IDC HERZLIYA

Final Project: Big mart sales

Predict product sales in supermarkets

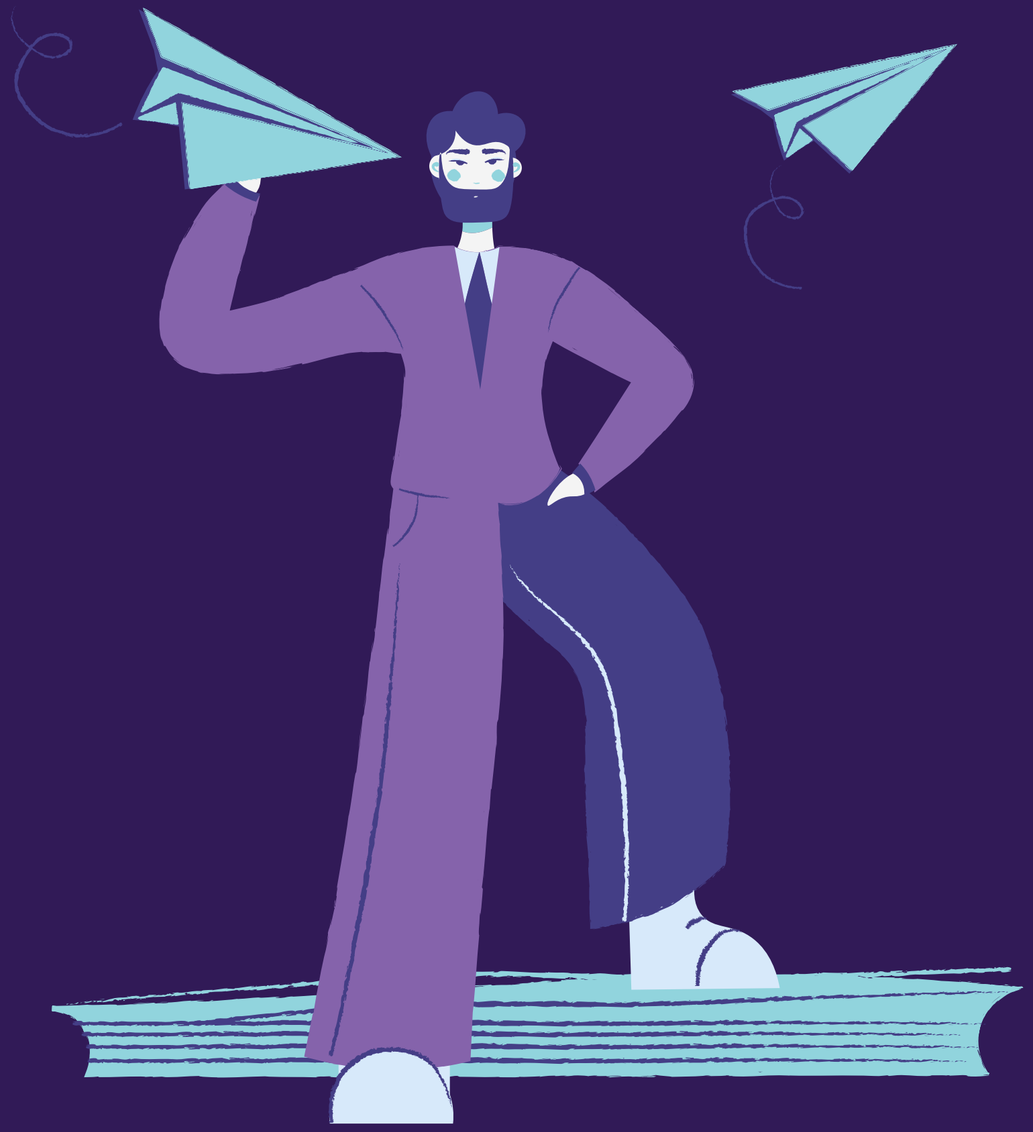
The Team

IDC HERZLIYA

Ofir Meir
203342324

Guy Koifman
311359806

Samuel Szpilman
336319538



The Problem:

- We want to predict *the sales of each product at a particular store*
- *For that we have a dataset of 8523 instances which is comprised of 1559 products in 10 differents stores.*
- *We will try to understand which features of the data make better predicion Sales*

Evaluation:

We evaluated the performance of the algorithm using Rooted Mean Squared Error.

Rooted Mean Squared Error is widely used to evaluate regression algorithm performance.

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}}$$

Evaluation:

We choose randomly 75% of the data to train our models. The rest 25% of the data for estimating the models preformence.

Data Description

1. HOW MANY EXAMPLES

- Train - 6392
- Test - 2131

2. FEATURES

- 11 features in the original data set
- 35 features after get_dummies
- 8 feature for analysis after cleaning

3. MISSING VALUES

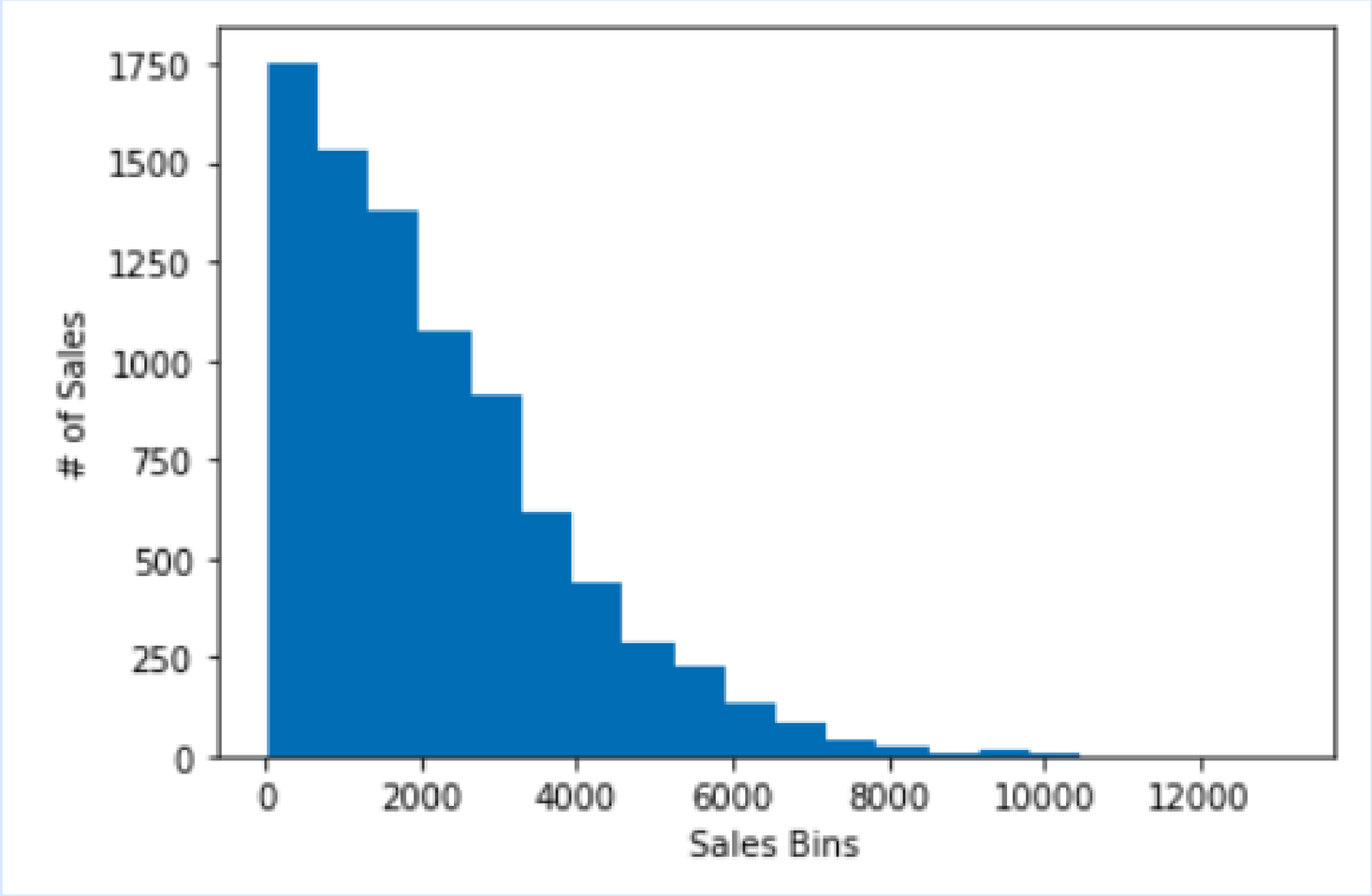
There are 2 features with missing values:

- Item_Weight
- Outlet_Size

Data Description

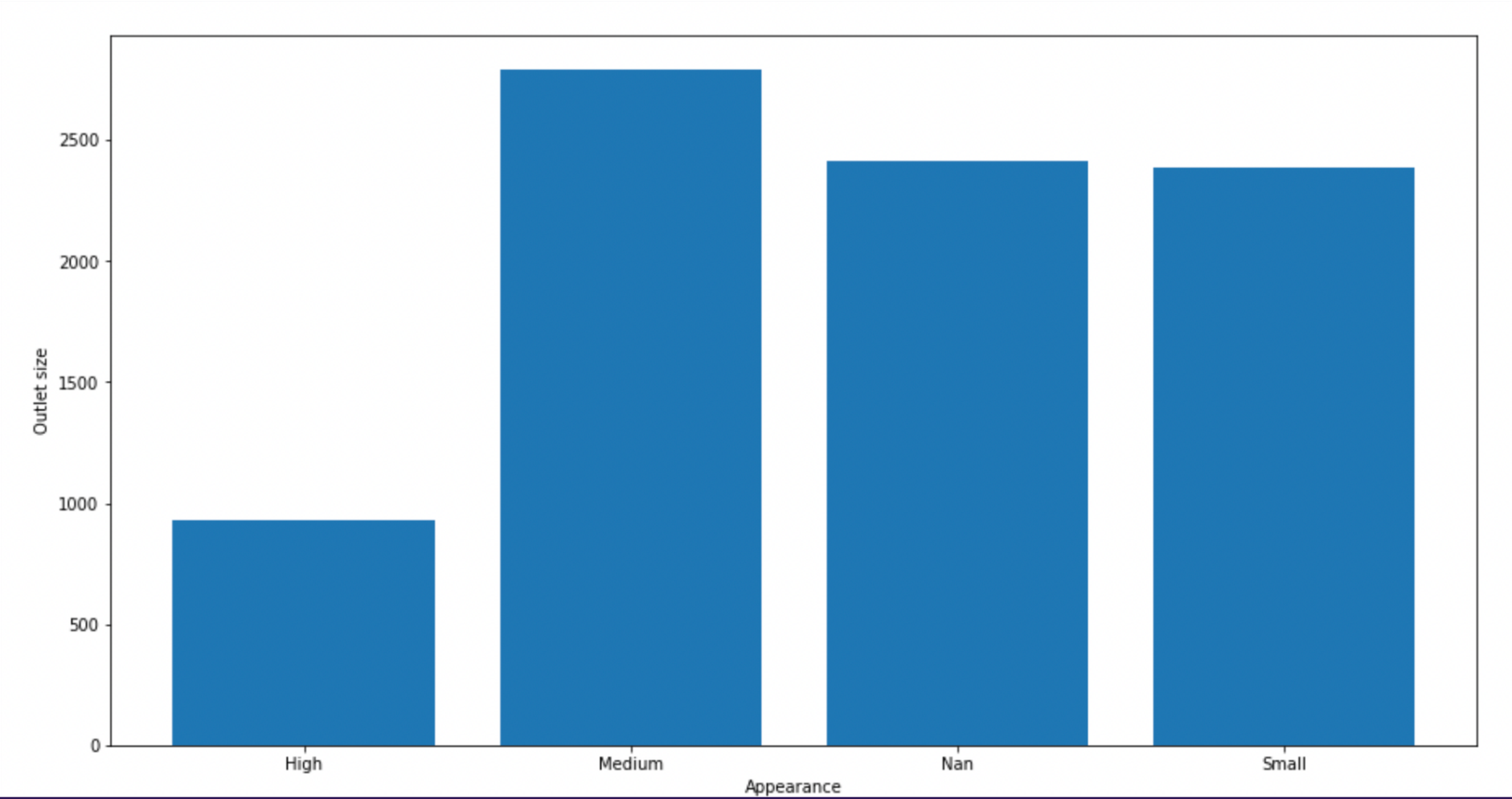
VALUES DISTRIBUTION

The Labels are the number of products sold. We can see that many products has low number of sales



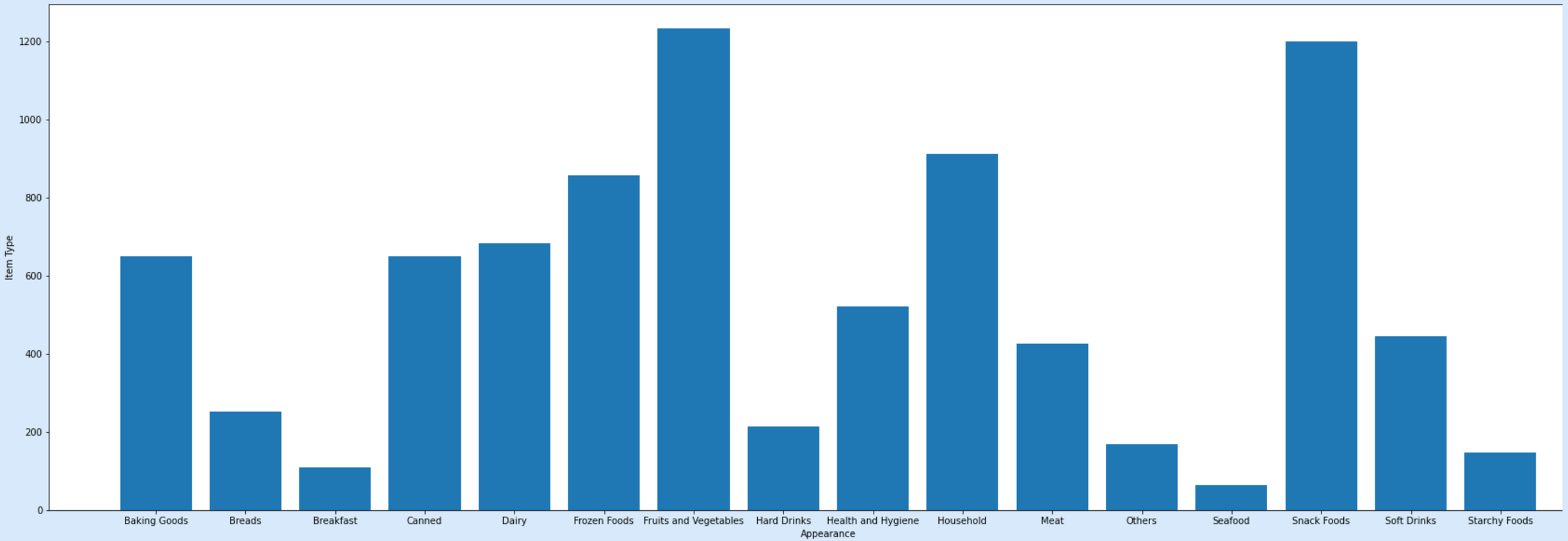
The amount of each Outlet_size in the data

*This feature has the highest number of missing values in the data

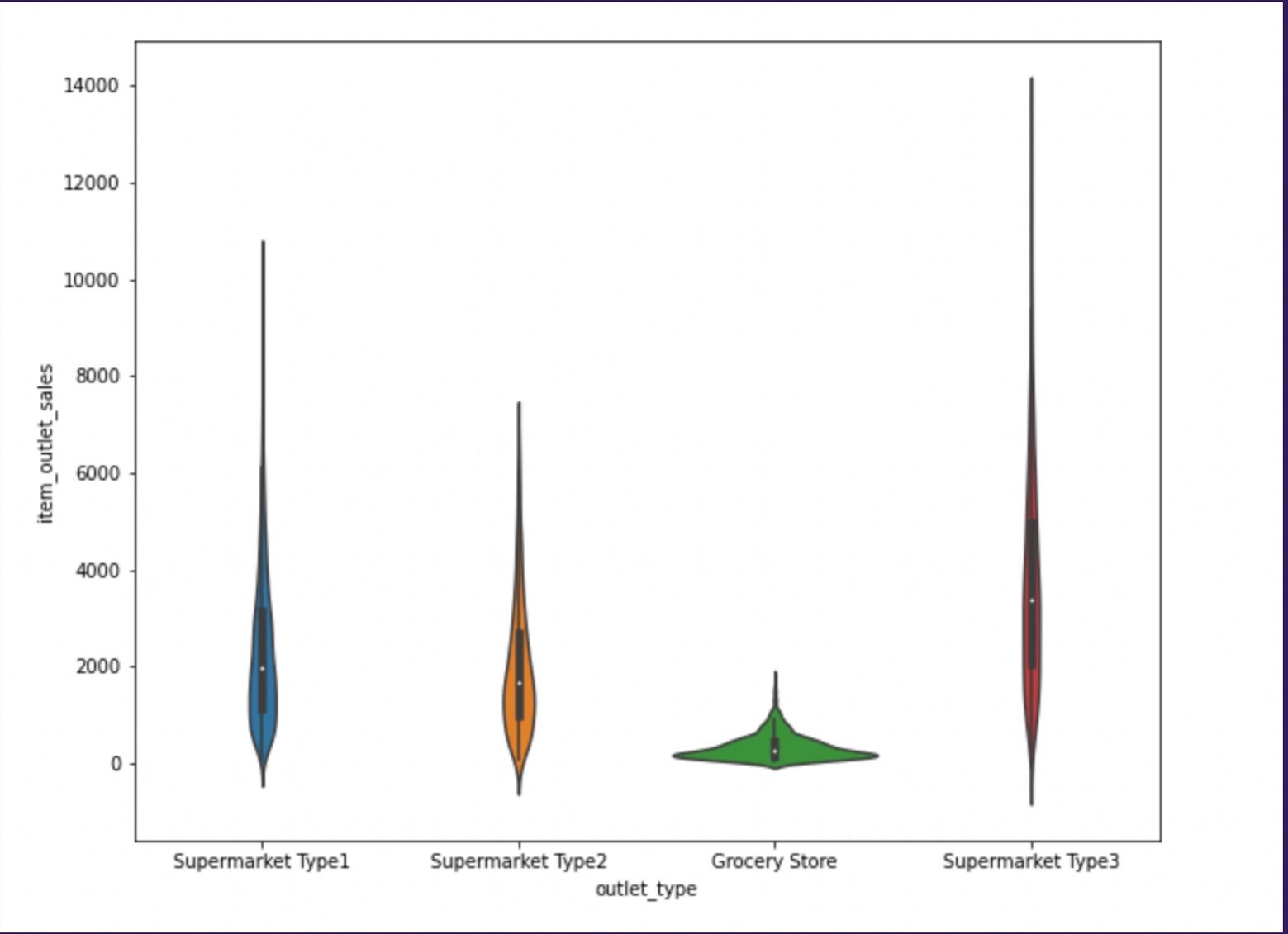


The amount of each Item_type in the data

*This feature has 16 different values



Given the outlet type what is the item sales distribution



Data engineering - Removed features:

After examining the feature importance (AdaBoost) and Lasso Coefficient, the following features where found to be less significant for the prediction:

- item_type_identifier_Drinks
- item_type_identifier_Food
- item_type_identifier_Non-Consumable
- item_fat_content_Low Fat
- item_fat_content_Regular
- item_type_Baking Goods
- item_type_Breads
- item_type_Breakfast
- item_type_Canned
- item_type_Dairy
- item_type_Frozen Foods
- item_type_Fruits and Vegetables
- item_type_Hard Drinks
- item_type_Health and Hygiene
- item_type_Household
- item_type_Meat
- item_type_Others
- item_type_Seafood
- item_type_Snack Foods
- item_type_Soft Drinks
- item_type_Starchy Foods
- outlet_size_High
- outlet_size_Medium
- outlet_size_Small
- outlet_location_type_Tier 1
- outlet_location_type_Tier 2
- outlet_location_type_Tier 3

Data engineering - Removed features:

Performance Before removing :

RandomForestRegressor	1104.5413973978161
DecisionTreeRegressor	1117.0207628330252
Lasso	1148.8051406390823
LinearRegression	1148.8086694328676
AdaBoostRegressor	1154.346821659374
KNeighborsRegressor (Scaled)	1224.9379970846983
KNeighborsRegressor	1263.5906321635198
ConstantBaseline	1730.9446399475419

Performance After removing :

RandomForestRegressor	1102.854845676016
DecisionTreeRegressor	1113.5722868612604
Lasso	1147.671176969377
LinearRegression	1147.673415508388
KNeighborsRegressor (Scaled)	1156.3217921671453
AdaBoostRegressor	1156.808144364127
KNeighborsRegressor	1326.7003229786337
ConstantBaseline	1730.9446399475419

Data engineering - Added features:

The data set contained the feature Outlet_Establishment_Years That vary from 1985 to 2009. We decided that the values might not be appropriate in that form , so we converted them to how old the particular store is.

Data engineering - Missing Values:

We saw that 'item_weight' and 'outlet_size' has lots of missing values.

checking which columns have Nan values

```
data.isna().mean()
```

```
item_type_identifier          0.000000
item_weight                   0.171653
item_fat_content              0.000000
item_visibility               0.000000
item_type                     0.000000
item_mrp                      0.000000
outlet_establishment_year     0.000000
outlet_size                   0.282764
outlet_location_type          0.000000
outlet_type                   0.000000
item_outlet_sales             0.000000
dtype: float64
```

Data engineering - item_weight:

- We imputed the missing values of 'item_weght'with the mean value of the entire weight data.
- Notice that from the right the item weight Avg given the item type doesn`t vary much.
- Note that the feature importance of item_weight is insignificante
- Therefore we decided to change all messing values in the item_weight column to the Avg of the entire column.

```
np.mean(data['item_weight'])
```

```
12.857645184135976
```

item_type	item_weight
Baking Goods	12.277108
Breads	11.346936
Breakfast	12.768202
Canned	12.305705
Dairy	13.426069
Frozen Foods	12.867061
Fruits and Vegetables	13.224769
Hard Drinks	11.400328
Health and Hygiene	13.142314
Household	13.384736
Meat	12.817344
Others	13.853285
Seafood	12.552843
Snack Foods	12.987880
Soft Drinks	11.847460
Starchy Foods	13.690731

Data engineering - outlet_size:

We used the most common size value using 'mode' for each 'outlet_type'. We looked at the instances that have missing values at 'outlet_size' and imputed them with the matching mode value of their outlet_type.

```
Mode for each Outlet_Type:
outlet_type Grocery Store Supermarket Type1 Supermarket Type2  \
outlet_size         Small              Small             Medium

outlet_type Supermarket Type3
outlet_size            Medium

Orignal #missing: 2410


Changed #missing: 0
```

Algorithms performance

Constant Baseline

RMSE:

1698.13	1730.94
Train	Test

Algorithms performance

k_list = [4, 7, 11]

Best k = 7

KNeighborsRegressor

Best k RMSE:

1096.80	1326.70
Train	Test

Algorithms performance

k_list = [3, 7, 9]

Best k = 9

KNeighborsRegressor
Scaled (StandardScaler)

Best k RMSE:

1004.22	1156.32
Train	Test

Algorithms performance

max_depth_list = [2,6,9]

Best max_depth = 6

Decision Tree Regressor

Best max_depth RMSE:

1052.53	1113.57
Train	Test

Algorithms performance

max_depth_list = [2,6,9]

Best max_depth = 6
n_estimators = 100

Random Forest Regressor

Best max_depth RMSE:

1040.14	1102.85
Train	Test

Algorithms performance

max_depth_list = [3,7,11]
Best max_depth = 11
n_estimators = 100

AdaBoost Regressor

Best max_depth RMSE:

664.07	1156.80
Train	Test

Algorithms performance

alpha_list = [0.000001,0.0001 ,0.01]

Best alpha = 0.01

Lasso regression Scaled (MinMaxScaler)

Best alpha RMSE:

1123.10	1147.67
Train	Test

Algorithms performance

A comparison of all Algorithms:

	Test RMSE
RandomForestRegressor	1102.854845676016
DecisionTreeRegressor	1113.5722868612604
Lasso	1147.671176969377
LinearRegression	1147.673415508388
KNeighborsRegressor (Scaled)	1156.3217921671453
AdaBoostRegressor	1156.808144364127
KNeighborsRegressor	1326.7003229786337
ConstantBaseline	1730.9446399475419

Algorithms introspection

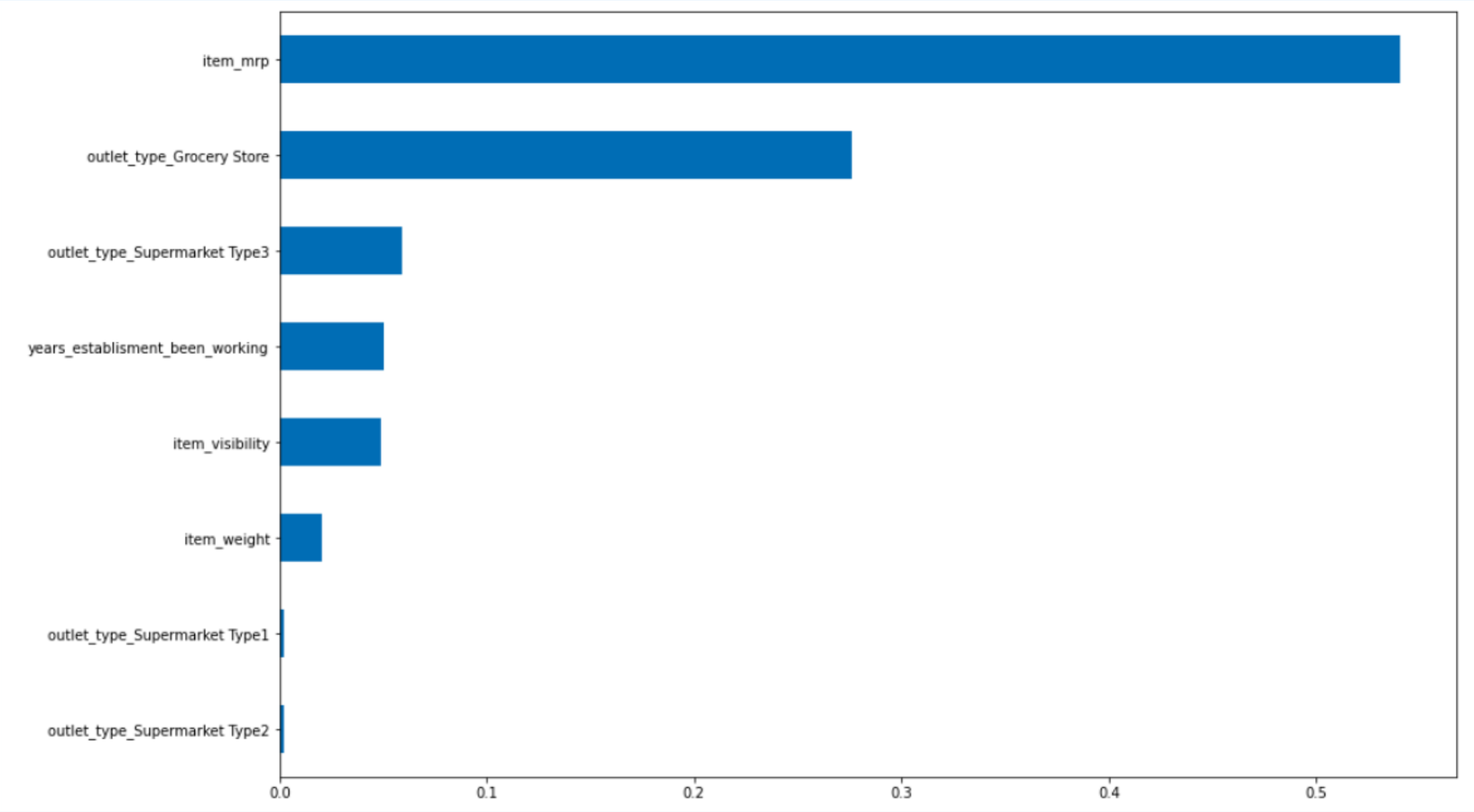
Weights of the Lasso coefficients:

item_mrp - maximum retail price for item

outlet_type_grocery_store:

Algorithms introspection

Random Forest Feature Importance



Hyperparameters

Random Forest Regressor Best Hyperparameters:

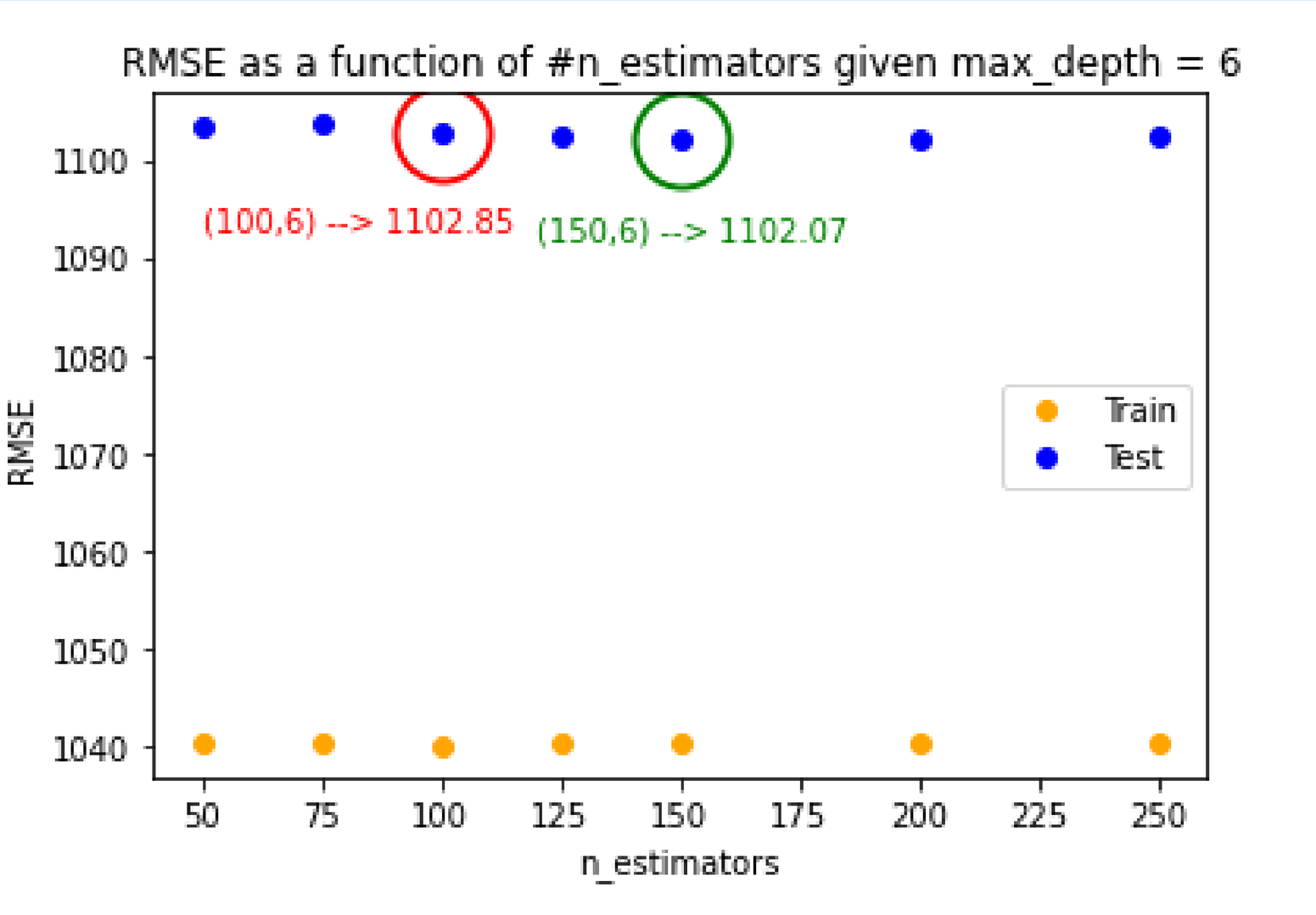
n_estimators_list = [50,75,100,125,150,200,250]
max_depth_list = [1,2,3,4,5,6,7]

Best n_estimators = 150
Best max_depth = 6

Hyperparameters

Random Forest Regressor Best Hyperparameters:

n_estimators = 150
max_depth = 6



Additional Analysis:

Performance vs. amount of data:

There is no need to collect more data. As we see in the graph, after using more than 50% from the training data the test RMSE is not decreasing .