ITC 6003-Applied machine learning

- Final Project -

Petros Tamvakis (240760) - Vasileios Filipppidis (243989) - Anastasios Katsaounis (244455)

April 11, 2020

https://github.com/petamva/Project-AML

Part 1 Classification

The first part of our project is a classification problem. Given the arrhythmia dataset which consists of 452 entries, 279 features and 16 classes we are asked to create a model that accurately predicts which of these classes best represents a new unknown entry (patient). There is a class that represents healthy patients while the rest of them represent a certain disease (e.g. Coronary Artery Disease)

First of all, we realized that we have to deal with a dataset that is in a very high degree unbalanced. Training a model to predict all these classes would yield to very unreliable results. Taking into consideration that we deal with medical data there is no room for mistake, because there is the risk of classifying an unhealthy patient as healthy or even giving wrong prescription. For this reason, we decided to keep only 2 classes. Class 0 (positive) for healthy patients and class 1 (negative), which results by grouping the rest of the classes, for unhealthy. In this sense, the model could be used to decide if a patient should visit a cardiologist or not.

Table 1: Classes of the dataset

Class Code	Class	Number of instances	Instances % per Class
1	Normal	245	***54.2%***
2	Ischemic changes (Coronary Artery Disease)	44	***9.7%***
3	Old Anterior Myocardial Infarction	15	***3.3%***
4	Old Inferior Myocardial Infarction	15	***3.3%***
5	Sinus tachycardy	13	***2.9%***
6	Sinus bradycardy	25	***5.5%***
7	Ventricular Premature Contraction (PVC)	3	***0.7%***
8	Supraventricular Premature Contraction	2	***0.4%***
9	Left bundle branch block	9	***2.0%***
10	Right bundle branch block	50	***11.1%***
11	1st degree AtrioVentricular block	0	***0.0%***
12	2nd degree AV block	0	***0.0%***
13	3rd degree AV block	0	***0.0%***
14	Left ventricule hypertrophy	4	***0.9%***
15	Atrial Fibrillation or Flutter	5	***1.1%***
16	Others	22	***4.9%***
Total		**452**	***100%***

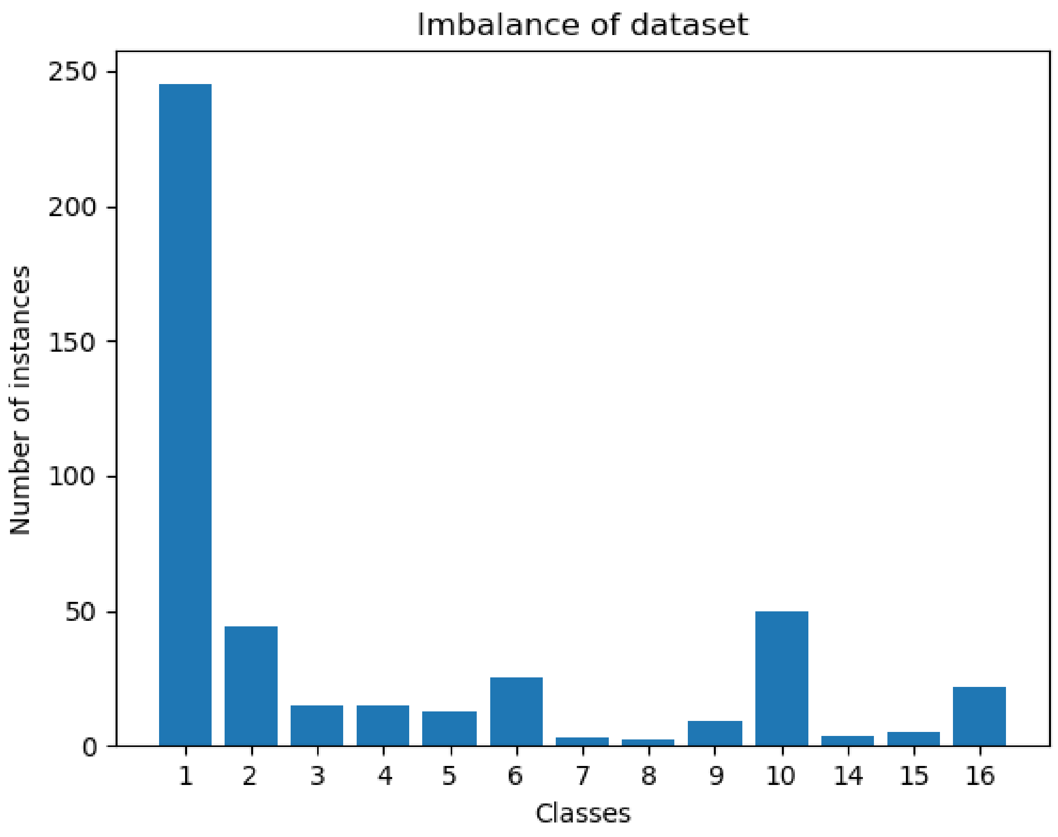


Figure 1: Number of instances per class

In the arrhythmia dataset unknown values are denoted by “?”. We decided to drop column 13 because more than 80% of its values where “?”. We then replaced the remaining nans with the median value of the corresponding feature.

The data are already in numeric form and there is no need to perform one hot encoding.

After dealing with nan values, we decided to reduce the number of features. This is a critical point because having so many features makes it hard to train a model, especially when the dataset grows in size by adding new entries. We used Principal Component Analysis (PCA) to engineer 30 features, which retain 92% of the variance (explained variance). The downside of using PCA is that it prevents the interpretation of the original features, as well as the magnitude of their impact, because eigenvectors are not meaningful. Of course, we do not have domain knowledge as neither of us is a doctor so this is a price we are willing to pay.

Afterwards, we unveiled the outliers of each and every feature. Assuming normal distribution, we evaluated as outlier every value greater than 2.5 * σ (standard deviation) and replaced it with the median value of the corresponding feature. In Figure 2 we can see the distribution of each principal component and the outliers. Also we can verify some PCA theory, as principal PC0 has the most variance followed by PC1 and so on.

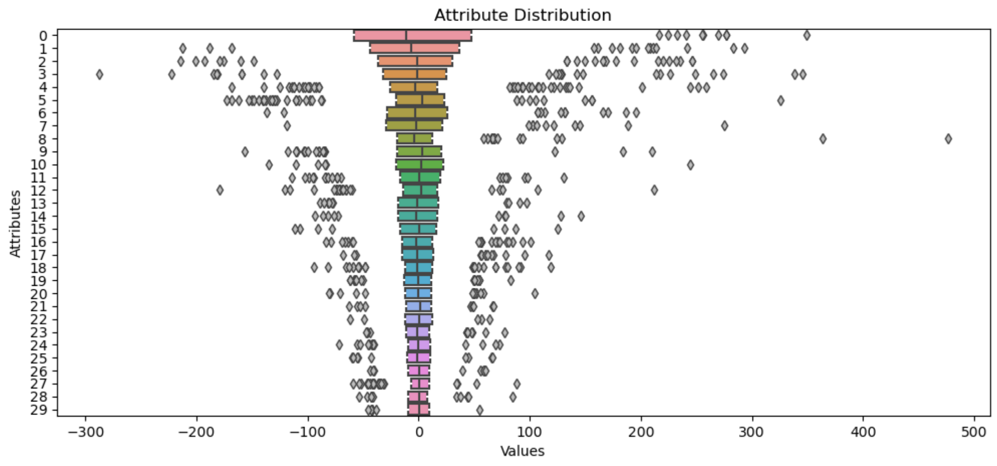


Figure 2: PCA Engineered features and their outliers

At this point we are done with preprocessing and our data are ready to be imported to the classification algorithms and build our models. Here is a list of the classification techniques that we used:

- Decision Trees
- Nearest Neighbors
- Naïve Bayes
- Artificial Neural Networks
- Support Vector Machine (SVM)

We tried several train/test ratios and different hyperparameters for each classifier to yield to the best results possible. We also took advantage of the grid search technique that scikit-learn library provides to help us reach our goal.

In order to compare our models and choose the best we need evaluation metrics. We took account of:

- Precision
- Recall
- Macro F1 score

Of course, predicting correctly unhealthy patients (True Negatives) is much more important than predicting that a healthy person is not healthy (False Negatives). For this reason, we chose Recall over Precision together with Macro F1 score to avoid overfitting.

Table 2: Evaluation Metrics

Classifier	Macro Precision	Recall	F1-score
Decision Tree	0.640	0.632	0.630
Decision Tree 2	0.597	0.597	0.597
Naïve Bayes	0.729	0.723	0.724
Support Vector Machines	0.747	0.733	0.734
Nearest Neighbors	0.701	0.621	0.592
Nearest Neighbors 2	0.648	0.521	0.408
Artificial Neural Networks	0.694	0.688	0.688
Artificial Neural Nets 2	0.646	0.638	0.637

Since we have converted the problem at hand into a binary one it is easier to choose the best classifier by visualizing them using the receiver operating characteristic (ROC) curve.

Figure 3 provides the ROC curves of the classifiers we used and Table 3 the corresponding Area Under Curve.

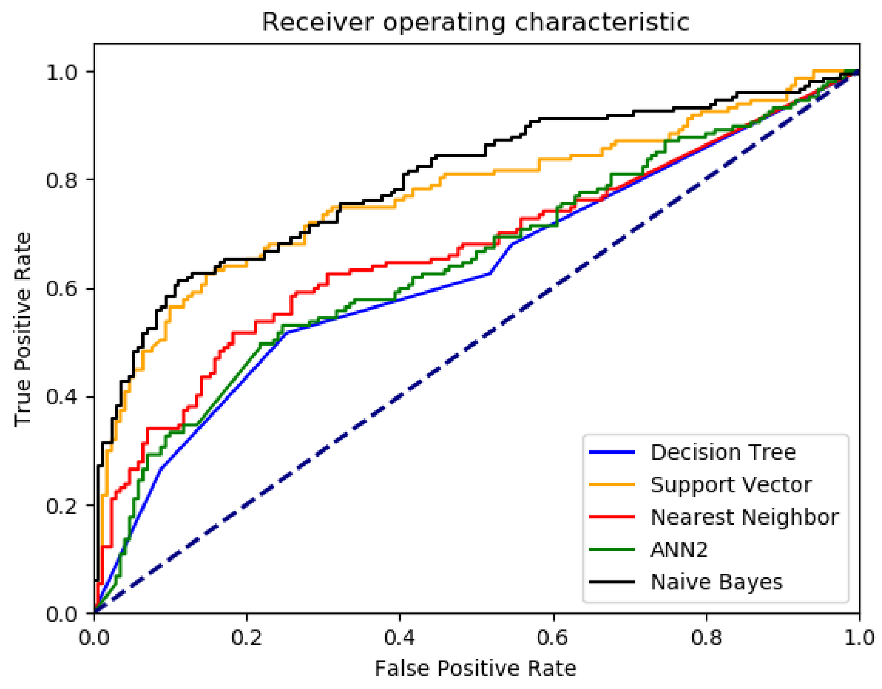


Figure 3: ROC curves of the classifiers

Table 3: Area Under Curve

Area Under Curve	
Decision Tree	0.627
Support Vector Machines	0.769
Neural Networks	0.645
Nearest Neighbors	0.669
Naïve Bayes	0.798

Now we can clearly see that if we were to choose a classifier we would choose Naïve Bayes or Support Vector Machines.

PART B: Clustering

Cluster analysis groups data objects based only on information found in the data that describes the objects and their relationships. The goal is that the objects within a group be similar (or related) to one another and different from (or unrelated to) the objects in other groups. The greater the similarity (or homogeneity) within a group and the greater the difference between groups, the better or more distinct the clustering.[1]

Cluster analysis is related to other techniques that are used to divide data objects into groups. For instance, clustering can be regarded as a form of classification in that it creates a labeling of objects with class (cluster) labels. However, it derives these labels only from the data. In contrast, classification is supervised classification; i.e., new, unlabeled objects are assigned a class label using a model developed from objects with known class labels. For this reason, cluster analysis is sometimes referred to as unsupervised classification.[1]

In this part of the project the task is to perform clustering algorithms and evaluate their parameters. The assigned dataset was downloaded from `https://archive.ics.uci.edu/ml/datasets/Wholesale+customers` and refers to clients of a wholesale distributor. It includes 440 records describing the annual spending in monetary units (m.u.) on diverse product categories such as : Fresh, Milk, Grocery, Frozen, Detergents-Paper and Delicatessen.

1 Preparing the data

'Channel' and 'Region' categories were dropped as irrelevant and not contributing any information to the task at hand.

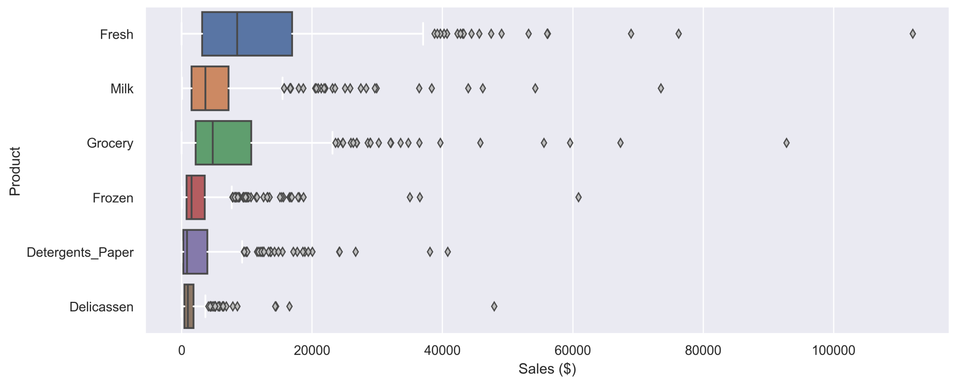


Figure 1: Annual spending distributions

Because of the annual spending's big distribution range, data were transformed to a logarithmic scale:

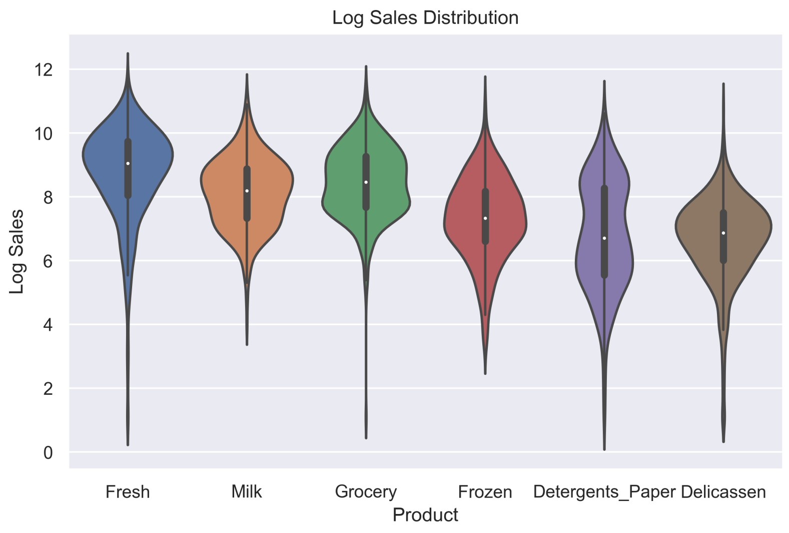
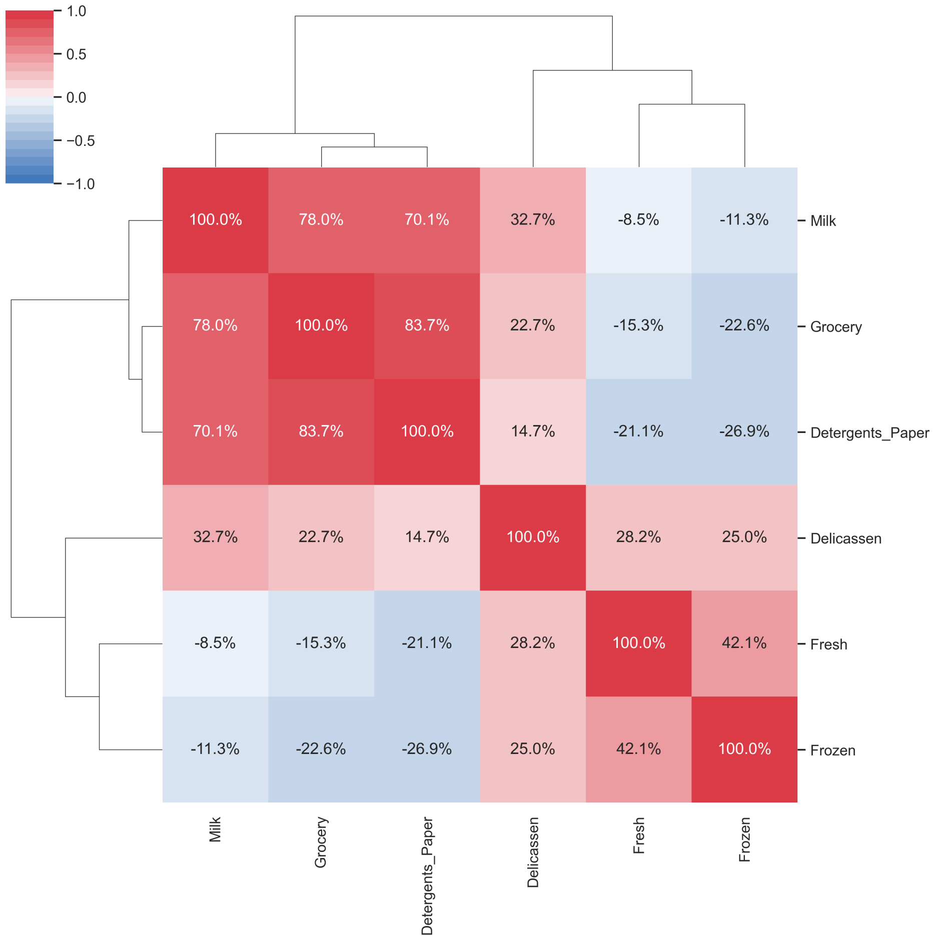


Figure 2: Scaled data

Both diagrams clearly show outliers in our data. To avoid infliction of a negative effect to the clustering techniques, outliers were removed using the Local Outlier Factor setting n-neighbors=20 and contamination factor as 0.05 resulting to a reduced dataset of 418 records.

2 Principal Component Analysis

After outlier removal a correlation heatmap was constructed to further explore the relationship between the products and potentially reduce data dimensions through PCA. From the heatmap it is clear that two major



	Milk	Grocery	Detergents_Paper	Delicassen	Fresh	Frozen
Milk	100.0%	78.0%	70.1%	32.7%	-8.5%	-11.3%
Grocery	78.0%	100.0%	83.7%	22.7%	-15.3%	-22.6%
Detergents_Paper	70.1%	83.7%	100.0%	14.7%	-21.1%	-26.9%
Delicassen	32.7%	22.7%	14.7%	100.0%	28.2%	25.0%
Fresh	-8.5%	-15.3%	-21.1%	28.2%	100.0%	42.1%
Frozen	-11.3%	-22.6%	-26.9%	25.0%	42.1%	100.0%

Figure 3: Correlation heatmap

groups (or clusters) exist in our data: the first group is: Milk, Grocery and Detergents-Paper, and the other : Frozen, Fresh and Delicatessen. Further analysis reveals that the two groups (components) capture 0.74% of total data variance directing to a dimensionality reduction, retaining the specific components.

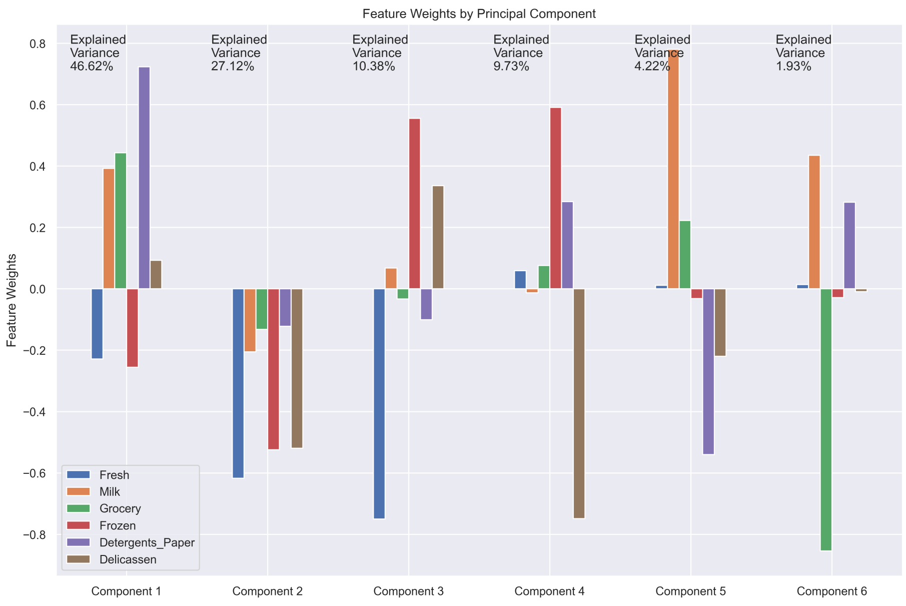


Figure 4: Feature weights

Below we present a joint plot of the data distribution (scatterplot and histograms) based on the new dimensions. On the diagram we can also see the initial component vector with respect to the new dimensions.

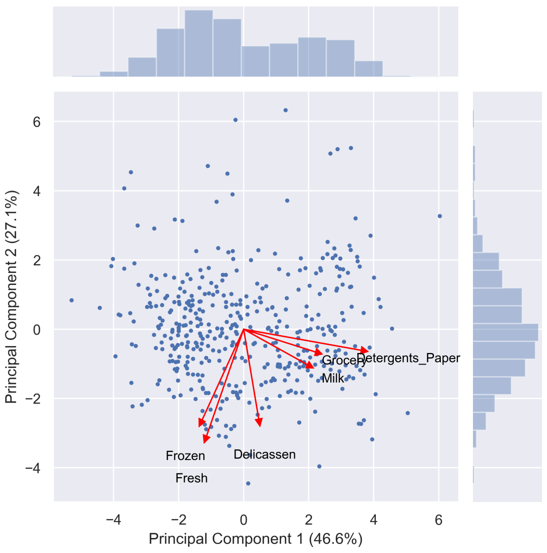


Figure 5: Feature weights

3 Clustering algorithms

3.1 K means

This is a prototype-based, partitional clustering technique that attempts to find a user-specified number of clusters (K), which are represented by their centroids. K-means defines a prototype in terms of a centroid, which is usually the mean of a group of points, and is typically applied to objects in a continuous n-dimensional space.[1]

For research purposes K-means algorithm was tested on the reduced dataset for n=(2,11) clusters. Evaluation metrics (inertia,silhouette) were calculated and their respective diagrams along with cluster depictions and a result table for n=4,5,6,7 are presented below:

K means	**Clusters**			
	4	**5**	**6**	**7**
Inertia	869.25	710.43	593.83	503.99
Silhouette	0.344	0.365	0.378	0.370

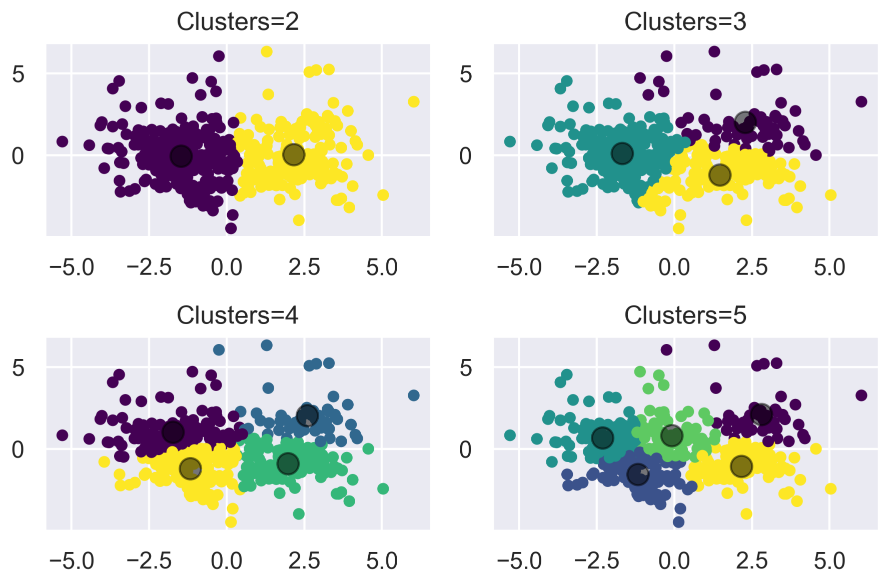


Figure 6: K-means

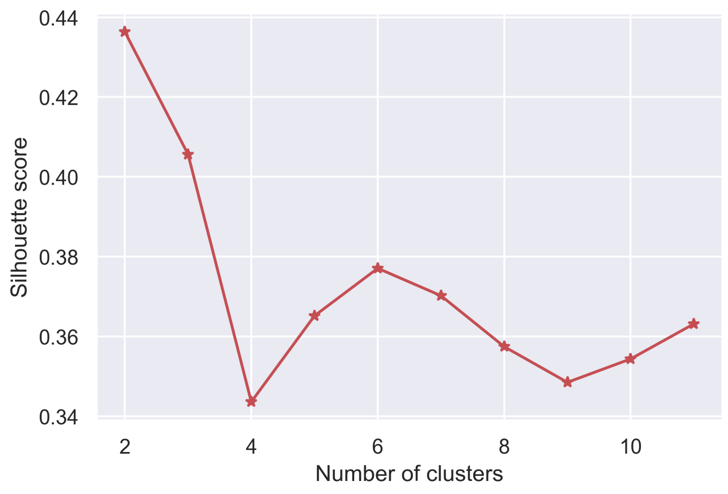


Figure 7: Silhouette

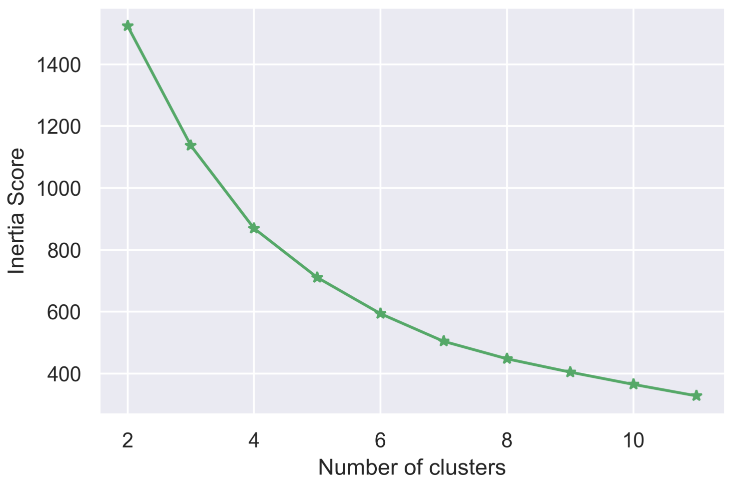


Figure 8: Inertia

From the inertia ”elbow” diagram a remark to be made is that the optimum number of clusters for our analysis is 6. For 6 clusters the slope of the diagram starts to decrease. Also the value of silhouette is higher than 5 or 7 clusters and beyond. One has to keep in mind that the dataset has already been reduced by PCA to only two components.

3.2 Gaussian mixtures

A Gaussian mixture model (GMM) attempts to find a mixture of multi-dimensional Gaussian probability distributions that best model any input dataset. In the simplest case, GMMs can be used for finding clusters in the same manner as k-means.[1]

GMM algorithm was implemented to the reduced dataset for the same number of clusters (n=2,11) and the results for the log-likelihood metric are summarized in the table below.

Log-likelihood

Clusters	2	3	4	5	6
	-3.946	**-3.943**	**-3.898**	**-3.889**	**-3.866**
Clusters	7	8	9	10	11
	-3.86	**-3.823**	**-3.831**	**-3.823**	**-3.799**

For the evaluation the silhouette metric was initially used. Six clusters seem to be the optimal number for this algorithm too although the results for other numbers of clusters have marginal differences.

Silhouette

Clusters	2	3	4	5	6
	0.436	**0.322**	**0.262**	**0.239**	**0.351**
Clusters	7	8	9	10	11
	0.338	**0.328**	**0.312**	**0.333**	**0.314**

The Akaike and Bayesian information criteria were also used for the current model and although the two-cluster model yields the lowest values , AIC plot seems to have a local minimum for six clusters also.

AIC

Clusters	2	3	4	5	6
	3321	**3333**	**3305**	**3309**	**3300**
Clusters	7	8	9	10	11
	3299	**3293**	**3322**	**3307**	**3310**

BIC

Clusters	2	3	4	5	6
	3365	**3401**	**3398**	**3426**	**3442**
Clusters	7	8	9	10	11
	3465	**3483**	**3536**	**3545**	**3572**

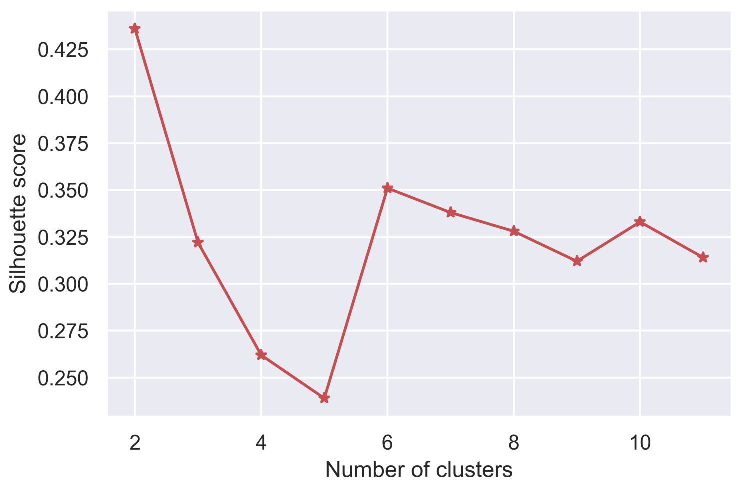


Figure 9: Silhouette((GMM)

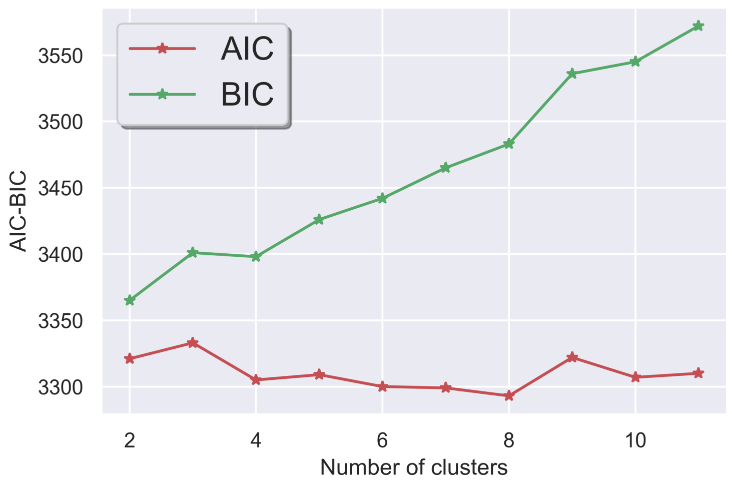


Figure 10: AIC-BIC

3.3 DBSCAN

This is a density-based clustering algorithm that produces a partitional clustering, in which the number of clusters is automatically determined by the algorithm. Points in low-density regions are classified as noise and omitted; thus, DBSCAN does not produce a complete clustering.[1]

In DBSCAN the parameters that have to be considered are the maximum radius of a cluster (eps) and the minimum number of points that must be included in the cluster (min-samples). Adjusting these parameters the conclusion was reached that based on the silhouette criterion the optimum number of clusters for DBSCAN algorithm is 1 or 2.

DBSCAN	*eps=1,min=20*	*eps=0.5,min=5*
	cluster=1	**clusters=2**
Silhouette	0.418	0.255

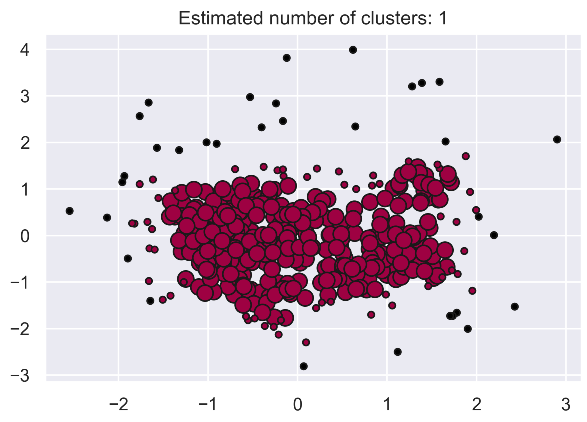


Figure 11: DBSCAN 1 cluster

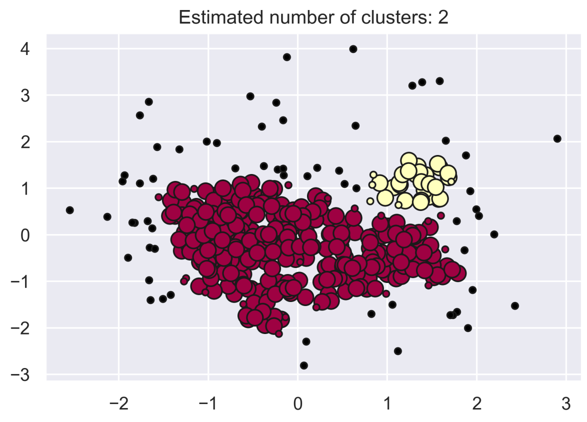


Figure 12: DBSCAN 2 clusters

Extensive analysis showed that for the majority of eps and min-sample combinations, DBSCAN clustered the samples in one big cluster. Only for very small cluster radius and minimum samples did the algorithm yield two clusters distinguishing DBSCAN from the other clustering techniques.

4 Performing clustering to the non-reduced dataset

A question that arises is: had we kept the dataset as a whole and its features hadn't been reduced would the clustering algorithms perform better or worst? All three algorithms were applied again this time to the whole dataset keeping all six original features.Results are summarized below.

4.1 K means

K means algorithm yielded almost identical results for the whole dataset resulting to 6 clusters as an optimum number for the analysis.

K means	**Clusters**			
	4	**5**	**6**	**7**
Inertia	1867.91	1703.90	1562.00	1463.08
Silhouette	0.211	0.209	0.218	0.215

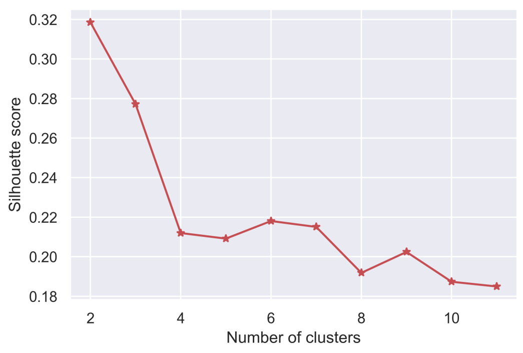


Figure 13: Silhouette

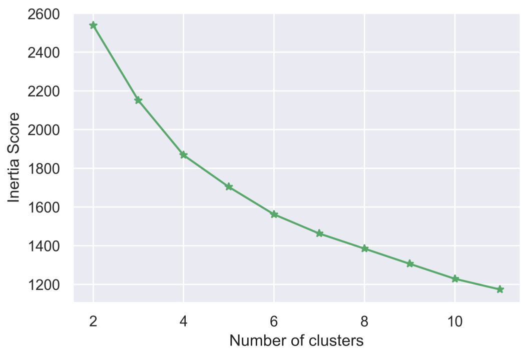


Figure 14: Inertia

4.2 Gaussian mixtures

GMM algorithm implementation:

log-likelihood

Clusters	2	3	4	5	6
	-7.99	**-7.74**	**-7.625**	**-7.582**	**-7.422**
Clusters	7	8	9	10	11
	-7.387	**-7.328**	**-7.255**	**-7.212**	**-7.101**

Silhouette

Clusters	2	3	4	5	6
	0.309	**0.221**	**0.148**	**0.116**	**0.14**
Clusters	7	8	9	10	11
	0.156	**0.085**	**0.1**	**0.083**	**0.151**

4.3 DBSCAN

The majority of eps and min-samples combinations resulted to the same result: DBSCAN clustered the samples to one big cluster.

DBSCAN	*eps=1,min=10*	*eps=1.2,min=10*
	cluster=1	**clusters=2**
Silhouette	0.32	0.079

Part C: Regression

Our aim in this part of the project was to find a suitable dataset and apply various Machine Learning Algorithms for Regression. The dataset that we chose was the “House Sales in King County, USA” and it contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015. (House Sales in King County, USA, 2020).

After searching for some time to find a good dataset where the class we are trying to predict has continuous values and its features have meaningful data we found the aforementioned dataset. It didn’t have any missing values also which made it a very good choice for this part of the project.

The dataset has 21 columns from which one represents the price of the houses and the rest the various features of the houses like the number of bedrooms, bathrooms, the square feet of the various rooms, the zip code and more.

Our goal is to use various Regression algorithms to predict the Price (dependent variable) based on the rest 20 features of the dataset (independent variables).

After importing the dataset in a pandas DataFrame we created a heatmap to see the correlation between its features:

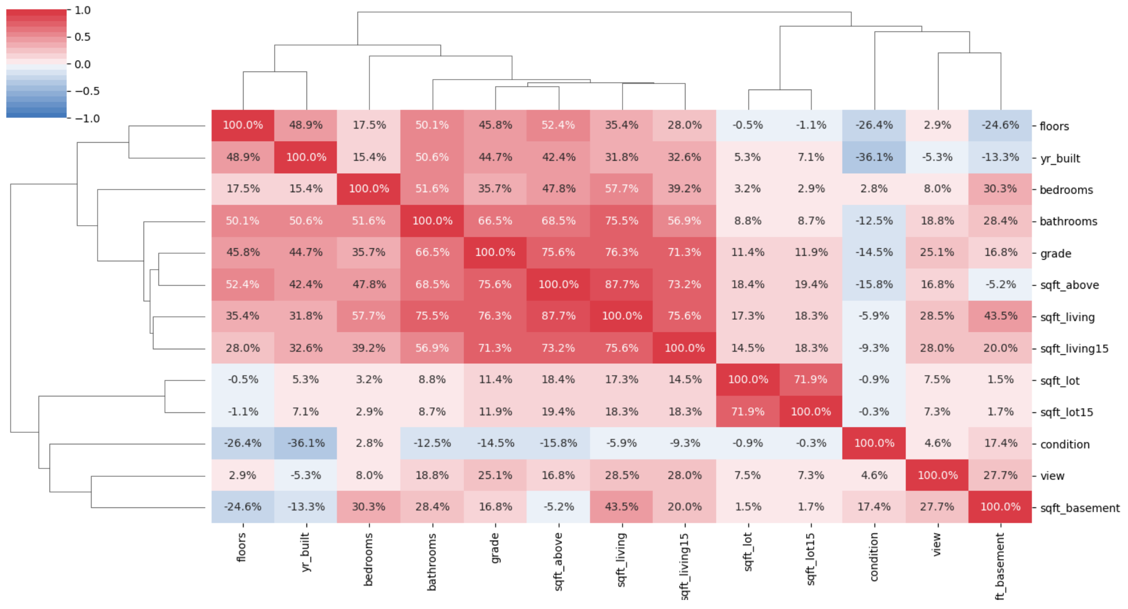


Figure 1 of Part 3: Heatmap showing correlation between features

We then removed the outliers from our dataset and made a matrix with scatter plots to get a better understanding of how price is related to the rest of the features:

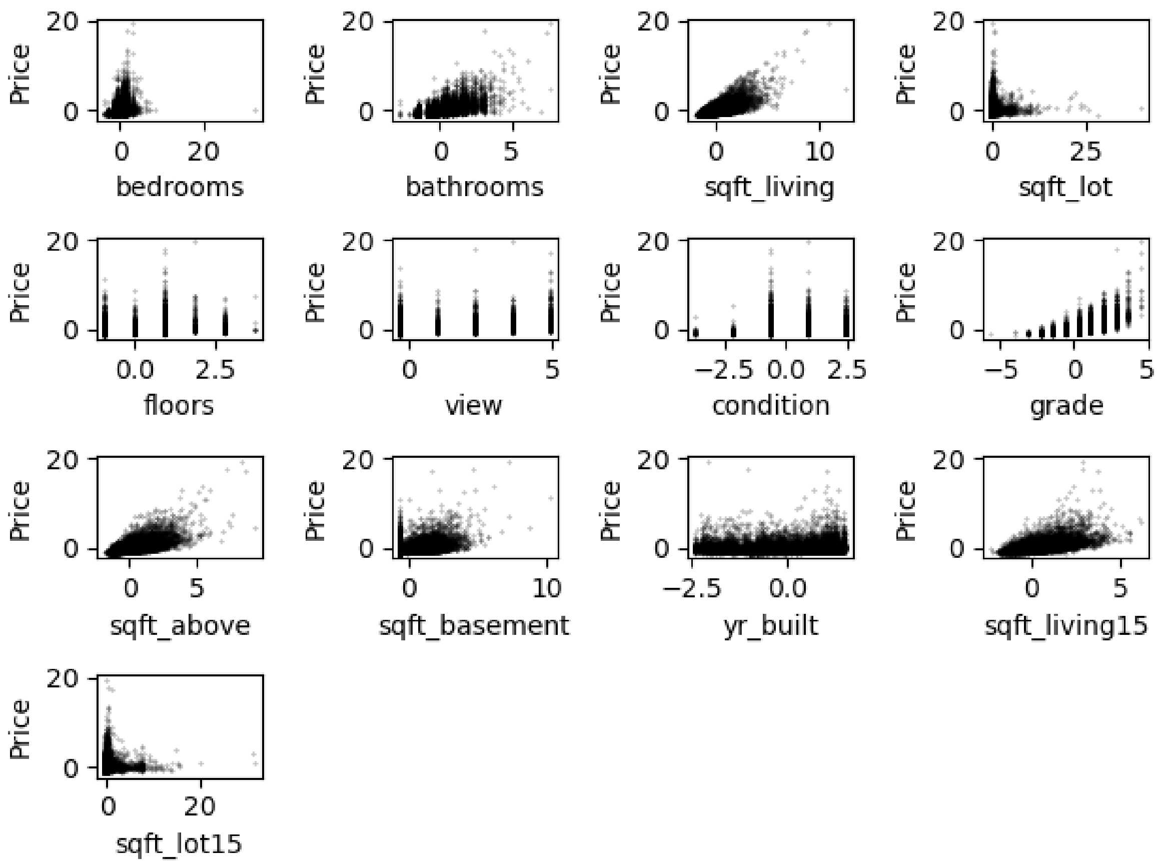


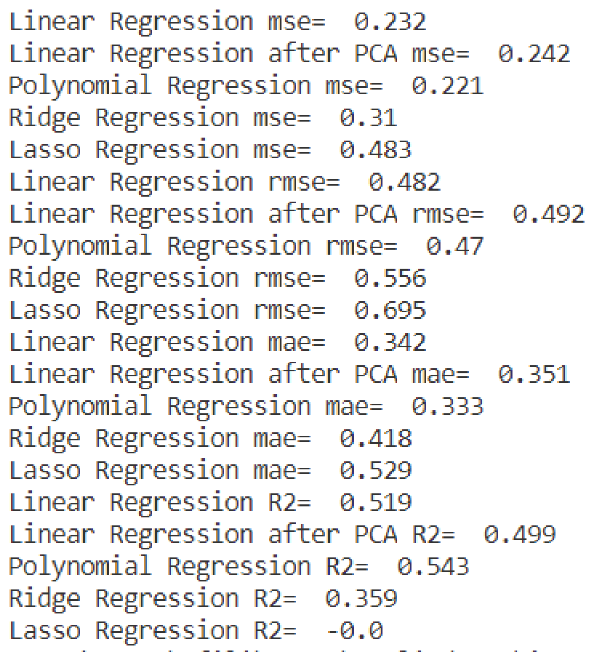
Figure 2 of part 3: Scatter plot of Price vs features

After that we performed feature selection using Principal component analysis and we decided to use 7 components which led us to an Explained Variance of 0.895 .

We then split the dataset with all the features and the dataset after performing PCA with a 50-50 train test split and we run 4 Machine Learning Regression algorithms:

- Linear Regression
- Polynomial Regression
- Ridge Regression
- Lasso Regression

We then measured the Mean Square Error(MSE), Root Mean Square Error(RMSE), Mean Absolute Error(MAE) and the R2 Score.



```
Linear Regression mse=  0.232
Linear Regression after PCA mse=  0.242
Polynomial Regression mse=  0.221
Ridge Regression mse=  0.31
Lasso Regression mse=  0.483
Linear Regression rmse=  0.482
Linear Regression after PCA rmse=  0.492
Polynomial Regression rmse=  0.47
Ridge Regression rmse=  0.556
Lasso Regression rmse=  0.695
Linear Regression mae=  0.342
Linear Regression after PCA mae=  0.351
Polynomial Regression mae=  0.333
Ridge Regression mae=  0.418
Lasso Regression mae=  0.529
Linear Regression R2=  0.519
Linear Regression after PCA R2=  0.499
Polynomial Regression R2=  0.543
Ridge Regression R2=  0.359
Lasso Regression R2=  -0.0
```

Figure 3: MSE, RMSE, MAE and R2 Score

From the above measures we can clearly see that the Polynomial Regression had the best prediction for the value of the Price

We then created two plots to visualize the residuals and the coefficients:

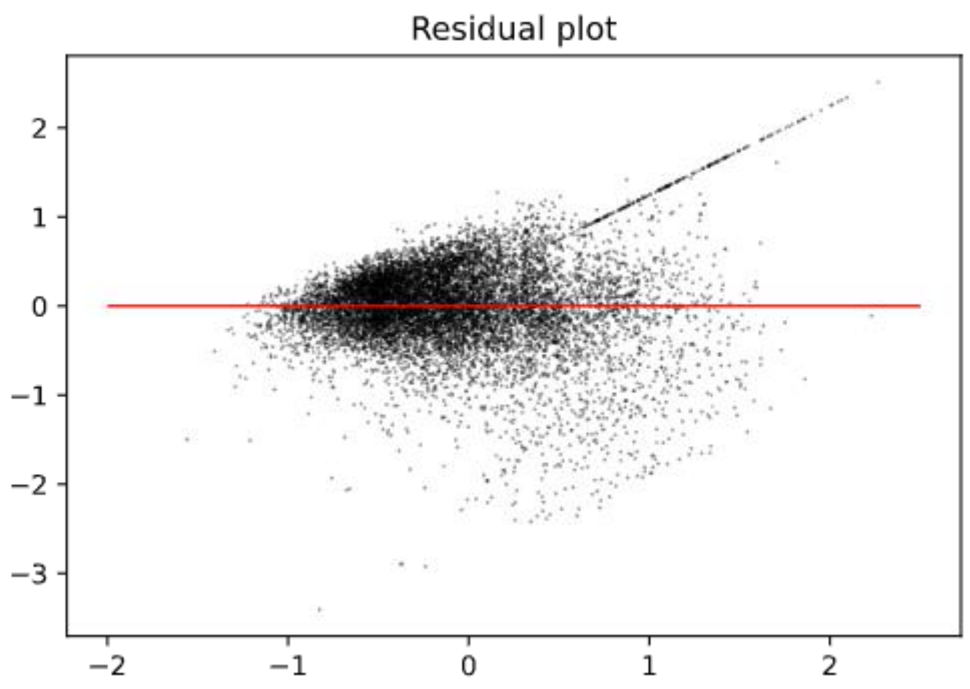


Figure 4 of part 3: Residual plot

In the residual plot of Figure 4 we have the predicted values of the price as the x-axis and the predicted values minus the actual values as the y-axis. The values on the red line, that is the 0 for the y-axis, are the ones that the predicted value is the same as the actual value. The values above the red line, positive y- axis values, show that the predicted values were lower than the actual values where the values that are below the red line, negative y-axis values, show that the predicted values were higher than the actual values.

In our case, Figure 4 exhibits "heteroscedasticity", meaning that as the prediction value increases so does the residuals. That indicates that we might have to transform a variable or that a variable from our dataset is missing.

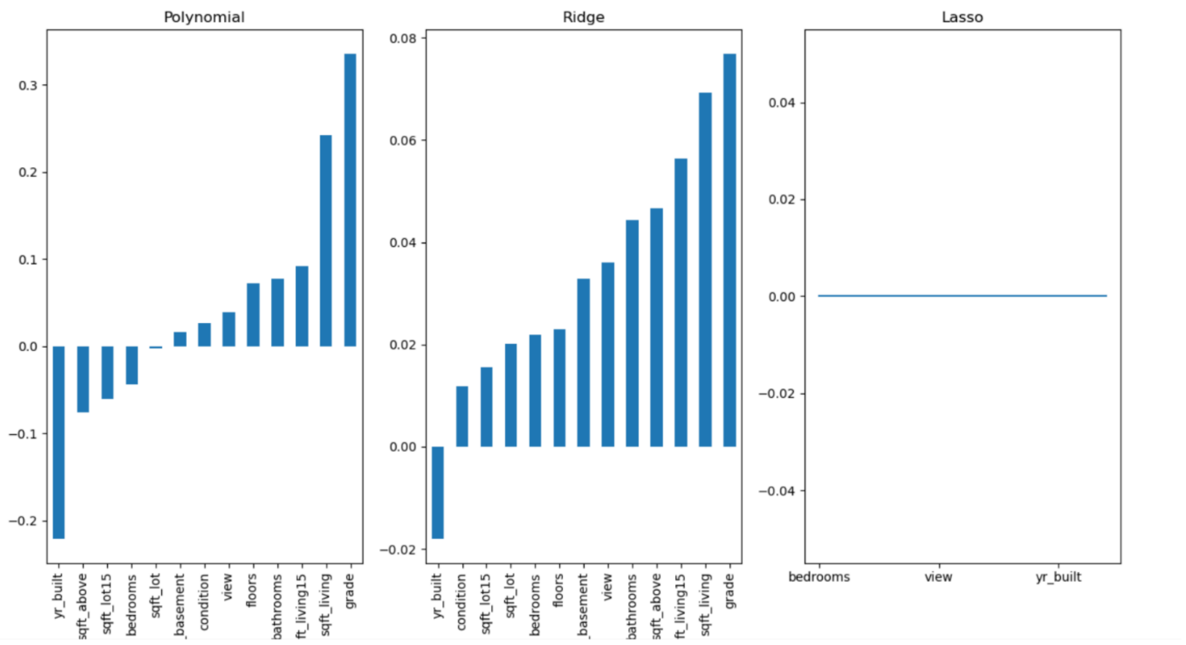


Figure 5 of part 3: Coefficients plot

In Figure 5 we can see the different values of the coefficients based on the type of regression. We can see that for the same coefficient ridge regression is penalizing the coefficients that take large values but almost leaves the same the coefficients with smaller values compared to Polynomial Regression. That is because Ridge Regression is performing a form of regularization and favors small value parameters. On the other hand, we can see that Lasso's Regression coefficient values are zero because Lasso works very similar to Ridge Regression but with a big difference. They both have a penalizing factor and favor small value parameters but, in contrast to Ridge Regression, Lasso Regression can turn many parameter values to zero. By doing that it excludes all the useless parameters and keeps the most useful ones.

PART D: Predicting buys

In part D the goal is to predict whether a user will buy a product or not based on his/her online behavior, and in particular his/her clicks during a session. There are two datases which were downloaded from `https://2015.recsyschallenge.com/challenge.html`.

yoochoose-clicks.dat: Click events. Each record/line in the file has the following fields:

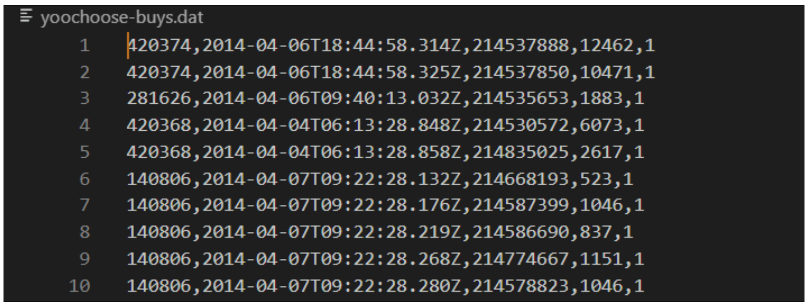
- Session ID – the id of the session. In one session there are one or many clicks.
- Timestamp – the time when the click occurred.
- Item ID – the unique identifier of the item.
- Category – the category of the item.

yoochoose-buys.dat: Buy events. Each record/line in the file has the following fields:

- Session ID – the id of the session. In one session there are one or many clicks.
- Timestamp – the time when the click occurred.
- Item ID – the unique identifier of the item.
- Price – item's price
- Quantity – how many items were bought.

Tasks to perform are:

- Build a data set that can be used in classifier to decide whether someone will buy or not.
- Preprocess the data and perform classification


yoochoose-buys.dat

```
420374,2014-04-06T18:44:58.314Z,214537888,12462,1
420374,2014-04-06T18:44:58.325Z,214537850,10471,1
281626,2014-04-06T09:40:13.032Z,214535653,1883,1
420368,2014-04-04T06:13:28.848Z,214530572,6073,1
420368,2014-04-04T06:13:28.858Z,214835025,2617,1
140806,2014-04-07T09:22:28.132Z,214668193,523,1
140806,2014-04-07T09:22:28.176Z,214587399,1046,1
140806,2014-04-07T09:22:28.219Z,214586690,837,1
140806,2014-04-07T09:22:28.268Z,214774667,1151,1
140806,2014-04-07T09:22:28.280Z,214578823,1046,1
```

Figure 15: CSV format

1 Preprocess the data

Perhaps the most challenging part was to prepare the two datasets. Both sets combined occupied memory size of 3GB and consisted of about 34 million rows. Decision was made to drop 'Category' and 'Price' columns based on the assumption that they play no major role to the number of clicks or the decision to buy or not.

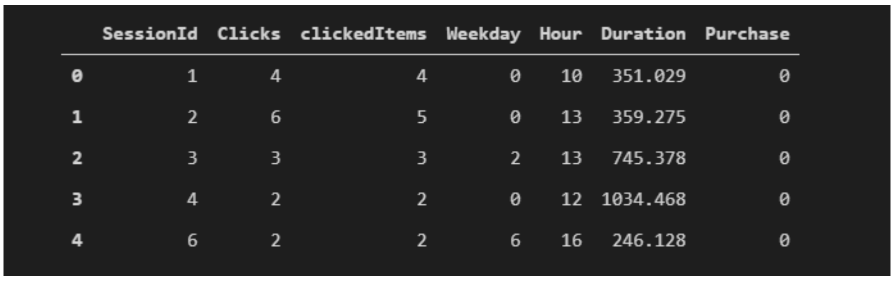
To preprocess the data and in order for the dataset to reach its final form the following steps were necessary:

1. **Dataset merging:** The two datasets were appended and indexed correctly. NaN values were replaced with zeros.
2. **Parse Timestamp column:** Timestamp feature was parsed and transformed to python datetime object.
3. **Unique items in session:** Dataset grouped-by sessions and number of unique items clicked per session was saved as a new column
4. **Aggregation:** Dataset grouped by sessions,column "Clicks" created (total number of clicks in session),column "Purchase" created. Columns "sessionStart" and "sessionEnd" created.
5. **Date-Time columns:** Weekday and hour the session commenced were extracted from the "sessionStart" using a custom function that was applied to the datasets. Furthermore the duration of session was calculated by subtracting session's "startTime" from session's

"endTime" and recalculated into seconds.

6. **Class column:** "Purchase" column was transformed to a binary class feature: no-buy:0,buy:1.

7. **Feature drop:** unecessary/no-interest features dropped.



```
   SessionId  Clicks  clickedItems  Weekday  Hour  Duration  Purchase
0          1       4             4        0    10   351.029         0
1          2       6             5        0    13   359.275         0
2          3       3             3        2    13   745.378         0
3          4       2             2        0    12  1034.468         0
4          6       2             2        6    16   246.128         0
```

Figure 16: Final dataframe

Analyzing the newly formed dataset we were able to calculate the buying average per weekday, per number of clicks and per hour of day.

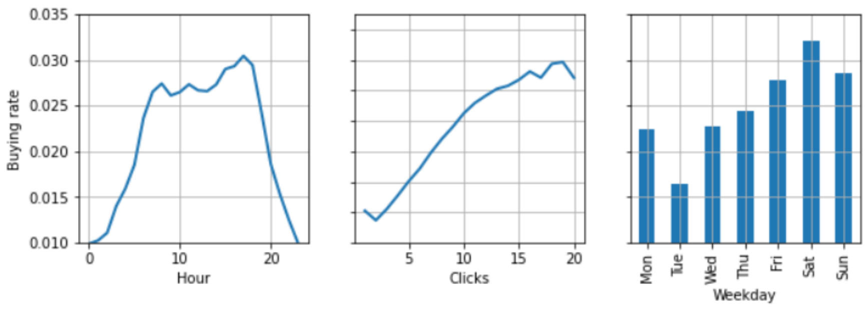


Figure 17: Buying rate

It is clear from the diagrams that people prefer to make purchases mostly on the weekends. Buying activity reaches its peak in noon and afternoon hours. Lastly more-clicks-sessions seem to end up in buys more than less-click sessions.

2 Prepare the data

After preprocessing the dataset was prepared for classification analysis. This was a highly imbalanced set with a predominant no-buy class (buy to no-buy ratio: 0.261%). Decision was made to downsample the no-buy class to produce a balanced set of equal sized classes. This resulted to a reduced dataset of 440K rows. Normalization was also performed (mean=0, std=1) to all the features for classifier optimization. One-hot encoding was initially not performed to "Weekday" and "Hour" attributes.

3 Classification

After data preprocessing and preparation the dataset was split in train and test sets (50%) in order to run classification techniques and evaluate their performance on correctly predicting buys. Classifiers chosen were:

- **Naive Bayes**
- **Decision Tree classifier** (max depth=10)
- **Neural Network** (one hidden layer/5 nodes, max-iterations:20)

Evaluation metrics are presented below:

	Evaluation metrics		
	Macro Precision	**Recall**	**F1 score**
Decision Tree	0.836	0.825	0.824
Naive Bayes	0.691	0.637	0.609
ANN	0.835	0.827	0.826

In general all techniques performed adequately. Decision tree and ANN were on par with each other resulting in outstanding predictive performance.

ROC curves are plotted for comparison against each other on the same plot depicting predictive accuracy of each classifier:

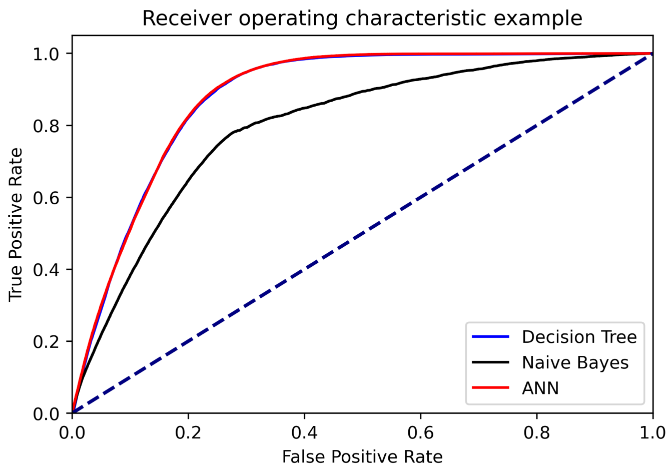


Figure 18: ROC curve

Future work includes building an SVM classifier for imbalanced classes or a k-fold validation technique to run on the whole dataset in order to take advantage of all data information. Another matter that needs further investigation is the effect of not one-hot encoding ”Weekday” and ”Hour” columns. These two attributes are ordinal (one could argue that they are categorical) but they were treated as numeric. Whether this played or not a significant role to the classification remains to be seen.

References

[1] Pang-Ning Tan, Michael Steinbach and Vipin Kumar. *Introduction to data mining*. Pearson Education Ltd, Edinburgh Gate, 2014.