TECHNOCOLABS DATA SCIENCE INTERNSHIP

PROJECT REPORT

TITLE: BigMart Sales Prediction

AIM:

To build a predictive model and find out the sales of each product at a particular store.

PROBLEM STATEMENT:

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. Using this model, BigMart will try to understand the properties of products and stores which play a key role in increasing sales.

OVERVIEW:

- The Dataset
- Exploratory Data Analysis
- Data Preprocessing
- Feature Engineering
- Data Modelling

DATASET:

The dataset contains train (8523,12) input and output variable(s) and test (5681,11) input variables and need to predict the sales for test data set.

Train file: CSV containing the item outlet information with sales value

Test file: CSV containing item outlet combinations for which sales need to be forecasted

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.

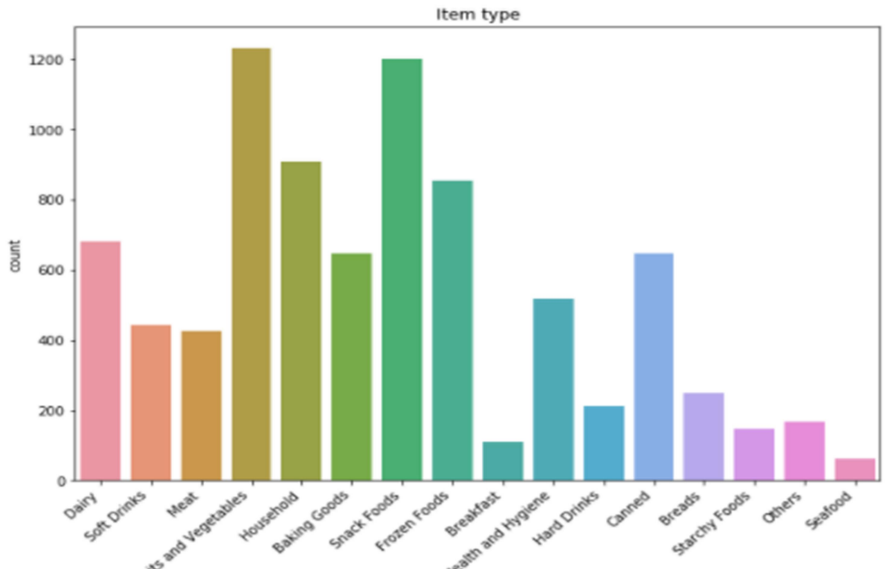
EXPLORATORY DATA ANALYSIS:

In this project, to analyse the data in csv format, I have created a dataframe using pandas and will try to understand the features/variables in the dataset through pandas function like describe(), info() and plot graphs using seaborn and matplotlib python library.

UNIVARIATE ANALYSIS

In univariate analysis we try to understand how each variable/feature has the influence on the target variable and get to know the whether the input variables really impact output variable or not.

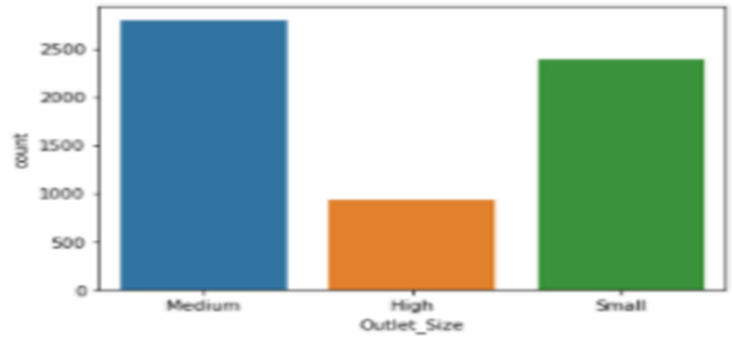
```
In [19]: plt.figure(figsize=(10,7))
         graph1=sns.countplot(x="Item_Type",data=train)
         graph1.set_xticklabels(graph1.get_xticklabels(), rotation = 45,horizontalalignment='right')
         plt.title(" Item type ")
Out[19]: Text(0.5, 1.0, ' Item type ')
```



In [12]:
```
print(train['Outlet_Size'].value_counts())
sns.countplot(x="Outlet_Size",data=train)
```

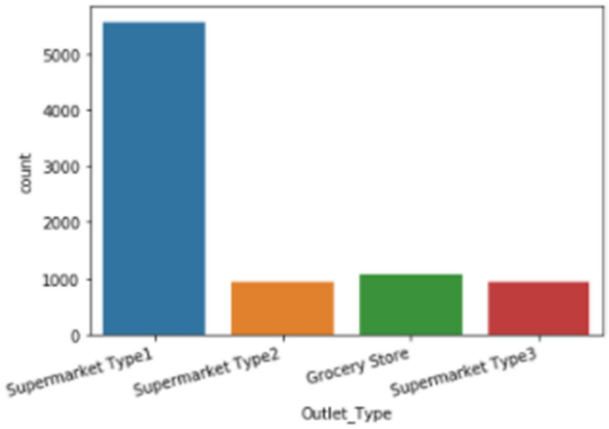
```
Medium    2793
Small     2388
High       932
Name: Outlet_Size, dtype: int64
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x266eab54108>
```



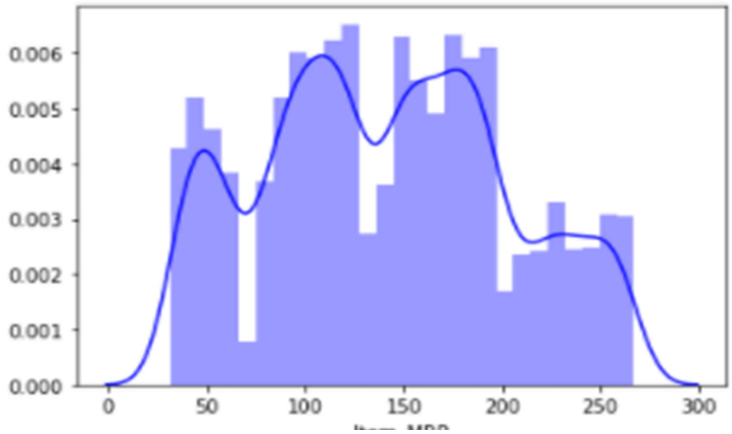
In [14]:
```
g2=sns.countplot(x="Outlet_Type",data=train)
g2.set_xticklabels(g2.get_xticklabels(), rotation = 15,horizontalalignment='right')
print(train['Outlet_Type'].value_counts())
```

```
Supermarket Type1    5577
Grocery Store        1083
Supermarket Type3     935
Supermarket Type2     928
Name: Outlet_Type, dtype: int64
```



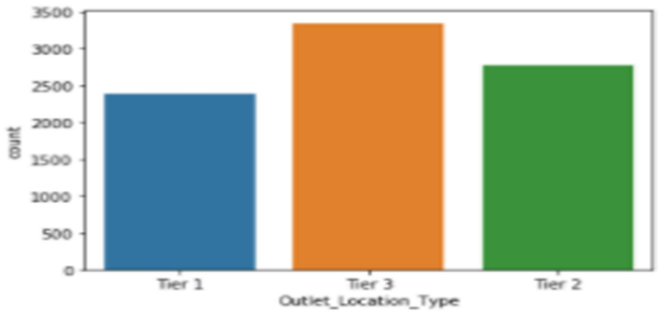
In [17]:
```
sns.distplot(train.Item_MRP,kde=True,color="b")
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x266eaf2dd08>
```



```
In [13]: sns.countplot(x="Outlet_Location_Type",data=train)
         print(train['Outlet_Location_Type'].value_counts())
```

```
Tier 3    3350
Tier 2    2785
Tier 1    2388
Name: Outlet_Location_Type, dtype: int64
```

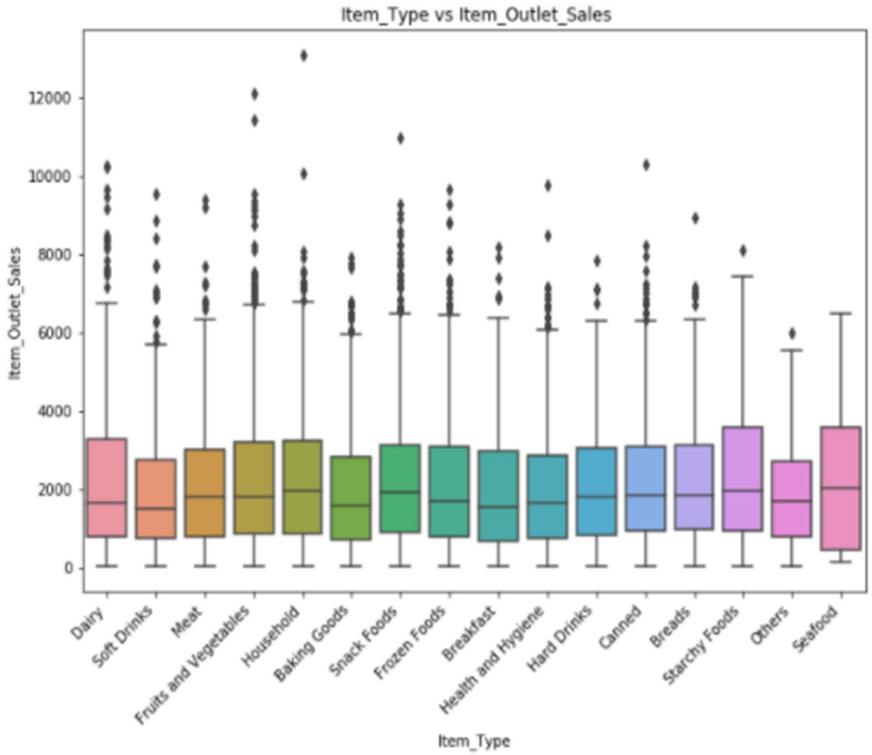


➢ BIVARIATE ANALYSIS:

In bivariate analysis we will analyse how different features/variable are related to each other and how much impact they do have between them.

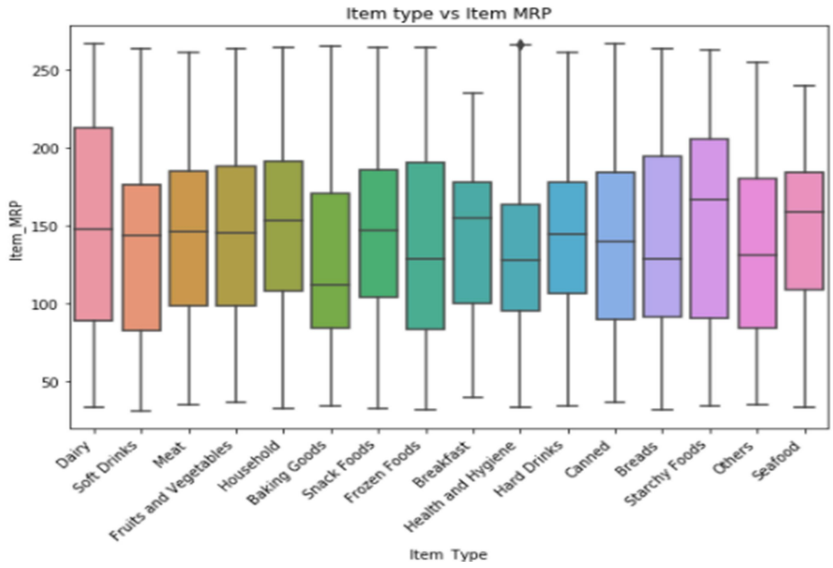
```
In [18]: plt.figure(figsize=(10,7))
         graph1=sns.boxplot(x="Item_Type",y="Item_Outlet_Sales",data=train)
         graph1.set_xticklabels(graph1.get_xticklabels(), rotation = 45,horizontalalignment='right')
         plt.title(" Item_Type vs Item_Outlet_Sales")
```

```
Out[18]: Text(0.5, 1.0, ' Item_Type vs Item_Outlet_Sales')
```

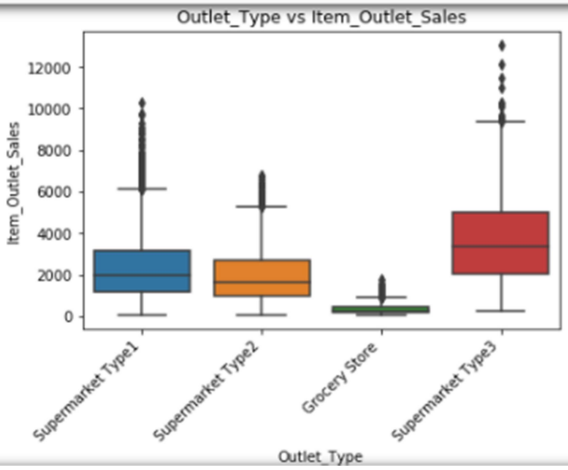


In [16]:
```
plt.figure(figsize=(9,6))
g2=sns.boxplot(x="Item_Type", y="Item_MRP",data=train)
g2.set_xticklabels(g2.get_xticklabels(),rotation = 45,horizontalalignment='right')
plt.title(" Item type vs Item MRP ")
```

Out[16]: Text(0.5, 1.0, ' Item type vs Item MRP ')

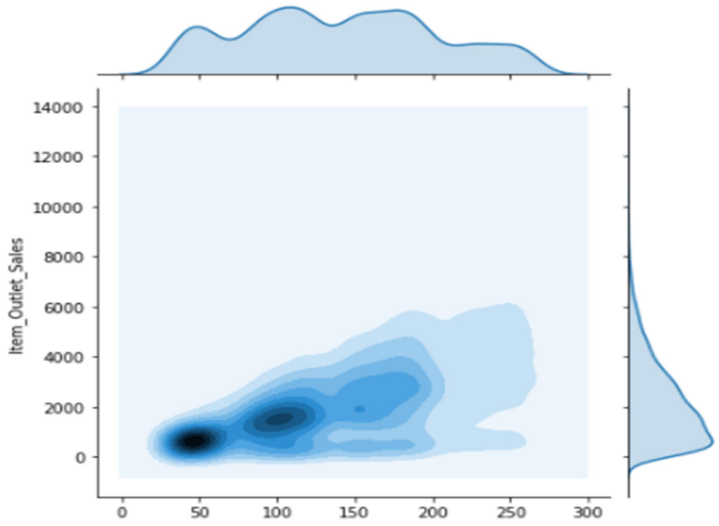


In [20]:
```
plt.figure(figsize=(6,4))
graph1=sns.boxplot(x="Outlet_Type",y="Item_Outlet_Sales",data=train)
graph1.set_xticklabels(graph1.get_xticklabels(), rotation = 45,horizontalalignment='right')
plt.title("Outlet_Type vs Item_Outlet_Sales")
```



In [15]:
```
sns.jointplot("Item_MRP", "Item_Outlet_Sales",data=train,kind='kde')
```

Out[15]: <seaborn.axisgrid.JointGrid at 0x266eaba0588>



DATA PREPROCESSING:

Now we try to impute missing values by different methods like by replacing nan values with either mean or mode and check the number of missing values by isna() function.

```
In [21]: sales['Item_Weight']=sales['Item_Weight'].fillna(sales['Item_Weight'].mean())

In [22]: from scipy.stats import mode

         sales['Outlet_Size']=sales['Outlet_Size'].fillna(sales['Outlet_Size'].mode().iloc[0])

In [23]: sales.isna().sum()

Out[23]: 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          5681
         category                      0
         dtype: int64
```

FEATURE ENGINEERING:

In feature engineering we transform raw data into new variables/features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.

➢ One Hot Encoding

```
In [29]: sales=pd.get_dummies(sales,columns=['Item_Type','Outlet_Location_Type','Outlet_Type','Item_Fat_Content','Outlet_Size'])
```

From our dataset we have converted the variables item type, outlet_location_type, outlet_type, item_fat_content and outlet_size to one hot code variables for better individual understanding.

➢ Label encoding

```
In [31]: #Item_Identifier and outlet_Identifier are also useful for making prediction
         from sklearn.preprocessing import LabelEncoder
         num=LabelEncoder()
         sales['Outlet']=num.fit_transform(sales['Outlet_Identifier'])
         sales['Identifier']=num.fit_transform(sales['Item_Identifier'])
```

Variables Outlet_identifier and Item_identifier are converted to labels like 1,2,3… for better interpretability.

➢ Splitting the train and test data after modification:

```
In [36]: #Divide into test and train:
         train = sales.loc[sales['category']=="train"]
         test = sales.loc[sales['category']=="test"]

         #Drop unnecessary columns:
         test.drop(['Item_Outlet_Sales','category'],axis=1,inplace=True)
         train.drop(['category'],axis=1,inplace=True)

         #Export files as modified versions:
         train.to_csv("train_modified.csv",index=False)
         test.to_csv("test_modified.csv",index=False)
```

DATA MODELLING

Now using the training data we will train our model using different machine learning algorithms to predict the sales price. First we will split our data using train_test_split method and will use Linear Regression , Regularized linear regression, Random Forest and XGBoost algorithms and let's see which model will give the lowest RMSE value which will become our preferable model to predict the sales price more accurately.

```
In [41]: X=train_model.drop(['Item_Identifier','Outlet_Identifier','Item_Outlet_Sales'],axis=1)

In [42]: y=train_model['Item_Outlet_Sales']

In [43]: y

Out[43]: 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

In [44]: from sklearn.model_selection import KFold,train_test_split
         from sklearn.linear_model import LinearRegression
         from sklearn.svm import SVR
```

➢ Linear Regression:

In [46]:
```
model=LinearRegression()
model.fit(X_train,y_train)
```

Out[46]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [47]:
```
y_pred=model.predict(X_test)
```

In [48]:
```
y_pred
```

Out[48]:
```
array([1354.26089453,  703.16781773,  847.79410303, ...,  820.09919209,
        591.60548781, 1682.33613198])
```

In [49]:
```
from sklearn import metrics

np.sqrt(metrics.mean_squared_error(y_test,y_pred))
```

Out[49]:
```
1069.7153798211723
```

➢ Regularized Linear Regression:

In [51]:
```
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
```

In [52]:
```
rr = Ridge(alpha=0.01)
rr.fit(X_train, y_train)
pred_train_rr= rr.predict(X_train)
print(np.sqrt(metrics.mean_squared_error(y_train,pred_train_rr)))

pred_test_rr= rr.predict(X_test)
print(np.sqrt(metrics.mean_squared_error(y_test,pred_test_rr)))
```

```
1141.3987914916424
1069.7134265872414
```

In [53]:
```
model_lasso = Lasso(alpha=0.05)
model_lasso.fit(X_train, y_train)
pred_train_lasso= model_lasso.predict(X_train)
print(np.sqrt(metrics.mean_squared_error(y_train,pred_train_lasso)))

pred_test_lasso= model_lasso.predict(X_test)
print(np.sqrt(metrics.mean_squared_error(y_test,pred_test_lasso)))
```

```
1141.411688263805
1069.5857085417967
```

➢ Random Forest:

In [54]:
```
from sklearn.ensemble import RandomForestRegressor

rfg=RandomForestRegressor()
rfg.fit(X_train,y_train)
rfg_train_predict=rfg.predict(X_train)
print(np.sqrt(metrics.mean_squared_error(y_train,rfg_train_predict)))

rfg_test_predict=rfg.predict(X_test)
print(np.sqrt(metrics.mean_squared_error(y_test,rfg_test_predict)))
```

```
430.3685966907478
1093.1632935509497
```

➢ XGBoost :

```
In [55]: from xgboost import XGBRegressor

         model1 = XGBRegressor(n_estimators=1000, learning_rate=0.02,random_state=42)
         model1.fit(X_train,y_train)
```

```
Out[55]: XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                      colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
                      importance_type='gain', interaction_constraints='',
                      learning_rate=0.02, max_delta_step=0, max_depth=6,
                      min_child_weight=1, missing=nan, monotone_constraints='()',
                      n_estimators=1000, n_jobs=0, num_parallel_tree=1,
                      objective='reg:squarederror', random_state=42, reg_alpha=0,
                      reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
                      validate_parameters=1, verbosity=None)
```

```
In [56]: y_pred1=model1.predict(X_test)
```

```
In [57]: np.sqrt(metrics.mean_squared_error(y_pred1,y_test))
```

```
Out[57]: 1086.0191469257325
```

➢ Final Predictions on the Test Dataset:

Loading Test Data

```
In [58]: test_model=pd.read_csv("test_modified.csv")
```

```
In [59]: predictors=test_model.drop(["Item_Identifier","Outlet_Identifier"],axis=1)
```

Predicting with lowest rmse model i.e Linear Regression

```
In [60]: y_pred2=model.predict(predictors)
```

```
In [61]: y_pred2
```

```
Out[61]: array([1858.40092322, 1496.87896143, 1844.56693591, ..., 1865.37930995,
                3567.15088817, 1317.25405714])
```

➢ SUMMARY

Finally we have implemented all the algorithms and saw that we got best results from linear regression model giving root mean squared value of 1069 which is the lowest among all other algorithms and from this model BigMart can understand the properties of products and outlets which play a major role for increasing the sales price and will help them for their company's growth.

Prepared by: RACHIT AGARWAL