Deree
School of Graduate and Professional Education

THE AMERICAN COLLEGE OF GREECE
Pierce
Deree
Alba
1875

SUBJECT: APPLIED MACHINE LEARNING - ITC 6103

MASTER OF SCIENCE (MS) IN DATA SCIENCE

Students

Lykos Stamatios

Prozymas Konstantinos

Kollarou Elisavet-Agapi

Professor: Dr. Eleni Chatzimichali

1.Classification

The dataset has a list of people and several attributes that we will use to train our model. The goal of our model is to predict whether an individual is earning more or less than 50K a year.

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

Utilizing the .info() method yields the following result:

```
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64
 5   marital-status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64
 11  capital-loss    32561 non-null  int64
 12  hours-per-week  32561 non-null  int64
 13  native-country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

The dataset comprises 9 categorical attributes—namely workclass, education, marital-status, occupation, relationship, race, sex, native-country, and income—each represented as an object data type. Meanwhile, numerical attributes such as age, fnlwgt, education-num, capital-gain, capital-loss, and hours-per-week are encoded as int64 data type.

Using the .info() method in both files, we observed that there are zero non-null values, indicating a complete dataset. But we can see some '?' in our data sets.

In order to fix this we used the .replace() method to replace occurrences of '?' with NaN. By using the .isna().any() we can see that this time we have missing values on the columns workclass, occupation and native country.

```
age               False
workclass          True
fnlwgt            False
education         False
education-num     False
marital-status    False
occupation         True
relationship      False
race              False
sex               False
capital-gain      False
capital-loss      False
hours-per-week    False
native-country     True
income            False
dtype: bool
```

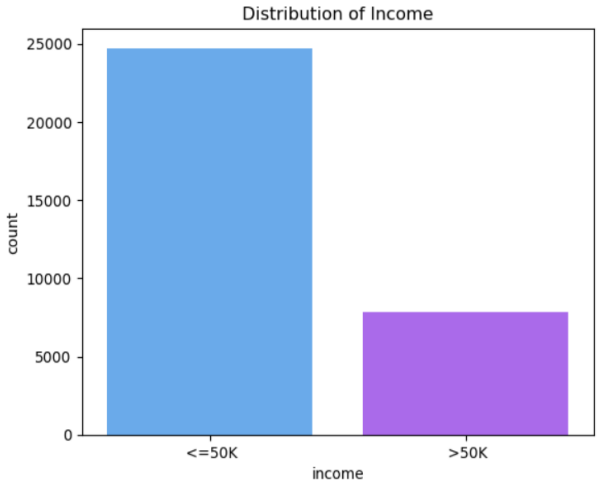
```
age               False
workclass          True
fnlwgt            False
education         False
education-num     False
marital-status    False
occupation         True
relationship      False
race              False
sex               False
capital-gain      False
capital-loss      False
hours-per-week    False
native-country     True
income            False
dtype: bool
```

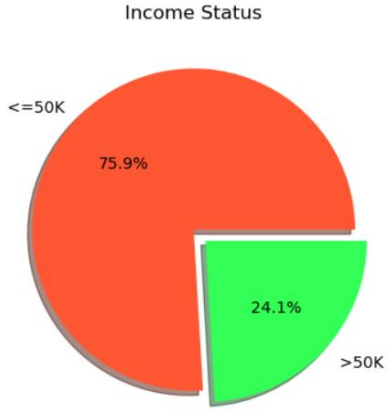
To address the presence of missing values within the dataset, we employed the 'SimpleImputer' tool. Utilizing this approach on both the training and testing sets, we opted to replace NaN values with the most frequently occurring value in each respective column.

EXPANATORY DATA ANALYSIS & GROUPINGS

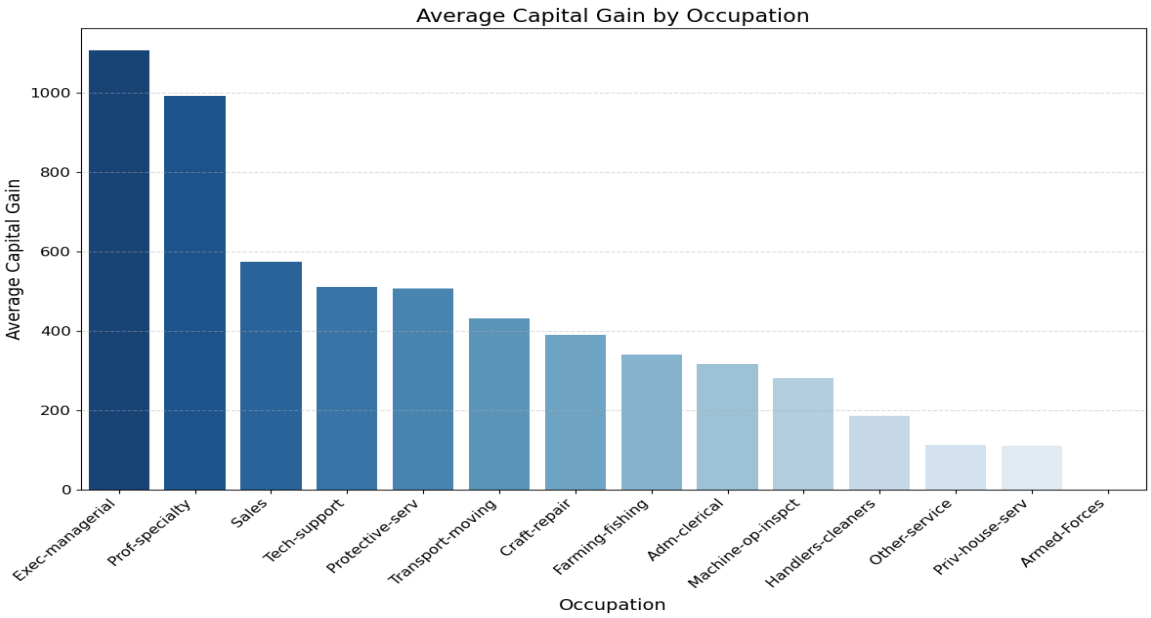
Further exploration into our dataset reveals a notable trend: the majority of adults (24,720 individuals) exhibit an income of less than or equal to 50,000.

This observation is aptly depicted through the following diagrams.

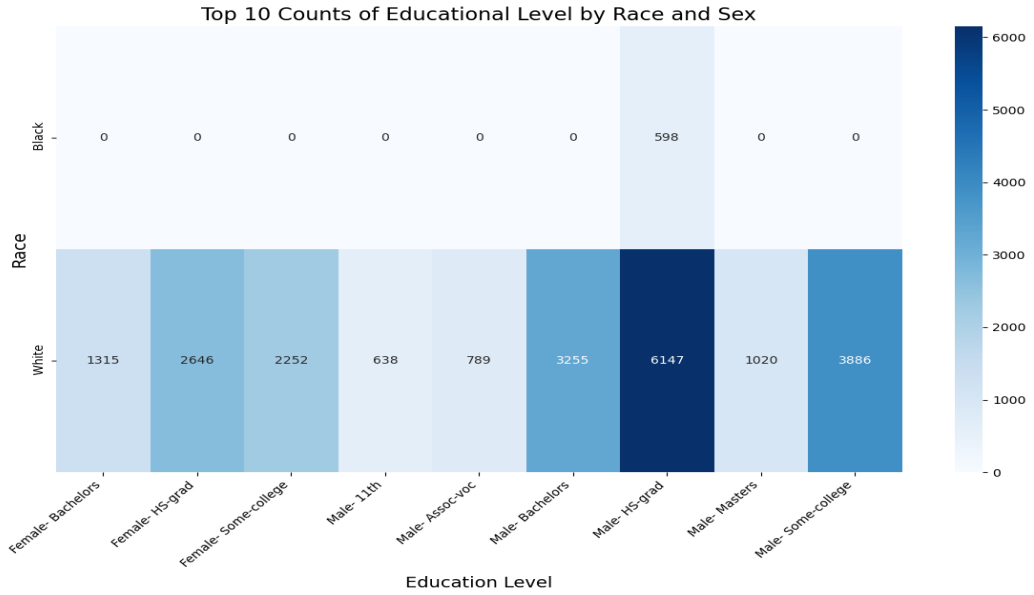




In addition, as it can be seen from the diagram below, we can see that Executive managerial and Proffesion speciality have a big diference in the Average Capital Gain than the rest occupations, which could make them potential outliers.

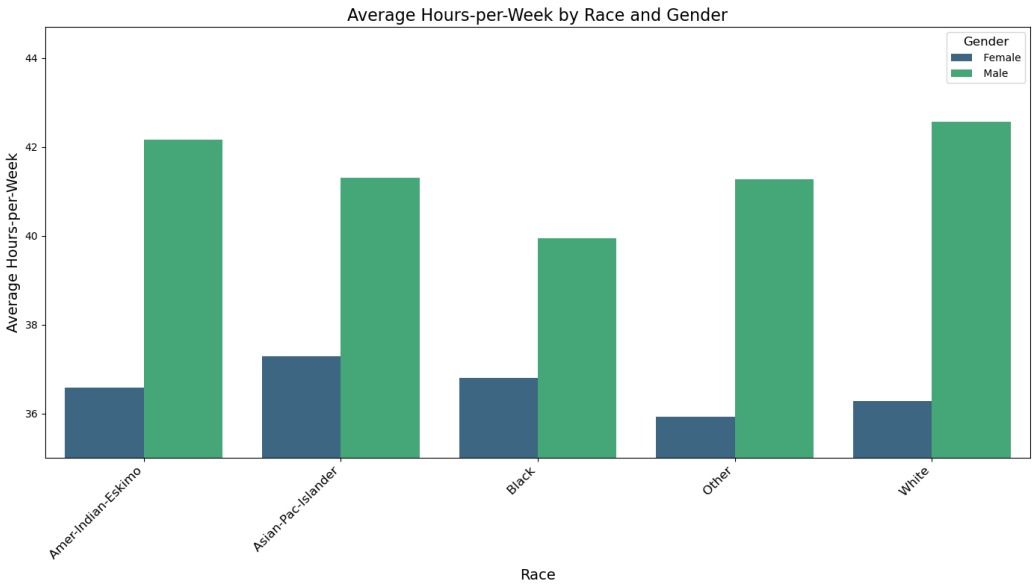


Furthermore, the following heatmap visualizes the distribution of educational levels across different races and genders. Each cell represents the count of indivisuals belonging to a specific combination of race, gender and education level.

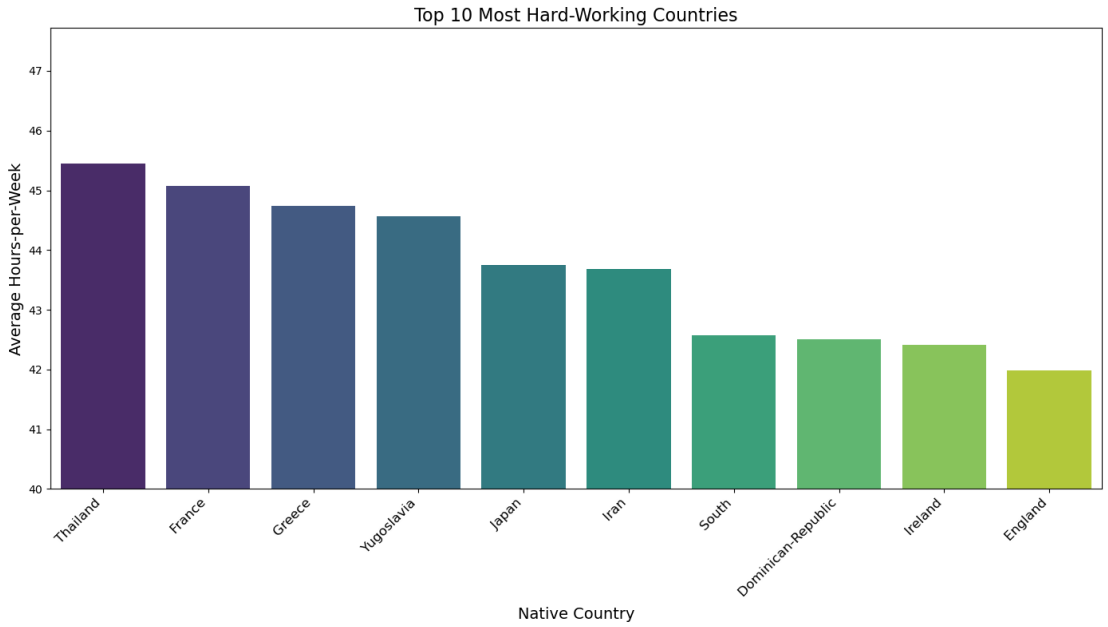


When it comes to males, the majority of counts for both rare highschool graduates. Meanwhile, on our set we have 0 Black females and the majority of White females are as well, highschool graduates.

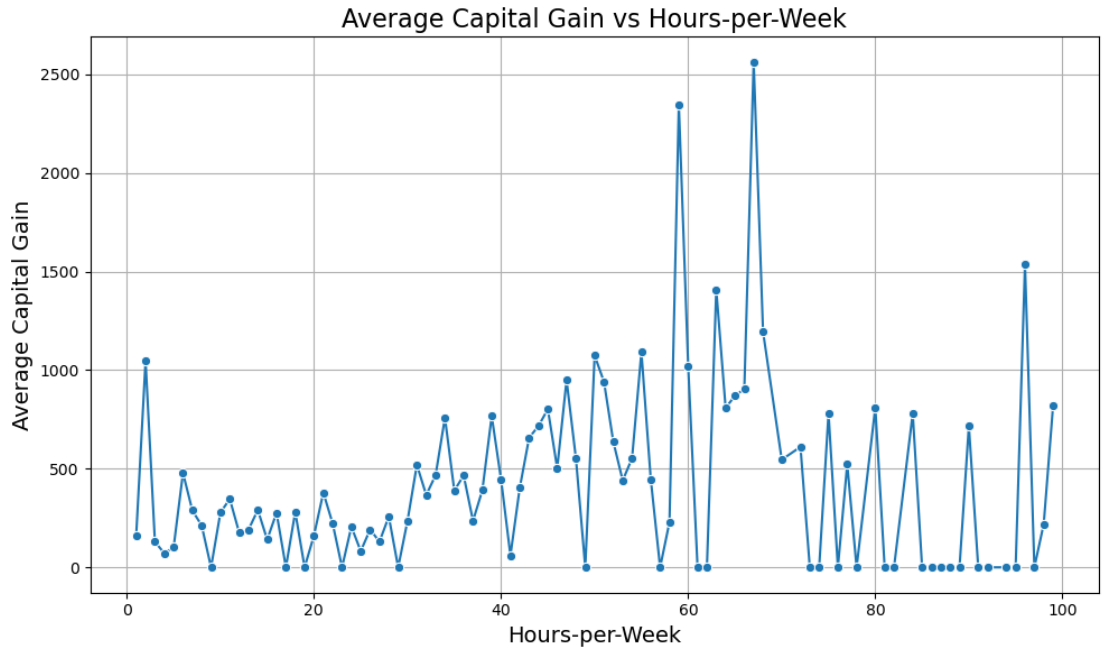
Additonaly, the bar chart below, demonstrates that men work almost 6 hours more than women do on a weekly basis. White men work more that the rest of the men of the other races but Asian women work more hours that the rest of the women.



The next bar charts, shows us the top 10 Most working countries, with number one being Thailand.



Last but not least, the following graph, demonstrates the Capital Gain vs. Working Hours. The Capital Gain maximizes at around 65 hours and its second maximun prize is around 58 working hours.



To incorporate the categorical variables into our analysis, we used One Hot Encoding. This transformation facilitates the conversion of categorical variables into binary vectors, with each category represented as a binary feature column.

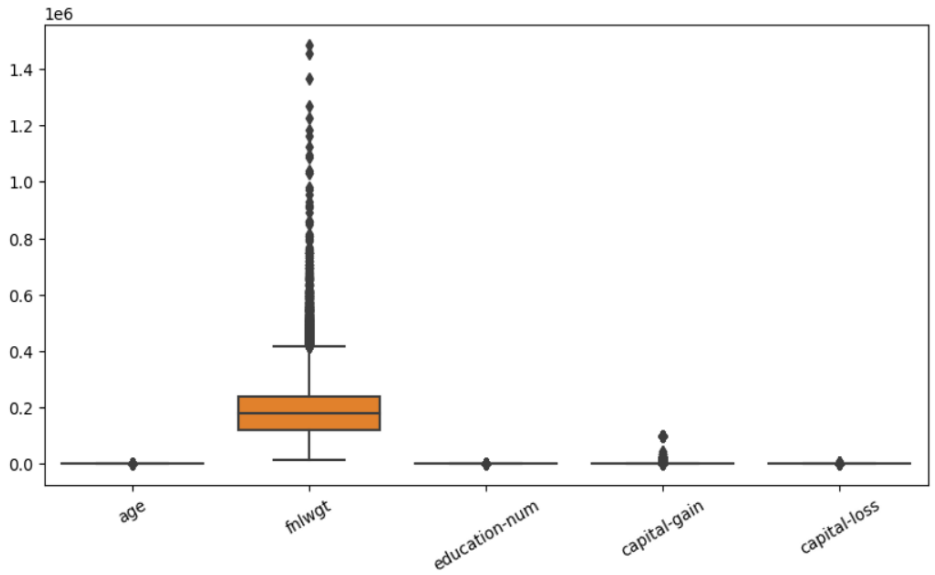
With our data now preprocessed, it's time to split it into sets for training and testing our machine learning models. This separation is crucial, as it enables us to feed the independent variables into the model for training while keeping the dependent variable aside for evaluation.

In this step, we dropped the 'income' column from both X_train and X_test DataFrames, ensuring they contain all the features except the target variable. On the other hand, y_train and y_test exclusively contain the 'income' column, serving as the dependent variable we aim to predict.

This separation ensures that our data is appropriately structured for training and testing machine learning models, facilitating accurate predictions and evaluation of model performance.

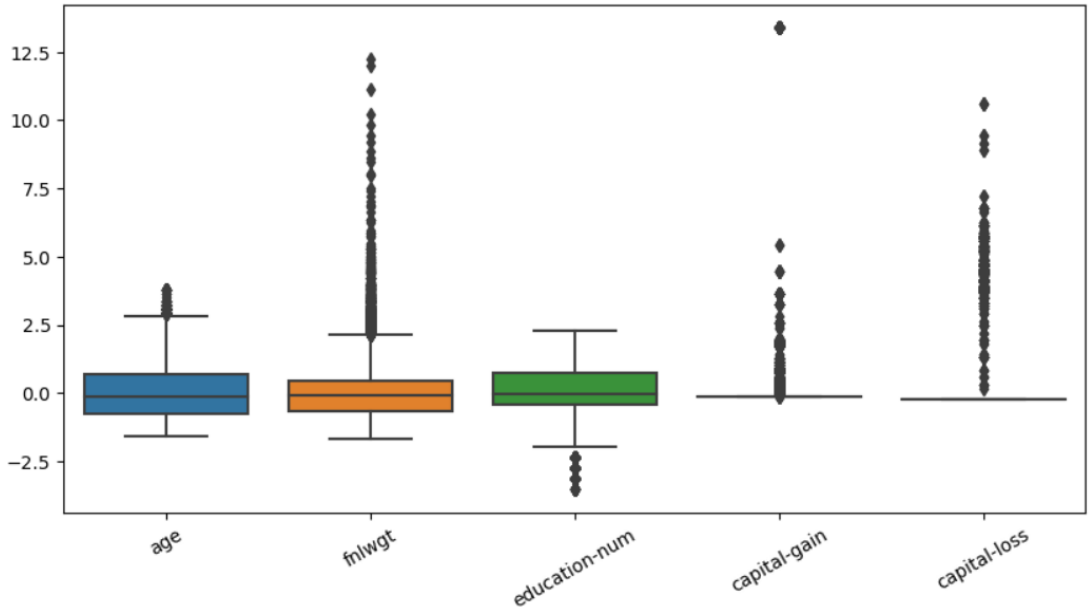
Scaling

Boxplot 1: Before Scaling



As it can be seen, if we did not proceed with the scaling of our data, fnlwgt would dominate our research.

Boxplot 2: After Scaling



The changes are expected and reflect the new distributions of the feautures after standardization.

After splitting our data into training and testing sets, we proceed with classifiers such as Random Forest (RF), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), linear SVM, Logistic Regression and Neural Networks to build predictive models. Finally, we decided to use Logistic Regression and Neural networks which are both very different methods, the one being great for simple problems with linear relationships while the other being greater for nonlinear and complex problems.

Random Forest

Accuracy for Random Forest is 0.85

	precision	recall	f1-score	support
0	0.88	0.93	0.90	12435
1	0.71	0.60	0.65	3846
accuracy			0.85	16281
macro avg	0.80	0.76	0.78	16281
weighted avg	0.84	0.85	0.84	16281

```
Best Parameters using grid search:
 {'n_estimators': 90, 'max_features': 'sqrt', 'max_depth': 20}
```

We see that the model performed well when predicting the values of class 0 but not quite well predicting class 1.

We then used the Optimal Random Forest, to assess the performance of the Random Forest classifier with optimized hyperparameters on the test set and the results are the following:

Test set accuracy: 0.862

	precision	recall	f1-score	support
0	0.88	0.95	0.91	12435
1	0.77	0.59	0.67	3846
accuracy			0.86	16281
macro avg	0.83	0.77	0.79	16281
weighted avg	0.86	0.86	0.86	16281

Overall, as expected the optimal Random Forest seems to perform better as it achieved higher test accuracy, better precision for both classes, improved recall, and f1-score.

K- Nearest Neighbors

Test set accuracy: 0.825

	precision	recall	f1-score	support
0	0.87	0.90	0.89	12435
1	0.65	0.56	0.60	3846
accuracy			0.82	16281
macro avg	0.76	0.73	0.75	16281
weighted avg	0.82	0.82	0.82	16281

We have a lower accuracy with an even lower F1 score.

Next, we utilized the optimized KNN classifier to evaluate its performance on the test set. Below the results:

```
knn_param_grid = dict(n_neighbors = np.arange(1, 21, 5),
                      weights = ['uniform', 'distance'],
                      metric = ['minkowski','euclidean','manhattan'])

knn_grid_cv = GridSearchCV(KNeighborsClassifier(),
                           knn_param_grid,
                           cv=5)

knn_grid_cv.fit(X_train, y_train)

print("Best parameters: n_neighbors=", knn_grid_cv.best_params_['n_neighbors'])
print("Best parameters: weights =", knn_grid_cv.best_params_['weights'])
print("Best parameters: metric =", knn_grid_cv.best_params_['metric'])
```

```
Best parameters: n_neighbors= 16
Best parameters: weights = uniform
Best parameters: metric = minkowski
```

Next, we utilized the optimized KNN classifier to evaluate its performance on the test set. Below the results:

```
Test set accuracy:  0.836

              precision    recall  f1-score   support

           0       0.86      0.93      0.90     12435
           1       0.71      0.52      0.60      3846

    accuracy                           0.84     16281
   macro avg       0.79      0.73      0.75     16281
weighted avg       0.83      0.84      0.83     16281
```

We observe slightly better performance especially during classes' 1 precision,

---

Support Vector Machine

```
Accuracy for RBF SVM is: 0.851

              precision    recall  f1-score   support

           0       0.88      0.94      0.91     12435
           1       0.74      0.57      0.64      3846

    accuracy                           0.85     16281
   macro avg       0.81      0.75      0.77     16281
weighted avg       0.84      0.85      0.84     16281
```

Similarly we did with all classifiers and we will analyze below

SVM with linear kernel

Accuracy for Linear SVM is: 0.849

	precision	recall	f1-score	support
0	0.87	0.94	0.90	12435
1	0.74	0.57	0.64	3846
accuracy			0.85	16281
macro avg	0.81	0.75	0.77	16281
weighted avg	0.84	0.85	0.84	16281

The best parameters for SVM using grid search, are kernel = rbf, gamma = 0.001, c = 100.

The results of Optimal SVM are the following with accuracy 0.854, recall 0.94 and F1 score 0.91.

Logistic Regression

Test set accuracy: 0.854

	precision	recall	f1-score	support
0	0.88	0.93	0.91	12331
1	0.72	0.59	0.65	3656
accuracy			0.85	15987
macro avg	0.80	0.76	0.78	15987
weighted avg	0.85	0.85	0.85	15987

Best Parameters using Randomized Search CV: {'solver': 'liblinear', 'penalty': 'l2', 'C': 0.001}

The results below refer to the optimized model's performance using the best parameters.

Test set accuracy: 0.854

	precision	recall	f1-score	support
0	0.89	0.93	0.91	12331
1	0.71	0.61	0.65	3656
accuracy			0.85	15987
macro avg	0.80	0.77	0.78	15987
weighted avg	0.85	0.85	0.85	15987

Neural Networks

Accuracy for Neural Network is 0.832

	precision	recall	f1-score	support
0	0.88	0.91	0.89	12331
1	0.65	0.59	0.62	3656
accuracy			0.83	15987
macro avg	0.76	0.75	0.75	15987
weighted avg	0.83	0.83	0.83	15987

The best parameters are solver = sgd, learning rate = adaptive, hidden layer sizes = (100.0), alpha = 0.0001, activation = logistic.

Test set accuracy: 0.853

	precision	recall	f1-score	support
0	0.89	0.93	0.91	12331
1	0.72	0.59	0.65	3656
accuracy			0.85	15987
macro avg	0.80	0.76	0.78	15987
weighted avg	0.85	0.85	0.85	15987

```
Test set accuracy:  0.854

              precision    recall  f1-score   support

           0       0.88      0.94      0.91     12331
           1       0.73      0.57      0.64      3656

    accuracy                           0.85     15987
   macro avg       0.81      0.75      0.78     15987
weighted avg       0.85      0.85      0.85     15987
```

Based on the results above and after storing based on both accuracy and f1 score, the following conclusions can be drawn:

	Model	Accuracy	F1
0	RF Opt	0.866	0.665
0	Logistic Regression Opt	0.854	0.654
0	Logistic Regression	0.854	0.647
0	SVM Opt	0.854	0.642
0	SVM rbf	0.854	0.637
0	NN Opt	0.853	0.648
0	SVM Linear	0.852	0.637
0	RF	0.850	0.644
0	KNN Opt	0.841	0.604
0	NN	0.832	0.616
0	KNN	0.832	0.611

Random Forest optimized has the highest accuracy and the highest F1 score. Logistic Regression optimized has the second highest F1 score of 0.654 but its accuracy is lower at 0.854. Overall, based on these metrics, the Random Forest after optimization appears to be the best performing model.

Ensemble

Moving on we created an ensemble classifier that combines multiple machine learning models to potentially improve classification accuracy and robustness.

We used 5 classifiers (KNN, RF, SVM, Logistic Regression, MLP) with the best hyperparameters for each.

The results are the following:

```
The accuracy for ensembled model is: 0.8583223869393882

              precision    recall  f1-score   support

           0       0.88      0.94      0.91     12331
           1       0.75      0.58      0.65      3656

    accuracy                           0.86     15987
   macro avg       0.81      0.76      0.78     15987
weighted avg       0.85      0.86      0.85     15987
```

We note the accuracy is slightly lower than the Optimized Random Forest but its F1 score is much higher, and we feel much more confident with our classification. This ensemble model makes fewer false negatives, even if the lower accuracy suggests that it makes mistakes elsewhere.

2.Clustering

For clustering we are provided with a different dataset, which is called Wholesale distributors data. This dataset has 440 rows and 8 columns.

```
(440, 8)
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

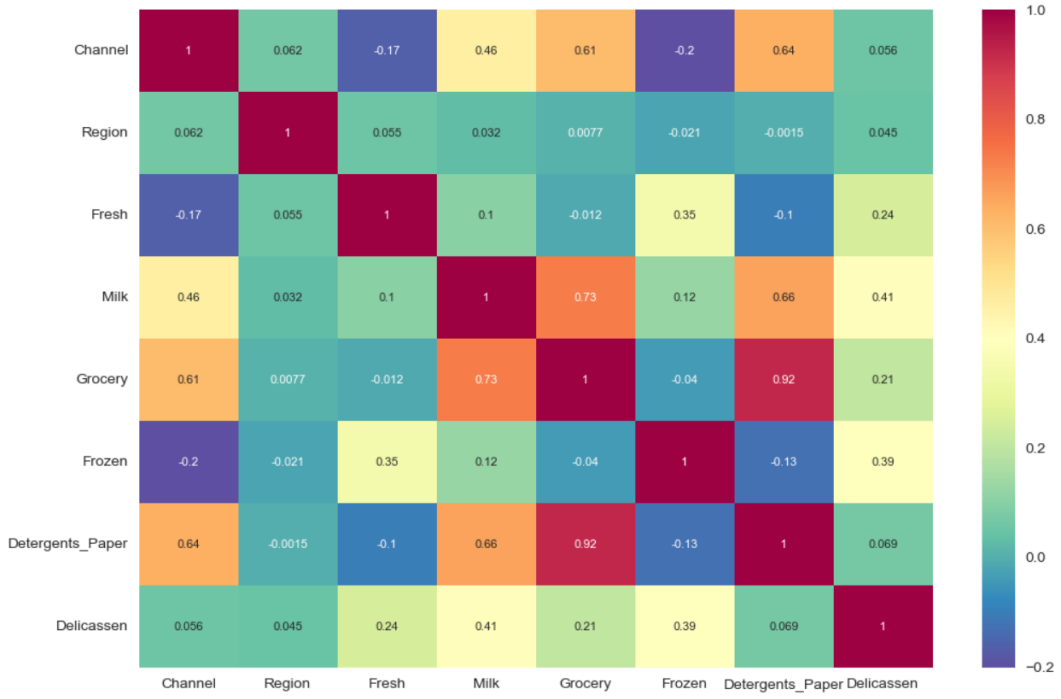
Next, and taking into advantage the .info() we get some information for our dataset. Firstly, we see that all the columns are of the same type (int 64) and secondly there are no missing values.

```
RangeIndex: 440 entries, 0 to 439
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Channel           440 non-null    int64
 1   Region            440 non-null    int64
 2   Fresh             440 non-null    int64
 3   Milk              440 non-null    int64
 4   Grocery           440 non-null    int64
 5   Frozen            440 non-null    int64
 6   Detergents_Paper  440 non-null    int64
 7   Delicassen        440 non-null    int64
dtypes: int64(8)
memory usage: 27.6 KB
```

Then using the .describe() we get some statistics for our dataset, such as the average value, the standard deviation, the minimum and maximum value for each column.

	Channel	Region	Fresh	Milk	Grocery	Frozen
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000
mean	1.322727	2.543182	12000.297727	5796.265909	7951.277273	3071.931818
std	0.468052	0.774272	12647.328865	7380.377175	9503.162829	4854.673333
min	1.000000	1.000000	3.000000	55.000000	3.000000	25.000000
25%	1.000000	2.000000	3127.750000	1533.000000	2153.000000	742.250000
50%	1.000000	3.000000	8504.000000	3627.000000	4755.500000	1526.000000
75%	2.000000	3.000000	16933.750000	7190.250000	10655.750000	3554.250000
max	2.000000	3.000000	112151.000000	73498.000000	92780.000000	60869.000000

Fon instance, the Milk column shows that, on average, customers spend 5796.26$ annually on milk products but there is a range in spending habits. Some customers spend as low as 55$, while others spend up to 73498.00$.



Color Scale: The color intensity in the heatmap represents the correlation coefficient between two features. Warmer colors (reds and oranges) indicate positive correlations, meaning when one value increases, the other tends to increase as well. Conversely, cooler colors (blues and greens) represent negative correlations, where

an increase in one value is associated with a decrease in the other. The closer the color is to red or blue, the stronger the correlation.

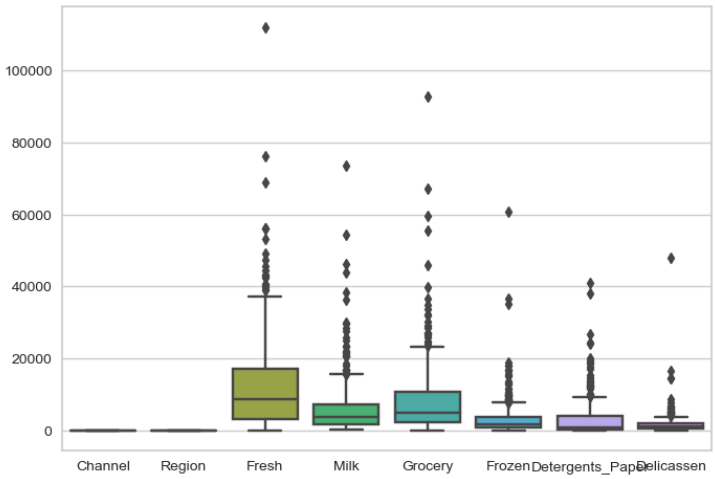
Positive Correlations**:** For example, Grocery and Frozen might indicate that customers who spend more on groceries also tend to spend more on frozen products.

Negative Correlations: There are also negative correlations. For instance, the negative correlation between Channel and Fresh suggests that customers who buy through certain channels tend to spend less on fresh products.

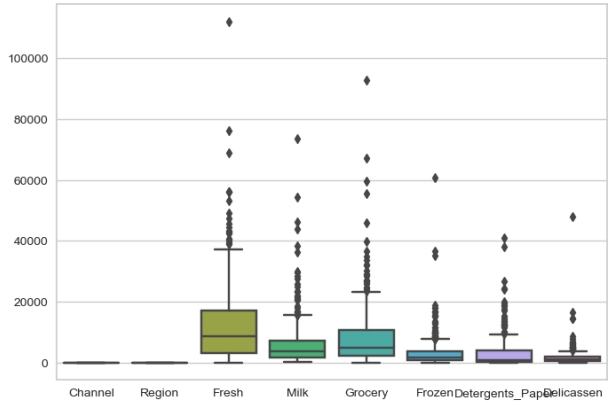
Scaling

Before we went further with our analysis, we performed data scaling using StandardScaler to avoid having the model dominated by a specific attribute with a bigger range.

Boxplot 1: Before Scaling



Boxplot 2: After Scaling



K-Means

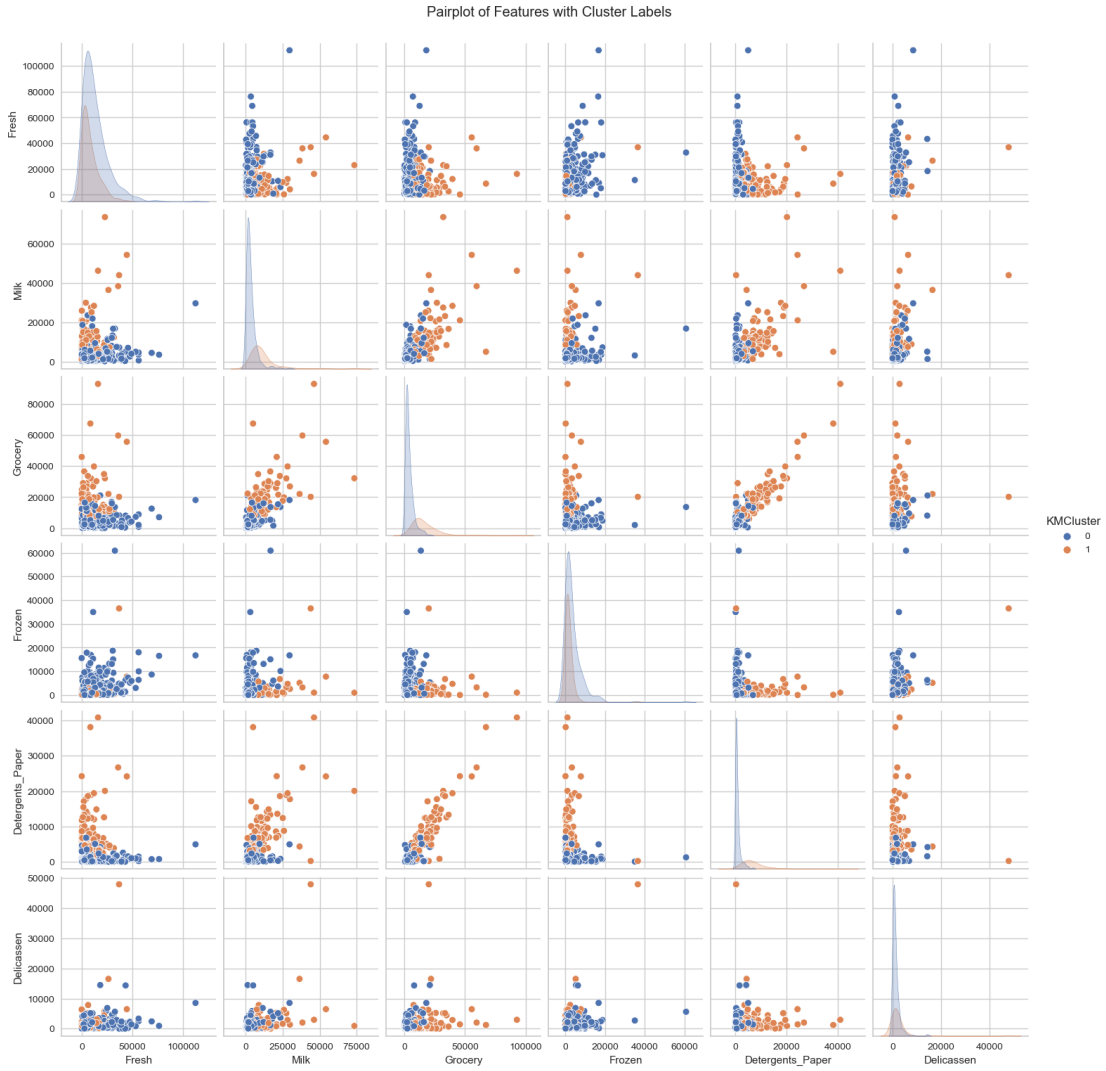
We proceed by performing K-means clustering and group our data into a minimum of 2 clusters. We can notice that most data seem closer to each other and the clustering into two groups works quite well.

```
array([1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0])
```

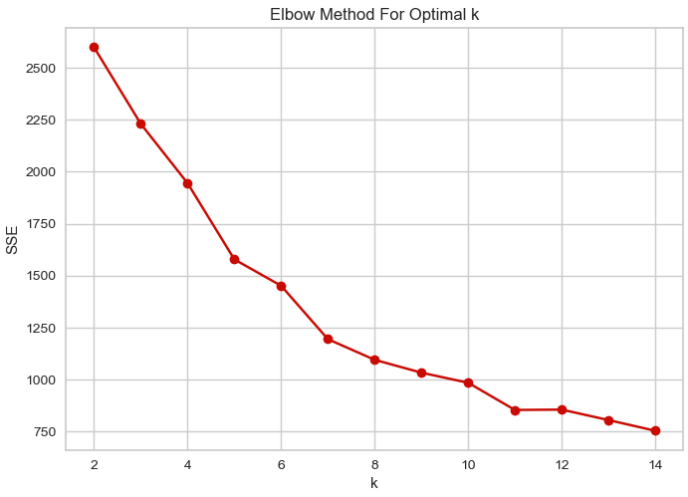
Next, we added a new column, in which we have incorporated the cluster information directly into the DataFrame, to make it easier to analyze and visualize the customer data segmented by their assigned clusters.

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	KMCluster
0	2	3	12669	9656	7561	214	2674	1338	1
1	2	3	7057	9810	9568	1762	3293	1776	1
2	2	3	6353	8808	7684	2405	3516	7844	1
3	1	3	13265	1196	4221	6404	507	1788	0
4	2	3	22615	5410	7198	3915	1777	5185	1
5	2	3	9413	8259	5126	666	1795	1451	1
6	2	3	12126	3199	6975	480	3140	545	1
7	2	3	7579	4956	9426	1669	3321	2566	1
8	1	3	5963	3648	6192	425	1716	750	0
9	2	3	6006	11093	18881	1159	7425	2098	1

This pair plot demonstrates the relationships between features in our data, colored by the cluster labels assigned by K-Means clustering.

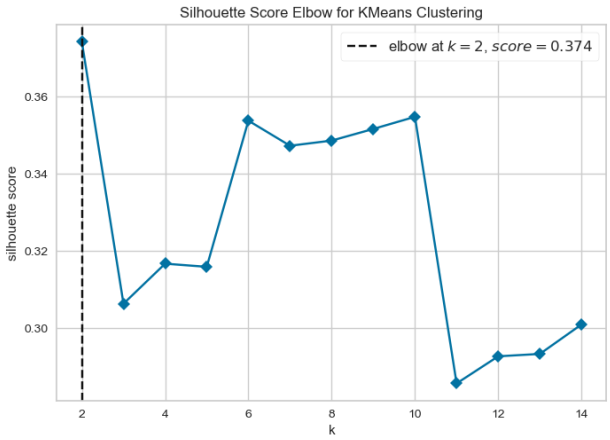


Utilizing the pair plot, in the pair Fresh – Milk, as it can be seen both clusters show a positive correlation, however the slope in cluster 1 is steeper. This means that both groups buy fresh and milk products together, but cluster 1 tends to do it more often.



Through this graph we identify the optimal k value where the total square distance starts to decrease slowly. In this particulars graph, the elbow occurs around k=4. This suggests that 4 clusters may be the optimal number of clusters.

This Silhouette score is used to assess the quality of k-means clustering for different numbers of clusters.



The plot seems that k =2 would be the most well-defined clusters and lies far away from neighboring clusters contrary to k=4, but still not with a great score.

Then, we calculated the centroids of the clusters, that represent the average characteristics of each cluster-based on the features.

```
array([[-0.05206239, -1.50525144, -0.10755472, -0.06250666,  0.01474244,
         0.06691965,  0.0341074 , -0.08660556],
       [ 0.02042954,  0.59066829,  0.04220502,  0.02452793, -0.00578501,
        -0.02625961, -0.01338392,  0.03398446]])
```

PCA

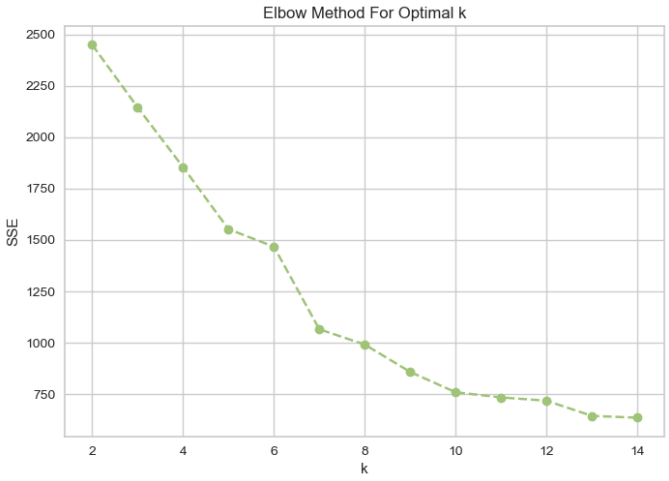
We perform PCA to reduce the dimensionality of our data. The following array displays the variance ratio for each principal component, for example it indicates that the first principal component explains approximately 38.75% of the total variance of our data. The first two components explain 61.12% of our data, so all 6 components explain around 96% of our data.

```
array([0.38750123, 0.61124711, 0.73771884, 0.83001788, 0.89959693,
       0.95701047])
```

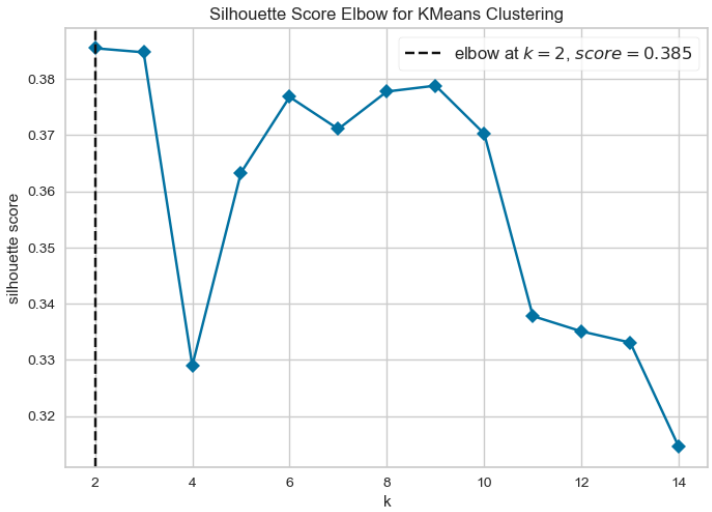
In the table below, the values in each cell represent the coordinates of a particular data point.

	PC1	PC2	PC3	PC4	PC5	PC6
0	0.843939	-0.515351	-0.767632	-0.044215	-0.446234	-0.939441
1	1.062676	-0.484601	-0.672975	0.401372	-0.130458	-0.867227
2	1.269141	0.682055	-0.664095	1.634953	-1.193813	-1.078442
3	-1.056782	0.610821	-0.505654	0.196005	0.457855	0.116959
4	0.634030	0.974199	-0.771209	0.186374	-0.813877	-1.505372

In the first rows we have values like 0.8439, -0.5153 etc. This means that the first data point has a higher score on PC1 and a lower score on PC2.



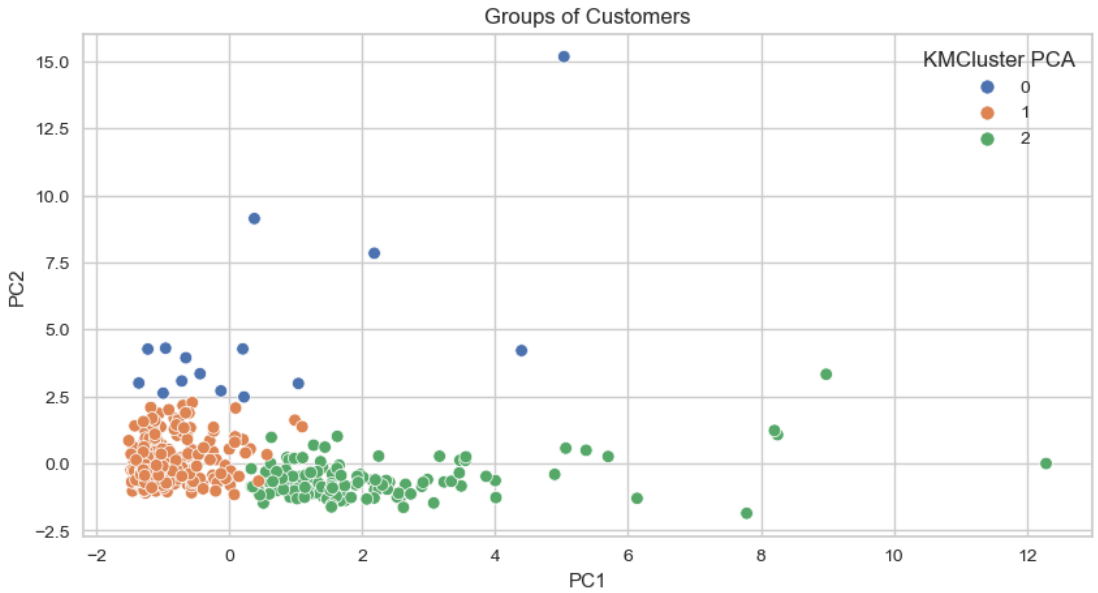
Based on the graph above we see that there are not any very distinct elbow points.



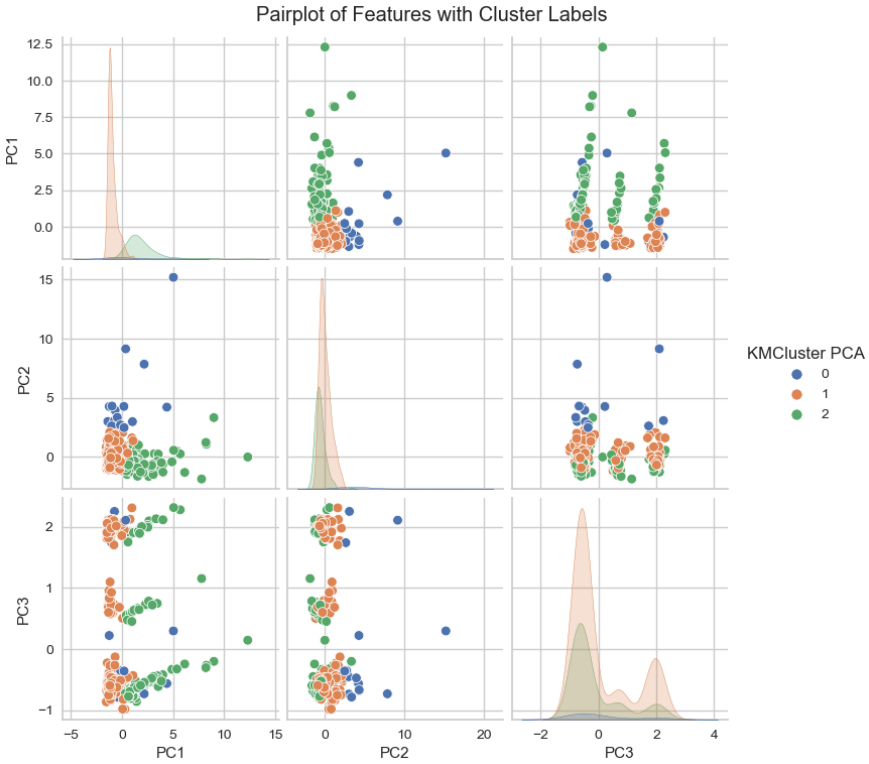
Utilizing the silhouette score we decided to separate our customer into k =3 clusters.

The following table is a result of coding that performs 3 Means Clustering on our data using PCA.

	PC1	PC2	PC3	PC4	PC5	PC6	KMCluster PCA
0	0.843939	-0.515351	-0.767632	-0.044215	-0.446234	-0.939441	2
1	1.062676	-0.484601	-0.672975	0.401372	-0.130458	-0.867227	2
2	1.269141	0.682055	-0.664095	1.634953	-1.193813	-1.078442	2
3	-1.056782	0.610821	-0.505654	0.196005	0.457855	0.116959	1
4	0.634030	0.974199	-0.771209	0.186374	-0.813877	-1.505372	2



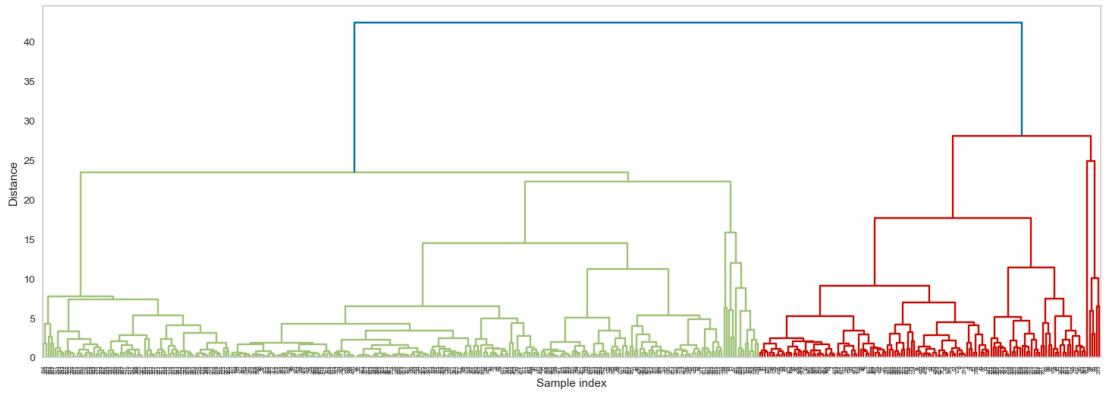
This scatter plot visualizes the above table. It represents how the data points are separated into the 3 clusters.



If we look on the smaller upper left plot, we can see that PC1 is concentrated in the lower left area, while the PC2 is spread across the entire range and PC1 shows a wider spread compared to the other clusters.

HIERARCHICAL CLUSTERING

Dendrogram



Customers are represented by leaves at the bottom of the dendrogram. As we move upwards, clusters are merged based on similarity. We can notice that using hierarchical clustering still the customers break into two clusters.

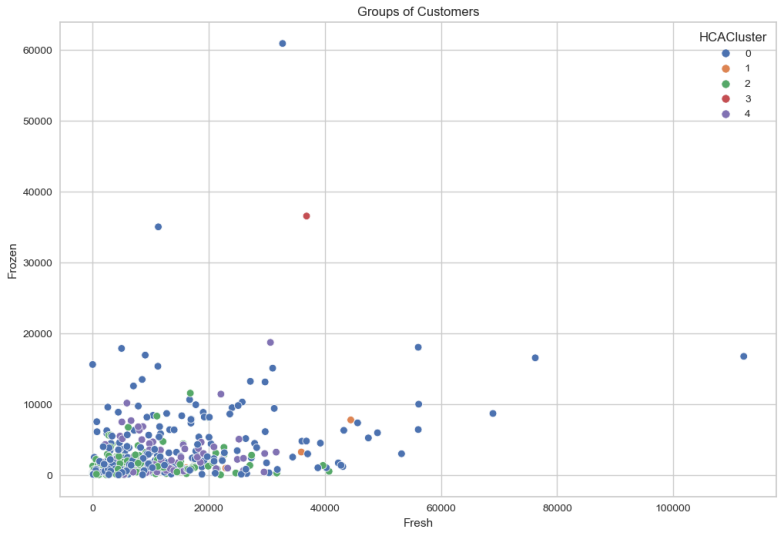
Agglomerative Hierarchical Clustering

We performed hierarchical clustering with Ward's method and Euclidean distance on our scaled data. 5 clusters were defined.

```
array([2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2,
       0, 0, 2, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 0, 0, 2, 2,
       2, 2, 2, 1, 2, 2, 0, 0, 2, 2, 0, 0, 2, 2, 0, 0, 2, 1, 2, 2, 0, 2,
       0, 2, 0, 0, 0, 0, 0, 2, 2, 0, 0, 2, 0, 0, 0, 2, 2, 0, 2, 1, 1, 0,
       0, 0, 0, 0, 2, 0, 2, 0, 2, 0, 0, 0, 2, 2, 2, 0, 0, 0, 2, 2, 2, 2,
       0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 2,
       0, 2, 0, 2, 2, 2, 2, 0, 0, 2, 2, 2, 0, 0, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 2, 2, 0, 0, 0, 2, 0, 0, 4, 2,
       4, 4, 2, 2, 4, 4, 4, 2, 4, 2, 4, 2, 4, 2, 4, 4, 2, 4, 2, 4, 2, 4,
       4, 4, 4, 2, 4, 4, 2, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 4, 4, 0, 4, 4,
       4, 4, 4, 2, 4, 4, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 0, 0, 4, 4, 4, 4,
       2, 4, 2, 4, 2, 4, 4, 4, 4, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 2, 4, 2, 4, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 4,
       4, 2, 0, 4, 2, 4, 4, 2, 4, 4, 4, 2, 4, 4, 4, 4, 4, 0, 4, 4, 0, 4,
       4, 2, 4, 1, 2, 2, 4, 0, 0, 0, 2, 2, 0, 2, 0, 0, 2, 2, 0, 2, 0, 2,
       0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 2,
       0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 2, 2, 0,
       2, 0, 0, 2, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0],
      dtype=int64)
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	KMCluster	HCACluster
0	2	3	12669	9656	7561	214	2674	1338	1	2
1	2	3	7057	9810	9568	1762	3293	1776	1	2
2	2	3	6353	8808	7684	2405	3516	7844	1	2
3	1	3	13265	1196	4221	6404	507	1788	1	0
4	2	3	22615	5410	7198	3915	1777	5185	1	2
5	2	3	9413	8259	5126	666	1795	1451	1	2
6	2	3	12126	3199	6975	480	3140	545	1	2
7	2	3	7579	4956	9426	1669	3321	2566	1	2
8	1	3	5963	3648	6192	425	1716	750	1	0
9	2	3	6006	11093	18881	1159	7425	2098	1	2

Below is a scatter plot, visualizing the distribution of the 5 clusters:



The scatter plot can give us some information about our data.

For instance, clusters 0,2 and 4 spend less on both fresh and frozen groceries. Meanwhile cluster 1 spends more on both categories.

DBSCAN

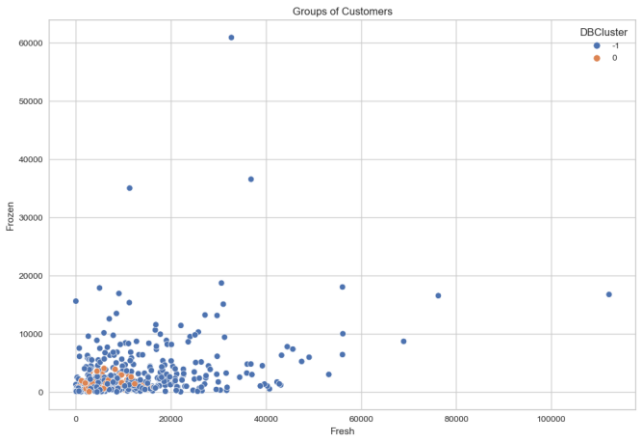
We used DBSCAN to identify meaningful clusters in our data considering the potential for uneven densities and noise points and an unknown number of clusters.

```
array([-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  0, -1,
       -1, -1, -1, -1,  0, -1, -1, -1, -1,  0,  0, -1, -1, -1, -1, -1, -1,
        0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1,  0, -1, -1, -1, -1, -1, -1, -1, -1,  0,  0,  0, -1, -1, -1, -1,
       -1, -1, -1, -1, -1,  0, -1, -1, -1, -1, -1,  0,  0, -1, -1, -1,
       -1, -1,  0, -1, -1, -1, -1, -1,  0, -1, -1, -1, -1,  0,  0, -1, -1,
        0, -1,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1,  0,  0,  0,  0,  0,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  0,  0,  0, -1, -1,  0, -1,
       -1,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  0,  0,  0,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  0, -1,  0,
       -1, -1, -1, -1,  0,  0, -1,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1,  0,  0, -1, -1, -1, -1,  0, -1, -1, -1, -1, -1,  0, -1,  0,
       -1,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1,  0, -1, -1, -1,  0, -1,  0, -1,  0, -1, -1,  0, -1,
       -1, -1, -1, -1,  0,  0, -1,  0, -1,  0,  0, -1,  0, -1, -1, -1, -1,
        0,  0, -1, -1,  0, -1, -1, -1, -1, -1, -1,  0,  0, -1,  0,  0, -1,
        0, -1, -1,  0,  0, -1, -1, -1,  0,  0, -1, -1, -1, -1,  0, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1,  0, -1, -1, -1, -1,  0],
      dtype=int64)
```

```
Number of clusters found 2
Clusters found [-1  0]
```

The value of -1 represents noise points that DBSCAN did not assign to any cluster.

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	KMCluster	HCACluster	DBCluster
0	2	3	12669	9656	7561	214	2674	1338	1	2	-1
1	2	3	7057	9810	9568	1762	3293	1776	1	2	-1
2	2	3	6353	8808	7684	2405	3516	7844	1	2	-1
3	1	3	13265	1196	4221	6404	507	1788	1	0	-1
4	2	3	22615	5410	7198	3915	1777	5185	1	2	-1
5	2	3	9413	8259	5126	666	1795	1451	1	2	-1
6	2	3	12126	3199	6975	480	3140	545	1	2	-1
7	2	3	7579	4956	9426	1669	3321	2566	1	2	-1
8	1	3	5963	3648	6192	425	1716	750	1	0	-1
9	2	3	6006	11093	18881	1159	7425	2098	1	2	-1



Moving on, we explored the impact of different parameter values on DBSCAN clustering for our data.

We defined two sets of values:

- Epsilon ranging from 0.3 to 2.0 and
- Minimum samples ranging from 3 to 48 with step of 2

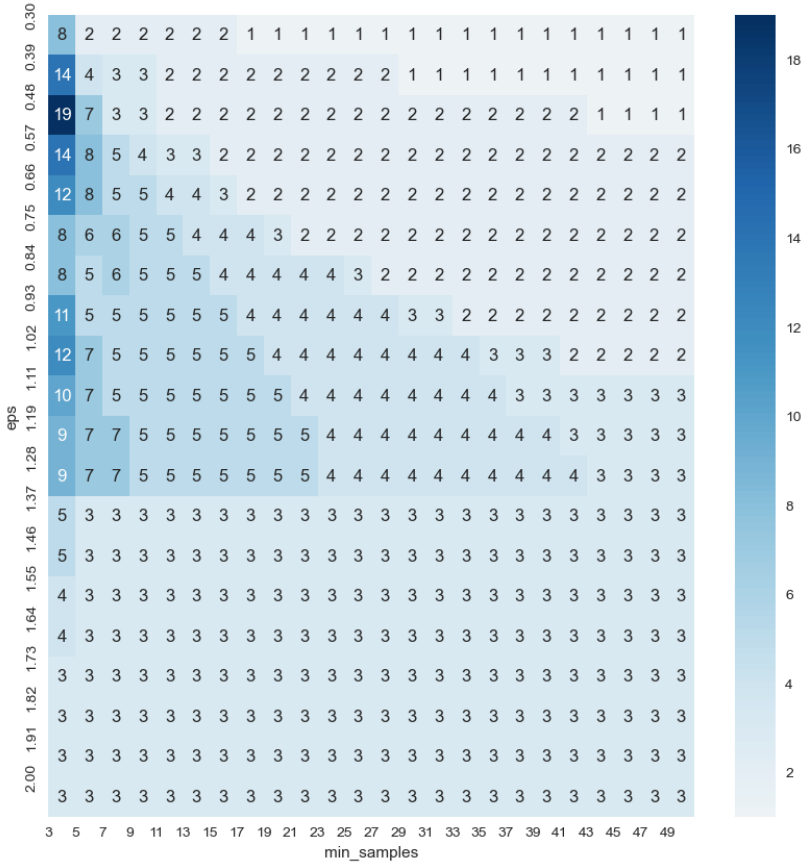
So, the code at this part explores how varying the minimum density and distance threshold in DBSCAN affects the resulting number of clusters.

The heatmap below is a visualization of the code explained before.

Color Scale: the colors of the heatmap represent the number of clusters for each combination of epsilon and minimum samples values. Dark blue shows a low number of clusters, while yellow shows a higher number of clusters.

In areas with dark blue shades, DBSCAN found very few clusters. This happens when epsilon is high or minimum samples is high.

Meanwhile, in areas with yellow, DBSCAN found a higher number of clusters. This happens when epsilon is low and minimum samples is low as well.



3. Regression

Our goal is to predict the final price of each home in our unknown dataset, using our data which include explanatory variables describing almost every aspect of residential homes in Ames, Iowa

(1460, 81)

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	8	2007	WD
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0	2	2010	WD
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	GdPrv	Shed	2500	5	2010	WD
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	4	2010	WD
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	6	2008	WD

In our dataset we have 1460 houses, 79 explanatory variables to use as input, the id of each house and our dependent variable SalePrice.

We also have the same input in the unknown dataset, and we need to predict the output.

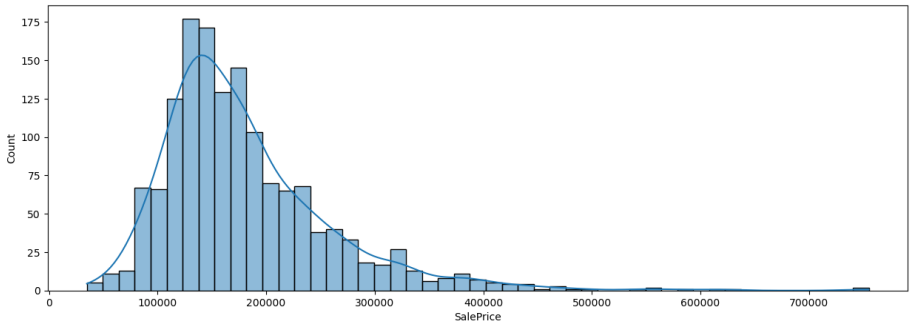
(1459, 80)

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold
1454	2915	160	RM	21.0	1936	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	NaN	0	6	2006
1455	2916	160	RM	21.0	1894	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	NaN	0	4	2006
1456	2917	20	RL	160.0	20000	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	NaN	0	9	2006
1457	2918	85	RL	62.0	10441	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	MnPrv	Shed	700	7	2006
1458	2919	60	RL	74.0	9627	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	NaN	0	11	2006

Moving on, we set Id column as the index on both the sets.

Then we drop any duplicates, and the shape becomes the following for train and test sets respectively.

(1460, 80) (1459, 79)



In the histogram the x- axis represents the Sale Price and the y-axis the number of houses sold. The distribution is positively skewed, which means there are more houses sold at lower prices. After investigating further using the describe function we found that the cheapest house cost 34.900 $ while the most expensive 755.000$. We want to use this range to test if the prediction would fall within and make us feel more confident with our predictions.

Then we proceed with checking for missing values, and we realized that there are many attributes with a few missing values and a few attributes with most of their values being null. Below you can see the attributes that we need to impute.

	Missing Values Flag
LotFrontage	True
Alley	True
MasVnrType	True
MasVnrArea	True
BsmtQual	True
BsmtCond	True
BsmtExposure	True
BsmtFinType1	True
BsmtFinType2	True
Electrical	True
FireplaceQu	True
GarageType	True
GarageYrBlt	True
GarageFinish	True
GarageQual	True
GarageCond	True
PoolQC	True
Fence	True
MiscFeature	True

```
train=train.drop(['Alley', 'PoolQC', 'Fence', 'MiscFeature'], axis=1)
train.shape
```

```
(1460, 76)
```

```
test=test.drop(['Alley', 'PoolQC', 'Fence', 'MiscFeature'], axis=1)
test.head()
```

```
#Attributes with so many values or which are not nominal and will not be able and help our model if mapped
additional_categoricals = [
    'MSSubClass', 'Condition1', 'Condition2', 'HouseStyle', 'RoofStyle',
    'RoofMatl', 'Exterior1st', 'Exterior2nd', 'Foundation', 'BsmtFinType2',
    'Heating', 'Electrical', 'GarageType', 'SaleType', 'Utilities', 'Neighborhood'
]
```

Python

```
#Drop those with so many values or are not nominal and will not be able and help our model if mapped

train = train.drop(additional_categoricals, axis=1)
print(train.shape)
test = test.drop(additional_categoricals, axis=1)
print(test.shape)
```

Python

```
(1460, 60)
(1459, 59)
```

```python
#Map the categorical variables
categorical_variables = [
    'MSZoning', 'Street', 'LotShape', 'LandContour', 'LotConfig',
    'LandSlope', 'BldgType', 'MasVnrType', 'ExterQual', 'ExterCond',
    'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'HeatingQC',
    'CentralAir', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageFinish',
    'GarageQual', 'GarageCond', 'PavedDrive', 'SaleCondition'
]

mapping_dict = {
    'MSZoning': {'A': 0, 'C': 1, 'FV': 2, 'I': 3, 'RH': 4, 'RL': 5, 'RP': 6, 'RM': 7},
    'Street': {'Grvl': 0, 'Pave': 1},
    'LotShape': {'Reg': 0, 'IR1': 1, 'IR2': 2, 'IR3': 3},
    'LandContour': {'Lvl': 0, 'Bnk': 1, 'HLS': 2, 'Low': 3},
    'LotConfig': {'Inside': 0, 'Corner': 1, 'CulDSac': 2, 'FR2': 3, 'FR3': 4},
    'LandSlope': {'Gtl': 0, 'Mod': 1, 'Sev': 2},
    'BldgType': {'1Fam': 0, '2FmCon': 1, 'Duplx': 2, 'TwnhsE': 3, 'TwnhsI': 4},
    'MasVnrType': {'BrkCmn': 0, 'BrkFace': 1, 'CBlock': 2, 'None': 3, 'Stone': 4},
    'ExterQual': {'Ex': 5, 'Gd': 4, 'TA': 3, 'Fa': 2, 'Po': 1},
    'ExterCond': {'Ex': 5, 'Gd': 4, 'TA': 3, 'Fa': 2, 'Po': 1},
    'BsmtQual': {'Ex': 5, 'Gd': 4, 'TA': 3, 'Fa': 2, 'Po': 1, 'NA': 0},
    'BsmtCond': {'Ex': 5, 'Gd': 4, 'TA': 3, 'Fa': 2, 'Po': 1, 'NA': 0},
    'BsmtExposure': {'Gd': 4, 'Av': 3, 'Mn': 2, 'No': 1, 'NA': 0},
    'BsmtFinType1': {'GLQ': 5, 'ALQ': 4, 'BLQ': 3, 'Rec': 2, 'LwQ': 1, 'Unf': 0, 'NA': 0},
    'HeatingQC': {'Ex': 5, 'Gd': 4, 'TA': 3, 'Fa': 2, 'Po': 1},
    'CentralAir': {'N': 0, 'Y': 1},
    'KitchenQual': {'Ex': 5, 'Gd': 4, 'TA': 3, 'Fa': 2, 'Po': 1},
    'Functional': {'Typ': 0, 'Min1': 1, 'Min2': 2, 'Mod': 3, 'Maj1': 4, 'Maj2': 5, 'Sev': 6, 'Sal': 7},
    'FireplaceQu': {'Ex': 5, 'Gd': 4, 'TA': 3, 'Fa': 2, 'Po': 1, 'NA': 0},
    'GarageFinish': {'Fin': 2, 'RFn': 1, 'Unf': 0, 'NA': 0},
    'GarageQual': {'Ex': 5, 'Gd': 4, 'TA': 3, 'Fa': 2, 'Po': 1, 'NA': 0},
    'GarageCond': {'Ex': 5, 'Gd': 4, 'TA': 3, 'Fa': 2, 'Po': 1, 'NA': 0},
    'PavedDrive': {'Y': 2, 'P': 1, 'N': 0},
    'SaleCondition': {'Normal': 0, 'Abnorml': 1, 'AdjLand': 2, 'Alloca': 3, 'Family': 4, 'Partial': 5}
}
```

```python
for var in categorical_variables:
    train[var] = train[var].map(mapping_dict[var])

print(train.shape)

for var in categorical_variables:
    test[var] = test[var].map(mapping_dict[var])

print(test.shape)
```

As per above you can see that our next steps were to drop any columns that either have many missing values or have many different values that would either increase a lot our dimensionality thought one hot encoding, or is not nominal to map it accordingly. Finally, we map all categorical variables we wanted to use in a manner that makes sense (e.g ‘Po’ = 1 and ‘Ex’=5)

Splitting

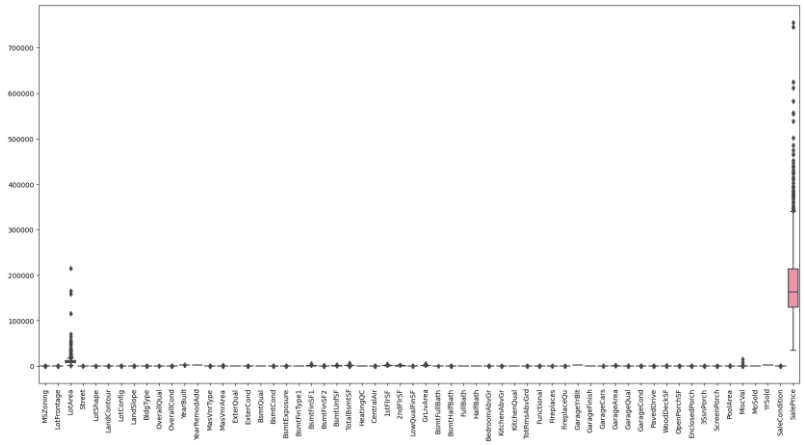
We split our train set with our target variable being SalePrice.

20% of the data will be used for the testing set and the remaining 80% for the training set.

The shapes are:

```
Shape of x_train and y_train:  (1168, 59) (1168,)
Shape of x_test and y_test:  (292, 59) (292,)
```

The boxplot below, is a visualization of our initial dataset before splitting and we can clearly see that some attributes may dominated the importance. So we have to scale our data accordingly.



Imputation

We used SimpleImputer() for numerical and categorical values, to impute missing values. In the case of numerical we replace the missing values with the median and in the case of categorical, the most frequently were used.

```
imp_cat = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
imp_num = SimpleImputer(missing_values=np.nan, strategy='median')
```

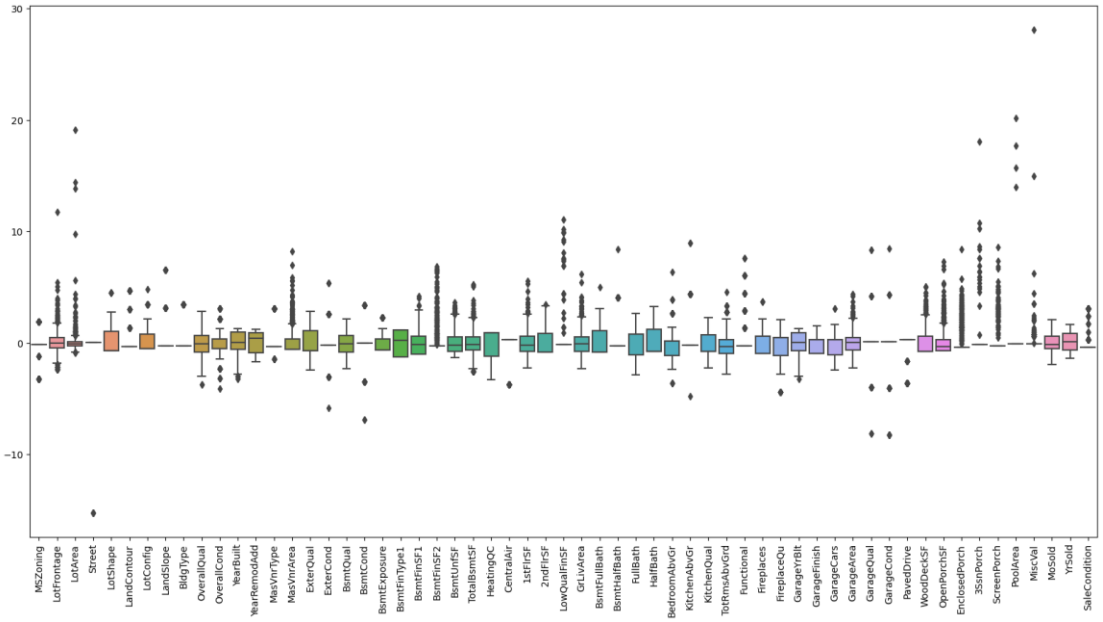
```
<class 'pandas.core.frame.DataFrame'>
Index: 1168 entries, 619 to 685
Data columns (total 59 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MSZoning       1168 non-null   float64
 1   LotFrontage    1168 non-null   float64
 2   LotArea        1168 non-null   int64
 3   Street         1168 non-null   int64
 4   LotShape       1168 non-null   int64
 5   LandContour    1168 non-null   int64
 6   LotConfig      1168 non-null   int64
 7   LandSlope      1168 non-null   int64
 8   BldgType       1168 non-null   float64
 9   OverallQual    1168 non-null   int64
 10  OverallCond    1168 non-null   int64
 11  YearBuilt      1168 non-null   int64
 12  YearRemodAdd   1168 non-null   int64
 13  MasVnrType     1168 non-null   float64
 14  MasVnrArea     1168 non-null   float64
 15  ExterQual      1168 non-null   int64
 16  ExterCond      1168 non-null   int64
 17  BsmtQual       1168 non-null   float64
 18  BsmtCond       1168 non-null   float64
 19  BsmtExposure   1168 non-null   float64
...
 57  YrSold         1168 non-null   int64
 58  SaleCondition  1168 non-null   int64
dtypes: float64(24), int64(35)
memory usage: 547.5 KB
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 292 entries, 530 to 62
Data columns (total 59 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MSZoning       292 non-null    float64
 1   LotFrontage    292 non-null    float64
 2   LotArea        292 non-null    int64
 3   Street         292 non-null    int64
 4   LotShape       292 non-null    int64
 5   LandContour    292 non-null    int64
 6   LotConfig      292 non-null    int64
 7   LandSlope      292 non-null    int64
 8   BldgType       292 non-null    float64
 9   OverallQual    292 non-null    int64
 10  OverallCond    292 non-null    int64
 11  YearBuilt      292 non-null    int64
 12  YearRemodAdd   292 non-null    int64
 13  MasVnrType     292 non-null    float64
 14  MasVnrArea     292 non-null    float64
 15  ExterQual      292 non-null    int64
 16  ExterCond      292 non-null    int64
 17  BsmtQual       292 non-null    float64
 18  BsmtCond       292 non-null    float64
 19  BsmtExposure   292 non-null    float64
...
 57  YrSold         292 non-null    int64
 58  SaleCondition  292 non-null    int64
dtypes: float64(24), int64(35)
memory usage: 136.9 KB
```

As we can observe, both X_train and X_test have no null-values after imputation and we can proceed with our scaling/

Scaling

After having split the data and imputed any missing values, using StandardScaler() we got the updated scatterplot of the input data:



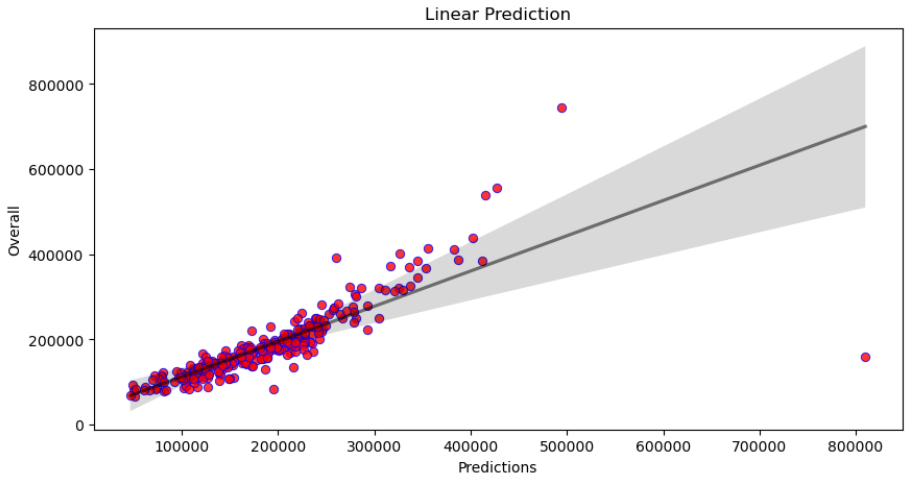
Prediction Regressors

Linear Regression models the relationship between a dependent variable and one or more variables.

```
r2 score (coefficient of determination): 0.653
RMSE : 48946.193
MAE : 21996.874
```

R^2: This model explains 65.3% of the variance in the SalePrice and there is a moderate positive relationship between the features and SalePrice.

Id	y_actual	y_predicted
530	200624	227551.540614
492	133000	142923.613145
460	110000	111270.887743
280	192000	218413.290434
656	88000	116466.161535



The graph illustrates the performance of a linear regression model in predicting the "Overall" values. The x-axis represents the predicted values, while the y-axis represents the actual "Overall" values. The scatter plot depicts individual data points, where each point represents a combination of the predicted and actual values. Additionally, a regression line is overlaid on the scatter plot, showing the overall trend of the relationship between the predicted and actual values. The regression line has been slightly faded (alpha = 0.5) to avoid overpowering the individual data points while still clearly showing the trend.

The coefficients indicate the relationship between the feature and the SalePrice. The signs show the direction of the relationship.

The magnitude of the coefficient indicates the strength of the relation. A bigger magnitude means a stronger influence.

	Columns	Coefficient Estimate
0	MSZoning	-2057.169023
1	LotFrontage	2349.249484
2	LotArea	3724.387596
3	Street	1701.210829
4	LotShape	1734.775282
5	LandContour	192.455823
6	LotConfig	169.273998
7	LandSlope	-228.383860
8	BldgType	-2525.696376
9	OverallQual	13491.937627

The intercept represents the predicted value of the SalePrice when all components in the model are zero.

```
Intercept: 180816.57940504467
```

Ordinary Least Squares

Firstly, we converted y_train in a NumPy array as OLS regression expects the target variable to be an Numpy array.

Then, we added a constant term (the intercept calculated before) to the begging of our training data features.

R^2: The model explains the 86.8% of the variance in SalePrice based on the features. There is a moderate-strong positive relationship.

The F-Statistic is 128.4 and its p-values is 0.00, which suggests that the model is statistically significant.

Also, the table shows the coefficients and their standard errors, t-statistics and p-values.

Regularization

A statistical method to reduce errors caused by overfitting on training data.

Ridge Regression

---

a method for estimating the coefficients of multiple regression.

To find the optimal alpha for Ridge Regression, we proceded with Redge Regression with cross validation.

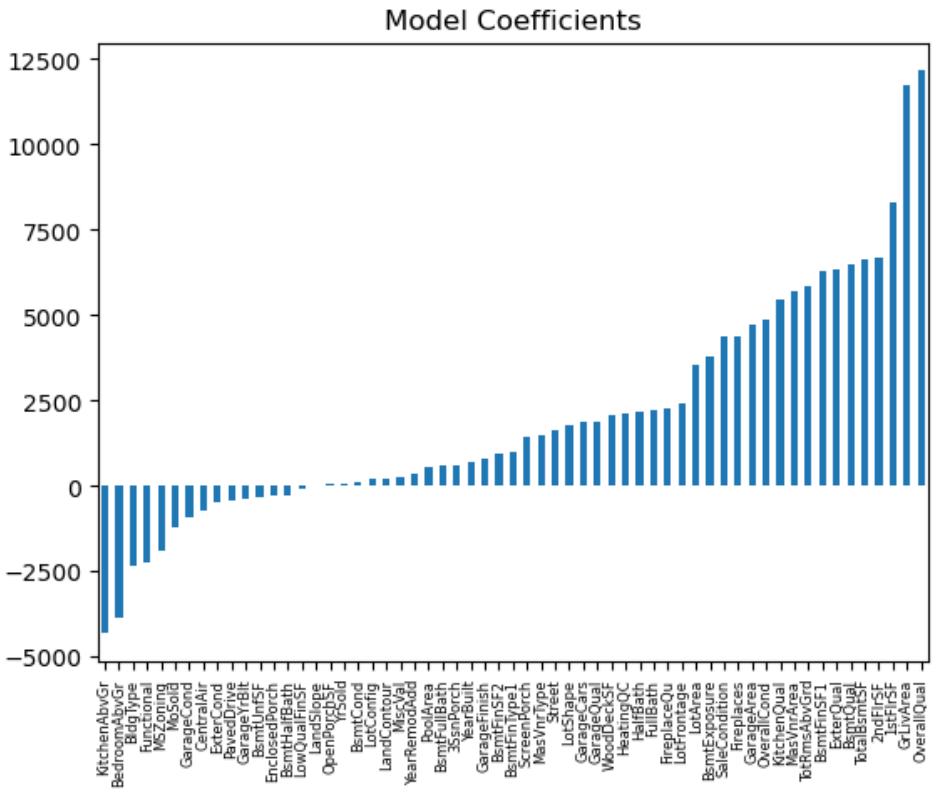
We used a sequence of 13 alpha values between -6, 6.

The metric used to evaluate the performance of the model is R2 and 10 folds.

The optimal alpha is: `100.0`

The results of Ridge Regression with the optimal alpha for the first 20 columns:

	var	coef
0	MSZoning	-1891.281220
1	LotFrontage	2404.060591
2	LotArea	3528.010524
3	Street	1626.335763
4	LotShape	1785.412248
5	LandContour	205.906688
6	LotConfig	185.085020
7	LandSlope	-6.907643
8	BldgType	-2327.728334
9	OverallQual	12152.009195
10	OverallCond	4870.879472
11	YearBuilt	717.983807
12	YearRemodAdd	331.541792
13	MasVnrType	1493.656513
14	MasVnrArea	5677.679489
15	ExterQual	6318.511802
16	ExterCond	-476.429040
17	BsmtQual	6482.480996
18	BsmtCond	91.081957
19	BsmtExposure	3762.122552
20	BsmtFinType1	983.706101



Each bar represents the coefficient value of a feature. The length of the bar indicates the magnitude of the coefficient. Positive coefficients are depicted with bars

extending to the right, while negative coefficients are shown with bars extending to the left. The features are sorted based on the magnitude of their coefficients. This sorting enables easy identification of the most influential features (largest coefficients) and the least influential features (smallest coefficients). We can see the the two most important coefficients in our model is the Overall quality and the squared meters of the houses Ground living area.

Lasso Regression

A regularization technique that applies a penalty to prevent overfitting and enhance the accuracy of statistical models.

```
Training score: 0.8683407957324363
Testing score: 0.6531171433592671
MAE of Lasso Regression: 21993.278299663554
```

$R^{2\ \text{training data}}$: This model explains 86.83% of the variance in the SalePrice.

$R^{2\ \text{test data}}$: This model explains 65.3% of the variance in the SalePrice.

This means that the model performs better on the data it was trained in.

Afterwards, to avoid overfitting with the previous model, we did Lasso Regression with cross validation.

We used a sequence of 13 alpha values between -6, 6 and 10 folds.

The optimal alpha is: 1000.0

The results of Lasso Regression with the optimal alpha for the model:

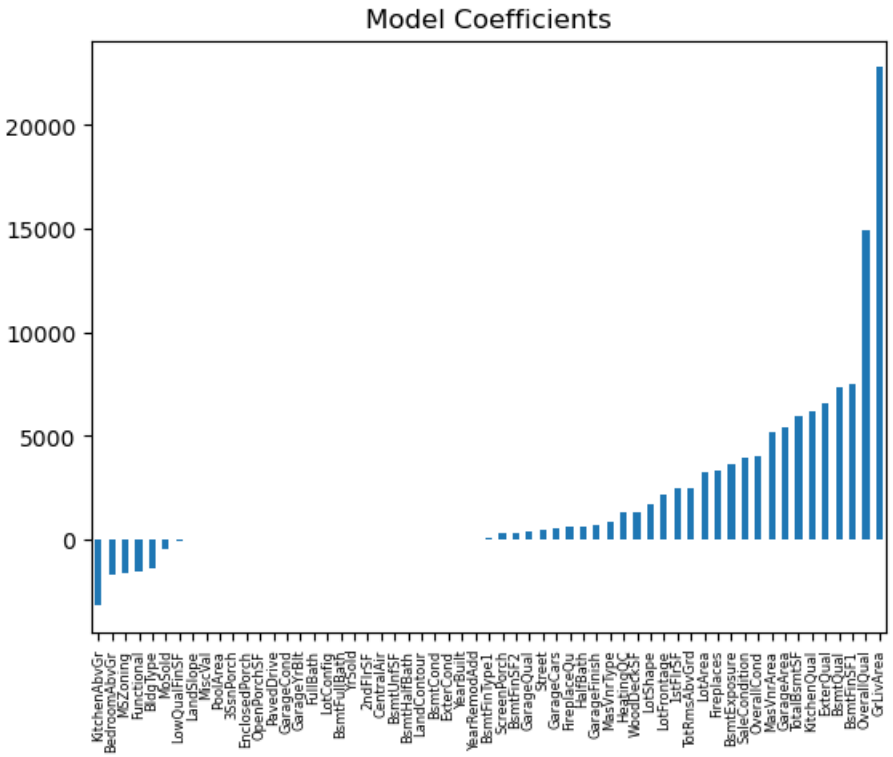
```
Training score: 0.8683407957324363
Testing score: 0.6531171433592671
MAE of Lasso Regression: 21993.278299663554
```

$R^{2\ \text{training data}}$: This model explains 86.83% of the variance in the SalePrice.

$R^{2\ \text{test data}}$: This model explains 65.3%1 of the variance in the SalePrice.

As we can see the R^2 and the MAE remained the same, even with the optimal alpha.

	var	coef
0	MSZoning	-1624.579590
1	LotFrontage	2126.881555
2	LotArea	3210.087856
3	Street	440.688650
4	LotShape	1694.770877
5	LandContour	0.000000
6	LotConfig	0.000000
7	LandSlope	0.000000
8	BldgType	-1399.906824
9	OverallQual	14938.194581
10	OverallCond	3982.717280
11	YearBuilt	0.000000
12	YearRemodAdd	5.104678
13	MasVnrType	818.150908
14	MasVnrArea	5188.514608
15	ExterQual	6554.652523
16	ExterCond	-0.000000
17	BsmtQual	7371.614582
18	BsmtCond	0.000000
19	BsmtExposure	3635.443805
20	BsmtFinType1	98.306878



Similarly with Lasso we have the same coefficients being the most important, with this time the squared feet of the ground living area being more important even from the overall quality.

Polynomial Regression

Is a form of regression in which the relationship between the independent variable and the dependent variable is modeled as an nth degree polynomial in x.

```
r2 score (coefficient of determination): -0.692
RMSE : 108086.976
MAE : 27001.161
```

R^2 with a negative value suggests that the model performs worse than predicted.

RMSE is also high, meaning that the average difference between the predicted and the actual price is 108086.

MAE is also high, meaning that an mean absolute average difference between the predicted and the actual price is 27000.

Random Forest Regressor

---

is a supervised learning algorithm and bagging technique that uses an ensemble learning method for regression.

```
r2 score (coefficient of determination): 0.844
RMSE : 32802.469
MAE : 17327.141
```

R^2: The model explains the 84.4% of the variance in SalePrice based on the features.

RMSE: the average difference between the predicted and the actual price is 32802.469.

MAE: a mean absolute average difference between the predicted and the actual price is 17327.141.

Then, we performed a hyperparameter tuning for the above Random Forest Regression model. The best parameters are:

```
{'n_estimators': 157,
 'min_samples_split': 3,
 'min_samples_leaf': 1,
 'max_depth': 10}
```

The results of the optimized Random Forest Regressor are:

```
r2 score (coefficient of determination): 0.841
RMSE : 33171.48
MAE : 17372.464
```

R^2: The optimized model explains the 84.1% of the variance in SalePrice based on the features.

RMSE: the average difference between the predicted and the actual price is 33171.48.

MAE: a mean absolute average difference between the predicted and the actual price is 17372.464.

Final Comparison

	Model	r2	RMSE	MAE
0	RFRegressor	0.841	33171.480	17372.464
0	Ridge	0.667	47973.488	21843.681
0	Lasso	0.663	48231.708	21908.029
0	Linear model (OLS)	0.653	48946.193	21996.874
0	Poly	-0.692	108086.976	27001.161

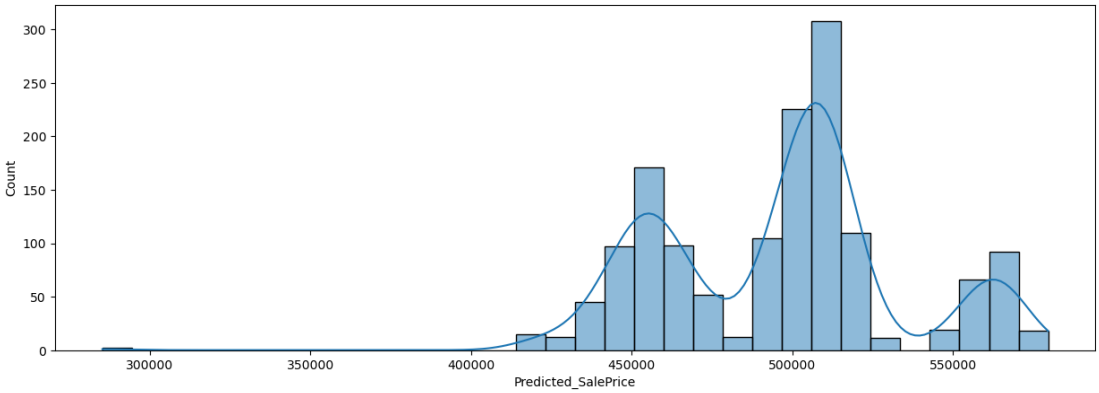
Random Forest Regressor appears to be the best performing model based on the R2 RMSE and MAE metrics. Furthermore, more underlying dynamics are captured in the data, which leads to more accurate predictions.

Lastly, we created a Random Forest Regression model to predict SalePrice.

Here is the prediction for the first 10 houses:

Predicted_SalePrice
456479.44
500407.47
501968.12
558882.80
435380.00
510617.75
456240.65
503680.27
458348.55
461320.24

The visualization of the predicted SalePrice distribution:



The kernel curve suggests a peak of around 500.000, indicating that this model predicts that most houses have sales prices at this price. Also, we have two other peaks in the 450.000 and around 570.000 respectively.

The spread of the graph suggests a variation in the predicted prices, which is inside the range of the houses used to train our model.