İZMİR EKONOMİ ÜNİVERSİTESİ

CE 477 - Data Science

2022 - 2023 Fall

Project Report

Burcu Koçulu
Mustafa Barış Demirörs

Data Set Link

Github Link

1. Introduction

Data set contains 60.000 customers' information(personal, shopping) and the convenience store information of the company Food Mart (CFM). Food Mart is a chain of convenience stores in the United States. The private company's headquarters are located in Mentor, Ohio, and there are currently approximately 325 stores located in the US. Convenient Food Mart operates on the franchise system.

2. Data Set Description

Along with the outputs of descriptions showing dataset statistics, the dataset has 60428 instances and 40 attributes including 23 numeric and 17 categorical.

There are 12 variables which are numeric but categoric and 4 variables are categorical but cardinal. We defined cardinal variables as the variables which have more than 20 classes.

2.1 Missing Value Analysis

According to our missing value analysis, there are no missing values in any of the attributes of this dataset. As we do not have missing values, we did not do missing value imputation.

food_category	0	SRP	0
food_department	0	gross_weight	0
food_family	0	net_weight	0
store_sales(in millions)	0	recyclable_package	0
store_cost(in millions)	0	low_fat	0
unit_sales(in millions)	0	units_per_case	0
promotion_name	0	store_type	0
sales_country	0	store_city	0
marital_status	0	store_state	0
gender	0	store_sqft	0
total_children	0	grocery_sqft	0
education	0	frozen_sqft	0
member_card	0	meat_sqft	0
occupation	0	coffee_bar	0
houseowner	0	video_store	0
avg_cars_at home(approx)	0	salad_bar	0
avg. yearly_income	0	prepared_food	0
num_children_at_home	0	florist	0
avg_cars_at home(approx).1	0	media_type	0
brand_name	0	cost	0

dtype: int64

Total : 0

Figure 1. Missing Values

2.2 Numeric Attributes Analysis

We checked the quantiles of numeric columns such as store_sales(in millions),store_cost(in millions),SRP, gross_weight etc. According to quantile analysis numeric variables do not have outliers.

2.2.1 Histograms

Histograms to check distribution of variables.

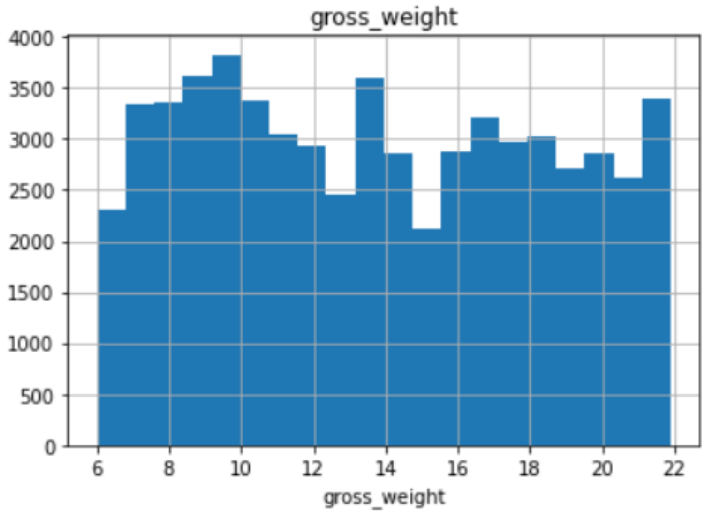


Figure 2. Histogram of "gross_weight" variable

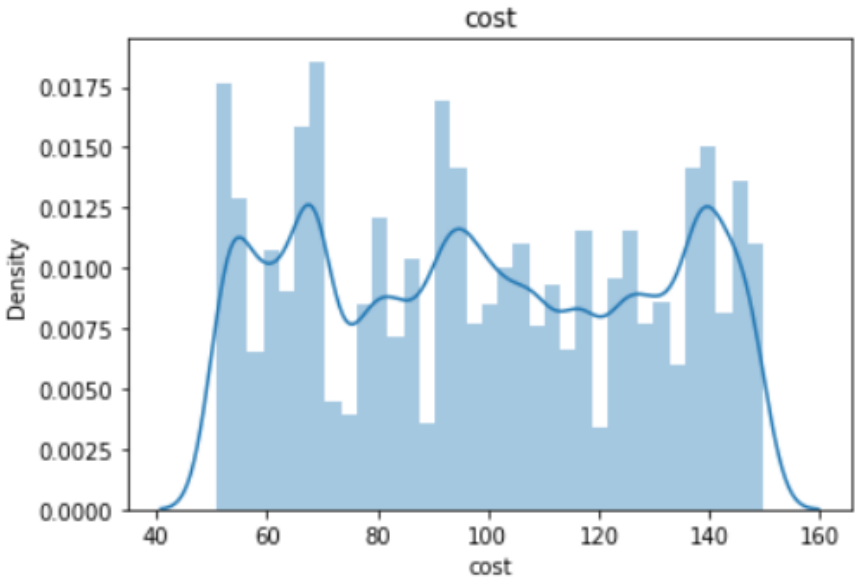


Figure 3. Histogram of "cost" variable

Store_sales and store_cost numeric variables are right skewed.

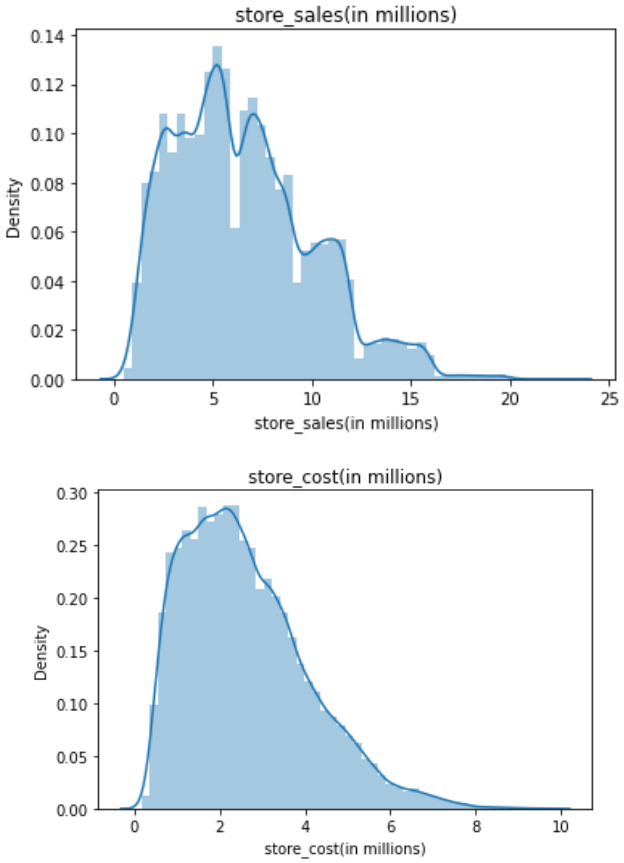


Figure 4. Histogram of "store_sales" and "store_cost" variables

2.2.2 Scatter Plots

1) SRP- Store Sales

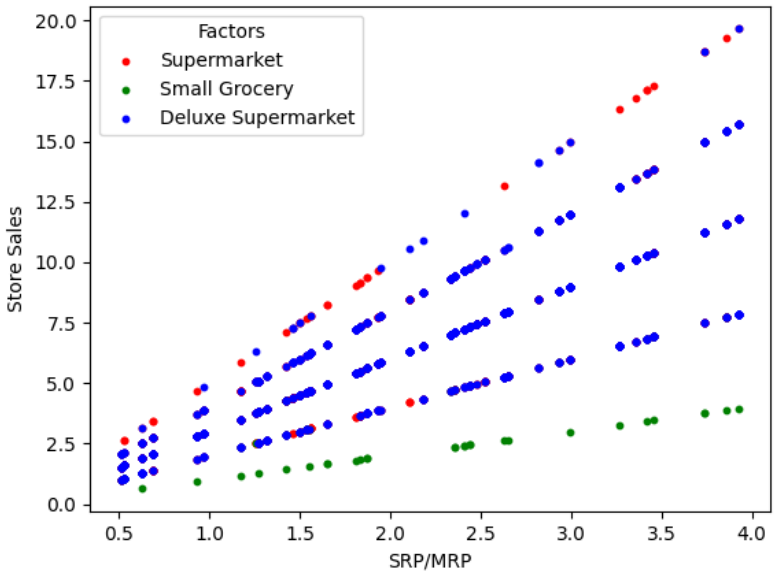


Figure 5. Scatter plot for SRP-Store Sales

2) Store Cost - Store Sale

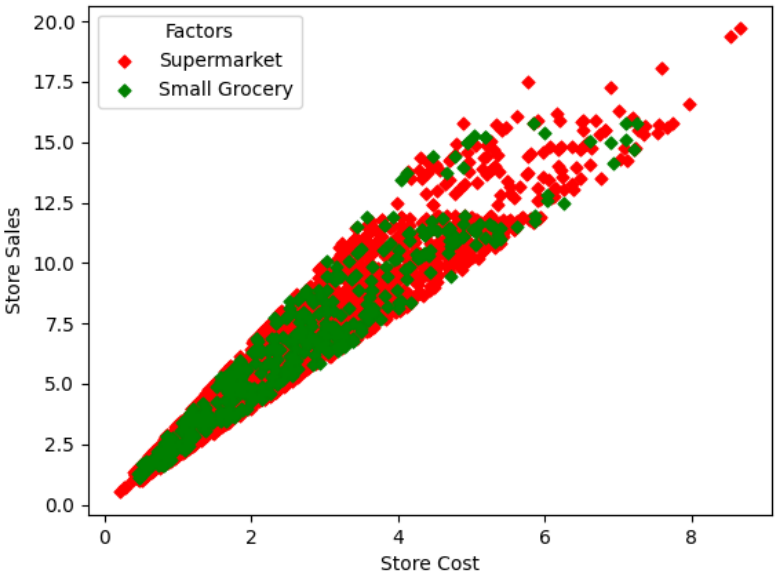


Figure 6. Scatter plot for Store Cost-Store Sales

3)Store Sales - Total Children in Home

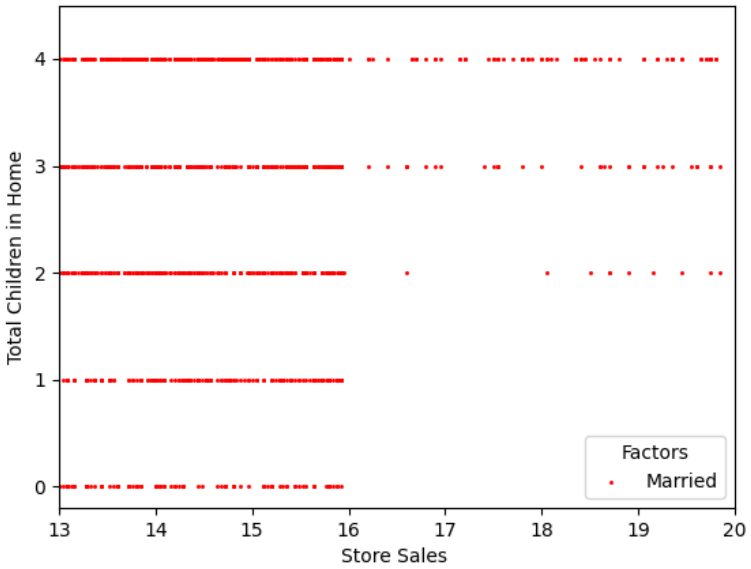


Figure 7. Scatter plot for Store Sales-Total Children

2.3 Correlation Analysis

For correlation analysis, seaborn library's heatmap method were used, the result can be seen in Figure 5. According to correlation matrix:

- There is a strong positive correlation between SRP, store_cost and store_sales variables.
- Unit_sales variable have strong positive correlation with store_cost and store_sales variables.
- There is a strong positive correlation between grocery_sqft, frozen_sqft and meat_sqft variables. Salad_bar and prepared_food variables have positive correlation with these variables.
- Video_store, salad_bar and prepared_food variables have strong positive correlation between each other.
- Grocery_sqft and coffee_bar variables have weak negative correlation.

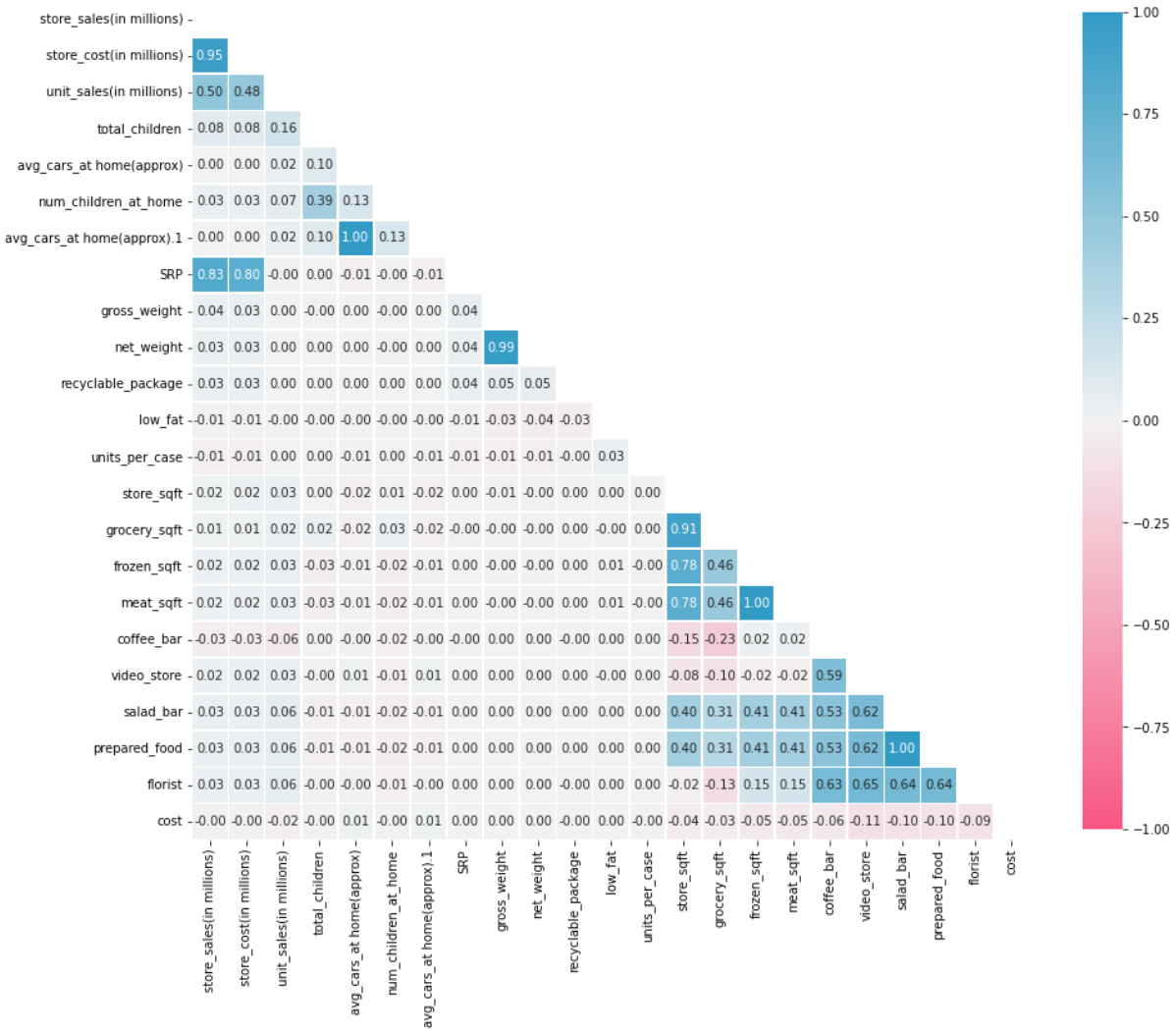


Figure 8. Correlation m9atrix

2.4 Outlier Analysis

For outlier analysis, IQR method were used. In notebook we used **Outlier_threshold** function to find upper and lower bound of every numeric variable. And according to those thresholds, **check_outlier** function controls if there any outlier values which is bigger than upper bound and smaller than lower bound of that variable. As a result there weren't any outliers.

```
for i in num_cols:
    print(check_outlier(data,i))
```

```
False
False
False
False
False
False
False
False
False
False
False
```

Figure 9. Outlier Analysis

2.4.1 Boxplot

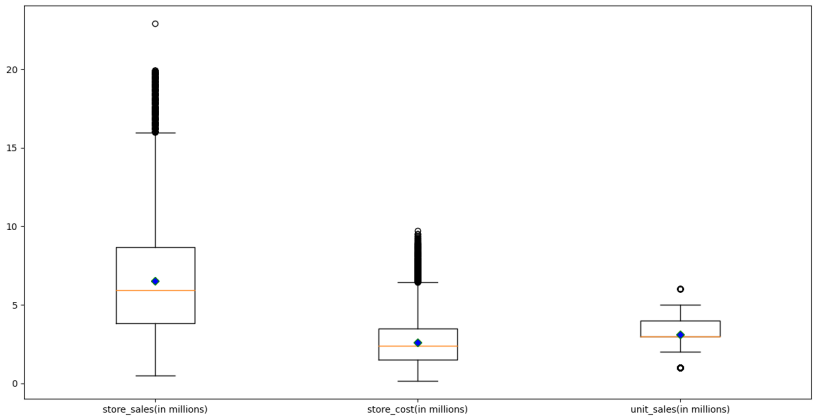
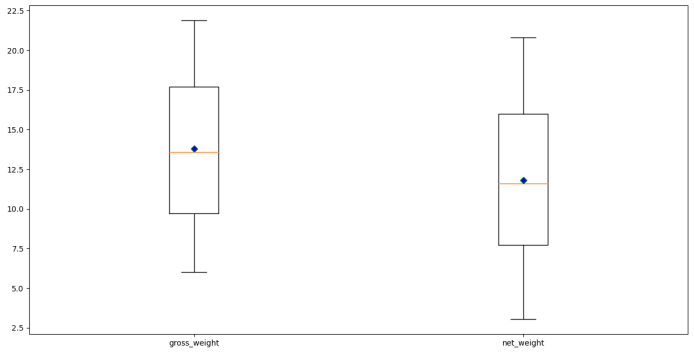


Figure 10. Store sales, store cost, unit sales boxplot



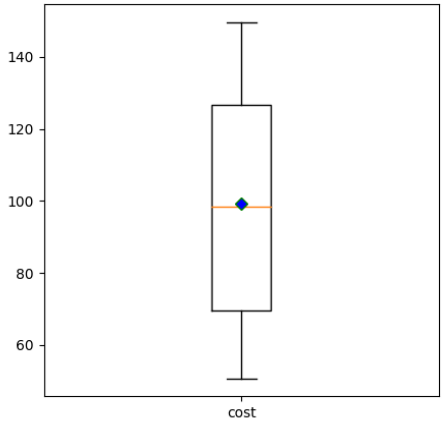


Figure 11. Product gross, net weight, Cost boxplot

3. Preprocessing

3.1 Discretization

Discretization of some numeric attributes using numpy library.

1) Cost of acquiring a customer

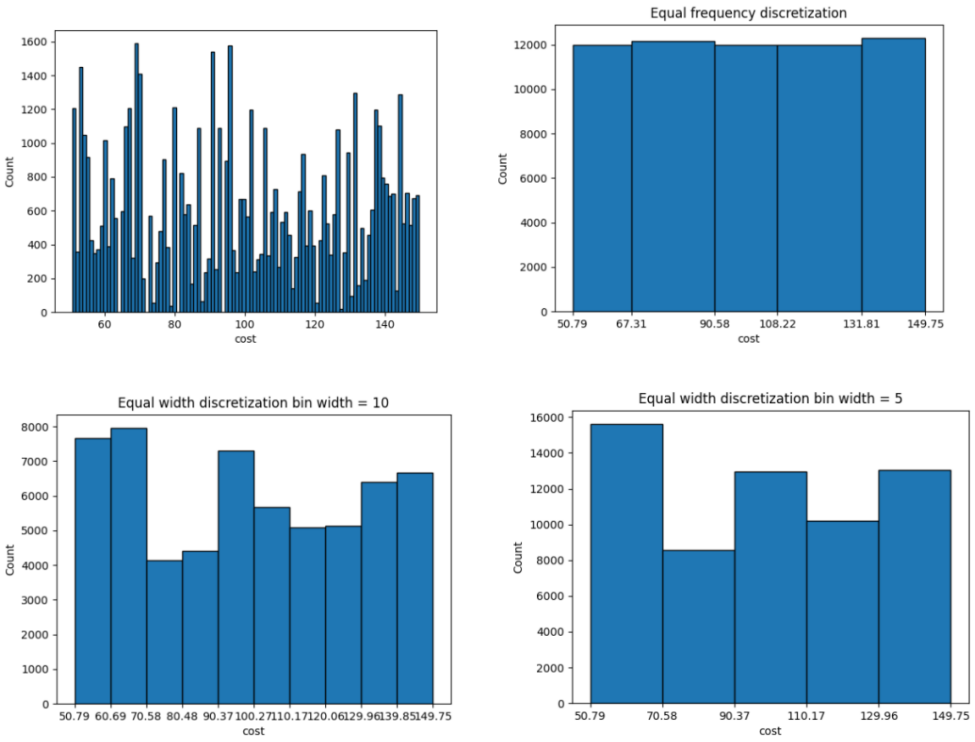


Figure 12. Discretization of cost

2) Store Sales

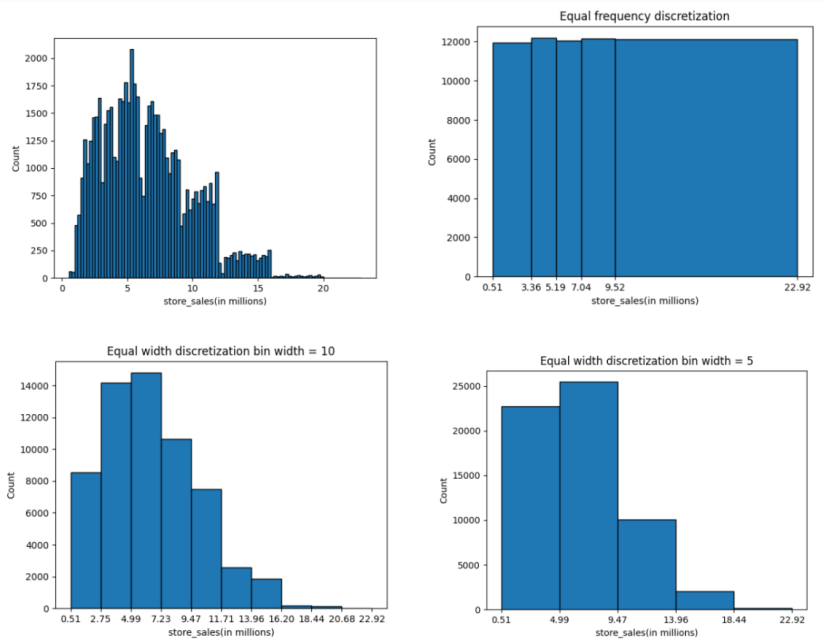


Figure 13. Discretization of store_sales variable

3.2 Scaling

For standardization, StandardScaler, and for normalization MinMaxScaler methods of Scikit-Learn library were used. Since numeric variables mostly do not have a normal or gaussian distribution, normalization is a preferable scaling technique for our project. Variables **which** has gaussian distribution such as gross_weight and net_weight, standardization is a preferable scaling technique.

	store_sales(in millions)	store_cost(in millions)	SRP	gross_weight	net_weight	units_per_case	store_sqft	grocery_sqft	frozen_sqft	meat_sqft	cost
0	0.306	0.268	0.385	0.862	0.825	0.457	0.381	0.315	0.440	0.440	0.766
1	0.224	0.254	0.385	0.862	0.825	0.457	0.381	0.315	0.440	0.440	0.092
2	0.141	0.125	0.385	0.862	0.825	0.457	0.381	0.315	0.440	0.440	0.337
3	0.141	0.106	0.385	0.862	0.825	0.457	0.381	0.315	0.440	0.440	0.455
4	0.159	0.132	0.247	0.070	0.116	0.800	0.381	0.315	0.440	0.440	0.000

Figure 14. Min-Max Scaled columns

3.3 Encoding

We applied one hot encoding to variables who have more than two classes and less than 10 classes because if we apply one hot encoding to the features which have more than 10 classes and cardinal features, we would work with high-dimensional data. Instead of one hot encoding, rare encoding is a better solution for these variables. One hot encoded variables can be seen on Figure 15.

```
ohe_cols
```

```
['food_family',
 'sales_country',
 'education',
 'member_card',
 'occupation',
 'avg. yearly_income',
 'store_type',
 'store_state',
 'unit_sales(in millions)',
 'total_children',
 'avg_cars_at home(approx)',
 'num_children_at_home',
 'avg_cars_at home(approx).1']
```

Figure 15. One hot encoded variables

food_family_Drink	food_family_Food	food_family_Non-Consumable	sales_country_Canada	sales_country_Mexico	sales_country_USA	ed
0	1	0	0	0	1	
0	1	0	0	0	1	
0	1	0	0	0	1	
0	1	0	0	0	1	
0	1	0	0	0	1	

Figure 16. One hot encoding

3.4 PCA

Visualization the data in 2 dimensions

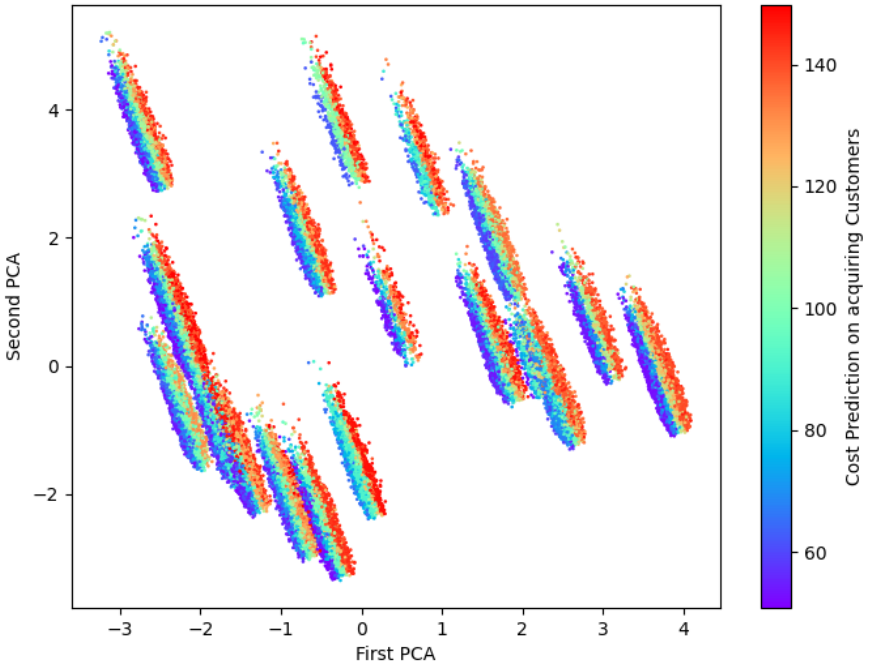


Figure 17. 2D PCA

4.Modeling

Aim of this project is to explore the data set and predict the “cost” dependent variable for regression which means how much a customer would pay for her/his shopping, using regression models and sales_country feature as a target for the classification

4.1 Classification

For classification, we predicted the "sales_country" dependent variable by using gradient boosting algorithms Decision Tree, Gradient Boosting and Random Forest.

Before modeling, we did feature engineering as encoding and dropping unnecessary cardinal columns. For encoding, one hot encoder and label encoder were used.

4.1.1 Model Evaluation

Classification metrics such as accuracy, recall, precision and F1 scores can be seen below. Overall some models were overfitted to data. We did grid search cross validation for the reason for the overfitting problem, and we saw that some parameters such as max_depth and n_estimators were the reason for the overfitting. Also Feature selection, using validation set in addition to train and test set, would be great solutions, too.

Decision Tree can be seen on Figure 1.

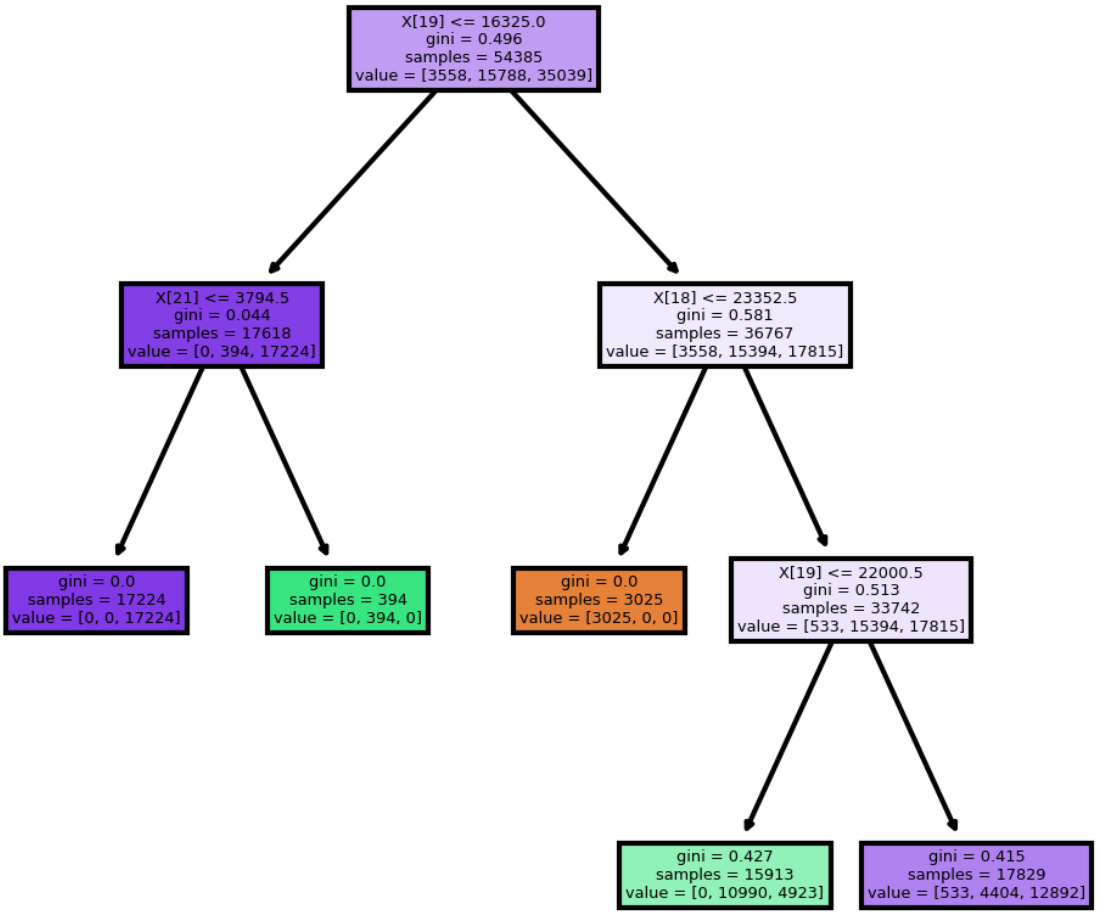


Figure 18. Decision Tree

For classification Accuracy, Recall, Precision and F1 Score metrics are used to evaluate the models. Results before cross validation can be seen below.

Before Grid Search CV:
DecisionTreeClassifier:
Accuracy: 82.0453%
Recall : 0.82045
Precision : 0.8234434
F1 Score : 0.8219457

RandomForestClassifier:

Accuracy: 98.9906%
Recall : 0.989
Precision : 0.990062996827962
F1 Score : 0.9899843301582946

GradientBoostingClassifier:
Accuracy: 100.0000%
Recall : 1.0
Precision : 1.0
F1 Score : 1.0

After GridSearch Cross Validation, we could see that, some parameters such as max_depth and n_estimators was the reason of the overfitting which can be seen on Figure 3.

```
({0: "mean:0.8819711279470747params{'criterion': 'gini', 'max_depth': 3, 'max_features': 'auto', 'n_estimators': 100}",
  1: "mean:0.8819711279470747params{'criterion': 'gini', 'max_depth': 3, 'max_features': 'auto', 'n_estimators': 150}",
  2: "mean:0.8819711279470747params{'criterion': 'gini', 'max_depth': 3, 'max_features': 'auto', 'n_estimators': 200}",
  3: "mean:0.8819711279470747params{'criterion': 'gini', 'max_depth': 3, 'max_features': 'auto', 'n_estimators': 250}",
  4: "mean:0.8819711279470747params{'criterion': 'gini', 'max_depth': 3, 'max_features': 'auto', 'n_estimators': 300}",
  5: "mean:0.8819895146939075params{'criterion': 'gini', 'max_depth': 3, 'max_features': 'auto', 'n_estimators': 350}",
  6: "mean:0.8829089037634912params{'criterion': 'gini', 'max_depth': 3, 'max_features': 'auto', 'n_estimators': 400}",
  7: "mean:0.8819711279470747params{'criterion': 'gini', 'max_depth': 3, 'max_features': 'auto', 'n_estimators': 450}",
  8: "mean:0.9795715006787273params{'criterion': 'gini', 'max_depth': 5, 'max_features': 'auto', 'n_estimators': 100}",
  9: "mean:0.9840029551624166params{'criterion': 'gini', 'max_depth': 5, 'max_features': 'auto', 'n_estimators': 150}",
  10: "mean:0.983175563726217params{'criterion': 'gini', 'max_depth': 5, 'max_features': 'auto', 'n_estimators': 200}",
  11: "mean:0.9818331739950511params{'criterion': 'gini', 'max_depth': 5, 'max_features': 'auto', 'n_estimators': 250}",
  12: "mean:0.9843891437879467params{'criterion': 'gini', 'max_depth': 5, 'max_features': 'auto', 'n_estimators': 300}",
  13: "mean:0.9813551601626024params{'criterion': 'gini', 'max_depth': 5, 'max_features': 'auto', 'n_estimators': 350}",
  14: "mean:0.9841501262218525params{'criterion': 'gini', 'max_depth': 5, 'max_features': 'auto', 'n_estimators': 400}",
  15: "mean:0.9837087195422444params{'criterion': 'gini', 'max_depth': 5, 'max_features': 'auto', 'n_estimators': 450}",
  16: "mean:1.0params{'criterion': 'gini', 'max_depth': 7, 'max_features': 'auto', 'n_estimators': 100}",
  17: "mean:1.0params{'criterion': 'gini', 'max_depth': 7, 'max_features': 'auto', 'n_estimators': 150}",
  18: "mean:1.0params{'criterion': 'gini', 'max_depth': 7, 'max_features': 'auto', 'n_estimators': 200}",
  19: "mean:0.9999264530126685params{'criterion': 'gini', 'max_depth': 7, 'max_features': 'auto', 'n_estimators': 250}",
  20: "mean:1.0params{'criterion': 'gini', 'max_depth': 7, 'max_features': 'auto', 'n_estimators': 300}",
  21: "mean:1.0params{'criterion': 'gini', 'max_depth': 7, 'max_features': 'auto', 'n_estimators': 350}",
  22: "mean:1.0params{'criterion': 'gini', 'max_depth': 7, 'max_features': 'auto', 'n_estimators': 400}",
  23: "mean:1.0params{'criterion': 'gini', 'max_depth': 7, 'max_features': 'auto', 'n_estimators': 450}"},
 {'best_mean': 1.0,
  'best_param': {'criterion': 'gini',
   'max_depth': 7,
   'max_features': 'auto',
   'n_estimators': 100}})
```

Figure 19. Random Forest GridSearch CV

4.1.2 Feature Importance

grocery_sqft, store_sqft, salad_bar, coffee_bar, meat_sqft features were the most important feature for both gradient boosting and random forest algorithm. grocery_sqft, store_sqft features' importance were high for decision tree algorithm.

4.2 Regression

For regression, we predicted the "cost" dependent variable by using gradient boosting algorithms XGBoost and LightGBM.

Before modeling, encoding and scaling applied. For encoding, one hot encoder and label encoder were used. And for the scaling, min max scaler (normalization) applied to the numeric columns to put them on an equal footing.

After these steps, we divided data into train and test sets to see model performance.
For the evaluation metric, Root mean squared error is used. In order to create the best model, we decided to do hyperparameter tuning for the models, and we used GridSearch for it. Before cross validation, Lightgbm's RMSE was 18.001. For the CV we used GridSearchCV to see best parameters of the model. After using the best parameters RMSE decreased to 17.078. XGBoost RMSE score was 22.13 . Overall, Lightgbm is the most successful model and XGBoost is also performed quite well but the algorithm was very slow to train that's why we did not apply hyperparameter tuning.

Also we decided to check results without scaling, and results were 5.2 and we thought that our model overfitted therefore we decided to apply scaling to prevent our model from overfitting.

In order to see the most important feature for our model, we used the feature importance method which can only be used by ensemble methods. Feature importance of LightGBM can be seen on Figure 20 and Feature importance of XGBoost can be seen on Figure 21.

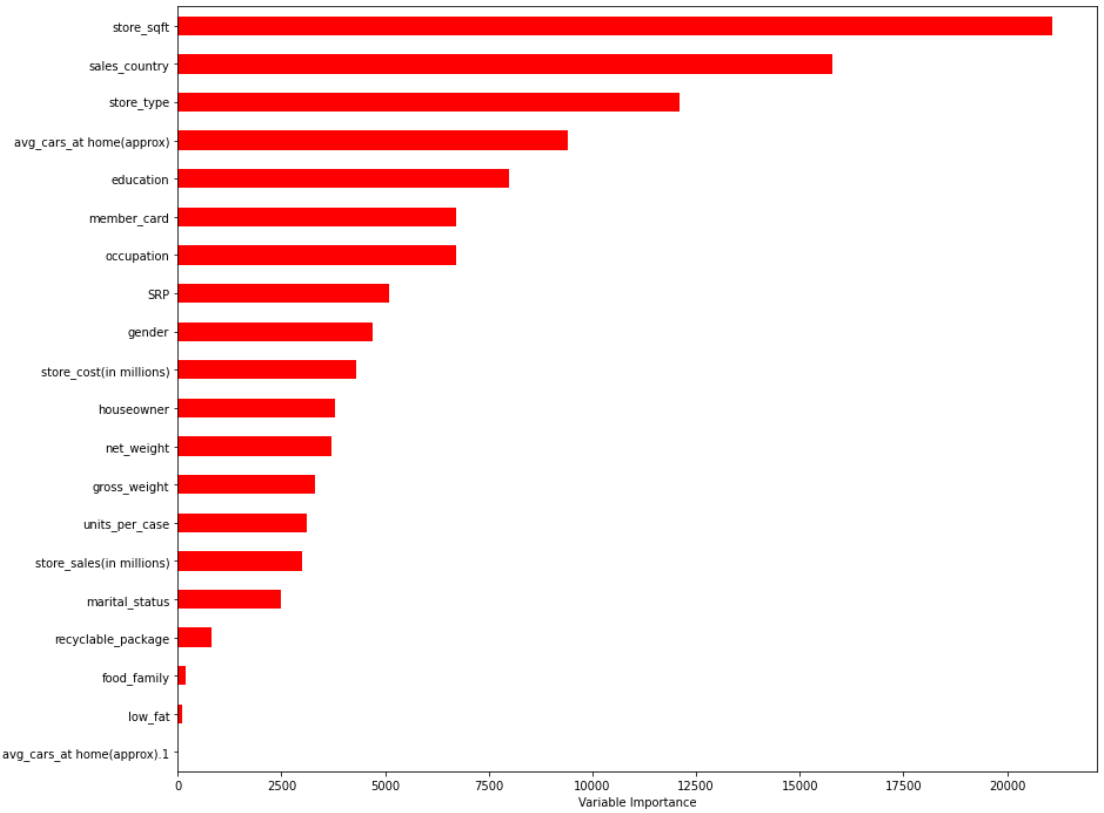


Figure 20. Lightgbm Feature importance

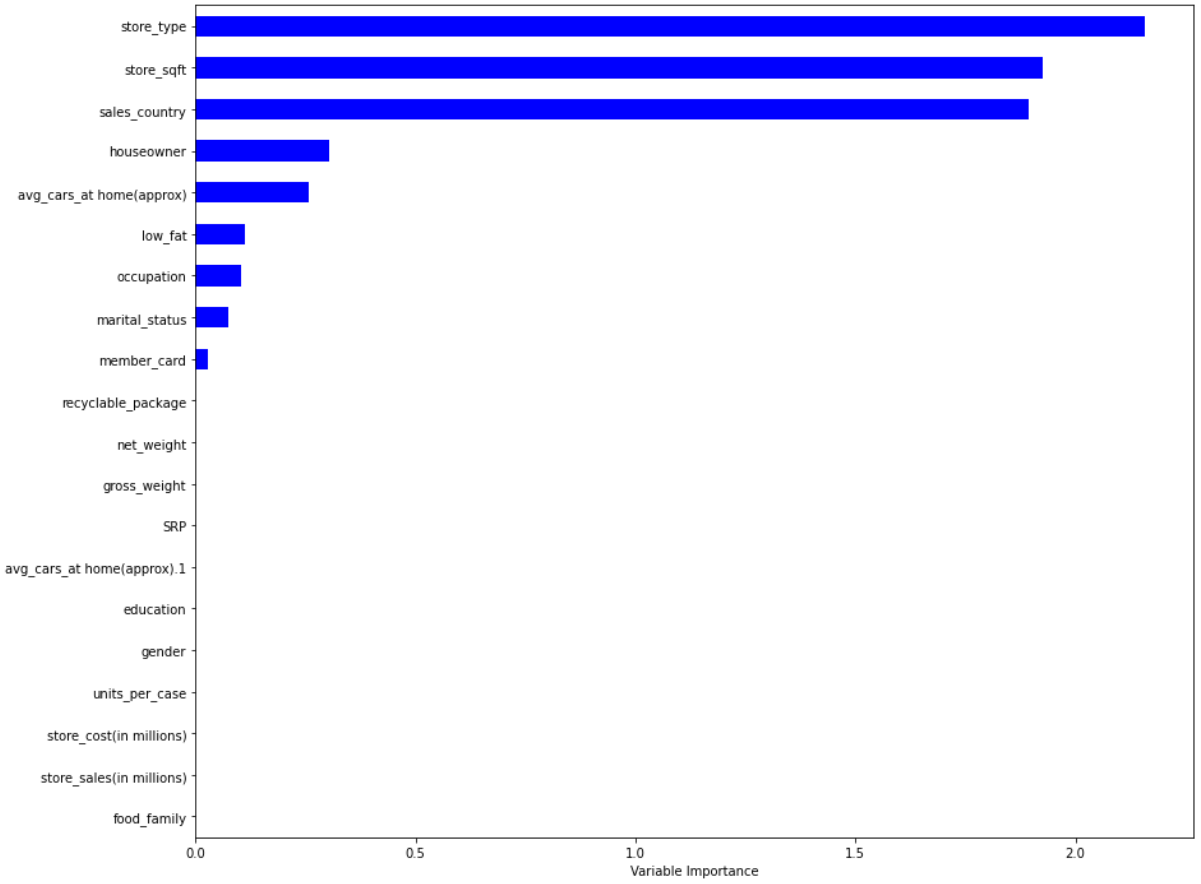


Figure 21. XGBoost Feature importance

5.Clustering

In the final task, for clustering we used K-Means, HCA and DBSCAN algorithms. For classification with ensemble methods, we used the houseowner feature as target.

5.1 K-Means

For K-means clustering, The most important parameter for the Kmean algorithm is number of clusters. We used the elbow method for deciding the best parameters with k values between 2 and 20 we got elbow at k=7. This method calculates the variance between data points within a cluster using the Sum of Squared Error. The best value of k to select is the point of inflection on the curve.

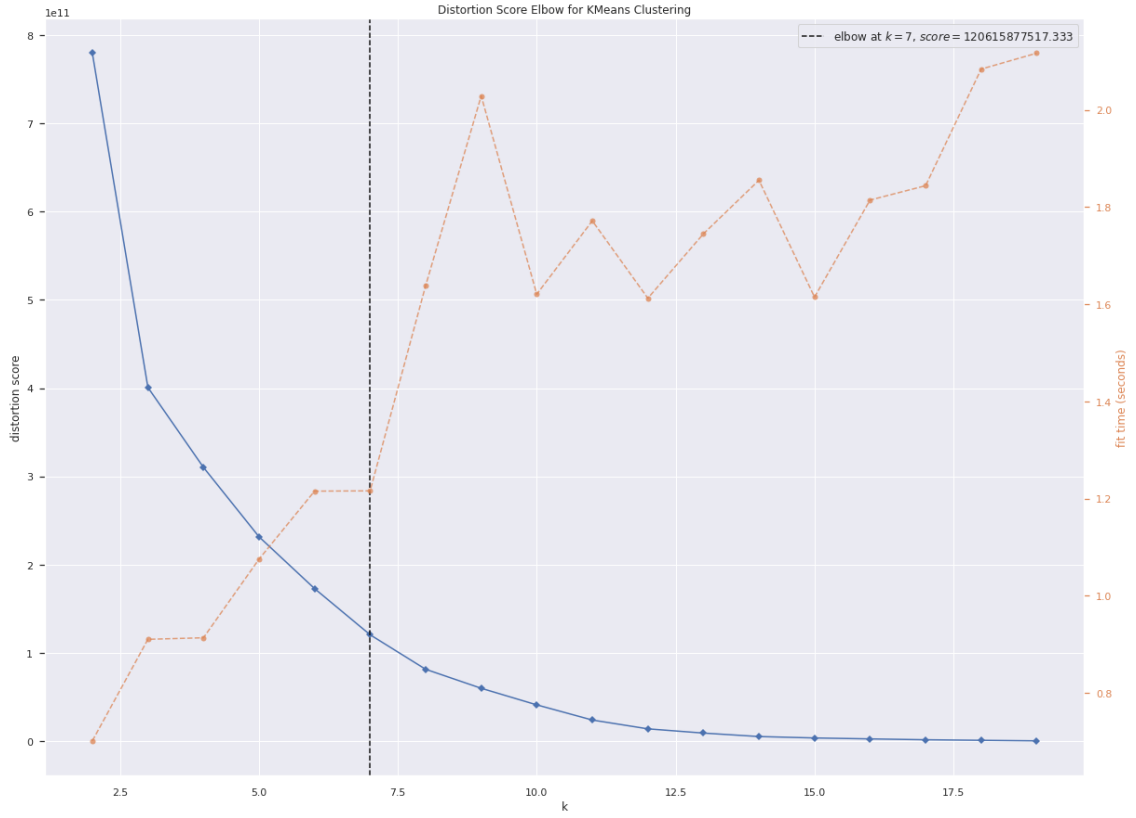


Figure 22. k means elbow score

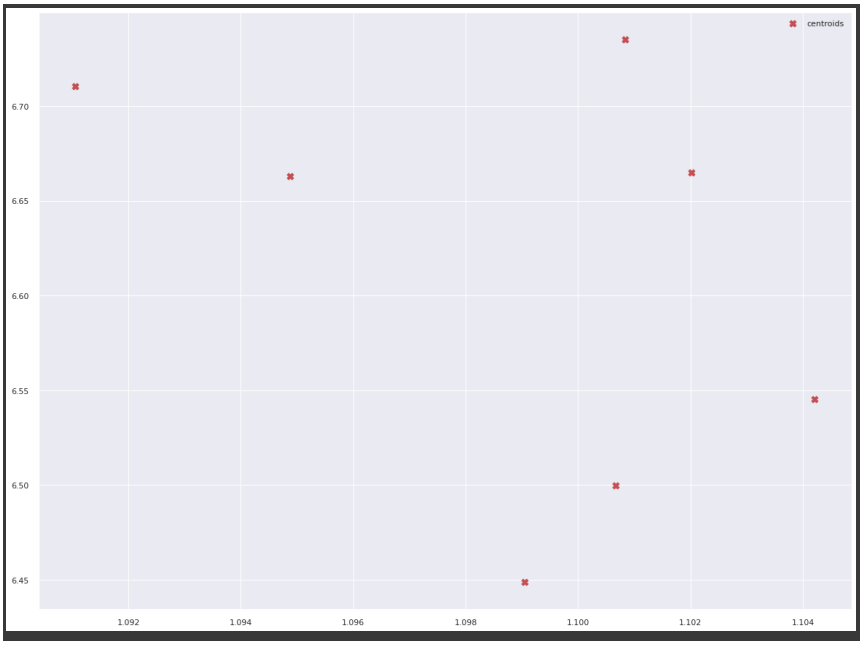


Figure 23. K Means cluster centers

Cluster sizes after applying algorithm :

1	4923
2	12178
3	4641
4	3108
5	19465
6	6381
7	3689

5.2 HCA (Hierarchical Clustering)

Hierarchical Clustering (HCA) is an algorithm which is easy to implement. There are two approaches, Agglomerative and Divisive. Agglomerative approach starts off with each data point being its own group. Next, the two groups closest to each other become a single group. It uses some distance methods like Euclidean, Manhattan, etc distances. Unlike Kmeans we did not have to choose a value. Divisive approach starts off splitting the whole data recursively until individual data have been split into singleton clusters.

Single linkage uses the shortest distance between groups, while complete linkage would use the largest distance between a group. Dendograms can be seen below in Figure 23 and 24.

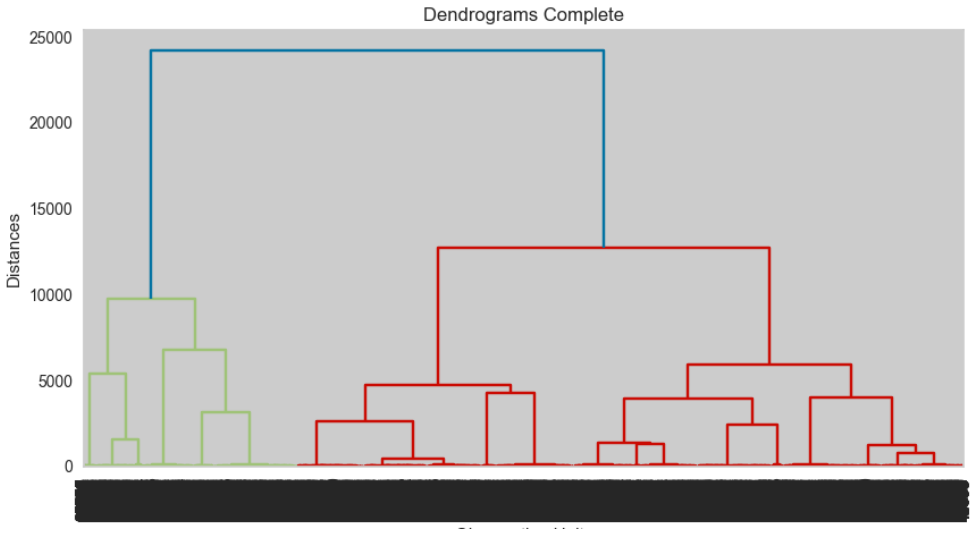


Figure 24. Complete linkage dendogram

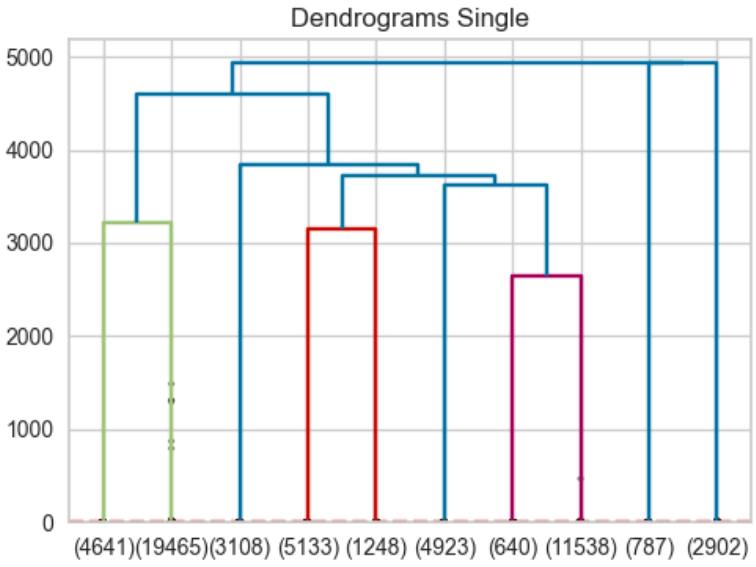


Figure 25. Single linkage dendogram

5.3 DBSCAN

For detecting outliers and anomalies in our dataset DBSCAN(density-based spatial clustering of applications with noise) is the most efficient. The two determining parameters of DBSCAN are the eps and min_samples. The eps is the distance that determines a data point's neighbour. Points are neighbours if the distance between them is less than or equal to eps. To determine the right value of eps, we used K-means results.

First, we applied DBSCAN algorithm to net_weight,cost, units_per_case columns.
After trying some values, the optimal value for the eps is 1.538 which gives us 7 clusters.
Cluster -1 is outliers. 124 outliers were detected.

	Cluster
0	36733
2	14170
1	9374
-1	124
4	15
5	5
6	4
3	3

Figure 26. DBSCAN Clusters

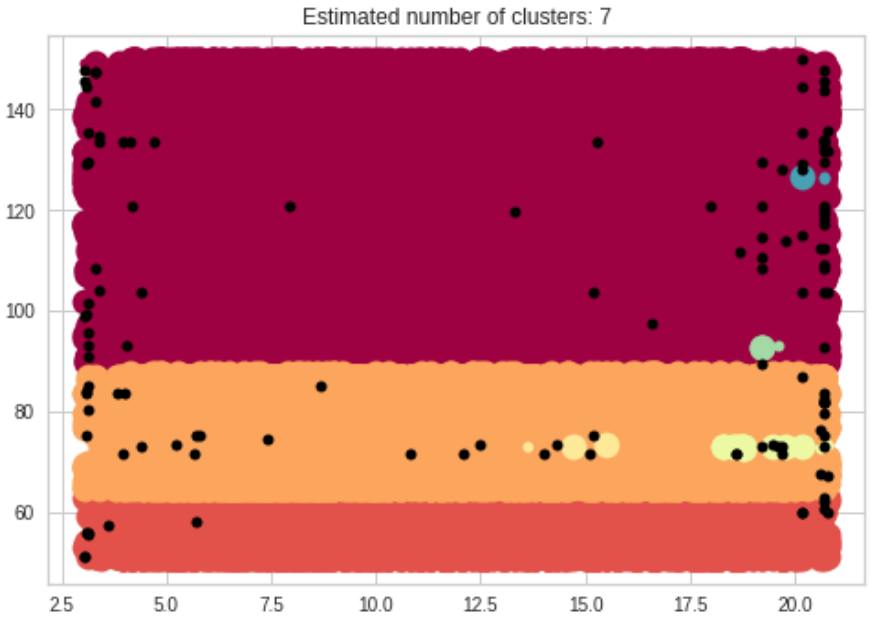


Figure 27. DBSCAN Cluster Visualization

6. Ensemble Learning

6.1 Classification with Decision Tree

For classification, we predicted the "homeowner" variable which contains two classes (yes,no) by using Decision Tree algorithm.

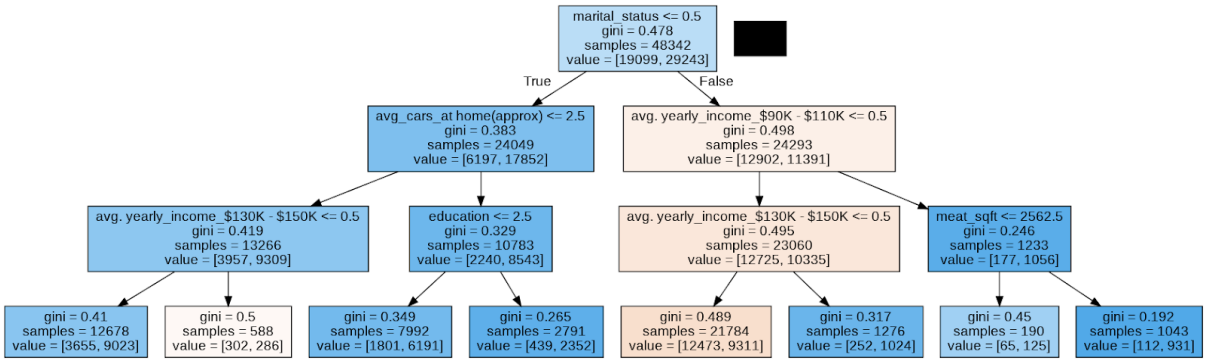


Figure 28. Decision Tree

Decision Tree classification results:
Accuracy: 0.6740
Recall: 0.6744
Precision: 0.7569

6.2 Applying Bagging to DT

BaggingClassifier() method will follow all the bagging steps and build an optimized model. The BaggingClassifier will fit the weak/base learners on the randomly sampled subsets. Next, it will use the voting techniques to produce an aggregated final model. Finally, we used the DecisionTreeClassifier algorithm as our weak/base learner. Bagging classifier did not improve the model's accuracy score.

Accuracy: 0.67334

6.3 Applying Adaboost to DT

AdaBoostClassifier uses boosting technique by assigning higher weights to misclassified data, lower weights to correctly classified data and sampling again to train the next classifier. In this project as a base estimator for Adaboost algorithm, decision tree algorithm is used. Boosting technique helped in reducing variance, as a result our accuracy score increased.

Model test Score: 0.683, Model training Score: 0.682

After cross validation and getting the best parameter for the adaboost, results got better.
Final results:
Accuracy: 0.6955
F1 Score: 0.7434
ROC-AUC: 0.752

6.4 Random Forest

Random Forest algorithm has several decision trees trained on the different subsets. Random Forest uses bagging underneath to sample the dataset with replacement randomly. Random Forest samples both columns and rows. It also follows the bagging steps to create an aggregated final model. After applying grid search cross validation and getting best parameters for our Random forest model, results can be seen below.

Accuracy: 0.714
F1: 0.762
ROC AUC: 0.8020

6.5 Results

Finally, the Random Forest algorithm gave the best results for our project with the 0.80 Roc-auc score and 0.714 accuracy. After that, Adaboost also gave slightly better results than baggingclassifier. Performance-wise Random Forest was the easiest and fastest approach and it also gave the best results.

7. Conclusion

There are several methods for encoding, scaling, modeling. After understanding data, We tried to choose best methods for encoding and scaling such as label encoding, one hot encoding, rare encoding and min max scaling (normalization). After modeling with algorithms such as decision tree, xgboost and lightgbm we had to deal with overfitting, we found best parameters to avoid overfitting by using grid search cross validation method. Also we did clustering by using different methods such as, K-means, Hierarchical and DBSCAN. Finally, we applied classification by using bagging classifier, adaboost classifier and random forest for ensemble learning. We realized the difference between those algorithms as we compared the differences of them.