All files","*.*")])
    content_text.delete('1.0',END)
    root.title(os.path.basename(file.name) + " - Silk v1.0 Text Editor")
    if file != None:
        content = file.read()
        content_text.insert('1.0',content)
        file.close()

def save_():
    file = filedialog.asksaveasfile(mode="w",defaultextension=".txt",filetypes=[("HTML file","*.html"),("Text file","*.txt"),("All files","*.*'
    if file != None:
        saved = content_text.get('1.0',END+'-1c')
        file.write(saved)
        file.close()

def about_():
    lbl = messagebox.showinfo("About","Silk Text Editor by Saurav Verma. All Rights Reserved. ")
    lbl.pack()

```

Some Menu definitions and styling:

```
128
129 #-------------------------------MENU-OPTIONS-----------------------------------------------------------------
130 menu = Menu(root)
131 root.config(menu=menu)
132 fileMenu = Menu(menu)
133 menu.add_cascade(label="File",menu=fileMenu)
134 fileMenu.add_command(label="New",accelerator='New',command=new_,image=new_file_icon)
135 fileMenu.add_command(label="Open",accelerator='Open',command=open_,image=open_file_icon)
136 fileMenu.add_command(label="Save",accelerator='Save',command=save_,image=save_file_icon)
137 fileMenu.add_separator()
138 fileMenu.add_command(label="Exit",accelerator='Exit',command=exit_,image=exit_icon)
139
140 #-------------------------------EDIT-MENU--------------------------------------------------------------------
141 editMenu = Menu(menu)
142 menu.add_cascade(label="Edit",menu=editMenu)
143 editMenu.add_command(label = "Find Text",accelerator='Find Text',underline=0,command=find_,image=find_icon)
144 editMenu.add_command(label = "Undo",accelerator='Undo',underline=0,command=undo_,image=undo_icon)
145 editMenu.add_command(label = "Redo",accelerator='Redo',underline=0,command=redo_,image=redo_icon)
146
147 show_line_number=IntVar()
148 show_line_number.set(1)
149 show_cursor_info=IntVar()
150 show_cursor_info.set(1)
151
152 #-------------------------------FONTS-MENU-------------------------------------------------------------------
153 fontMenu = Menu(menu)
154 menu.add_cascade(label="Fonts",menu=fontMenu)
155 helvetica=IntVar()
156 courier=IntVar()
157 fontMenu.add_checkbutton(label="Courier",command=FontCourier)
158 fontMenu.add_checkbutton(label="Helvetica",command=FontHelvetica)
159 fontMenu.add_checkbutton(label="Times New Roman", command=FontTimes)
160 fontMenu.add_checkbutton(label="Arial", command=FontArial)
161 fontMenu.add_checkbutton(label="Chiller", command=FontChiller)
162
163 #-------------------------------THEME-MENU-------------------------------------------------------------------
164 themeMenu = Menu(menu)
```

KEY LEARNINGS:

- GUIs can be built with Tkinter windows and widgets. Tkinter arranges label, entry, and button widgets in a window using a grid layout. The button widgets can be linked to functions and the data in entry widgets can be extracted for use elsewhere.
- Various Python files can interact with each other as modules. This allows for the principle of abstraction, where code can be used without knowing precisely how it was implemented. For this project, the frontend and backend were developed independently and later connected.

LINK TO FULL PROJECT:

https://github.com/vermageu/Silk-Text-Editor

PART 2

MACHINE LEARNING

INTRODUCTION

Machine learning is an application of **artificial intelligence** (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning** focuses on the development of computer programs that can access data and use it learn for themselves.

At Acadview I was introduced to the given topics of Machine Learning followed by the final project submission:

- Linear Algebra
- Probability
- Visualization
- Exploratory data analysis
- Feature engineering
- Regression Techniques
- KNN Algorithm
- Naïve-Bayes Algorithm
- Support Vector Machine Algorithm
- Trees
- Ensemble Models
- KMeans Clustering, etc.

PROJECT REPORT

OBJECTIVE:

The data scientists at BigMart have collected 2013 sales data for 1559 products across 10 stores in different cities. Also, certain attributes of each product and store have been defined. The aim is to build a predictive model and find out the sales of each product at a particular store.
Using this model, BigMart will try to understand the properties of products and stores which play a key role in increasing sales.

DATASET:

The dataset is in the form of a csv file and the link to download is given below:

Link:
https://drive.google.com/open?id=1sukL3ljrJIhgW3NbsnKoayGd7Pfy-gqo

DATASET DESCRIPTION:

The dataset has 8524 entries with 13 columns.The description of each columns is given below:

Variable	Description
Item_Identifier	Unique product ID
Item_Weight	Weight of product
Item_Fat_Content	Whether the product is low fat or not
Item_Visibility	The % of total display area of all products in a store allocated to the particular product
Item_Type	The category to which the product belongs
Item_MRP	Maximum Retail Price (list price) of the product
Outlet_Identifier	Unique store ID
Outlet_Establishment_Year	The year in which store was established
Outlet_Size	The size of the store in terms of ground area covered
Outlet_Location_Type	The type of city in which the store is located
Outlet_Type	Whether the outlet is just a grocery store or some sort of supermarket
Item_Outlet_Sales	Sales of the product in the particular store. This is the outcome variable to be predicted.

PREREQUISITE(S):

- Data Visualization:

  Data visualization is viewed by many disciplines as a modern equivalent of visual communication. It involves the creation and study of the visual representation of data. To communicate information clearly and efficiently, data visualization uses statistical graphics, plots, information graphics and other tools.

- Feature Engineering:

  Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive

- Linear Regression:

  In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables. The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression

- Gradient Boosting:

  Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

- Decision Tree Regression:

  Decision tree builds regression or classification models in the form of a tree structure. It brakes downs a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

- Random Forest Regression:

  A random forest is a meta-estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

WORKFLOW:

- The workflow for the project is described in steps given below:
- Perform data cleaning using pandas library. Which includes replacing the miscoded information and handling missing data.
- Make an Exploratory Data Analysis on the data using pandas.
- Visualize distributions and correlation of features using seaborn and pandas
- Build a linear regression model taking the selected features through feature engineering
- Predict the item_outlet_sales for the test data

CODE SNIPPETS:

Understanding the structure of dataset.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('C:/Users/Intel/Desktop/Big_mart.csv')
```

In [2]:
```
#Print first five rows and correspondingly all columns form the dataset
print(df.head())
```

```
  Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility  \
0           FDA15         9.30          Low Fat         0.016047
1           DRC01         5.92          Regular         0.019278
2           FDN15        17.50          Low Fat         0.016760
3           FDX07        19.20          Regular         0.000000
4           NCD19         8.93          Low Fat         0.000000

              Item_Type  Item_MRP Outlet_Identifier  \
0                 Dairy  249.8092            OUT049
1           Soft Drinks   48.2692            OUT018
2                  Meat  141.6180            OUT049
3  Fruits and Vegetables  182.0950            OUT010
4             Household   53.8614            OUT013

   Outlet_Establishment_Year Outlet_Size Outlet_Location_Type  \
0                       1999      Medium               Tier 1
1                       2009      Medium               Tier 3
2                       1999      Medium               Tier 1
3                       1998         NaN               Tier 3
4                       1987        High               Tier 3

        Outlet_Type  Item_Outlet_Sales
0  Supermarket Type1          3735.1380
1  Supermarket Type2           443.4228
2  Supermarket Type1          2097.2700
3      Grocery Store           732.3800
4  Supermarket Type1           994.7052
```

In [3]:
```
#Print shape of the dataset
print(df.shape)
```

```
(8523, 12)
```

Finding the amount of missing data.

```
In [6]: print(df.isnull().sum())
```

```
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
```

```
In [7]: #Since, there are many null values in Item_Weight and Outlet_size so we have to manage these null v
        alues

        list(df.columns)
```

```
Out[7]: ['Item_Identifier',
         'Item_Weight',
         'Item_Fat_Content',
         'Item_Visibility',
         'Item_Type',
         'Item_MRP',
         'Outlet_Identifier',
         'Outlet_Establishment_Year',
         'Outlet_Size',
         'Outlet_Location_Type',
         'Outlet_Type',
         'Item_Outlet_Sales']
```

Eliminating the missing values with mean of their column values

```
In [10]: #Filling the column Outlet_size with missing

         outletSizeMode = {}
         for i in set(df['Outlet_Type']):
           outletSizeMode[i] = df[df['Outlet_Type'] == i]['Outlet_Size'].mode()[0]
         print(outletSizeMode)
         miss_bool = df['Outlet_Size'].isnull()
         print ('\nOrignal missing value in Outlet_Size: %d'% sum(miss_bool))
         df.loc[miss_bool,'Outlet_Size'] = df.loc[miss_bool,'Outlet_Type'].apply(lambda x: outletSizeMode[x
         ])
         print ("New missing value count in Outlet_Size:",sum(df['Outlet_Size'].isnull()))
```

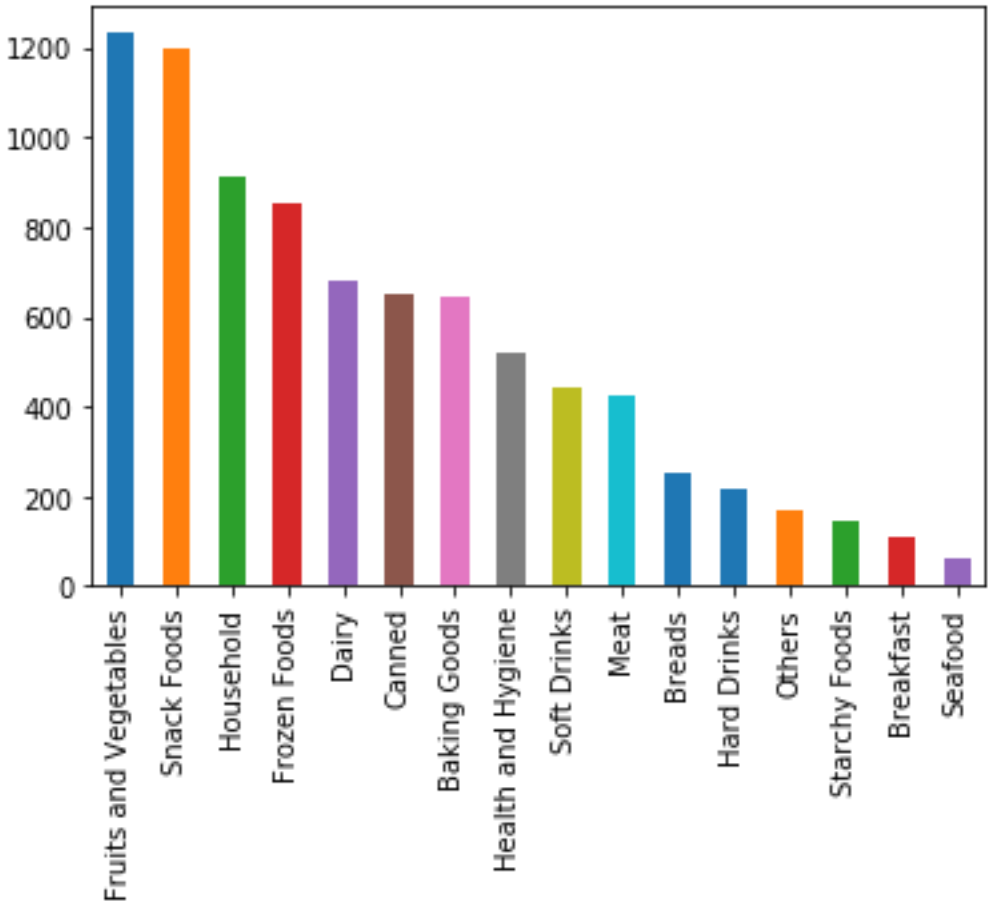
```
{'Supermarket Type2': 'Medium', 'Supermarket Type1': 'Small', 'Supermarket Type3': 'Medium', 'Groce
ry Store': 'Small'}

Orignal missing value in Outlet_Size: 2410
New missing value count in Outlet_Size: 0
```

```
In [11]: 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
```

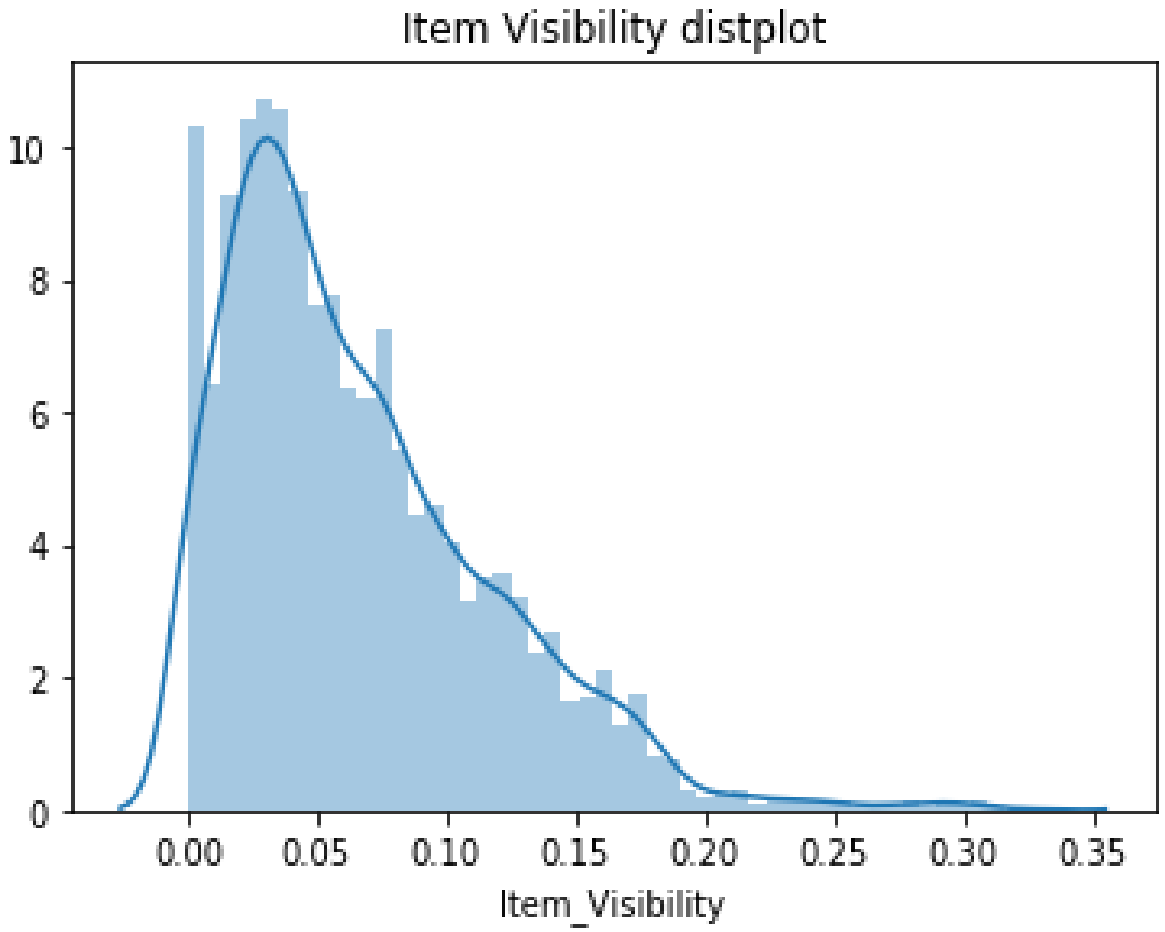
Now we have fulfilled all the null values. For better visualization I will take both numerical and categorical approach.

UNIVARIATE ANALYSIS:



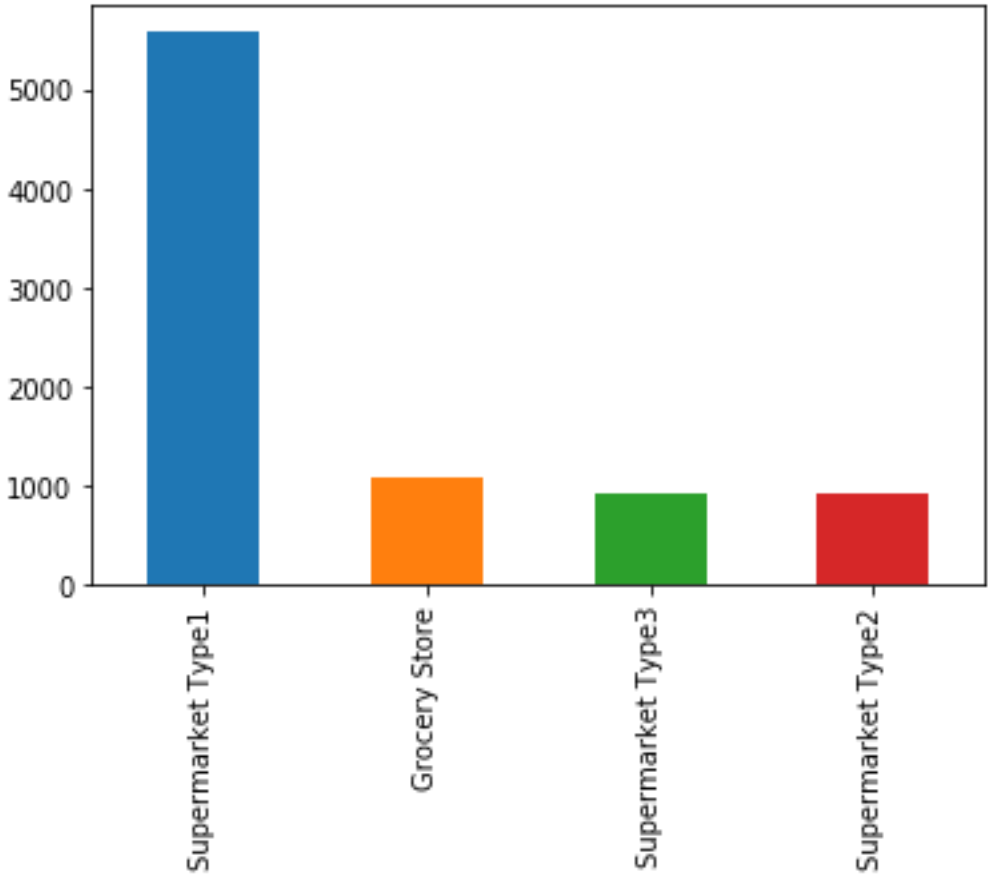
Observations:

- Major items are Fruits and vegetable, snack, household, dairy,canned and baking goods
- There are very less breakfast, starchy and seafood items Displot of Item_Visibility



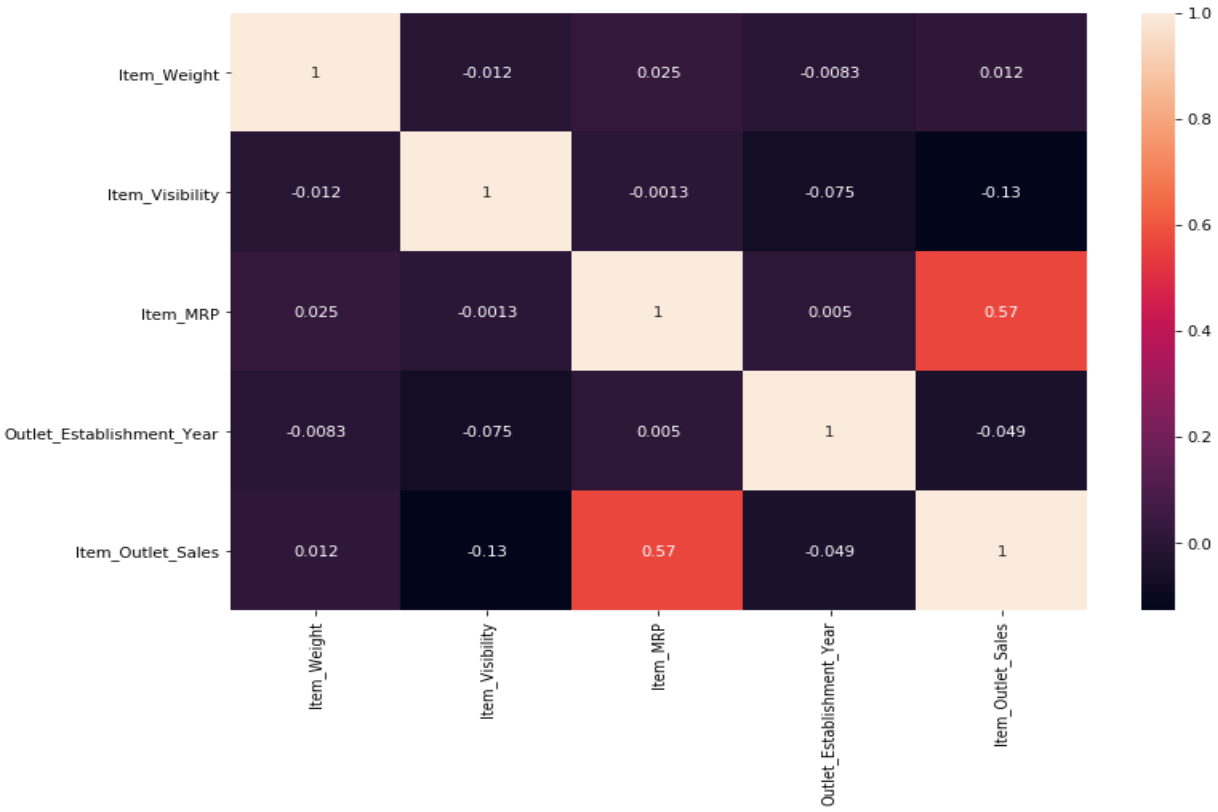
Observations:

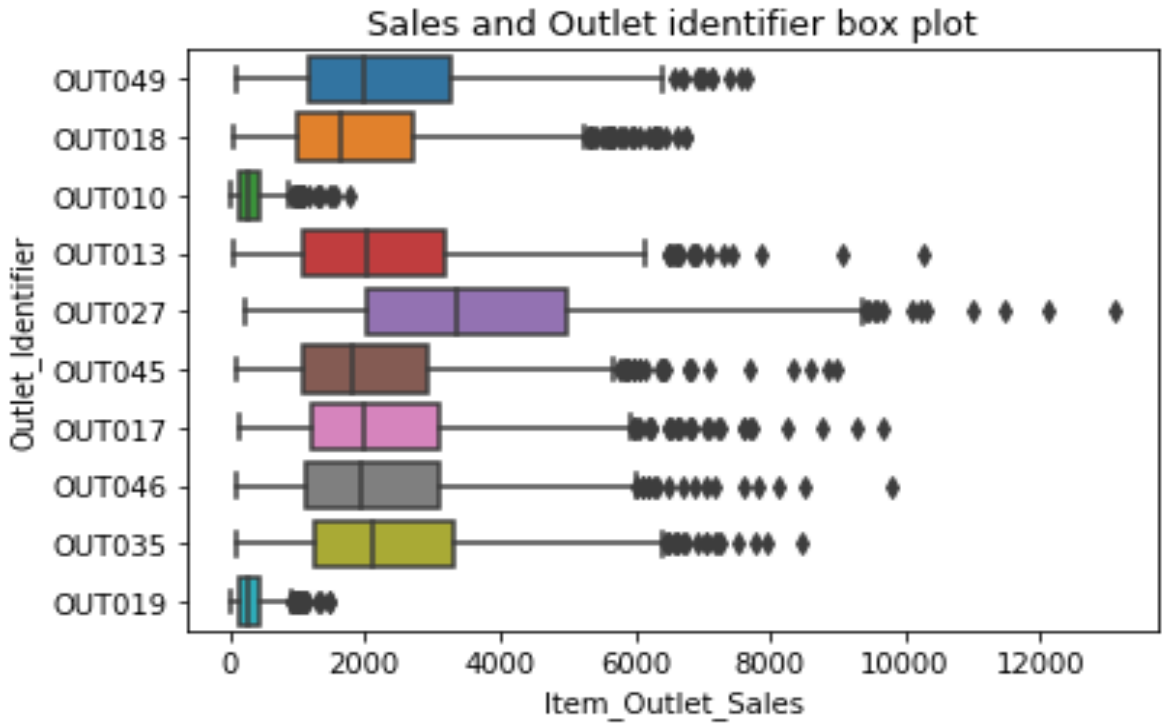
- Most of the product have visibility less than .20
- The graph is having skewness and there are some outliers present

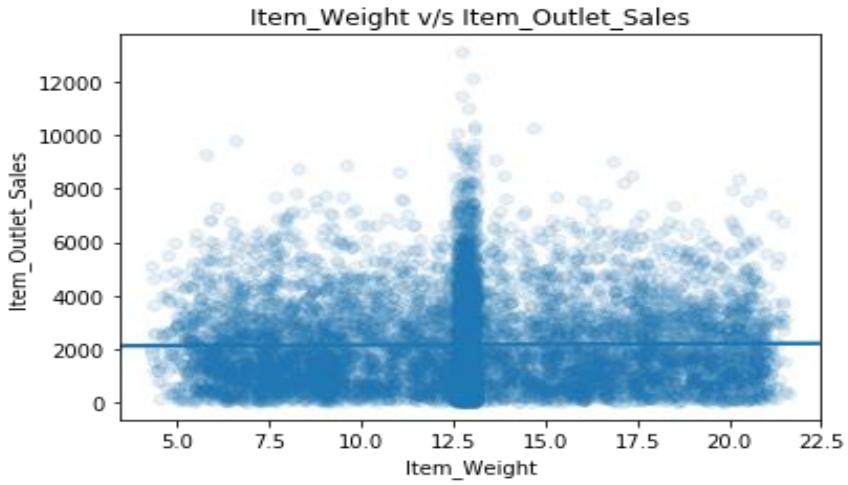


Observations: Super market type-1 is very large in number. Rest 3 are almost same in number

BIVARIATE ANALYSIS:



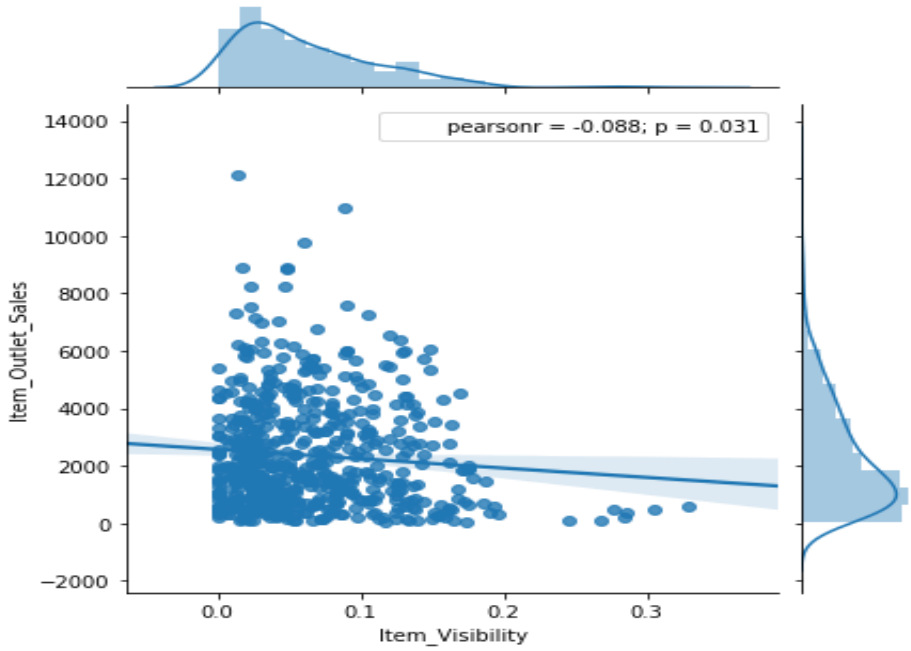




Observations:

-High number of product have weight near 12.5

-Major sales is less than 2000



As Visibiltiy % increases sales decreases

Now some features will be modified in the feature engineering phase:

In [45]:
```
'''
Now I will modify Item Visibility.
As visibilty of some item is '0' which makes no sense so we will replace it with mean of visibilty
'''
data['Item_Visibility'].replace(0,data['Item_Visibility'].mean(),inplace=True)
print ('Number of 0 values after modification: %d'%sum(data['Item_Visibility'] == 0))
data['Outlet_Years'] = 2013 - data['Outlet_Establishment_Year']
```

```
Number of 0 values after modification: 0
```

In [46]:
```
'''
Let's modify Establishment Year As we know that we are having the data of sales collected in 2013.
Older the store greater the sales as they have faith on the store
'''

print(data['Outlet_Years'].describe())
```

```
count    8523.000000
mean       15.168133
std         8.371760
min         4.000000
25%         9.000000
50%        14.000000
75%        26.000000
max        28.000000
Name: Outlet_Years, dtype: float64
```

In [47]:
```
#Mark non-consumables as separate category in low_fat:

data.loc[data['CombineItemType']=="NonConsumable",'Item_Fat_Content'] = "Non-Edible"
print(data['Item_Fat_Content'].value_counts())
```

```
Low Fat       3918
Regular       3006
Non-Edible    1599
Name: Item_Fat_Content, dtype: int64
```

At last the training and testing of the modified dataset will be performed on various algorithms to find the best fit.

1. Linear Regression:

```
In [51]: #Here, I finished with the feature part. Now I will train and split our dataset
         from  sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=101)
         print(x_train.shape)
         print(x_test.shape)
         print(y_train.shape)
         print(y_test.shape)

         (6818, 20)
         (1705, 20)
         (6818,)
         (1705,)

In [53]: #Now we will apply various model on our dataset and get the accuracy score

                                        #Linear Regression
         from sklearn.linear_model import LinearRegression
         lr=LinearRegression()
         lr.fit(x_train, y_train)
         pred=lr.predict(x_test)
         lr.score(x_train, y_train)
         from  sklearn import metrics
         print('MeanAbsoluteError:',  metrics.mean_absolute_error(y_test, pred))

         MeanAbsoluteError: 824.732790921

In [54]: print('MeanSquareError:', metrics.mean_squared_error(y_test, pred))

         MeanSquareError: 1206361.86224

In [55]: print('RootMeanSquareError:', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

2. Decision Tree Regression:

```
In [60]: #DecisionTreeRegressor
         from sklearn.tree import DecisionTreeRegressor
         DTR= DecisionTreeRegressor()
         DTR.fit(x_train,y_train)
         #FINDING Accuracy score
         print(DTR.score(x_test,y_test))

         0.11221671728

In [61]: #Since it is a very low accuracy score so we will not this algorithm here

         dpred=DTR.predict(x_test)
         from  sklearn import metrics
         print('MeanAbsoluteError:',  metrics.mean_absolute_error(y_test, dpred))

         MeanAbsoluteError: 1087.48726111

In [62]: print('MeanSquareError:', metrics.mean_squared_error(y_test, dpred))

         MeanSquareError: 2419533.25522

In [63]: print('RootMeanSquareError:', np.sqrt(metrics.mean_squared_error(y_test, dpred)))

         RootMeanSquareError: 1555.48489392
```

3. Random Rainforest Regression:

In [64]:
```
#RandomForestRegressor
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(x_train, y_train)
print(rfr.score(x_test,y_test))
```

```
0.497313056359
```

In [65]:
```
rpred=rfr.predict(x_test)
from sklearn import metrics
print('MeanAbsoluteError:',  metrics.mean_absolute_error(y_test, rpred))
```

```
MeanAbsoluteError: 823.471397713
```

In [66]:
```
print('MeanSquareError:', metrics.mean_squared_error(y_test, rpred))
```

```
MeanSquareError: 1370005.2713
```

In [67]:
```
print('RootMeanSquareError:', np.sqrt(metrics.mean_squared_error(y_test, dpred)))
```

```
RootMeanSquareError: 1555.48489392
```

In [68]:
```
'''
            GridSearchCV to find the best parameter for RandomforestRegressor
'''
from sklearn.model_selection import GridSearchCV
grid_para={'n_estimators':[50,100,150,200,500],'max_depth':[2,4,5,6,7,8]}
gsb=GridSearchCV(RandomForestRegressor(),grid_para)
gsb.fit(x_train,y_train)
print(gsb.best_params_)
```

```
{'max_depth': 5, 'n_estimators': 200}
```

In [69]:
```
grid_rbr=RandomForestRegressor(max_depth= 5, n_estimators= 200)
grid_rbr.fit(x_train,y_train)
print(grid_rbr.score(x_test,y_test))
```

```
0.598506093468
```

CONCLUSION:

After training and splitting the data and performing various Algorithms such as Random Forest, Gradient Boosting, Linear Regression and Gradient Tree Regressor, we calculated the Accuracy score and came to the conclusion that Gradient Boosting Regressor along with Random Forest Regression has the best result that is near to 60% which is not quite good ideally but still id better than other approaches.

FULL PROJECT LINK:

https://github.com/vermageu/Big-Mart-Sales-Analysis

THANK YOU