Statistics for Business Analytics II

Project II - 2022-2023

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ SCHOOL OF BUSINESS

ΤΜΗΜΑ ΔΙΟΙΚΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ & ΤΕΧΝΟΛΟΓΙΑΣ DEPARTMENT OF MANAGEMENT SCIENCE & TECHNOLOGY

Dimitris Matsanganis, f2822212

Contents

Table of Figures

Assignment

The current assignment involves two parts:

Part I:

The goal is to create a predictive model to classify whether Donald Trump received more than 50% of the votes in the 2016 United States Primaries Election. The dataset includes various demographic and economic-related variables for each county in the United States. At least three distinct methods should be used to create the predictive model, and the models should be assessed based on how good the predictions are made by them.

Part II:

In this part, the focus is on clustering the counties based on their demographic-related variables. The variables used for clustering are the following:

PST045214, PST040210, PST120214, POP010210, AGE135214, AGE295214, AGE775214, SEX255214, RHI125214, RHI225214, RHI325214, RHI425214, RHI525214, RHI625214, RHI725214, RHI825214, POP715213, POP645213, POP815213, EDU635213, EDU685213, and VET605213.

Then, the economic-related variables will be used to describe the clusters that are found. The chosen clustering method and variables used for the analysis should be explained in detail.

Provided Excel file: 'stat BA II project I.xlsx'[1]

1. Introduction - Description of the Problem

The provided data of this report are based on three Excel sheets that consist of 24611 election votes results from the Democratic and Republican 2016 primaries in the US, broken down by county and each presidential candidate. The data includes socioeconomic information about 3195 counties that voted in the primaries and includes 54 variables, covering voting patterns and demographic information among others. The purpose of this report is to develop a predictive model that classifies whether Donald Trump received more than 50% of the votes in the US Presidential Election. In the current report we will utilize at least three distinct methods and assess the accuracy of each model's predictions. By doing so, we aim also to provide insights into the effectiveness of various modeling techniques and their ability to predict electoral outcomes accurately - whether Trump got over the 50% of the Republicans votes.

The second part of the problem pertains to the clustering of counties based on their demographic characteristics. The variables have been categorized into two distinct groups - demographic and economic. The demographic variables, including PST045214, PST040210, PST120214, POP010210, AGE135214, AGE295214, AGE775214, SEX255214, RHI125214, RHI225214, RHI325214, RHI425214, RHI525214, RHI625214, RHI725214, RHI825214, POP715213, POP645213, POP815213, EDU635213, EDU685213, and VET605213, will be used to cluster the counties. The objective is to employ clustering techniques to group the counties based on the demographic variables and subsequently utilize the economic variables to describe the clusters. There are no constraints on the clustering methodology, and the approach will be elaborated upon comprehensively in the ensuing sections.

2. Data Cleaning and Data Transformation

In the Data Cleaning and Transformation stage of the study, a preliminary analysis of the *'votes'* dataset was performed using both MS Excel and R. The dataset initially contained 24611 observations for both the Republican and Democratic parties. To construct a model that can explain the behavior of voters in counties where Trump received more than 50% of the Republican votes, the dataset was filtered to exclude observations related to the Democratic party and candidates. As a result of this filtering, the number of observations in the dataset was reduced to 15652, with some counties being completely removed because they only contained data for Democratic candidates (e.g. Colorado, Maine, North Dakota). With this, the *'votes'* dataset has been limited to showcasing the Republican primary results, featuring *Donald Trump* and 10 other candidates, namely *Ted Cruz, Marco Rubio, John Kasich, Ben Carson, Jeb Bush, Rand Paul, Mike Huckabee, Carly Fiorina, Chris Christie, and Jim Gilmore*.

Further analysis of the *'votes'* dataset revealed the absence of 20 Federal Information Processing Standards (FIPS) county codes for the 4 candidates in the New Hampshire (NH) state, a total of 80 missing values. As these FIPS codes are already present in the *'county_facts'* dataset, it is feasible to selectively import the missing information by accurately identifying the New Hampshire counties which are missing the FIPS code's fields. To provide a clear visual representation of the missing FIPS codes, a figure has been included in the appendix which specifically highlights the case of New Hampshire and a sample of the 80 missing values in total (*see Appendix-Figure 1*).

Furthermore, it has been noted that some states have a distinct but unknown method of assigning codes to their counties, utilizing an 8-digit encoding format instead of the standard 5-digit (or 4-digit after being processed by MS Excel) FIPS codes. During our web-based research, we were unable to locate any specific information regarding these states and their non-FIPS county codes. As a result, we focused on the 'county' column, which displays the name of each county within the state. However, for the 10 states with the non-FIPS code issue, the county column did not always provide the county name and different alternative approaches were recognized. The 10 states with the non-FIPS code issue are the following: Alaska (AK), Connecticut (CT), Illinois (IL), Kansas (KS), Maine (ME), Massachusetts (MA), North Dakota (ND), Rhode Island (RI), Vermont (VT), and Wyoming (WY).

In the case of Illinois (IL), only two entries were found to have non-proper FIPS codes. Upon further examination, it was determined that these two entries, 'Chicago' and 'Cook Suburbs', belong to Cook County. The correct FIPS code, 17031, was then discovered through the 'county_facts' dataset and assigned to both of these entries. It is worth pointing out that the representation of Cook County in Illinois in the *'votes'* dataset did not previously have a specified FIPS code (*see Appendix-Figure 9*).

The counties within the states of Maine (ME) and North Dakota (ND) (as well as Colorado (CO) - which has correct FIPS codes), were found to not have a proper FIPS code representation. However, due to the fact that these states only contain data for Democratic candidates, they were considered to have limited significance for the current analysis. As a result, it was decided to not pursue further investigation into these states and to exclude them from the analysis during the filtering stage, as they were found to only contain data for the Democratic party (*see Appendix-Figure 5, 6, and 8*).

The counties belonging to the states of Connecticut (CT), Massachusetts (MA), Rhode Island (RI), and Vermont (VT) were found to be inaccurately represented as cities/towns, instead of counties, in the *'votes'* dataset. To resolve this issue, four separate CSV files were assembled through web-based research that contained the relationships between counties and cities/towns. These CSV files were used to correctly assign the counties to the appropriate FIPS codes using R and the *'county_facts'* and *'votes'* datasets (*see Datasets Archive for the four CSVs*). This process was repeated four times, one for each state, resulting in a grouped and summarized dataframe for each state. This was similar to the procedure applied to the rest of the data, and the final dataframe for each state was merged vertically with the '*votes*' dataframe (using the rbind command) to create the final *'votes'* dataframe (*see Datasets Archive, us_elections_2016.csv and Appendix-Figure 2*, that depicts the Vermont (VT) case).

For the states of Alaska (AK), Kansas (KS), and Wyoming (WY), it was not possible to accurately assign the votes to individual counties due to the nature of the provided data collection. The data for Alaska pertained to State House districts, while the data for Kansas pertained to Congressional districts, both of which can encompass multiple counties. As a result, it was not possible to determine a definitive assignment of the votes to individual counties within these districts. For example, it is unclear how to allocate the votes collected for Kansas Congressional District 1, which comprises the counties of Allen, Anderson, Bourbon, Chautauqua, Cherokee, Crawford, Labette, Linn, Miami, Montgomery, Neosho, and Wilson. Finally, in the case of Wyoming (WY), the data provided pertains to county pairs, such as *Uinta-Lincoln*. Thus, the results based on county pairs may not accurately represent a unified vote outcome, as they do not take into account individual county results. Therefore, it has been decided to exclude the data for these three states from the final FIPS code per county level dataset, as it is not possible to retain them in a manner that meets the criteria for inclusion (*see Appendix-Figure 3, 4, and 7*).

After creating separate data frames for Connecticut (CT), Massachusetts (MA), Rhode Island (RI), and Vermont (VT) that contain data related to counties and their corresponding FIPS codes, we would proceed to process the remaining counties in the *'votes'* dataset that have a valid FIPS code (we have created a copy of *votes* as a dataframe, named *us_elections_2016*). Then through R and with the assistance of the *dplyr* library, we select the columns *'fips'*, *'candidate'*, and *'votes'* from the *us_elections_2016* dataframe using the select() function. Next, with the group_by() function we group the data by the '*fips*' and *'candidate'* columns. We use the summarize() function to calculate the sum of *'votes'* for each group of *'fips'* and *'candidate'*. The result is a new dataframe (assigned to the same dataframe, *us_elections_2016*) with columns for *'fips'*, *'candidate'*, and *'Total_Votes'*. Finally, the spread() function is used to reshape the data set so that each unique value in the *'candidate'* column becomes a new column, with each candidates total votes per county. This is way we create a wider data format that is easier to read and analyze. To sum up, this procedure is preparing the data for further analysis by aggregating the total votes for each candidate in each county (as identified by *'fips'* column) and reshape the data in a more convenient format. We need to point out that the same procedure is used for each of the four separated dataframes of the four above-mentioned states. Thus, with a rbind() command we can combine the main dataframe, *us_elections_2016* with each of the separated four dataframes vertically (prior to the vertical merge we drop the observations with non-proper and more than 5-digits FIPS codes from the *us_elections_2016* dataframe).

At this point, the four separate dataframes containing state-level data were no longer necessary for our analysis and were removed. Next, we calculated the total number of votes cast for Republican candidates in each county. This was achieved using the *rowSums()* function, which summed the vote counts for each Republican candidate in the *us_elections_2016* dataframe, and stored the results in a new column called *'republicans_votes'*. Subsequently, a binary variable was computed to indicate whether Donald Trump won a majority of votes (over 50%) in each county. This was accomplished by using the *ifelse()* function to create a new column called '*trump_over_perc50*', which takes a value of 1 if the number of votes for Donald Trump exceeds 50% of the total votes for Republican candidates in that county, and 0 otherwise. Then, with the *select()* function is used to subset the data frame and keep only the specified columns and drop the unnecessary ones (the 11 Republicans candidates and the *republicans_votes*). The resulting data frame will have only the *'fips'* and *'trump_over_perc50'* columns, which contain the 5-digit FIPS codes and binary variable indicating whether Donald Trump won a majority of the Republicans votes (over 50%) in each county.

In the final step of data cleaning and data transformation phase, we merged the *'county_facts'* dataframe containing socioeconomic variables at the county level, with the *'us_elections_2016'* dataframe containing election results at the county level. The merge was conducted based on a common identifier, which is the 5-digit FIPS code (or the 4-digit FIPS code, where the leading zero was removed from the dataset creation tools, as non-significant digit of a regular number) for each county.

This merge resulted in the creation of a new dataframe called again for simplicity purposes, *'us_elections_2016'*. By combining these two dataframes, we can investigate the relationships between various socioeconomic factors and voting patterns in the 2016 US Republicans primaries election. Finally, the *'county_facts'* dataframe was no longer necessary for our analysis and was removed, while we export the *'us_elections_2016'* dataframe as *'us_elections_2016FINAL.csv' (see Datasets Archive)*.

3. Data Partitioning

Data Partitioning is the process of dividing a dataset into two or more subsets, usually a training set and a test set. The training set is used to train the model, while the test set is used to evaluate the performance of the model. The purpose of data partitioning is to avoid overfitting, which occurs when the model is too complex and fits the training data too closely, resulting in poor performance on new data.

By partitioning the data into a training set and a test set, we can evaluate the performance of the model on new, unseen data. However, by simply splitting the data into two sets can result in different results each time the model is run. To address this, we are using the trainControl function to set up a standardized and robust way to train and evaluate the models.

The method used here is repeated K-fold cross-validation, where the data is divided into K-folds, and the model is trained and evaluated k times. The results are then averaged to obtain a more reliable estimate of the model's performance.

Additionally, we are setting the random seed to ensure reproducibility of the analysis, which means that the same results will be obtained each time the code is run. This way, we can be more confident in the accuracy of the results and avoid overfitting of the models to the training data.

Overall, by partitioning the data and using techniques like cross-validation, we can ensure that our models are being trained and evaluated in a consistent and robust way, and the results are reliable and reproducible.

4. Part I: Classification

Classification is a fundamental task in machine learning, which involves assigning a label or a category to a given input or observation. The goal of classification is to build a model that can accurately predict the label of new, unseen data based on patterns and relationships identified in the training data. Classification has many practical applications in various fields, such as image recognition, spam filtering, credit scoring, medical diagnosis, and sentiment analysis.

To build a classification model, we typically start with a labeled dataset, where each data point is associated with a class label. We use this data to train a model that can learn the underlying patterns and relationships between the input features and the output labels. Once the model is trained, we can use it to predict the labels of new, unseen data and evaluate its performance through metrics such as accuracy, precision, recall, and ROC curve among others.

4.1. Logistic Regression

Logistic regression is a statistical method used for modeling the relationship between a binary dependent variable - a variable with two possible outcomes, and one or more independent variables. It is a type of generalized linear model that uses a logistic function to model the probability of the dependent variable being in one of the two categories, based on the values of the independent variables. The logistic function, also known as the sigmoid function, maps any input value to an output value between 0 and 1, representing the predicted probability of the dependent variable being in the positive class - usually 1.

The logistic regression model estimates the coefficients of the independent variables, which indicate the direction and magnitude of their effects on the log odds of the positive class. These coefficients can be exponentiated to obtain odds ratios, which represent the multiplicative change in the odds of the positive class for a one-unit increase in the corresponding independent variable. Logistic regression is widely used in various fields, such as medicine, epidemiology, social sciences, marketing, and finance, for analyzing binary outcomes and making predictions about future events. It has several advantages over other classification methods, such as simplicity, interpretability, and flexibility to handle both linear and nonlinear relationships between the independent and dependent variables.

Codewise, we start our procedure by constructing the full model using the *glm* function with binomial family, where *trump_over_perc50* is the dependent variable and all other variables in the train_data dataset are included as independent variables. Then, the stepwise selection procedure is used to select the best subset of variables based on the Akaike Information Criterion (AIC), which balances the model's goodness of fit with the complexity of the model. The step function is used to iteratively add or remove variables from the model based on the AIC criterion until the best model is found.

Afterwards, the *vif* function from the *car* package is used to check for multicollinearity issues among the independent variables in the model_aic. Multicollinearity occurs when two or more independent variables in a logistic regression model are highly correlated, which can cause them to share variation and make it difficult for the model to distinguish their unique effects on the dependent variable. This can result in inflated standard errors, reduced statistical power, and unstable coefficients, which can lead to incorrect inferences and predictions.

By addressing multicollinearity, you can improve the accuracy and reliability of the predictive model, by ensuring that the coefficients are more stable and interpretable, and that the model is able to capture the unique effects of each independent variable on the dependent variable. The Variance Inflation Factor (VIF) values for each independent variable are calculated, and those with the highest VIF values are sequentially removed from the model using the *update* function until all VIF values are below a threshold of 10 (*see Appendix-Figure 16*). To be more precise we removed the following variables (ordered by their removal sequence): PST040210, BZA010213, POP010210, BZA110213, POP645213, and INC910213. After the removal of the aforementioned variables the multicollinearity issue was resolved, and the model's formula is the following (*model_aic - for more details see Appendix-Figure 21*):

trump_over_perc50 ~ state_abbreviation + PST040210 + POP010210 + AGE295214 + RHI525214 + RHI625214 + POP645213 + POP815213 + EDU635213 + EDU685213 + VET605213 + HSG096213 + INC910213 + INC110213 + PVY020213 + BZA010213 + BZA110213 + NES010213 + SBO315207 + MAN450207 + AFN120207 + LND110210

Then, the model built previously is evaluated and metrics such as the Wald test, Likelihood Ratio Test, and pseudo-R2 (McFadden, CoxSnell, and Nagelkerke) are calculated. More specifically, the Wald test resulted in a chi-squared value of **128.7** with **17** degrees of freedom, and a p-value of **0**, indicates that at least one of the coefficients in the model is significantly different from zero. In other words, the results of a Wald test does not necessarily mean that all coefficients in the model are insignificant.

Then, the Likelihood Ratio Test (LRT) results indicate that the full model is a significantly better fit than the null model (which only includes the intercept), as the p-value for the overall model fit is very close to **0**. This means that the additional variables in the full model (as compared to the null model) are able to explain a significant amount of unexplained variability in the data. The p-value for the goodness of fit of the full model is **1**, indicating that the full model provides a good fit to the data. Overall, these results suggest that the full model is a good fit for the data and that it provides a significantly better fit than the null model (*see the results of LRT* Appendix-Figure 22).

Regarding the output of R2 computations the three different measures of pseudo R-squared were the McFadden's R-squared, Cox & Snell's R-squared, and Nagelkerke's R-squared. These are measures of how well the model fits the data compared to a null model. The McFadden's R-squared compares the log-likelihood of the full model to the log-likelihood of the null model. It ranges from 0 to 1, with higher values indicating a better fit. In this case, the McFadden's R-squared is 0.675, which means that the model explains **67.5%** of the variance in the data, compared to a null model.

The Cox & Snell's R-squared measure is similar to McFadden's R-squared, but is based on the maximum likelihood estimator of the standard deviation of the error term. It ranges from 0 to 1, with higher values indicating a better fit. In this case, the Cox & Snell's R-squared is 0.6, which means that the model explains **60%** of the variance in the data, compared to a null model. Finally, the Nagelkerke's R-squared measure is a rescaled version of Cox & Snell's R-squared, and ranges from 0 to 1. In this case, the Nagelkerke's R-squared is 0.808, which means that the model explains **80.8%** of the variance in the data, compared to a null model. To summarize the R-squared computations suggest that the model has a good fit and explains a substantial amount of the variance in the data, compared to a null model (*see the results of R2 computations* Appendix-Figure 23).

Afterwards, we are calculating odds ratios for the coefficients in the AIC model (model_aic). Odds ratios represent the multiplicative change in the odds of the outcome variable for a one-unit increase in the corresponding predictor variable, while holding all other predictors constant. We are also computing the odds ratios along with their 95% confidence intervals (*see the exponentiation of the coefficients computations* Appendix-Figure 24).

Then, the logistic regression model was trained using the *train* function to predict the response variable *trump_over_perc50* based on several predictor variables. The model was trained using a binomial family and glm method, and the training data was split into training and testing sets using the *trainControl* function - which mentioned on the previous section. The model's performance was further evaluated using three common metrics of model performance for binary classification tasks, and they can be used to assess the predictive ability of the model.

More specifically, RMSE (Root Mean Squared Error) is a measure of the average difference between the predicted probabilities and the actual binary outcomes. The lower the RMSE value, the better the model performance. In this case, the RMSE value is about **0.28**, which means that on average, the predicted probabilities are off by 0.28 units from the actual binary outcomes. Rsquared (Coefficient of Determination) is a measure of how well the model fits the data, and it ranges from 0 to 1, the Rsquared value of the current model is 0.6549, which means that the model explains **65.49%** of the variability in the data.

Finally, the MAE (Mean Absolute Error) is a measure of the average absolute difference between the predicted probabilities and the actual binary outcomes. The lower the MAE value, the better the model performance. In this case, the MAE value is **0.159**, which means that on average, the predicted probabilities are off by 0.159 units from the actual binary outcomes. Therefore, the performance of the model can be characterized as decent but not great, as there is room for improvement in all three metrics (*to have a look at the above mentioned along with other metrics see* Appendix-Figure 25).

To further evaluate the predictive ability of the logistic regression model we are using the *predict* function to generate predictions for the test set (*test_data*), and then converting these probabilities into binary classifications using a threshold of 0.5. We are then creating a confusion matrix using the predicted and actual classifications, and computing metrics such as accuracy and a more detailed confusion matrix using the *confusionMatrix* function.

To be more precise, the confusion matrix shows the classification results of the logistic regression model with actual versus predicted values. The model made a total of 828 predictions, with 462 true negatives (0,0), 271 true positives (1,1), 48 false negatives (1,0), and 47 false positives (0,1).

The confusion matrix and statistics provide information about the performance of a classification model. In this case, the model appears to have an accuracy of 0.8853, which means it correctly classified **88.53%** of the observations in the test data. The sensitivity of the model is 0.8522, which means it correctly identified **85.22%** of the positive class (*Trump to win the majority*) observations. The specificity of the model is 0.9059, which means it correctly identified **90.59%** of the negative class (class 0) observations.

The positive predictive value (PPV) of the model is 0.8495, which means that when the model predicts a positive class observation, it is correct **84.95%** of the time. The negative predictive value (NPV) of the model is 0.9077, which means that when the model predicts a negative class observation, it is correct **90.77%** of the time. The Kappa statistic is **0.7576**, which measures the agreement between the model's predictions and the actual labels. A Kappa value of 1 indicates perfect agreement, while a value of 0 indicates no agreement beyond chance. In summary, the logistic regression model performed well with a high accuracy, sensitivity, and specificity (for more details *see* Appendix-Figure 19).

Finally, we are generating a ROC curve using the *prediction* and *performance* functions, and plotting the curve. The purpose of the ROC curve is to evaluate the performance of the classifier across a range of threshold values, and the area under the curve (AUC) is often used as a summary metric of the overall performance of the model. The ROC curve follows below:

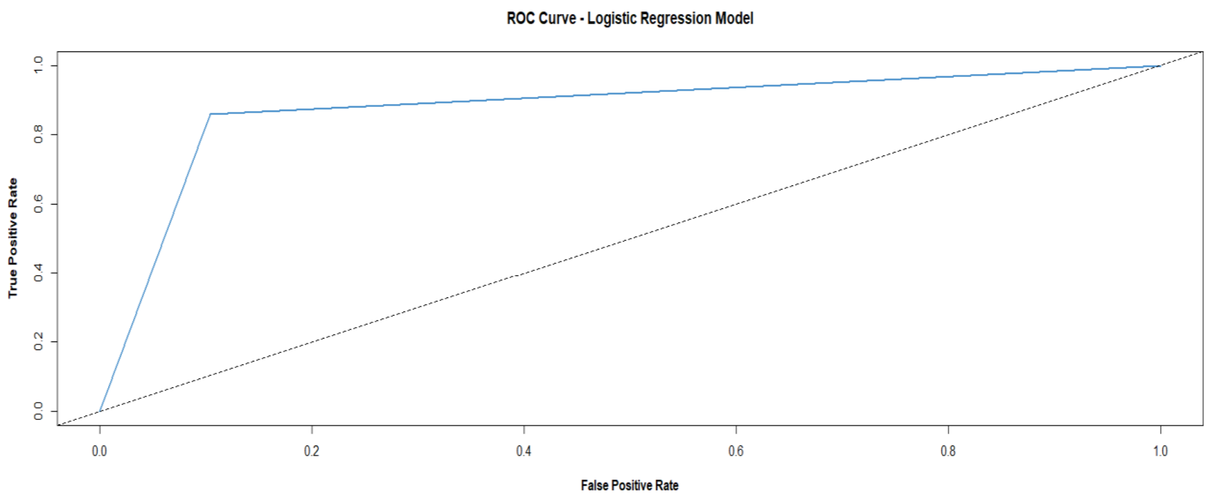


Figure 1: ROC curve of Logistic Regression model (model_log).

4.2. Decision Tree

Decision Trees are a popular machine learning technique used for solving classification and regression problems. They provide a visual and intuitive representation of the decision-making process by recursively partitioning the data based on the most relevant features. In this section, we will try and evaluate the Decision Tree model as a classification method.

The procedure starts with setting a random seed to ensure reproducibility of the followed procedure. The data is then divided into training and testing sets, with the training data used to fit the decision tree model, and the testing data used to evaluate model performance. A decision tree model is created using the training data, where the response variable is *trump_over_perc50*, and all other available variables are used as independent variables.

The complexity parameter table is printed to determine the optimal tree size and output the rules used to build the decision tree model. From it we can figure out that, the variables used in the model are EDU685213 (Bachelor's degree or higher, percent of persons age 25+), HSD310213 (Persons per household), INC110213 (Median household income), and state_abbreviation. The root node error is calculated as the number of misclassifications divided by the total number of observations, resulting in an error rate of about **0.41**. The output also shows the complexity parameter (CP) values and the relative error and cross-validation error rates associated with each split in the decision tree. The nsplit column indicates the number of splits, and the rel error column shows the relative error rate for each split. The xerror and xstd columns indicate the cross-validation error and its standard deviation. Based on this output, it appears that the second split with a CP value of 0.01875 is the most optimal for this decision tree model, as it results in the lowest cross-validation error rate of about 0.38 (*see* *Appendix-Figure 26*).

The importance of each independent variable in the model is also output. To be more precise, the importance of each covariate is expressed as a numeric value, where higher values indicate greater importance. The covariates with the highest importance value in this output is state_abbreviation, followed by LND110210 (Land area in square miles), HSG495213 (Median value of owner-occupied housing units), EDU685213 (Bachelor's degree or higher, percent of persons age 25+), and INC910213 (Per capita money income in past 12 months), in decreasing order of importance (*see* *Appendix-Figure 27*).

The model is then used to predict the outcomes of the test data, excluding the dependent variable, using the decision tree model. A 2x2 confusion matrix is created to evaluate the accuracy of the model by comparing the predicted outcomes to the actual outcomes in the test data. The confusion matrix presents the Decision Tree model's performance in more detail, showing that the model correctly predicted 447 observations as negative and 277 observations as positive. However, the model misclassified 41 observations as negative, which were actually positive, and 63 observations as positive, which were actually negative.

The decision tree model metrics achieved an overall accuracy of **0.8744**, with a 95% confidence interval ranging from **0.8499 to 0.8962**. This indicates that the model's predictions were correct for **87.44%** of the total observations. The Kappa score for the model was **0.7379**, which suggests substantial agreement between the predicted and observed outcomes. The sensitivity of the model is 0.8711, which means that the model correctly identified **87.11%** of the positive cases. The specificity of the model is 0.8765, which means that the model correctly identified **87.65%** of the negative cases. The positive predictive value (PPV) of the model is 0.8147, which means that out of all the predicted positive cases, **81.47%** were actually positive. The negative predictive value (NPV) of the model is 0.916, which means that out of all the predicted negative cases, **91.6%** were actually negative. (*see* *Appendix-Figure 20*).

Overall, the Decision Tree model showed - through its metrics a promising performance in predicting binary outcomes. The model achieved a high level of accuracy and showed substantial agreement with the observed outcomes. However, there is still room for improvement, especially in reducing the number of false negatives and false positives. Further refinement of the model and its parameters may be necessary to achieve even higher levels of accuracy and performance - this will not be further investigating during this assignment, and we will stay with the current model (*dt_model*).

A Decision Tree model's visual representation as well as the ROC curve follows below:

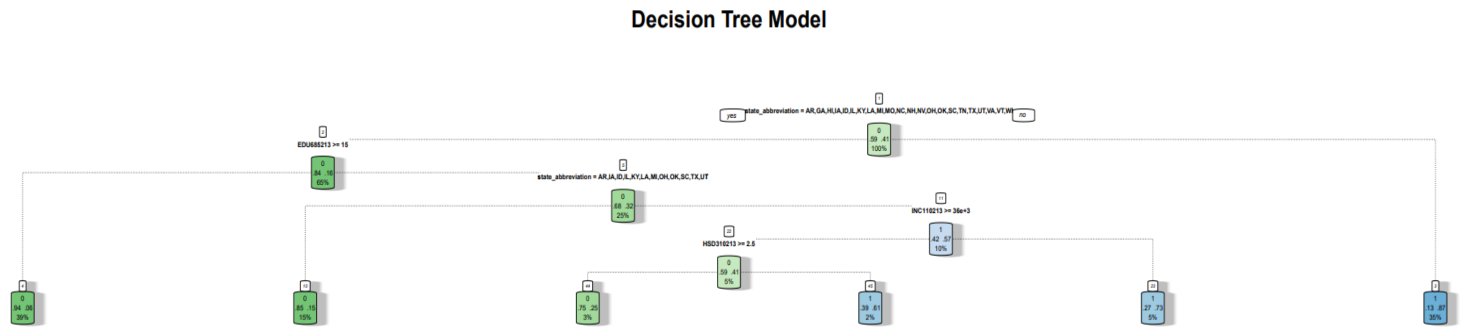


Figure 2: Decision Tree model (dt_model) visual representation.

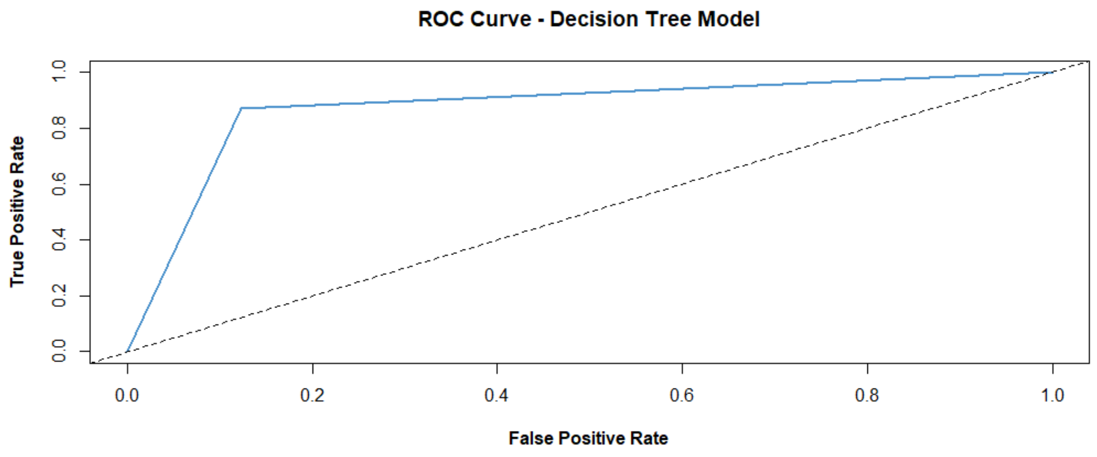


Figure 3: ROC curve of Decision Tree model (dt_model).

Finally, we can interpret the visual representation of the tree through the model's rules. More specifically, if the **state is AR, GA, HI, IA, ID, IL, KY, LA, MI, MO, NC, NH, NV, OH, OK, SC, TN, TX, UT, VA, VT, or WI**, and the percentage of people with a Bachelor's degree or higher (**EDU685213**) is **greater than or equal to 15%**, then *trump_over_perc50* is assigned a value of **0.06**.

If the **state is AR, IA, ID, IL, KY, LA, MI, OH, OK, SC, TX, or UT**, and the percentage of people with a Bachelor's degree or higher (**EDU685213**) is **less than 15%**, then *trump_over_perc50* is assigned a value of **0.15**.

If the **state is GA, MO, NC, NV, TN, VA, or WI**, and the percentage of people with a Bachelor's degree or higher (**EDU685213**) is **less than 15%**, the median household income (**INC110213**) is **greater than or equal to 35,805 dollars** (about 36,000 - as showed on the *Figure 2*), and the number of persons per household (**HSD310213**) is **greater than or equal to 2.5**, then *trump_over_perc50* is assigned a value of **0.25**.

If the **state is GA, MO, NC, NV, TN, VA, or WI**, and the percentage of people with a Bachelor's degree or higher (**EDU685213**) is **less than 15%**, the median household income (**INC110213**) is **greater than or equal to 35,805 dollars**, and the number of persons per household (**HSD310213**) is **less than 2.5**, then *trump_over_perc50* is assigned a value of **0.61**.

If the **state is GA, MO, NC, NV, TN, VA, or WI**, and the percentage of people with a Bachelor's degree or higher (**EDU685213**) is **less than 15%**, and the median household income (**INC110213**) is **less than 35,805 dollars**, then *trump_over_perc50* is assigned a value of **0.73**.

If the **state is AL, AZ, CA, CT, DE, FL, IN, MA, MD, MS, MT, NE, NJ, NM, NY, OR, PA, RI, SD, WA, or WV**, then *trump_over_perc50* is assigned a value of **0.87**. This happens probably since Trump win by a huge margin on these states.

Overall, from the *Figure 2* and the Decision Tree model's rules indicate that certain states and certain values of the covariates have a strong impact on the predicted value of *trump_over_perc50*. For example, a higher percentage of people with a Bachelor's degree or higher (**EDU685213**) **generally leads to a lower predicted value of trump_over_perc50**. Similarly, a higher median household income (**INC110213**) **generally leads to a lower predicted value of trump_over_perc50** (*for a better understanding see the text format of the rules Appendix-Figure 28*).

4.3. Naive Bayes

Afterwards, we are building a Naive Bayes classifier to predict the outcome of the 2016 US Presidential election. The Naive Bayes model is a probabilistic algorithm that works on the principle of Bayes' theorem, which calculates the probability of an event occurring based on prior knowledge of related events. Furthermore, Naive Bayes uses Bayes' theorem to predict the probability of a particular data point belonging to a certain class. It is a simple and efficient algorithm that works well in a variety of settings, including natural language processing, spam filtering, and image recognition. The "naive" in its name comes from the assumption that all features in the data are independent of each other, which is often not true in real-world data (**and in our case too**). Despite this limitation, Naive Bayes remains a popular and effective algorithm due to its simplicity and speed.

Codewise, we first constructed the Naive Bayes model using the *naiveBayes* function from the *e1071* library. The y argument is set to the response variable, which is whether Donald Trump got the majority of Republicans votes in each county, while the x argument is set to all the other variables in the dataset. The model is trained on the training set *train_data*, which was previously created using data partitioning techniques to ensure that the model is robust and accurate.

Based on the confusion matrix, the model correctly predicted 496 instances of class 0 and 308 instances of class 1, while making 10 false predictions for class 0 and 14 false predictions for class 1.

The accuracy of the model is 0.971, meaning that it correctly classified **97.1%** of the instances in the dataset. The 95% confidence interval for the accuracy is **(0.9572, 0.9813)**. The Kappa statistic is 0.9389, which measures the agreement between the predicted and actual classes while taking into account the agreement that would be expected by chance. To be more precise, the Kappa statistic indicates very high agreement between the predicted and actual classes.

The sensitivity of the model is 0.9686, indicating that it correctly identified **96.9%** of the positive instances in the dataset. The specificity is 0.9725, indicating that it correctly identified **97.2%** of the negative instances. The positive predictive value is 0.9565, indicating that when the model predicted a positive instance, it was correct **95.7%** of the time. The negative predictive value is 0.9802, indicating that when the model predicted a negative instance, it was correct about **98%** of the time.

Overall, it appears that the Naive Bayes model performed very well on this dataset, with high accuracy, Kappa statistic, and sensitivity/specificity values (*see Appendix-Figure 18*). Then, we output the ROC curve for this model:

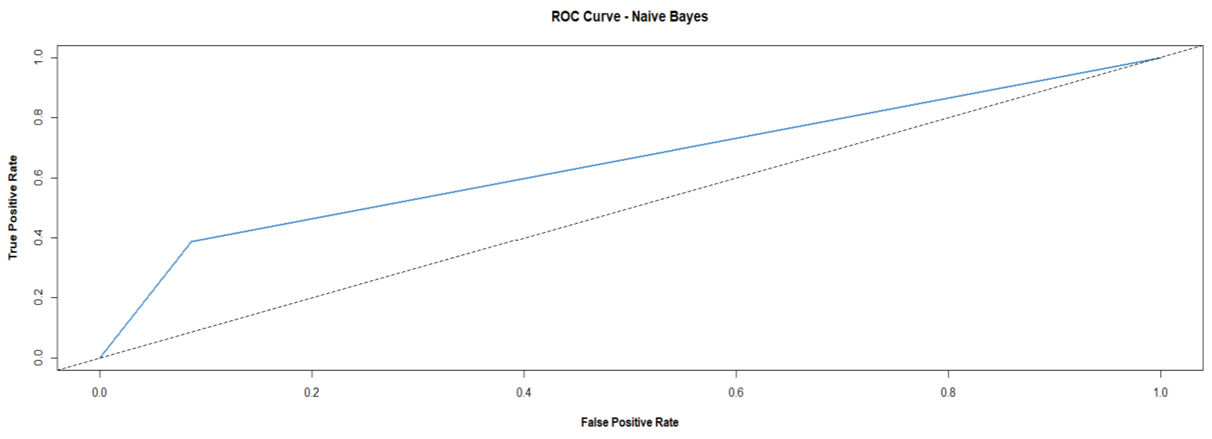


Figure 4: ROC curve of Naive Bayes model (nb_model).

Summarizing, the model seems to be performing fair enough, but there is certainly room for improvement in terms of specificity and predictive values. It may be worth exploring different feature sets or tuning the model hyperparameters to improve performance, which will not be done during this assignment.

Important note: There are several limitations and assumptions regarding the Naive Bayes classification algorithm that we accept in order to implement this simple applicable algorithm. To be more precise, the **independence assumption is violated** in our problem since **not** all features are independent of each other (true in many real-world scenarios). In such cases, the accuracy of the Naive Bayes model can be affected. Moreover, Naive Bayes requires a **relatively large amount of data** to build an accurate model we have **less than two thousand** (1935) in our train dataset.

But most important the Naïve Bayes **is very sensitive to outliers and normal assumption especially with the continuous variables is also violated** (especially due to natural limitations). To be more precise, Naive Bayes assumes that the data is normally distributed. For these reasons we should be very careful in the final models' comparisons.

4.4. Support Vector Machines (SVM)

For starters, Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It works by finding the best hyperplane in a high-dimensional space that separates different classes or groups of data points. The hyperplane that maximizes the margin between the classes is chosen as the optimal one. SVM has been widely used in various applications such as image classification, text classification, and bioinformatics.

The first thing we do prior to the construction of the SVM model is to check the dataset's dimensions and scale the data. The train and dataset's dimensions have been evaluated and fulfill the requirements of the algorithm (*see Appendix-Figure 10*), thus we move forward to scale the data. To be more precise, scaling the data prior to fitting an SVM model can be important for two main reasons. First of all, to ensure that all predictors are on the same scale. More specifically, SVM models work best when all predictors are on the same scale, as this prevents any one predictor from dominating the optimization of the SVM algorithm. In other words, predictors with larger numerical ranges will have a larger influence on the SVM model than those with smaller numerical ranges. Scaling the data ensures that all predictors are on the same scale, so they all have equal influence on the model.

Then a code procedure starts and trains the Support Vector Machine (SVM) model using the *svmLinear* method to classify data into two classes based on the response variable *trump_over_perc50*. The training data is used to train the model, while the unseen data is used to test the accuracy of the model. The response variable *trump_over_perc50* is converted to a factor. Also, the formula specifies that all the other variables in the data set should be used to predict the response variable. The *trainControl* argument is used to specify the training control options.

Afterwards, the trained SVM model is used to predict the class labels of the test data set (excluding the response variable) and confusion matrix is created to compare the predicted class labels with the actual class labels of the test data set. More analytically, the SVM model has made 828 predictions in total, with **463 true negatives** (predicted 0 and actually 0), **271 true positives** (predicted 1 and actually 1), **47 false positives** (predicted 1 but actually 0), and **47 false negatives** (predicted 0 but actually 1) *(for more details see Appendix-Figure 17)*.

The overall accuracy of the model is 0.8865, which means that the model correctly predicted the class of **88.65%** of the observations in the test set. The 95% confidence interval for the accuracy is **(0.8629, 0.9073)**. The "No Information Rate" is the accuracy that could be achieved by always predicting the majority class, which in this case is 0. The No Information Rate is **0.6159**, which means that the model is performing much better than simply guessing the majority class.

The Kappa statistic is a measure of agreement between the model's predictions and the actual classes, adjusted for chance agreement. The Kappa statistic for this model is **0.76**, indicating substantial agreement between the model's predictions and the actual classes.

The model has a high specificity of **90.78%**, which means that it correctly predicted the negative class label (0) for 90.78% of the samples. However, the sensitivity of the model is **85.22%**, which means that it correctly predicted the positive class label (1) for 85.22% of the samples. The positive predictive value (PPV) of the model is **85.22%**, which means that out of all the samples predicted to be positive by the model, 85.22% of them are actually positive. The negative predictive value (NPV) of the model is 90.78%, which means that out of all the samples predicted to be negative by the model, **90.78%** of them are actually negative. The prevalence of the positive class label in the dataset is **38.41%**, which means that 38.41% of the samples are labeled as positive. The detection rate of the model is **32.73%**, which means that the model correctly predicted 32.73% of the positive samples.

Overall, the SVM model has a high accuracy and specificity but a lower sensitivity, which means that it performs well in predicting the negative class label but not as well in predicting the positive class label. Furthermore, the ROC curve is plotted using the prediction and performance functions from the ROCR package. The ROC curve is a graphical representation of the performance of a binary classifier system as the discrimination threshold is varied. The true positive rate (TPR) is plotted on the y-axis, and the false positive rate (FPR) is plotted on the x-axis. The SVM model look like performing well according to the following ROC curve plot (*Figure 5*).

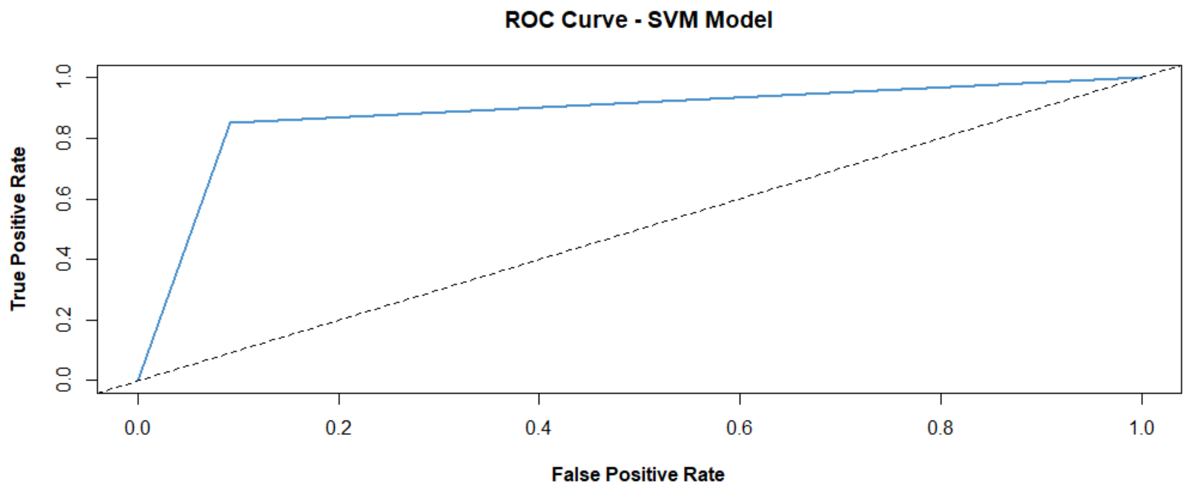


Figure 5: : ROC curve of SVM model.

4.5. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a popular machine learning algorithm used for classification tasks. It belongs to the category of instance-based learning algorithms, which means that it doesn't learn a model from the training data but instead memorizes the training dataset. KNN algorithm works by calculating the distance between the input data point and all the other data points in the training set. It then selects the K-nearest data points based on the calculated distance. The classification of the input data point is determined by the majority class of the K-nearest neighbors.

Furthermore, the KNN algorithm can be regarded as a simple and easy-to-understand algorithm. However, it can be computationally expensive for large datasets and doesn't work well with high-dimensional data. It also requires careful selection of the value of K to avoid underfitting or overfitting.

To be more practical, we are performing KNN classification. Thus, before applying KNN, we first checked the dimensions of the training and testing datasets to ensure they are equal (*see Appendix-Figure 10*). Then, we scaled the data to ensure that all the features are given equal weight in the distance metric, resulting in better performance of the KNN algorithm (excluding the factor and the response variable - state_abbreviation and trump_over_perc50).

Next, we selected the optimal number of nearest neighbors (NN) to use in the KNN algorithm. To do so, we defined a grid of K values ranging from 1 to 20 and used 10-fold cross-validation to evaluate the performance of the KNN algorithm for each value of K. We fit the KNN model with different values of K and tuned the hyperparameters using the *train* function from the *caret* library.

The results showed that the optimal number of nearest neighbors for the KNN algorithm is 5 (closely followed by 4 NN - *see the detailed results on the Appendix-Figure 11*). However, we will perform KNN classification with both scaled and unscaled datasets using 5 and 4 NN, and evaluate the performance of the KNN algorithm (*see Figure 6*).

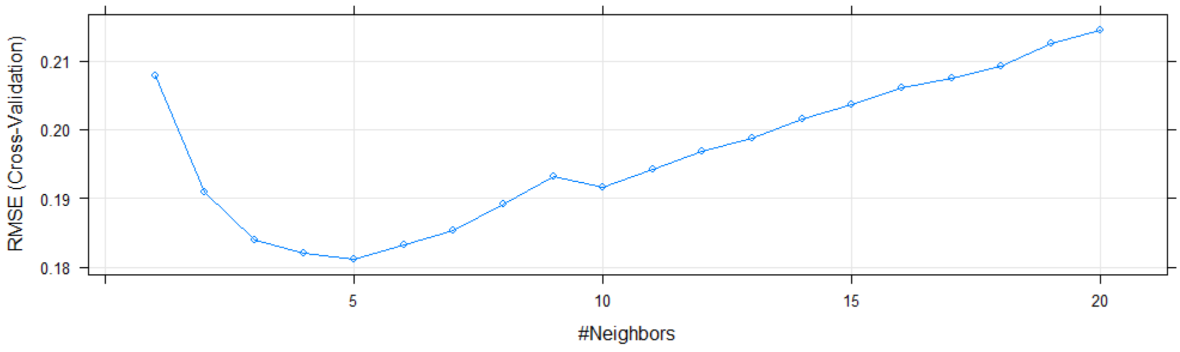


Figure 6: Select the optimal number of NN between 1-20 using a 10 - Fold CV.

K-Nearest Neighbors (KNN) with K = 5 on Scaled Data

We start by applying the KNN classification algorithm with k=5 to the scaled dataset. We first set a random seed to ensure reproducibility of the analysis (we used to set a seed during this assignment), and then fit the KNN model on the scaled training data and tested it on the scaled testing data, through the *knn* function.

To evaluate the performance of the KNN model, we created a confusion matrix to show the number of correct and incorrect predictions made by the model. Additionally, we computed the classification accuracy of the KNN model using the *confusionMatrix* function from the *caret* library. The KNN model with k=5 achieved an accuracy of **0.692**, with a 95% confidence interval of **(0.6593, 0.7233).** This means that the model correctly predicted the outcome for **69.2%** of the test data.

The classifier was better than the no information rate (NIR), which is the accuracy that would be obtained by always predicting the majority class. The NIR was **0.6159**, and the p-value for the difference between the accuracy and NIR was 2.961e-06, indicating that the difference was statistically significant. The kappa statistic measures the agreement between the classifier's predictions and the true values, taking into account the agreement that would be expected by chance. The kappa statistic was **0.3377**, indicating **fair agreement** between the classifier and the true values.

The sensitivity of the classifier was 0.5535, which means that it correctly identified about **55.4%** of the positive cases. The specificity was 0.7784, which means that it correctly identified **77.8%** of the negative cases. The positive predictive value (PPV) was 0.609, which means that when the classifier predicted a positive case, it was correct **60.9%** of the time. The negative predictive value (NPV) was 0.7365, which means that when the classifier predicted a negative case, it was correct **73.7%** of the time. The prevalence of the positive class (*Trump to win the majority vote*) in the dataset was 0.3841, (**38.41%**) and the detection rate was 0.2126, which means that the classifier detected **21.3%** of the positive cases. The detection prevalence was 0.349, which means that **34.9%** of the cases were predicted to be positive by the classifier.

The balanced accuracy takes into account the imbalance in the dataset and is the average of the sensitivity and specificity. The balanced accuracy was **0.6659**, which means that the classifier performed better than random guessing, but not as well as it would if it correctly classified both positive and negative cases equally well. (*see Appendix-Figure 12*).

In summary, the KNN model with k=5 achieved moderate accuracy and fair agreement with the actual outcomes. The specificity of the model was higher than its sensitivity, which means that it was better at identifying the negative cases than the positive cases.

Finally, we outputted the ROC curve for the KNN model with K = 5 (*see the following Figure*). As mentioned earlier, the ROC curve is a graphical representation of the performance of a binary classifier system as its discrimination threshold is varied. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. We used the *prediction* and *performance* functions from the *ROCR* library to compute and plot the ROC curve for the KNN model. The ROC curve usually helps us determine the optimal threshold for the KNN model to balance the tradeoff between TPR and FPR.

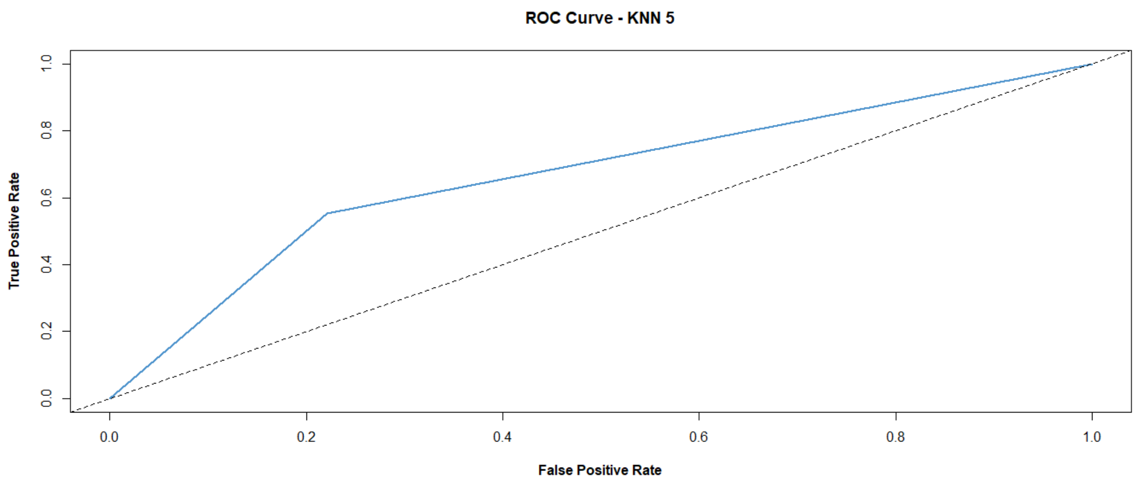


Figure 7: ROC curve of KNN with K = 5.

K-Nearest Neighbors (KNN) with K = 5 on Not Scaled Data

Now we are moving forward to explore the not scaled data of the KNN with K = 5. More specifically, we set a random seed to ensure reproducibility of the analysis. Then, we fit a KNN model on the not scaled data using the *knn* function in, with k=5 (using the five nearest neighbors to predict the outcome). Next, we create a confusion matrix to evaluate the accuracy of the model's predictions. The confusion matrix compares the predicted outcomes from the KNN model with the actual outcomes in the test data.

Based on the confusion matrix and statistics obtained from evaluating the KNN model, we can see that the model has an accuracy of 0.5604, which means that it correctly predicted the **56.04%** of the test data. The model made 358 true negative predictions (predicted 0 when the actual value was 0), 106 true positive predictions (predicted 1 when the actual value was 1), 212 false negative predictions (predicted 0 when the actual value was 1), and 152 false positive predictions (predicted 1 when the actual value was 0). The kappa coefficient is 0.0366, which indicates poor agreement between the predicted and actual classes.

The sensitivity of the model is 0.3333, which means that it correctly identified **33.33%** of the positive cases. The specificity is 0.702, which means that it correctly identified **70.2%** of the negative cases. The positive predictive value (PPV) is 0.4109, which means that when the model predicted a positive result, it was correct **41.09%** of the time. The negative predictive value (NPV) is 0.6281, which means that when the model predicted a negative result, it was correct **62.81%** of the time.

In summary, the KNN model with k=5 and unscaled data **has poor performance and low accuracy**, **sensitivity, and PPV**. It is **not a good model for predicting** the class of the samples. (*see Appendix-Figure 13*). The latest point can be further validated through the following ROC curve, and we will **not** continue to take into consideration for the final model comparisons that will follow.

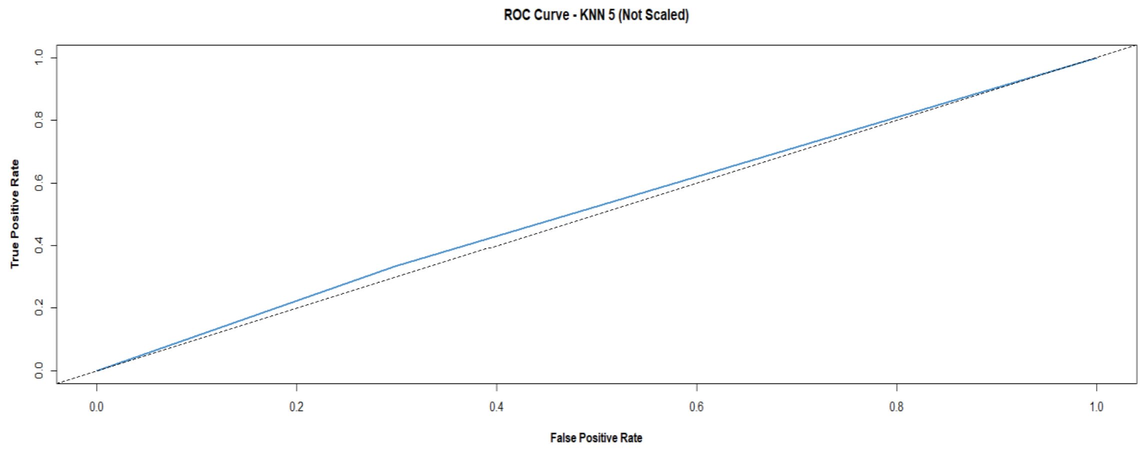


Figure 8: ROC curve of KNN with K = 5 (not scaled data).

K-Nearest Neighbors (KNN) with K = 4 on Scaled Data

Then, we construct a KNN model with K = 4 and follow the similar procedure. The model based on the confusion matrix and statistics, the KNN model with K = 4 has an accuracy of **0.7174 - 71.74% (95% CI: 0.6854, 0.7478)**, which is higher than the scaled KNN model with K = 5. The no information rate (NIR) is the accuracy that would be obtained by always predicting the most frequent class label. In this case, the NIR is **0.6159**, indicating that the model is significantly better than a naive prediction strategy.

The kappa statistic is a measure of the agreement between the model's predictions and the true class labels, adjusted for chance. A value of 1 indicates perfect agreement, while a value of 0 indicates agreement no better than chance. In this case, the kappa statistic is **0.3919**, indicating **moderate agreement** between the model's predictions and the true class labels.

The sensitivity and specificity of the model are reported as 0.5849 (**58.49%**) and 0.8 (**80%**), respectively. The positive predictive value (PPV) and negative predictive value (NPV) of the model are reported as 0.6458 (**64.58%**) and 0.7556 (**75.56%**), respectively. The prevalence of the positive class (*Trump to win the majority*) in the dataset is **0.3841**. The detection rate is the proportion of actual positive instances that were correctly identified by the model, while the detection prevalence is the proportion of predicted positive instances out of all instances. The balanced accuracy is the average of sensitivity and specificity, and in this case is **0.6925**.

From the above we can summarize that the KNN model with K = 4 appears to perform **reasonably well**, with an accuracy significantly better than the NIR and a moderate kappa statistic indicating reasonable agreement with the true class labels. The sensitivity and PPV are also relatively high, indicating that the model is effective at identifying the positive class. However, the specificity is somewhat lower, indicating that the model may be **less effective at identifying the negative class**. (*see* Appendix-*Figure 14*). This can be further validated through the ROC curve:

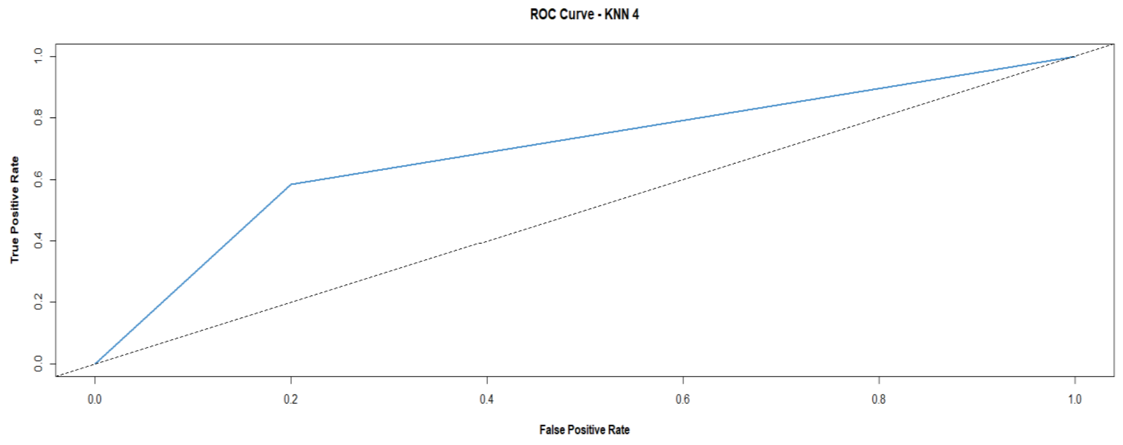


Figure 9: ROC curve of KNN with K = 4.

K-Nearest Neighbors (KNN) with K = 4 on Not Scaled Data

Finally, a KNN model with K = 4 was constructed with the not scaled data with the above procedure. Based on the confusion matrix and statistics, this model appears to perform poorly at predicting the outcome variable. The accuracy is 0.5507 (**55.07%**), which is **only slightly bette**r than the no information rate of 0.6159 (**61.59%**). The model has a sensitivity of 0.6667 (**66.67%**) and a specificity of 0.3648 (**36.78%**), indicating that the model is better at identifying true positives than true negatives (*see* *Appendix-Figure 15*).

The Kappa score is only 0.0321 (**3.21%**), indicating poor agreement between the model's predictions and the true class labels. The sensitivity and specificity of the model are reported as 0.3648 (**36.48%**) and 0.6667 (**66.67%**), respectively. Sensitivity is the proportion of true positives that were correctly identified by the model, while specificity is the proportion of true negatives that were correctly identified. The positive predictive value (PPV) and negative predictive value (NPV) of the model are reported as **0.4056** and **0.6273**, respectively. PPV is the proportion of positive predictions that were correct, while NPV is the proportion of negative predictions that were correct.

The prevalence of the positive class (*Trump to win the majority vote*) in the dataset is **0.3841**. The detection rate is the proportion of actual positive instances that were correctly identified by the model, while the detection prevalence is the proportion of predicted positive instances out of all instances. The balanced accuracy is the average of sensitivity and specificity, and in this case is **0.5157**.

Overall, the model does not appear to perform very well, with accuracy **only slightly better than the NIR and a low kappa statistic indicating poor agreement with the true class labels**. The sensitivity and PPV are also **quite low**, indicating that **the model is not very effective at identifying the positive class**.

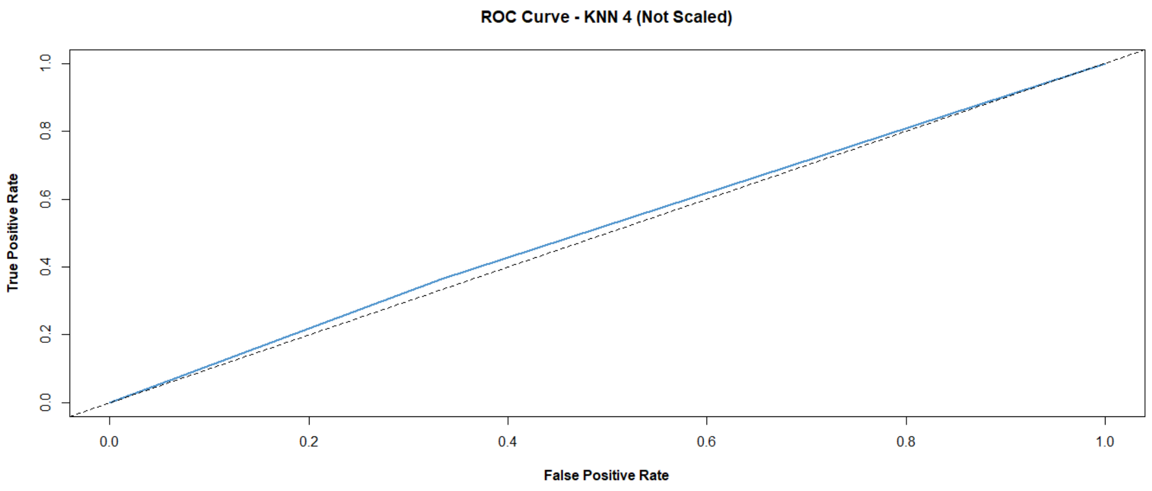


Figure 10: ROC curve of KNN with K = 4 (not scaled data).

4.6. Models Comparisons

Model comparison is an important step in machine learning and involves evaluating the performance of different models and selecting the best one for the task at hand. To be more precise, we will present various measures to select the best of the aforementioned models.

4.6.1. Dataframes balance/imbalance check - Measures selection

The first we need to do prior to the models comparisons stage is to check the balance of three dataframes: us_elections_2016 (total dataset), train_data, and test_data, by calculating the percentage of samples where Trump wins the majority of votes (*trump_over_perc50 == 1*) and where he loses (*trump_over_perc50 == 0*). The results show that all three dataframes are **somewhat imbalanced**, with **Trump winning** the majority in around **40-41%** of the samples, and **losing in around 59-61%** of the samples.

As the data is somewhat **imbalanced**, it is important to consider measures beyond accuracy when evaluating model performance. Sensitivity and specificity are relevant measures in this case, as they reflect the ability of the model to correctly identify Trump's victories and losses. In the case of imbalanced data, precision is also a relevant measure, as it reflects the model's ability to correctly identify Trump's victories among all predicted victories. Therefore, it is important to consider multiple performance metrics and not rely solely on **accuracy** in this case.

More specifically, **sensitivity** and **specificity** are important measures to consider in imbalanced data scenarios, as they help evaluate the ability of the model to correctly identify the positive and negative cases. Sensitivity, also known as recall or true positive rate, measures the proportion of actual positives that are correctly identified by the model. In an imbalanced data scenario, where the positive class is the minority class, a high sensitivity is desired to ensure that the model correctly identifies as many positive cases as possible.

Specificity measures the proportion of actual negatives that are correctly identified by the model. A high specificity is also desired in imbalanced data scenarios to ensure that the model correctly identifies the negative cases and does not misclassify them as positive cases.

Lastly, in addition to sensitivity and specificity, it is important to consider **precision**, which measures the proportion of true positives among all positive predictions. In imbalanced data, a high sensitivity or specificity alone may not be sufficient, as the model may be biased towards predicting the majority class. Therefore, it is important to also consider the precision of the model in identifying the minority class.

4.6.2. Models selection

Based on the individual model's ROC curves figures and since it not the suggested way to build the KNN models, the not scaled ones with four and five NN will be excluded from the model's comparisons. The above can be supported from the fact that their ROC curves (*see Figures 8 & 10*) and their AUC metrics suggest poor results.

To sum up, the models we are going to compare are the following ones:

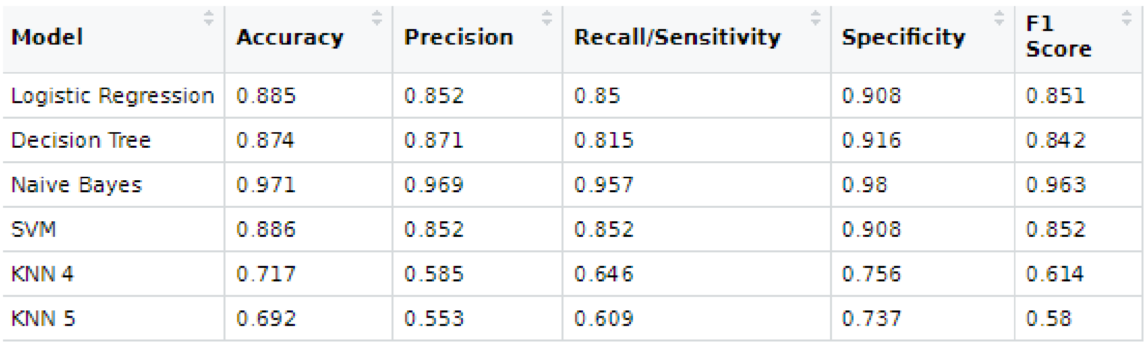
- Logistic Regression Model
- Decision Tree Model
- SVM Model
- Naïve Bayes Model
- KNN with K = 5 scaled Model
- KNN with K = 4 scaled Model

4.6.3. Statistics Measures

In this section we are going to interpret the statistics comparisons table. To be more precise, from the following table we can see the performance of six different classification models on a given dataset, based on several metrics. **Accuracy** is a crucial metric as it measures the overall performance of the model in correctly predicting the class of observations. It ranges from 0 to 1, with higher values indicating better performance.

Precision measures the proportion of true positives out of all positive predictions. A higher precision value means that the model is better at correctly identifying positive cases, but may have more false negatives. **Recall/Sensitivity**, on the other hand, measures the proportion of true positives out of all actual positive cases. A higher recall value means that the model is better at correctly identifying all positive cases, but may have more false positives.

Specificity measures the proportion of true negatives out of all actual negative cases. A higher specificity value means that the model is better at correctly identifying all negative cases, but may have more false positives. **F1 Score** is a combination of precision and recall, and it is a useful metric when the classes are imbalanced. It measures the harmonic mean of precision and recall and combines these two metrics into a single value that ranges from 0 to 1, with higher values indicating better performance.



Model	Accuracy	Precision	Recall/Sensitivity	Specificity	F1 Score
Logistic Regression	0.885	0.852	0.85	0.908	0.851
Decision Tree	0.874	0.871	0.815	0.916	0.842
Naive Bayes	0.971	0.969	0.957	0.98	0.963
SVM	0.886	0.852	0.852	0.908	0.852
KNN 4	0.717	0.585	0.646	0.756	0.614
KNN 5	0.692	0.553	0.609	0.737	0.58

Figure 11: Statistics Measures Table.

From the above table we can see that the Accuracy metric measures the overall performance of the model in correctly classifying the target variable. The highest Accuracy scores were obtained for Naive Bayes (0.971), followed by Logistic Regression (0.885) and SVM (0.886), indicating that these models perform well on this dataset.

The Precision metric measures the proportion of correctly predicted positive instances (True Positives) out of the total instances predicted as positive. Naive Bayes achieved the highest Precision score (0.969), followed by Logistic Regression (0.852) and SVM (0.852), indicating that these models are good at predicting the positive class. On this point we need to note that since the dataset is **slightly unbalanced over the negative class** (*Trump to lose the majority*) precision and recall are very important metrics.

The Recall/Sensitivity metric measures the proportion of correctly predicted positive instances out of the total positive instances in the dataset. Naive Bayes achieved the highest Recall score (0.957), followed by Logistic Regression (0.85) and SVM (0.852), indicating that these models are good at identifying the positive class.

The Specificity metric measures the proportion of correctly predicted negative instances (True Negatives) out of the total negative instances in the dataset. The highest Specificity scores were obtained for Naive Bayes (0.98) and Decision Tree (0.916), indicating that these models are good at identifying the negative class.

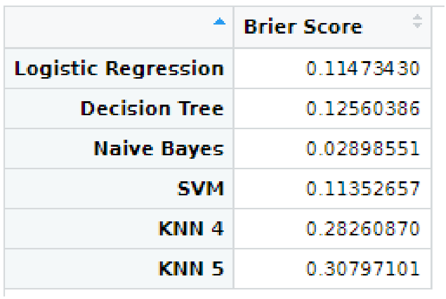
The F1 Score is the harmonic mean of Precision and Recall and provides a balanced measure of both metrics. The highest F1 Scores were obtained for Naive Bayes (0.963) and Logistic Regression (0.851), indicating that these models provide a good balance between Precision and Recall.

Overall, Naive Bayes appears to be the best-performing model on this dataset, achieving the highest scores in four out of five metrics evaluated. Based on the table, the Naive Bayes model appears to perform the best on this dataset, with the highest accuracy, precision, recall, and F1 score.

However, if we exclude the Naive Bayes model based on its limitations and its assumptions violation, the Logistic Regression and SVM models both have similar accuracy, precision, and F1 scores, with slightly better specificity than the other models. The Decision Tree model also has a high accuracy and F1 score, but its precision and recall values are slightly lower than the Logistic Regression and SVM models. Finally, the KNN models have the lowest accuracy and F1 scores, with the KNN 4 model performing slightly better than the KNN 5 model.

4.6.4. Brier Score

In this section we calculated the Brier Score for the six compared models. More specifically, the Brier score is a metric used to evaluate the accuracy of probabilistic predictions made by a classification model. It measures the average squared difference between the predicted probabilities and the actual outcomes. A lower Brier score indicates better performance, with a perfect score of 0 indicating perfect accuracy.



	Brier Score
Logistic Regression	0.11473430
Decision Tree	0.12560386
Naive Bayes	0.02898551
SVM	0.11352657
KNN 4	0.28260870
KNN 5	0.30797101

Figure 12: Brier Score Table.

Interpreting the Brier score table for the models, we can see that **Naive Bayes has the lowest Brier score of 0.0289**, indicating that it has the best performance in terms of predicting the outcomes accurately. This is followed by **Logistic Regression with a score of 0.1147 and SVM with a score of 0.1135**. Decision Tree also performs relatively well with a score of 0.1256. However, KNN 4 and KNN 5 have much higher Brier scores of **0.2826** and **0.308**, respectively, indicating that they have the worst performance in terms of predicting the outcomes accurately.

It is important to note that <u>Naive Bayes has certain assumptions and limitations, such as assuming that the features are independent and that all features are equally important</u>. Therefore, it is necessary to consider alternative models that do not make such assumptions, such as **Logistic Regression** and **SVM**.

Overall, the Brier Score provides additional insight into the performance of the classification models, complementing the other metrics such as accuracy, precision, recall, specificity, and F1 score that were performed earlier. **SVM and Logistic Regression models lead the comparisons so far** (excluding the Naïve Bayes model).

4.6.5. Lift Curves

Lift curves are a useful tool for evaluating the performance of classification models, especially in situations where the target variable is imbalanced. They provide a graphical representation of how much better a model performs compared to random chance, as the percentage of samples tested increases. In the procedure we followed we plot the lift curves of the top 4 models (Logistic Regression, Decision Tree, Naive Bayes, and SVM), which are compared using a lift curve comparison object. The *lift_df* dataframe is first created with the sorted lift curves, and the *lift_curve_com* object is then created using the *lift* function.

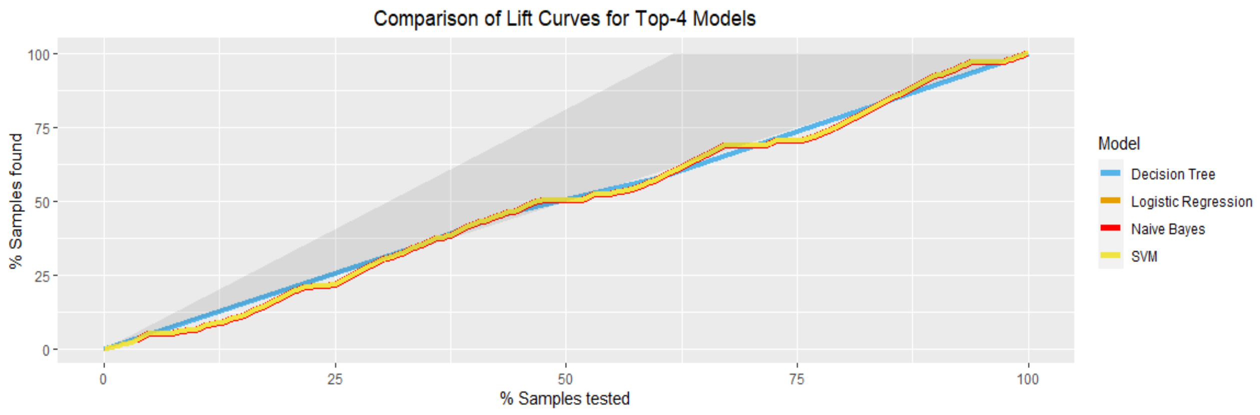


Figure 13: Lift Curves for Models Comparisons.

The resulting lift curves are plotted (*see Figure 13*) with each model represented by a different colored line. The x-axis represents the percentage of samples tested, while the y-axis represents the percentage of target events found. A polygon is also added to represent the lift of a random selection of samples. Regarding the above plot, it is evident that all curves, except the curve for the Decision Tree model, exhibit a similar pattern and generate comparable predictions despite the differences observed in the previous metrics. This indicates that all models possess very good and comparable predictive ability.

Furthermore, it is observed that the lift curves for all models, except for the Decision Tree model, are almost identical. Although the Decision Tree model performs slightly better, this difference is not significant for accurately predicting the counties that will vote for Trump among other Republican candidates, particularly in the first 50% of the most probable observations. Specifically, it is noted that selecting the first quarter of the most probable counties for Trump to prevail in the Republican party, according to the model's results, yields 25% more counties that will vote for Trump compared to randomly selecting counties.

In summary, the lift curves provide a visual representation of the performance of each model in predicting the outcome variable. From the plot, we can see that all models have a good and comparable predictive ability. Therefore, we cannot clearly conclude which model performs the best based solely on the lift curves.

4.6.6. ROC Curves & AUC values

ROC (Receiver Operating Characteristic) curves and AUC (Area Under the Curve) values are commonly used evaluation metrics in classification tasks. They help in assessing the performance of different models and selecting the best one for a specific problem. The ROC curve is a graphical representation of the performance of a binary classifier system as its discrimination threshold is varied. The AUC value measures the performance of the model across all possible classification thresholds, with a higher AUC indicating better overall performance. In this context, we have evaluated and compared the performance of six models: Logistic Regression, Decision Tree, SVM, KNN with K = 4, and KNN with K = 5, using ROC curves and AUC values.

By plotting the models ROC curve we can see (*Figure 14*) that the top three models are the Logistic Regression, SVM and Decision Tree models, with the Naïve Bayes and the two KNN models to fall behind with this metric.

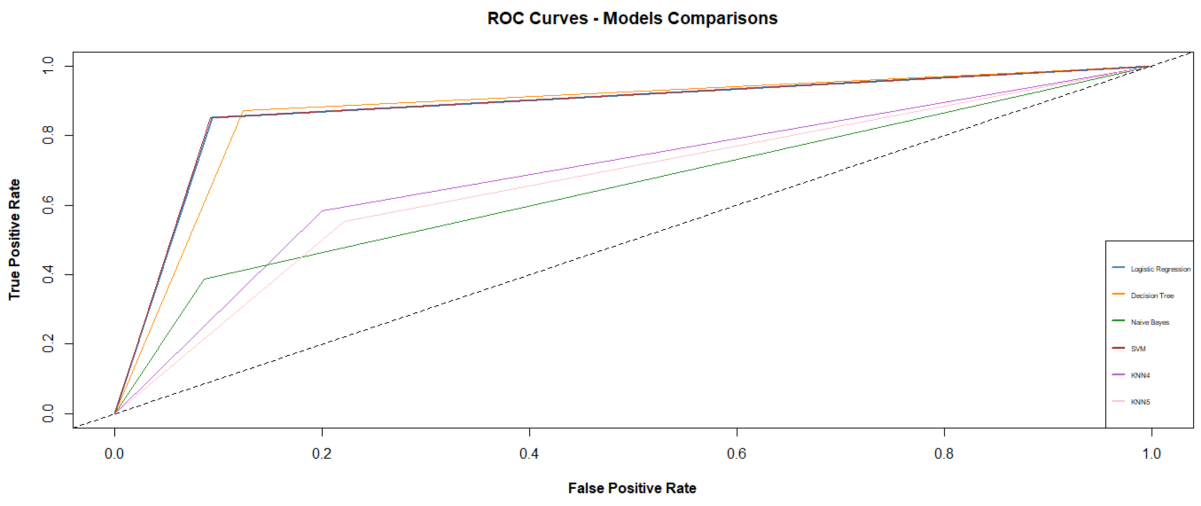
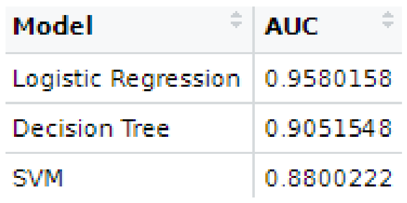


Figure 14: ROC Curves Comparisons.

We will conclude our evaluation by computing the AUC values of the top three models so far (and by excluding the Naïve Bayes one due to its limitations on a real world problem).



Model	AUC
Logistic Regression	0.9580158
Decision Tree	0.9051548
SVM	0.8800222

Figure 15: AUC values table (top three models).

From the above table (*Figure 15*), we can see that Logistic Regression model has the highest AUC value among all the models, which indicates that it is the best model for this particular task. AUC values closer to 1 indicate better performance in terms of correctly classifying the positive and negative samples. In contrast, the Decision Tree and SVM models have AUC values that are lower than the Logistic Regression model. This suggests that these models may not be as effective at classifying the response variable. However, the Decision Tree and the SVM model has a relatively high AUC value compared to the KNN models. Overall, the **Logistic Regression** model is the most effective model according to these metrics.

4.7. Conclusion

After taking all the aforementioned measures into consideration, and by excluding the Naïve Bayes models due to certain assumptions and limitations, such as assuming that the features are independent and that all features are equally important, we reach to the conclusion that the Logistic Regression and the SVM model are the leading two followed by the Decision Tree model. However since we are aiming for the best predictive model - and we do not care about the interpretations - the Logistic Regression model performs slightly better than the SVM and the Decision Tree models across to the multiple measures that were implemented above. Therefore, our final choice would be the **Logistic Regression** model.

5. Part II: Clustering

In the context of unsupervised learning, we will be exploring the use of clustering methods to extract valuable insights from our dataset. In contrast to supervised learning, the dataset lacks any labeled information, thus our objective is to discover hidden structures or groupings among the data points. Our aim is to partition the data into distinct clusters, where each cluster comprises of observations that exhibit similar characteristics or attributes. The clustering process is motivated by the goal of identifying homogeneity within clusters and heterogeneity between them, without any prior knowledge of the underlying labels or classes. Clustering is used in various domains such as marketing, image segmentation, bioinformatics, and social network analysis, among others.

Clustering algorithms can be broadly categorized into two types: hierarchical clustering and partitioning clustering. In hierarchical clustering, the data points are recursively grouped into a tree-like structure, known as a dendrogram, based on the similarity between them. The dendrogram is then pruned to obtain the final clusters. On the other hand, in partitioning clustering, the data points are divided into a fixed number of clusters based on a predefined criterion, such as minimizing the distance between the data points in the same cluster. The two most popular clustering algorithms – which will also be applied in the current analysis are the K-Means and the Hierarchical Clustering.

5.1. Dataset Preparations & Adjustments

The initial procedure followed is to prepare the dataset for clustering by separating the variables into two groups: demographics and economic-related variables. We load the given Excel file (*see Datasets Archive*) and remove rows with missing data (*states and us totals*). Then, we remove the columns that are not needed for clustering (*fips, county, state_abbreviation*). Finally we save and output the current CSV file as a checkpoint.

Afterwards, we load the data from the CSV file created in the first part (*us_elections_2016_clustering.csv - see Datasets Archive*) and create two separate data frames: one containing the **demographics-related** variables and the other containing the **economic-related** variables.

The variables selected for the demographics-related data frame are the following:

Variables	Description
PST045214	Population, 2014 estimate
PST040210	Population, 2010 (April 1) estimates base
PST120214	Population, percent change - April 1, 2010 to July 1, 2014
POP010210	Population, 2010
AGE135214	Persons under 5 years, percent, 2014
AGE295214	Persons under 18 years, percent, 2014
AGE775214	Persons 65 years and over, percent, 2014
SEX255214	Female persons, percent, 2014
RHI125214	White alone, percent, 2014
RHI225214	Black or African American alone, percent, 2014
RHI325214	American Indian and Alaska Native alone, percent, 2014
RHI425214	Asian alone, percent, 2014
RHI525214	Native Hawaiian and Other Pacific Islander alone, percent, 2014
RHI625214	Two or More Races, percent, 2014
RHI725214	Hispanic or Latino, percent, 2014

RHI825214	White alone, not Hispanic or Latino, percent, 2014
POP715213	Living in same house 1 year & over, percent, 2009-2013
POP645213	Foreign born persons, percent, 2009-2013
POP815213	Language other than English spoken at home, pct age 5+, 2009-2013
EDU635213	High school graduate or higher, percent of persons age 25+, 2009-2013
EDU685213	Bachelor's degree or higher, percent of persons age 25+, 2009-2013
VET605213	Veterans, 2009-2013

5.2. Variables Selection

Variable selection prior to clustering with correlation involves identifying variables that are highly correlated with each other and selecting a subset of variables that are not highly correlated to avoid redundancy and overfitting.

The first step is to calculate the correlation matrix of all variables in the dataset. Then, we determine the correlation coefficient cut-off value that will be used to define highly correlated variables, as the most common one of a cut-off of **0.7 or higher**. The cut-off is used to identify highly correlated variables. This way we are able to identify variables that are highly correlated with other variables. These variables may be redundant and can be removed to avoid overfitting and to simplify the clustering analysis.

In the end we will end up with a subset of variables that are not highly correlated with each other. These variables should have low or moderate correlation coefficients. Only these variables will be used as the input for the clustering algorithm that will follow.

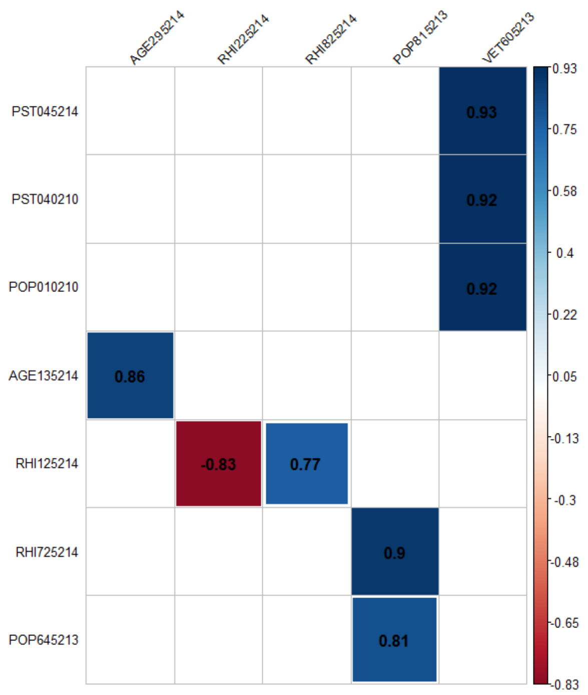


Figure 16: Correlation Plot between the demographics variables.

Based on the above figure (*Figure 16*), we have depicted the correlation coefficients between pairs of variables (highly correlated pairs - over 0.7). The first three correlations all involve VET605213. All three correlations have relatively high values above 0.9, which suggests a strong positive relationship between VET605213 and each of the other variables (PST045214, POP010210, and PST040210). To resolve this case we decide to remove the **VET605213** - this removal resolve these three pairs correlation case.

The correlation between RHI725214 and POP815213 is also relatively high, with a value of 0.899. This suggests a positive relationship between these variables. Also, the correlation between POP645213 and POP815213 is relatively high, with a value of 0.812. For these reasons we remove the common variable of these two pairs the **POP815213**.

Then, we focus on the correlation between RHI125214 and RHI22521, which is negative, with a value of -0.828. This suggests an inverse relationship between the percentage of the population that is white (RHI125214) and the percentage of the population that is black or African American (RHI225214). Moreover, there is a positive correlation between RHI125214 (again) and RHI825214, with a value of 0.769. Due to the fact that there is a common part to these two pairs we decide to remove the **RHI125214** variable.

Finally, the correlation between AGE135214 and AGE295214 is relatively high, with a value of 0.862. This suggests a positive relationship between the percentage of the population that is under 5 years old (AGE135214) and the percentage of the population that is 65 years old or older (AGE295214). Since both of the variables appeared only once in the top correlated pairs, with our understanding of the problem, we decide to remove the less significant variable (our decision) the **AGE135214**, which refers to persons under 5 years old - since we thought is less meaningful than the persons over 65 years old in a problem like this one.

To sum up, we decide to remove the following variables to resolve all the correlation issues:

Variables	Description
AGE135214	Persons under 5 years, percent, 2014
RHI125214	White alone, percent, 2014
POP815213	Language other than English spoken at home, pct age 5+, 2009-2013
VET605213	Veterans, 2009-2013

5.3. Variables Scaling

In this section we will mention the scaling process that followed our earlier - variables selection procedure. In general, scaling is important prior to clustering for several reasons, even if the variables are not correlated. To be more precise, since clustering algorithms are distance-based, and the distance between two data points is calculated based on the difference between their values in each variable. If the variables are on different scales or have different units, then some variables may dominate the distance calculation, leading to biased clustering results (*see variables that are on different scales or have different units, Appendix-Figure 29*).

Furthermore, this way we can identify outliers, since the scaling procedure helps to detect outliers in the data, which can have a significant impact on the clustering result. Outliers can distort the clustering result and make it difficult to identify meaningful clusters (***for example the K-Means can be trapped easily by outliers***). Standardization is another reason since scaling makes it easier to compare the contributions of different variables to the clustering result. By scaling the variables, we standardize their values, and the clustering algorithm treats them equally. This makes it easier to identify which variables are most important in forming the clusters.

In summary, scaling is important prior to clustering, even if the variables are not correlated, as it helps to standardize the variables, avoid bias in distance measures, and detect outliers. Therefore, after the removal of the correlated pairs we scale our dataset.

5.4. Hierarchical Clustering

Hierarchical clustering is a popular unsupervised learning method used to group similar data points into clusters. The procedure works by iteratively merging or dividing clusters based on their similarity, ultimately forming a hierarchical tree-like structure, called a dendrogram. The method is commonly used in data analysis, image segmentation, and information retrieval. Hierarchical clustering starts with all points in a single cluster and iteratively divides the most dissimilar clusters until each point is in its own cluster. The similarity between data points or clusters is often measured using a distance metric such as Euclidean or Manhattan distances. Different linkage criteria, such as complete, single, centroids, and ward linkage, can also be used to determine the similarity between clusters.

Hierarchical clustering can be visualized using a dendrogram, which displays the hierarchical relationships between clusters. The height of the dendrogram branches indicates the level of dissimilarity between clusters at each merging or dividing step. One advantage of hierarchical clustering is its flexibility, as it can handle both numerical and categorical data, and can adapt to different scales and data types. However, it can be computationally expensive, especially for large datasets, and the choice of distance metric and linkage criteria can strongly affect the clustering results.

To be more practical we perform hierarchical clustering on all four different methods (Complete, Centroid, Ward, and Single) with Euclidean distance as the similarity metric, initially. **Complete** linkage methods calculates the distance between clusters as the **maximum distance** between any two points in each cluster. It tends to produce clusters with more distinct boundaries and can be sensitive to outliers. **Single** linkage method calculates the distance between clusters as the **minimum distance** between any two points in each cluster. It tends to produce long, stringy clusters and can be sensitive to noise. **Centroid** linkage method calculates the distance between clusters as the **distance between** their **centroids**. It is less sensitive to outliers than complete linkage and less sensitive to noise than single linkage. However, it can produce clusters that are not well separated. Finally, the **Ward** method is a hierarchical clustering technique that aims to **minimize the sum of squared differences within each cluster**. It is one of the most popular methods for hierarchical clustering and is particularly useful when the goal is to create well-separated, homogeneous clusters. The Ward method tends to produce clusters that are relatively compact and spherical in shape. However, it can also be sensitive to outliers and can create unbalanced clusters if the dataset contains unevenly sized groups.

For all models we have plotted the dendrograms, indicated clusters using a red border, calculated the average silhouette coefficients for different numbers of clusters, and visualized cluster quality using silhouette plots (*regarding the Complete, Centroid, and Single models, see Appendix-Figures 32 - 37*). Based on the analysis of the clustering results using the ward, complete, centroids, and single methods, it was observed that the ward method presented a lower value of the average silhouette score compared to the other method. However, the **ward method** was found to produce **more homogeneous clusters**. It is important to note that the main objective of this clustering analysis was not solely focused on pattern recognition in the dataset, but rather on identifying distinct groups. Therefore, despite the lower average silhouette score, the ward method was deemed the most appropriate method to achieve the desired outcome. To summarize it, the Ward method was selected since it produced more homogenous clusters. The following two figures depict the procedure followed to select the optimal number of clusters based on silhouette coefficient and average width and the final results, regarding the silhouette plot with the optimal number of clusters (K = 8).

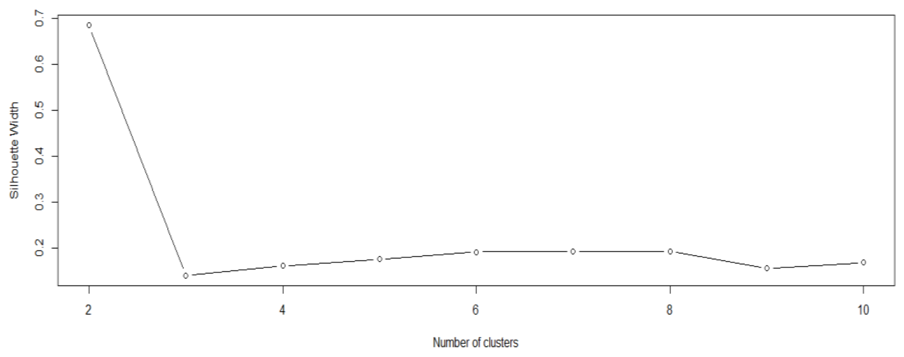


Figure 17: Silhouette Coefficient of Ward Method with Euclidean Distance for various clusters.

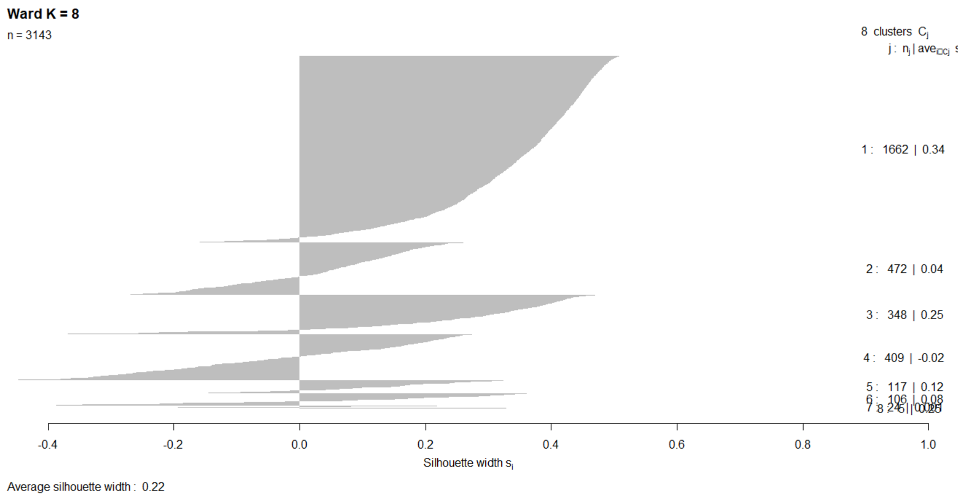


Figure 18: Silhouette Plot of Ward Method with Euclidean Distance.

In general, it is important to try different distance metrics, such as Euclidean and Manhattan distances, in hierarchical clustering as they can have a significant impact on the resulting clusters. While Euclidean distance measures the straight-line distance between two points in a multi-dimensional space, Manhattan distance measures the distance between two points by summing the absolute differences of their coordinates.

Furthermore, in some cases, Manhattan distance may be a more appropriate metric to use than Euclidean distance, such as when dealing with data that has a high level of variability or when the variables are on different scales. Therefore, we decide to construct a hierarchical clustering model on the best linkage method (ward) with the Manhattan distance. By using both Euclidean and Manhattan distances in hierarchical clustering, we can determine which distance metric results in more meaningful and homogeneous clusters for the given dataset. The two following to figures describe the selection procedure and output the results regarding the average silhouette width of the ward Manhattan with six clusters model.

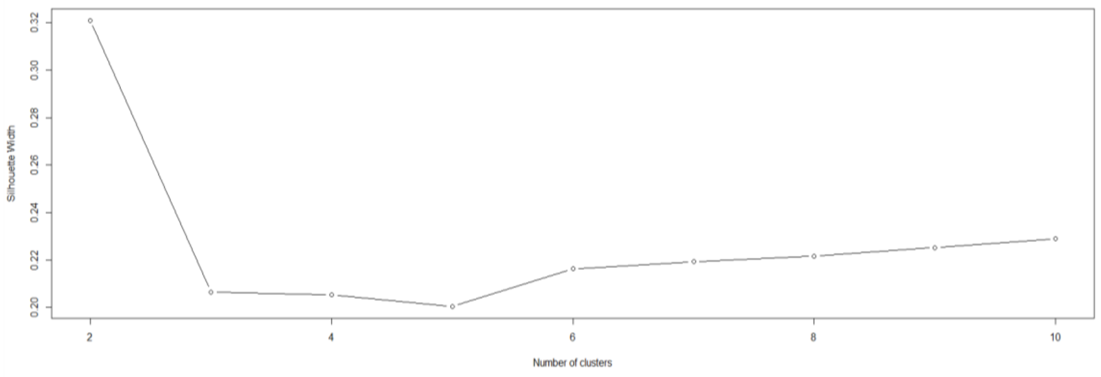


Figure 19: Silhouette Coefficient of Ward Method with Manhattan Distance for various clusters.

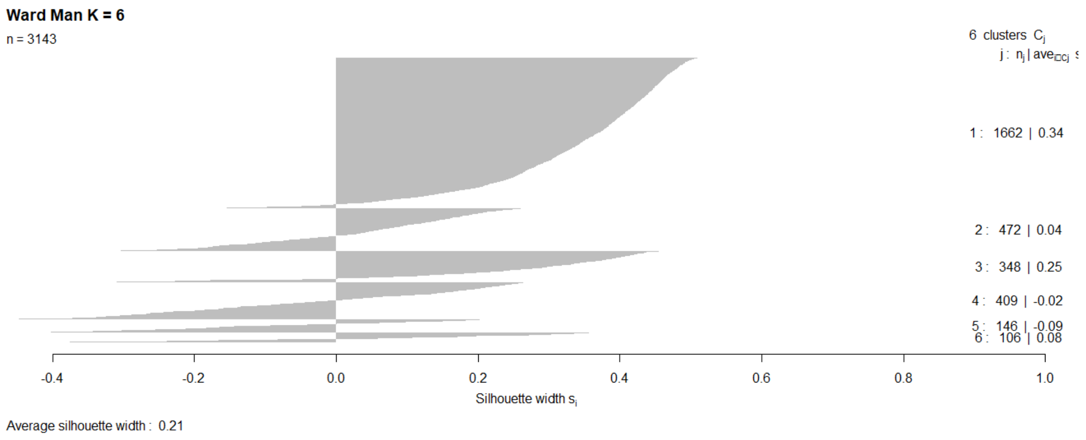


Figure 20: Silhouette Plot of Single Method with Manhattan Distance.

The Ward method with Euclidean distance resulted in eight clusters being the most appropriate option, with an average silhouette value of 0.19 indicating relatively homogeneous clusters. The Ward method with Manhattan distance, on the other hand, produced six clusters as the most appropriate option, with a higher average silhouette value of 0.21 indicating more homogeneity. Therefore, in this case, the analysis revealed that the **Manhattan Ward method** produced a **higher average silhouette value and more homogeneous clusters**, indicating that it may be a better choice for this particular dataset. The following figure depicts the dendrogram of the selected hierarchical clustering model.

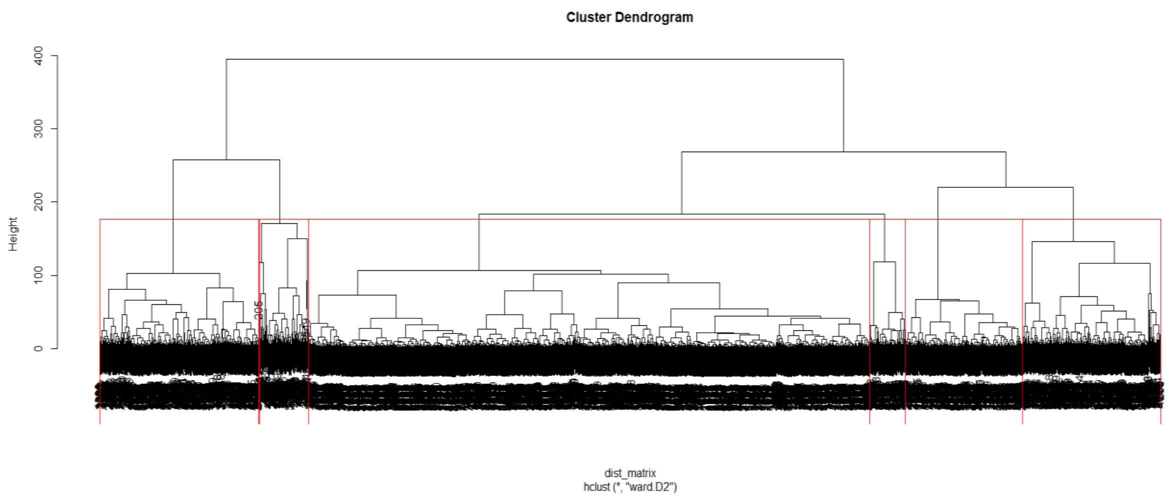


Figure 21: Dendrogram of Ward Method with Manhattan Distance.

5.5. K-Means Clustering

K-Means clustering is a popular unsupervised machine learning technique used for clustering similar data points into groups or clusters based on their similarity or distance from each other. The algorithm starts by assigning k centroids (where k is the number of desired clusters) to the data points. The algorithm then iteratively assigns each data point to the nearest centroid and re-computes the centroid based on the newly assigned data points. The iteration continues until there is no significant change in the assignment of data points to clusters or a predefined maximum number of iterations is reached.

The algorithm aims to minimize the sum of squared distances between each data point and its assigned centroid, which is also known as the within-cluster sum of squares (WSS). To determine the optimal number of clusters, one common method is to plot the WSS for different values of k and choose the value of k at the "elbow point," where the decrease in WSS starts to level off.

Thus, the next step is to calculate the WSS and create the elbow plot, in order to find the optimal number of clusters. The optimal number of clusters is typically chosen at the point where the WSS decreases sharply before leveling off. In our case, it appears that the "elbow point" is located at the point of 6 clusters.

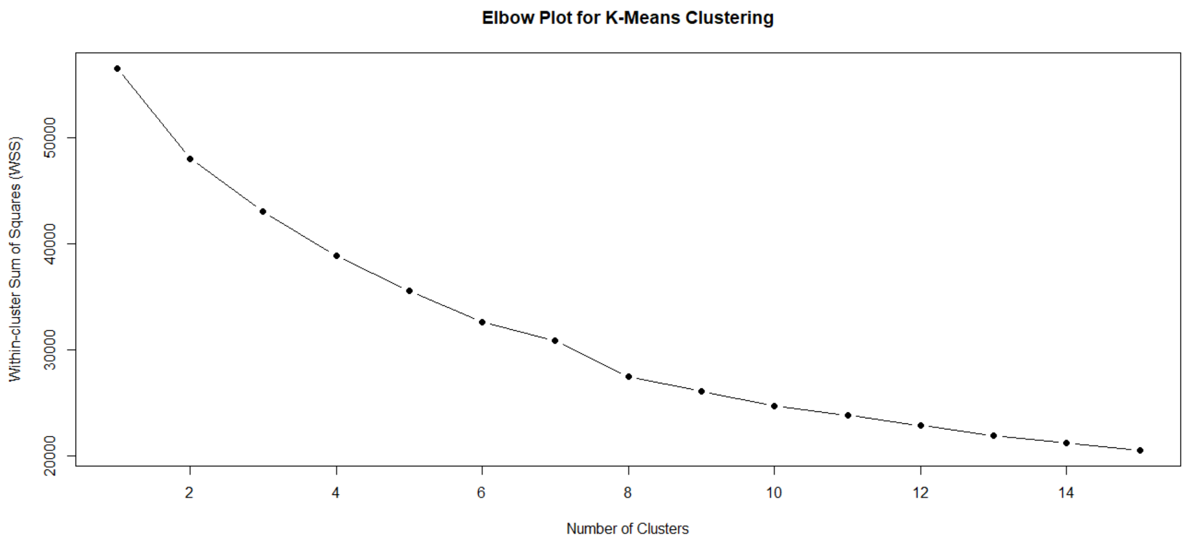


Figure 22: Elbow Plot for K-Means Clustering

By comparing the plot with the numbers of the WSS (*see* *Appendix-Figure 30*) we found that adding a seventh cluster did not result in significant improvement, and therefore we proceeded with six clusters. To be more precise, the analysis produced a table showing the detailed results of the cluster analysis, which indicated that the within-cluster sum of squares was about **33%**, indicating a low level of variability of observations within each cluster. The within-cluster sum of squares is a measure of the amount of variation within each cluster and is used to evaluate the quality of the clustering solution. A low within-cluster sum of squares indicates that the data points within each cluster are tightly packed together and therefore highly similar to each other. This suggests that the clustering algorithm has successfully identified distinct groups within the data based on their similarities, and that the resulting clusters are relatively **homogenous**.

Afterwards, we are performing K-Means clustering on the demographics scaled dataset (*demographics_scaled*) using three different values of K = 4, 5, and 6. We are trying these to validate the elbow plot and to be sure that the K-Means with the six clusters is the optimal one. At this point we need to clarify that prior of the construction of each model we set a random seed at the beginning of each clustering to ensure reproducibility of the results (*our academic id numbers*).

To be a little more practical, for each value of K, we create a K-Means object using the *kmeans* function and specify the number of clusters using the centers argument. We also set the number of random starts to 25 using the *nstart* argument to ensure that the algorithm converges to the global optimum.

We then aggregate the mean of each variable by cluster using the aggregate function, grouping by the cluster assignments stored in the K-Means object. Finally, we bind the cluster assignments to the original dataset.

Then and after constructing the three models, we are visualizing and comparing the results of K-Means clustering with the different numbers of clusters. We use the pairs plot function to visualize the relationships between the variables and assign colors to the data points based on their cluster assignments.

After visualizing the data, we plot silhouette diagrams for each of the K-Means models with 4, 5, and 6 clusters. The silhouette plot shows how similar an observation is to other observations in its own cluster compared to observations in other clusters. The average silhouette width indicates the overall quality of the clustering, with higher values indicating better-defined clusters (*see the following Figure*).

Based on this metric all three models have relevant results with **0.26** for the six clusters model and **0.25** for the other two. In addition to the prior, the K-Means model with 4 clusters has two negative values in its silhouette plot, indicating that some of the observations may be poorly clustered, while the other two have only one. Therefore, the optimum model according to the elbow plot seems to lead this metric too.

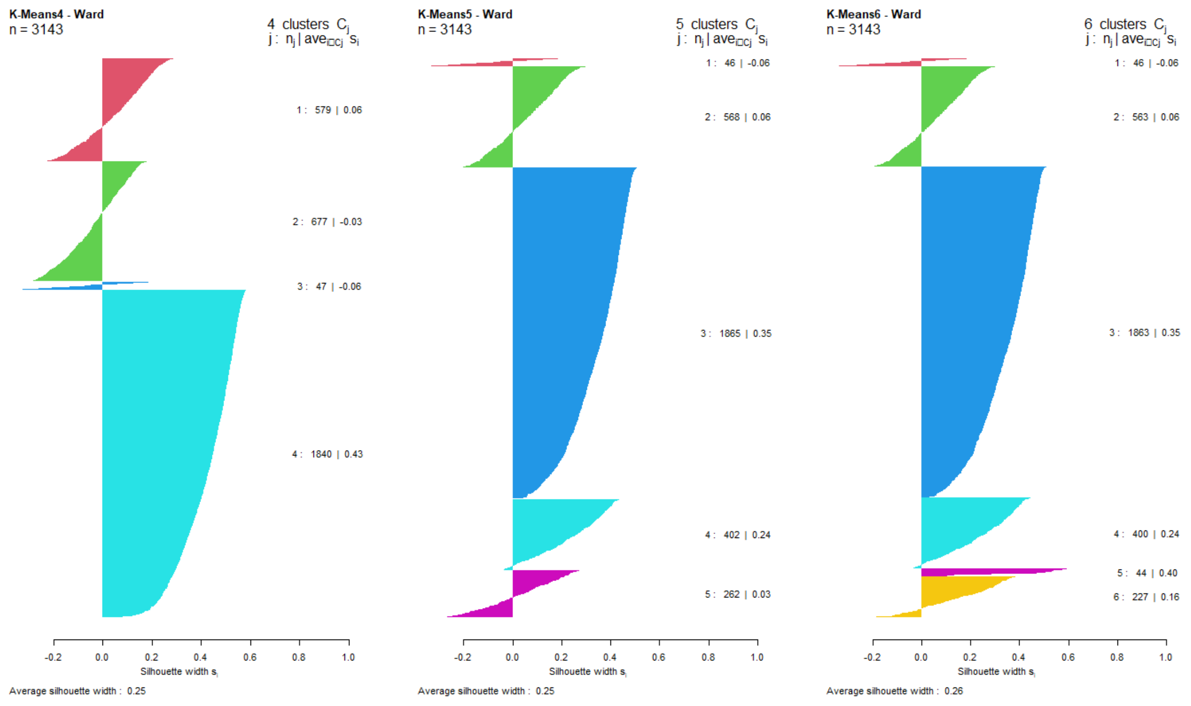


Figure 23: Silhouette diagrams for each of the K-Means models with 4, 5, and 6 clusters.

We also calculate the within-cluster sum of squares (WSS) for each of the K-Means models. WSS measures the total within-cluster variation and is used to evaluate the quality of clustering. Lower values of WSS indicate better-defined clusters. Based on the results, we conclude that the **K-Means model with 6 clusters performs better** than the other models, as it has the **highest average silhouette width and the lowest WSS value** (*see Appendix-Figure 30*).

Then we utilize the **Adjusted Rand Index (ARI)** as an evaluation metric. The Adjusted Rand Index measures the similarity between two clustering solutions, with a value ranging from -1 to 1, where 1 indicates perfect agreement between the two solutions. We calculated the ARI values for the three K-Means models, comparing each model against the other two. The ARI values were **0.8515** for K-Means with 4 clusters vs. K-Means with 5 clusters, **0.8483** for K-Means with 4 clusters vs. K-Means with 6 clusters, and **0.9831** for K-Means with 5 clusters vs. K-Means with 6 clusters.

Afterwards, we visualized the ARI values using a horizontal barplot, where each bar represents the comparison between two K-Means models. The barplot showed that K-Means with 6 clusters had the highest ARI value (0.9831) compared to the other models. Therefore, we can conclude that **the K-Means model with 6 clusters performed better than the other models** according to this metric.

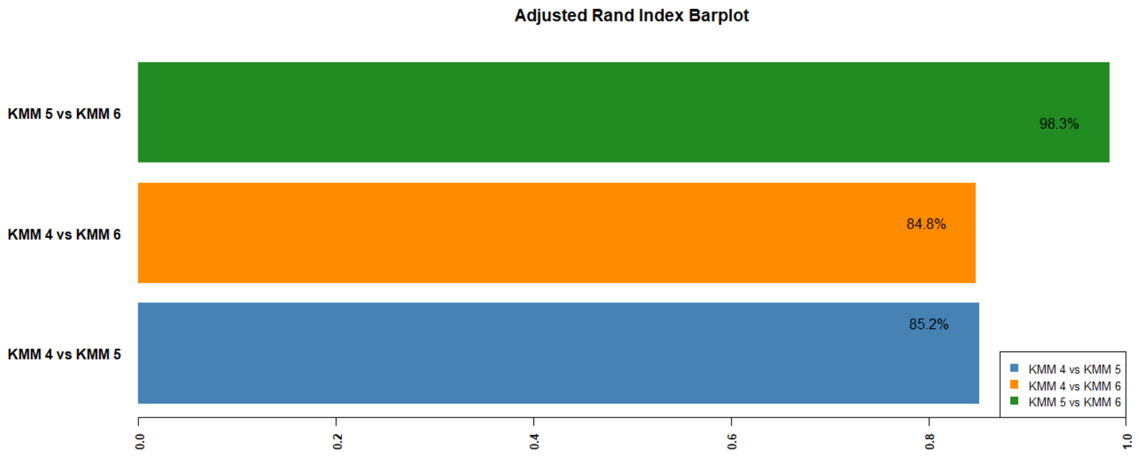


Figure 24: Adjusted Rand Index barplot (K-Means Clustering Models).

We also examined the contingency tables between the clusters produced by the K-Means models. The contingency tables showed how many observations were assigned to each cluster for each K-Means model. The tables provided additional information about the distribution of observations among the clusters, which can help to interpret the results and gain further insights into the underlying patterns in the data (*see Appendix-Figure 31*).

Finally, based on the evaluation metrics and the visualization of the clustering results, we can conclude that the **K-Means model with 6 clusters is the best one** for this dataset, as it was originally indicated from the **Elbow Plot**. It achieved the **highest silhouette score**, the **lowest WSS** and the **highest Adjusted Rand Index** compared to the other models. Additionally, the cluster plot for the K-Means model with 6 clusters **shows slightly more distinct and well-separated clusters** compared to the other models. Therefore, we recommend using the **K-Means model with 6 clusters** for further analysis and interpretation of the dataset.

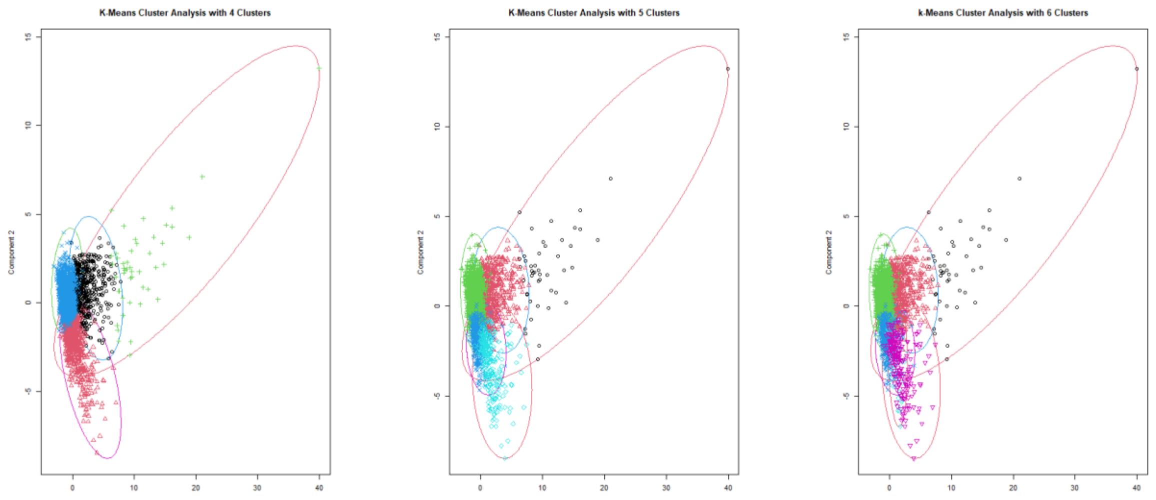


Figure 25: K-Means models visualization plots.

5.6. Models Comparisons

In this section we are going to compare the performance of K-Means and Hierarchical Clustering with the Ward Manhattan method using silhouette plots and clusplots. Silhouette plots measure how similar an observation is to its own cluster compared to other clusters. Clusplots visualize the clusters based on their pairwise distance.

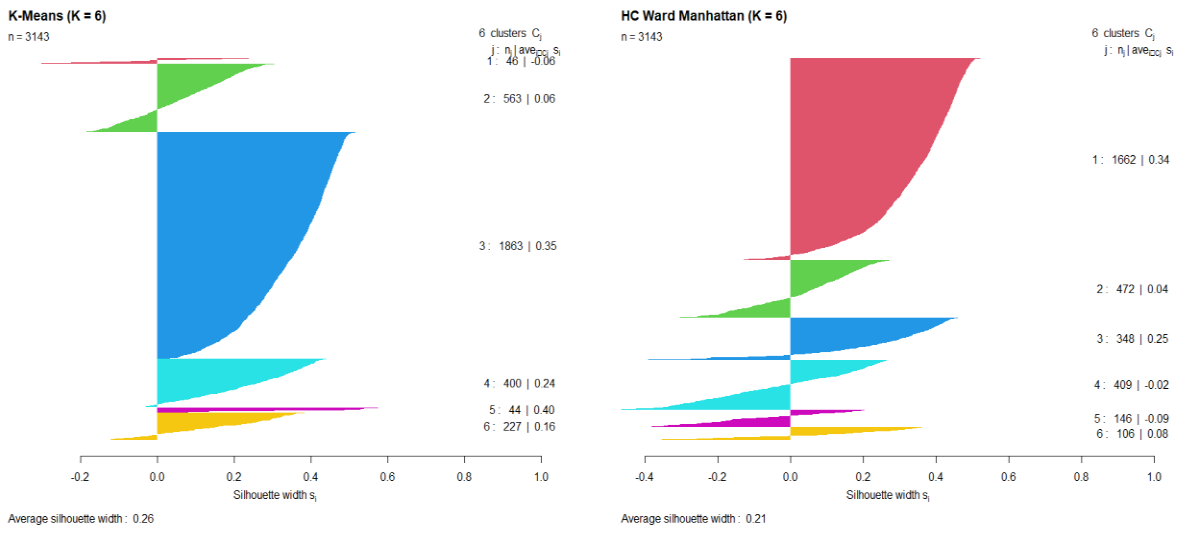


Figure 26: Top Clustering Models Comparisons - Silhouette Plots.

The silhouette plot shows **that K-Means with 6 clusters has a higher silhouette score** than Hierarchical Clustering with 6 clusters using the Ward Manhattan method. This suggests that **the K-Means clustering method produces more well-defined clusters than the hierarchical clustering one**.

From the created table that shows the count of observations in each cluster for both models. We can see that the K-Means and Hierarchical Clustering methods produce different clusterings, with some observations assigned to different clusters. We can summarize that the K-Means seems to be better at separating out some of the clusters (e.g., cluster 5) compared to the hierarchical clustering method with the Ward Manhattan method (*see Appendix-Figure 38*).

Finally, the clusplots show the clusters and their centroids in two-dimensional space (*displays the clusters in a scatter plot*). Each point represents an observation in the data, and its color indicates the cluster to which it belongs. The clusters are separated by ellipses that indicate the cluster boundaries.

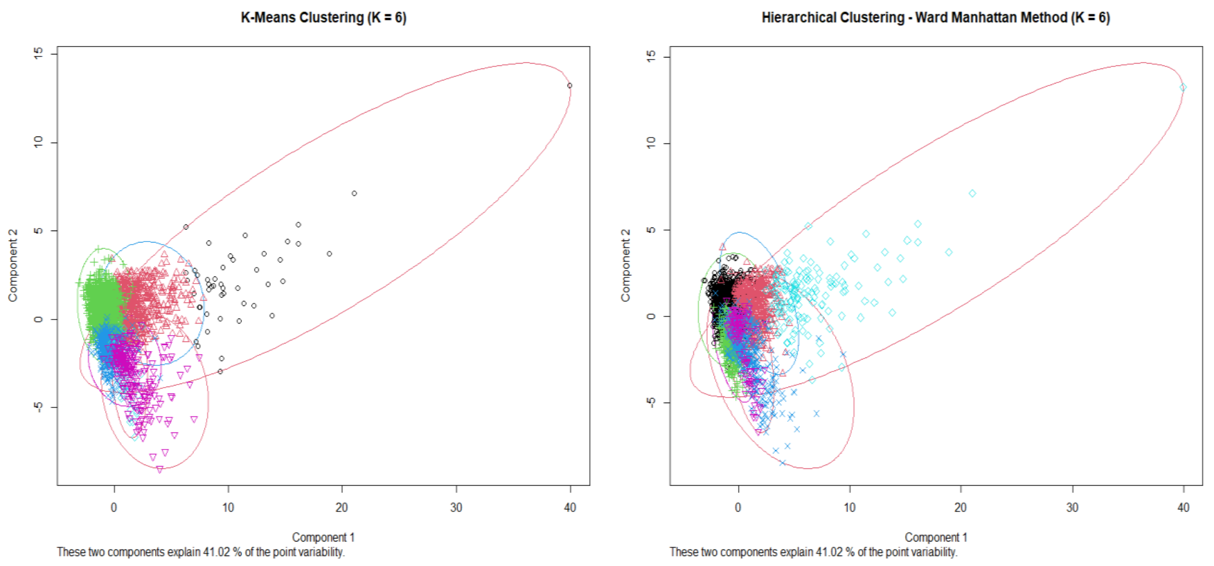


Figure 27: Top Clustering Models Comparisons - Clusplots.

The results of the two clustering methods can be interpreted from the plots. The K-Means plot shows **relatively well-defined, separated, and distinct clusters with slightly clearer boundaries**, whereas the Hierarchical plot shows **some overlap between the clusters, indicating less distinct cluster separation**. This suggests that the K-Means method with 6 clusters provides better clustering results than the Hierarchical method using the Ward Manhattan distance metric. Overall, these results suggest that **K-Means is a better clustering method** for this case compared to Hierarchical Clustering with the Ward Manhattan method.

5.7. ANOVA-based Variable Selection for Improved Clustering Analysis

In clustering, it is important to select variables that significantly contribute to the formation of clusters. Analyzing the contribution of variables to the clustering can be done through the analysis of variance (ANOVA) test. ANOVA helps to determine whether the means of the variables are significantly different across clusters. **By identifying variables that do not significantly contribute to the clustering, we can reduce noise and enhance the accuracy of the clustering**.

To perform ANOVA on our data, we first scaled the variables and added the cluster from the best clustering model (K-Means with K = 6). We then used the lapply function to apply the ANOVA test to each variable. The p-values of the ANOVA results were extracted and analyzed. **Variables with p-values greater than 0.05, indicating they do not significantly contribute to the clustering, were removed.**

The ANOVA test identified only one variable, "**AGE775214**," that did not significantly contribute to the clustering. Therefore, we removed it from our analysis. We re-ran the clustering models without this variable to further validate the clustering. No further removal was necessary. Overall, ANOVA is a useful technique for selecting variables that significantly contribute to clustering analysis. By removing variables that do not contribute to the clustering, we can improve the accuracy and robustness of the clustering. The re-run of clustering models after ANOVA validation helps to ensure the reliability of the clustering analysis.

5.8. Hierarchical Clustering on ANOVA-based Variable Selection

Based on the results of the ANOVA-based variable selection that was conducted earlier, we removed the variable "AGE775214" and continued with the remaining variables for our clustering analysis. Here, we applied the hierarchical clustering algorithm using two different methods - the better two from the earlier implementation (the others were also implemented but not presented since the results is again very poor), **Ward method with Euclidean distance** and **Ward method with Manhattan distance**, to identify the optimal number of clusters in our dataset.

For the Ward method with Euclidean distance, we first created a dendrogram to visualize the hierarchical relationship among the data points. We identified **9 clusters** as the most appropriate option, based on the silhouette coefficient plot, which measures the quality of clustering. The average silhouette value of **0.3 with homogeneous clusters was achieved**.

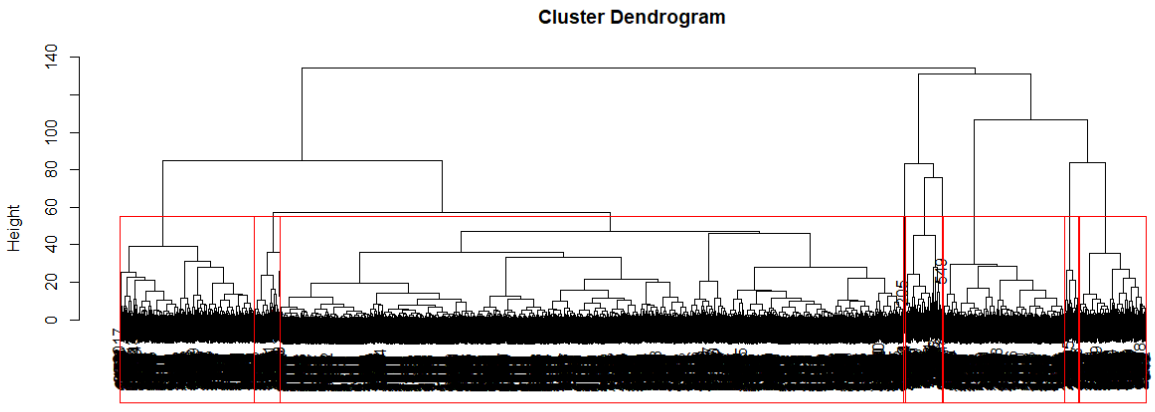


Figure 28: Hierarchical Clustering with 9 clusters Ward method with Euclidean distance dendrogram.

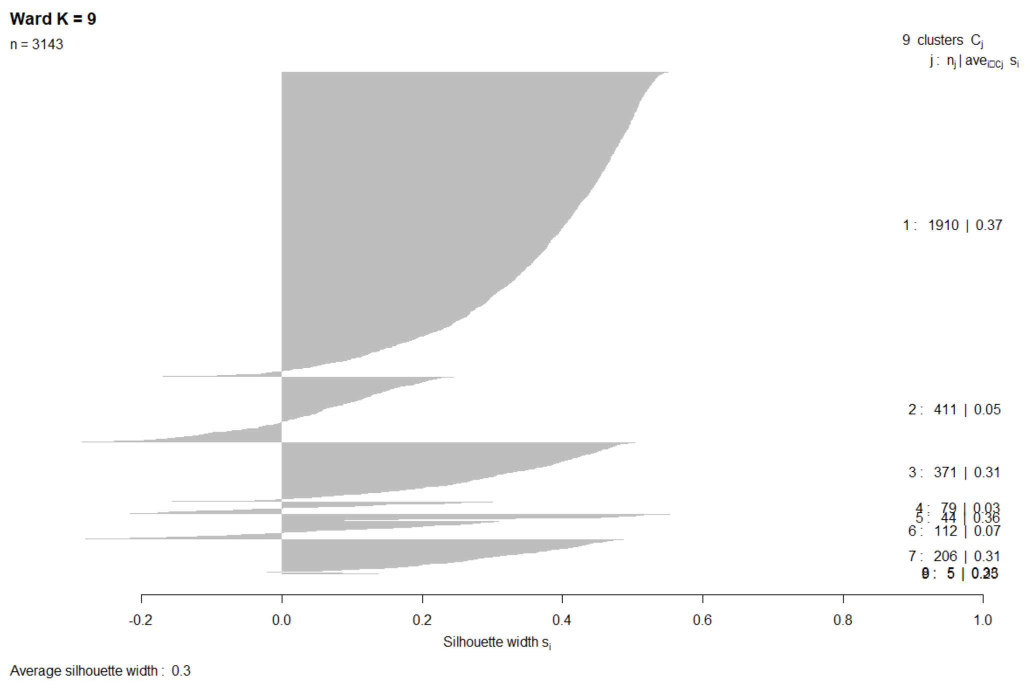


Figure 29: Hierarchical Clustering with 9 clusters Ward method with Euclidean distance - Silhouette Plots.

For the Ward method with Manhattan distance, we again created a dendrogram to visualize the hierarchical relationship among the data points. We identified **4 clusters** as the most appropriate option, based on the silhouette coefficient plot, which measures the quality of clustering. The average silhouette value of **0.34** was achieved. Thus, we select this as the **best hierarchical clustering model - so far**.

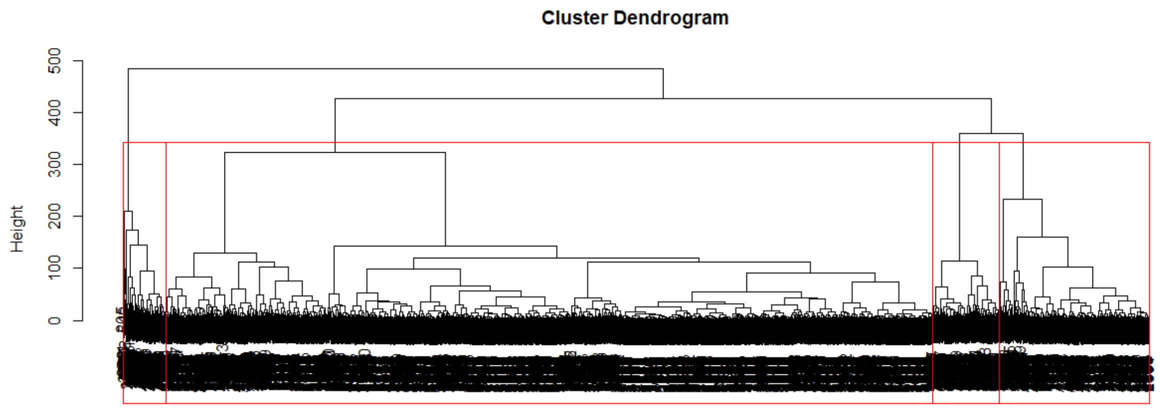


Figure 30: Hierarchical Clustering with 9 clusters Ward method with Manhattan distance dendrogram.

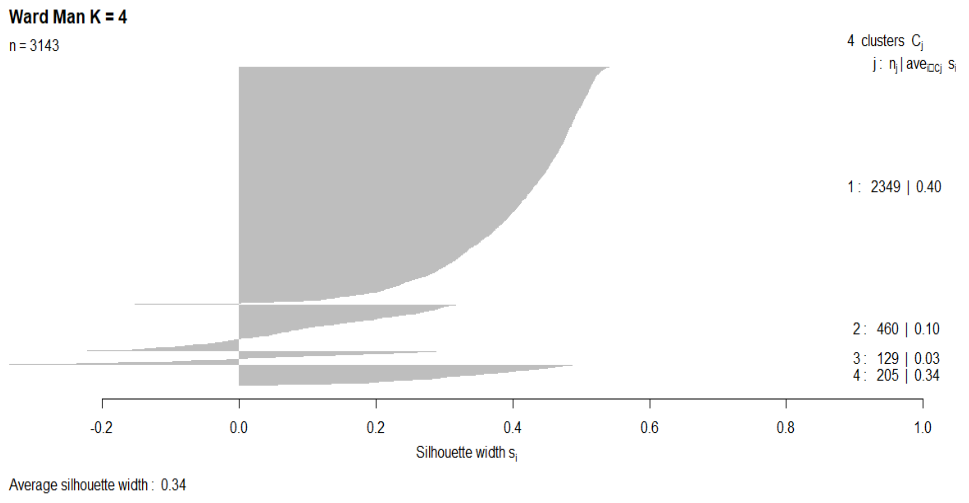


Figure 31: Hierarchical Clustering with 9 clusters Ward method with Manhattan distance - Silhouette Plots.

Based on the results obtained from hierarchical clustering using the Ward method with different distance metrics, the model with **Manhattan** distance yielded a **higher average silhouette value** compared to the model with Euclidean distance. The average silhouette value for the model with Manhattan distance was **0.34**, which indicates that the clusters are **more homogeneous compared** (also fewer negative values) to the clusters formed by the model with Euclidean distance (average silhouette value of 0.3).

The silhouette coefficient measures how well each data point fits into its assigned cluster based on the distance between the data point and other points within the cluster compared to points in other clusters. A higher silhouette coefficient value indicates that the data points within each cluster are more similar to each other and dissimilar to points in other clusters. Therefore, based on the silhouette coefficient and the dendrogram, the hierarchical clustering model using the Ward method with **Manhattan** distance and **four clusters** seems to be the most appropriate option for this dataset.

5.9. K-Means Clustering on ANOVA-based Variable Selection

Moving on to the K-Means clustering, we performed an elbow plot to determine the optimal number of clusters. We used the kmeans function to create clusters and computed the mean of each variable by cluster. We then created different clustering models to examine the consistency of the results. We used the aggregate function to compute the mean of each variable by cluster and bind the cluster assignments to the original data frame. The elbow plot suggested that the optimal number of clusters is 8. We ran K-Means algorithm with K = 4 - 9 to validate the results.

Then, we evaluate the quality of the clustering based on the silhouette plots. Based on the analysis, it seems that the K-Means algorithm with K=7 clusters is the best option, as it has the highest average silhouette width and does not have any negative values (also less miscalassifies), indicating good separation between clusters. K=4 and K=8 are also good options, K=4 seems to be a simpler model.

Through the Adjusted Rand Index (ARI) no summaries were outputted. To be more precise, Adjusted Rand Index measures the similarity between two clusterings. It ranges between -1 and 1, where a value of 1 indicates that the two clusterings are identical, while a value of 0 indicates that they are completely random.

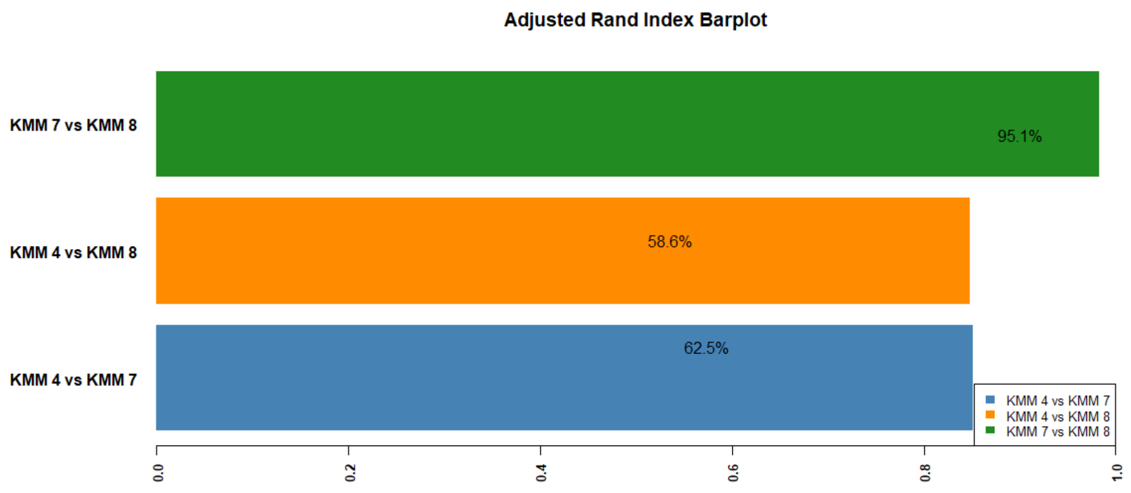


Figure 32: Adjusted Rand Index (ARI) for the top three K-Means models.

In our case, the ARI values for the K-Means models are:

- ARI between K-Means with 4 clusters and K-Means with 7 clusters: 0.62
- ARI between K-Means with 4 clusters and K-Means with 8 clusters: 0.59
- ARI between K-Means with 7 clusters and K-Means with 8 clusters: 0.95

These values suggest that the K-Means with 7 clusters is the most similar to the K-Means with 8 clusters, while the K-Means with 4 clusters is less similar to both. This is a summary that is logical since we expected the 4 clusters model to be different from 7 and 8 clusters one, thus we cannot conclude any significant summaries.

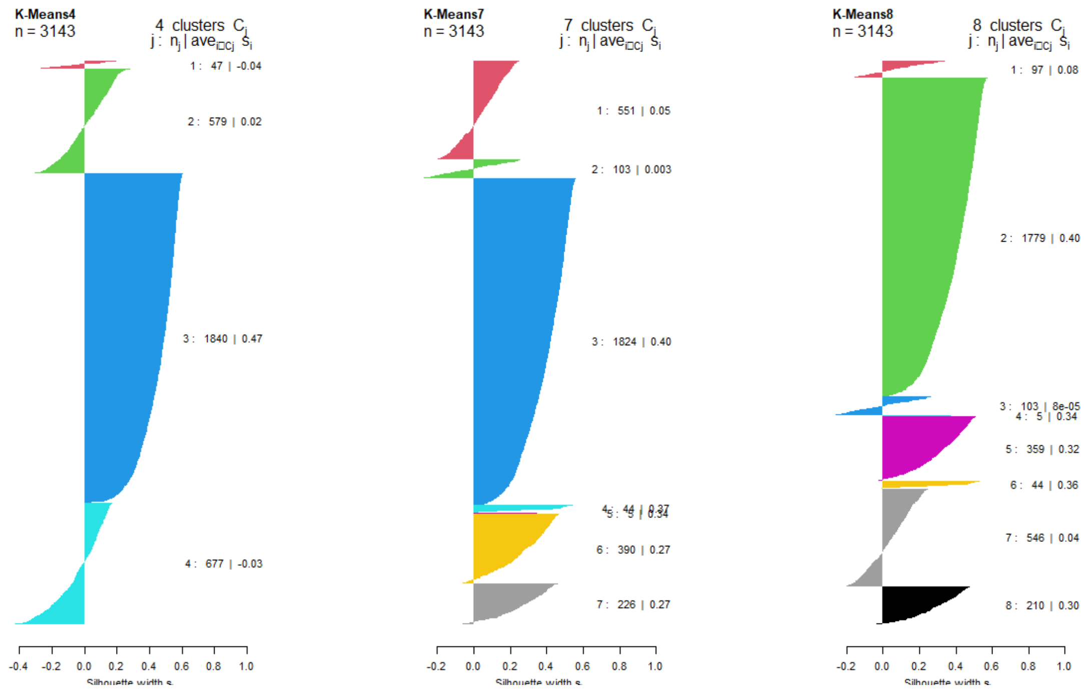


Figure 33: Top K-Means Clustering Models Comparisons - Silhouette Plots.

From the above silhouette plots, we can see that the model with the 7 clusters seems to perform better than the other two - which also have great performances. Regarding the silhouette width the 4 clusters model has **0.31**, while the other two have **0.3**. To sum up, based on our view we select the model with 7 clusters.

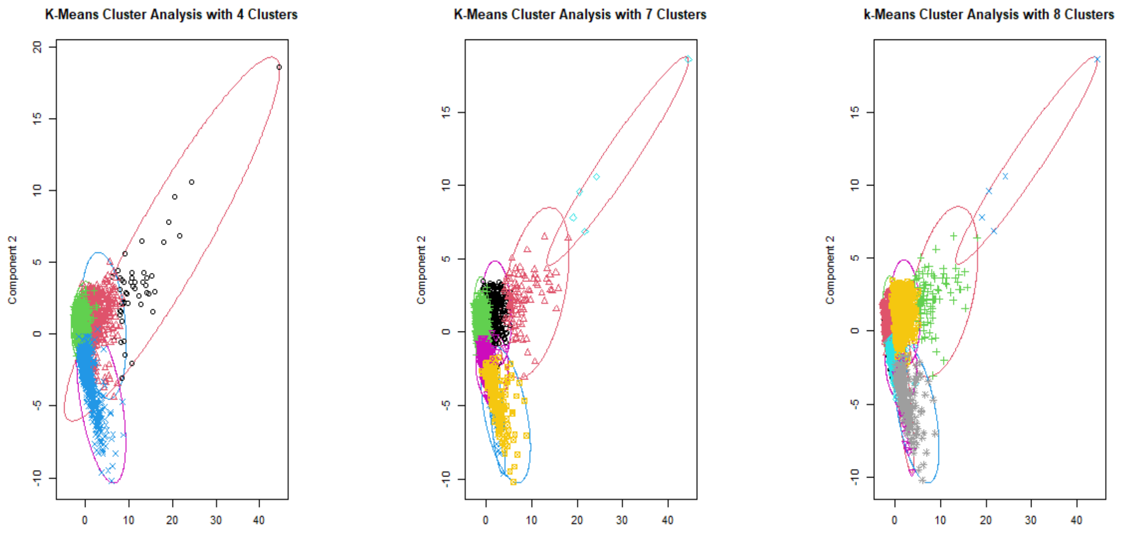


Figure 34: Top K-Means Clustering Models Comparisons - Clusplots.

Again, we have models with different number of clusters, therefore we expect the 7 and 8 to identify the outliers - as they do, and based on the above figure we conclude to **model with the 7 clusters is the better one**.

5.10. Models Comparisons on ANOVA-based Variable Selection

This section involves a comparison of K-Means and Hierarchical Clustering using the Ward Manhattan method by utilizing silhouette plots and clusplots. Silhouette plots help to evaluate the similarity of an observation to its own cluster in relation to other clusters. On the other hand, clusplots aid in visualizing the clusters based on their pairwise distance.

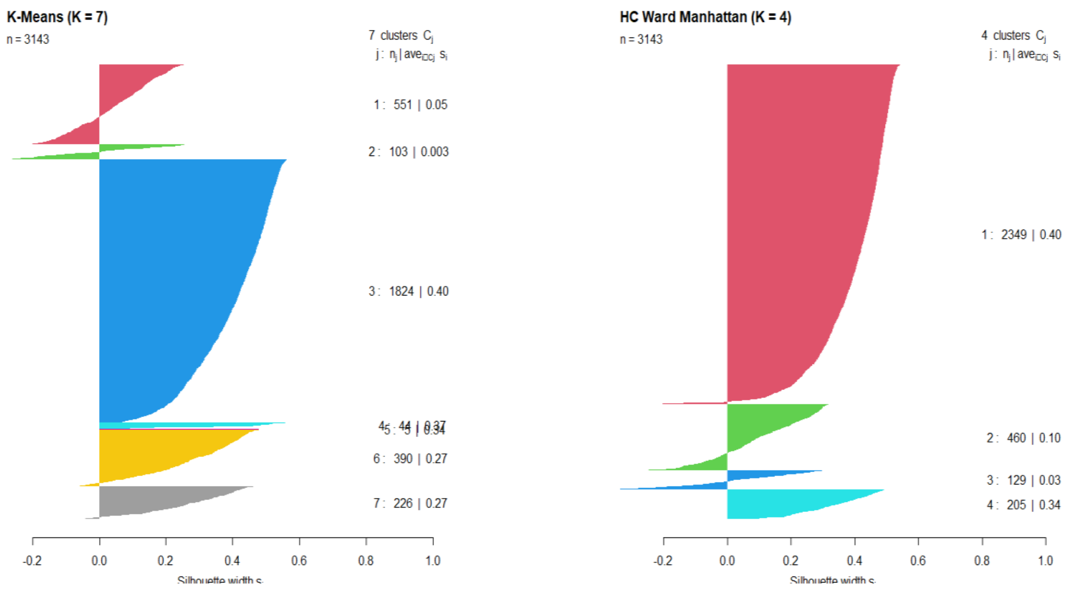


Figure 35: Top Clustering Models Comparisons on ANOVA-based Variable Selection - Silhouette Plots.

The silhouette plot shows that K-Means with 7 clusters has a lower silhouette score than Hierarchical Clustering with 4 clusters using the Ward Manhattan method. This suggests that the Hierarchical Clustering with 4 clusters using the Ward Manhattan method produces more well-defined clusters than the K-Means clustering one.

Finally, the clusplots show the clusters and their centroids in two-dimensional space (displays the clusters in a scatter plot). Each point represents an observation in the data, and its color indicates the cluster to which it belongs. The clusters are separated by ellipses that indicate the cluster boundaries.

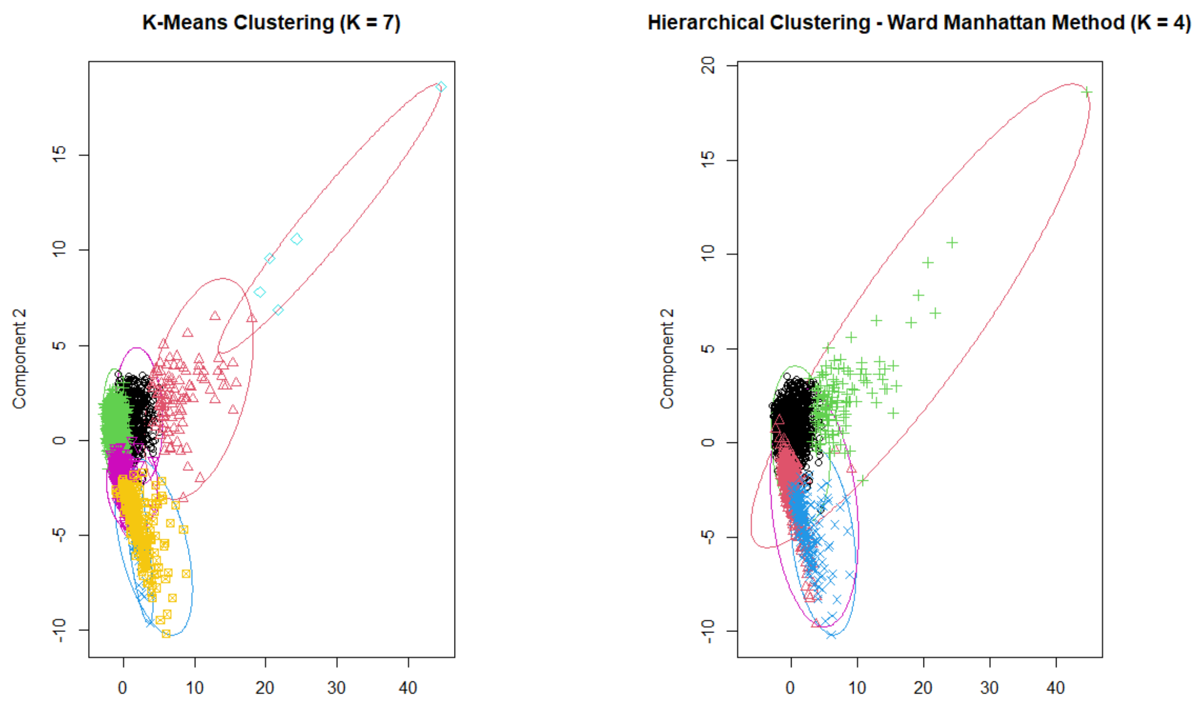


Figure 36: Top Clustering Models Comparisons on ANOVA-based Variable Selection - Clusplots.

The results of the two clustering methods can be interpreted from the plots. The K-Means plot shows relatively well-defined, separated, and distinct clusters with slightly clearer boundaries, whereas the Hierarchical plot shows some overlap between the clusters, indicating less distinct cluster separation. This suggests that the K-Means method with 7 clusters provides better clustering results than the Hierarchical method using the Ward Manhattan distance metric.

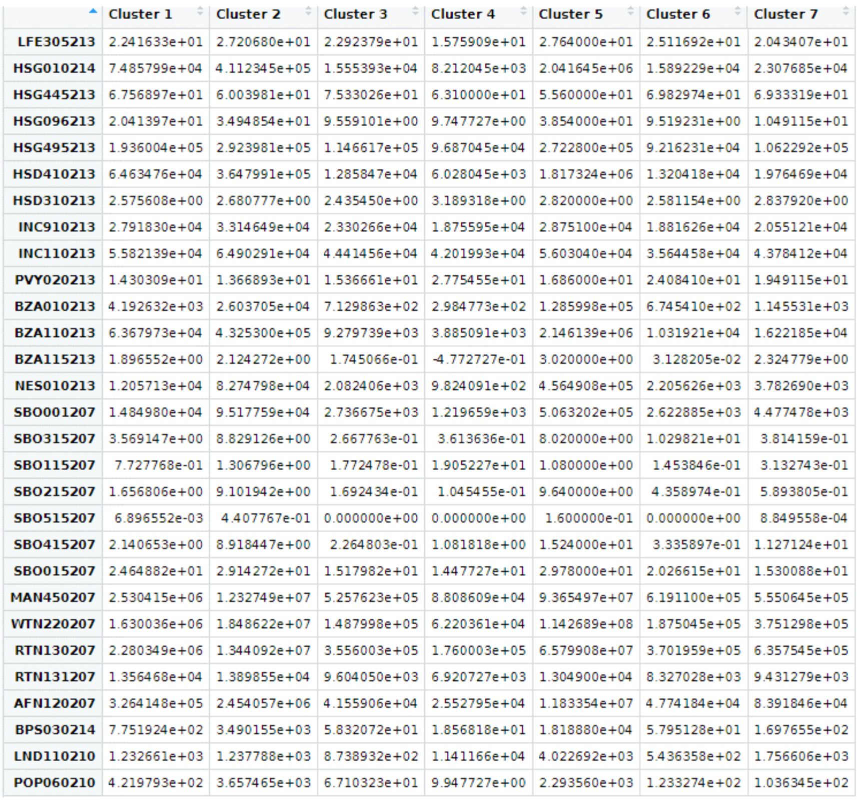
Overall, these results suggest that **K-Means is a better clustering method for this case compared to Hierarchical Clustering with the Ward Manhattan method, despite the lower average silhouette value (0.34 to 0.3) since the K-Means model has distinct clusters with slightly clearer boundaries and unique clusters for the outliers** (if we could exclude it the avg silhouette value would be closer or even greater with the Hierarchical one).

5.11. Conclusion - Profiling Clusters

Profiling clusters is a common task in data analysis, as it helps us understand the characteristics of different groups in the data. In our case, we have already identified seven clusters based on their demographic attributes (K-Means with 7 clusters). Now we want to profile these clusters using the economic variables.

To do this, we first add the cluster assignments to the economics dataset and compute the mean values of each variable for each cluster (we could use the median but since some variables have few data e.g. the firms own by variables, and the median does not make sense). We then create a parallel coordinate plot of the economics data with color-coded clusters, where each line represents the mean of each cluster, and each economic variable is shown as a vertical axis. The plot helps us visualize the differences between the clusters and identify the variables that distinguish them the most.

We use the GGally and plotly packages to create an interactive plot that allows us to explore the data more efficiently. By hovering over the lines, we can see the values of each variable for each county, and by selecting specific variables on the right-hand side, we can highlight the lines that have extreme values for these variables. This allows us to identify the outliers and investigate them further (you will be able to find the html outputted file in the drive - *Economics_explanation_clusters.html*). Below you can see all seven clusters mean values based on the economics variables and a visual representation of it.



	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
LFE305213	2.241633e+01	2.720680e+01	2.292379e+01	1.575909e+01	2.764000e+01	2.511692e+01	2.043407e+01
HSG010214	7.485799e+04	4.112345e+05	1.555393e+04	8.212045e+03	2.041645e+06	1.589229e+04	2.307685e+04
HSG445213	6.756897e+01	6.003981e+01	7.533026e+01	6.310000e+01	5.560000e+01	6.982974e+01	6.933319e+01
HSG096213	2.041397e+01	3.494854e+01	9.559101e+00	9.747727e+00	3.854000e+01	9.519231e+00	1.049115e+01
HSG495213	1.936004e+05	2.923981e+05	1.146617e+05	9.687045e+04	2.722800e+05	9.216231e+04	1.062292e+05
HSD410213	6.463476e+04	3.647991e+05	1.285847e+04	6.028045e+03	1.817324e+06	1.320418e+04	1.976469e+04
HSD310213	2.575608e+00	2.680777e+00	2.435450e+00	3.189318e+00	2.820000e+00	2.581154e+00	2.837920e+00
INC910213	2.791830e+04	3.314649e+04	2.330266e+04	1.875595e+04	2.875100e+04	1.881626e+04	2.055121e+04
INC110213	5.582139e+04	6.490291e+04	4.441456e+04	4.201993e+04	5.603040e+04	3.564458e+04	4.378412e+04
PVY020213	1.430309e+01	1.366893e+01	1.536661e+01	2.775455e+01	1.686000e+01	2.408410e+01	1.949115e+01
BZA010213	4.192632e+03	2.603705e+04	7.129863e+02	2.984773e+02	1.285998e+05	6.745410e+02	1.145531e+03
BZA110213	6.367973e+04	4.325300e+05	9.279739e+03	3.885091e+03	2.146139e+06	1.031921e+04	1.622185e+04
BZA115213	1.896552e+00	2.124272e+00	1.745066e-01	-4.772727e-01	3.020000e+00	3.128205e-02	2.324779e+00
NES010213	1.205713e+04	8.274798e+04	2.082406e+03	9.824091e+02	4.564908e+05	2.205626e+03	3.782690e+03
SBO001207	1.484980e+04	9.517759e+04	2.736675e+03	1.219659e+03	5.063202e+05	2.622885e+03	4.477478e+03
SBO315207	3.569147e+00	8.829126e+00	2.667763e-01	3.613636e-01	8.020000e+00	1.029821e+01	3.814159e-01
SBO115207	7.727768e-01	1.306796e+00	1.772478e-01	1.905227e+01	1.080000e+00	1.453846e-01	3.132743e-01
SBO215207	1.656806e+00	9.101942e+00	1.692434e-01	1.045455e-01	9.640000e+00	4.358974e-01	5.893805e-01
SBO515207	6.896552e-03	4.407767e-01	0.000000e+00	0.000000e+00	1.600000e-01	0.000000e+00	8.849558e-04
SBO415207	2.140653e+00	8.918447e+00	2.264803e-01	1.081818e+00	1.524000e+01	3.335897e-01	1.127124e+01
SBO015207	2.464882e+01	2.914272e+01	1.517982e+01	1.447727e+01	2.978000e+01	2.026615e+01	1.530088e+01
MAN450207	2.530415e+06	1.232749e+07	5.257623e+05	8.808609e+04	9.365497e+07	6.191100e+05	5.550645e+05
WTN220207	1.630036e+06	1.848622e+07	1.487998e+05	6.220361e+04	1.142689e+08	1.875045e+05	3.751298e+05
RTN130207	2.280349e+06	1.344092e+07	3.556003e+05	1.760003e+05	6.579908e+07	3.701959e+05	6.357545e+05
RTN131207	1.356468e+04	1.389855e+04	9.604050e+03	6.920727e+03	1.304900e+04	8.327028e+03	9.431279e+03
AFN120207	3.264148e+05	2.454057e+06	4.155906e+04	2.552795e+04	1.183354e+07	4.774184e+04	8.391846e+04
BPS030214	7.751924e+02	3.490155e+03	5.832072e+01	1.856818e+01	1.818880e+04	5.795128e+01	1.697655e+02
LND110210	1.232661e+03	1.237788e+03	8.738932e+02	1.141166e+04	4.022692e+03	5.436358e+02	1.756606e+03
POP060210	4.219793e+02	3.657465e+03	6.710323e+01	9.947727e+00	2.293560e+03	1.233274e+02	1.036345e+02

Figure 37: Clusters description table (mean) based on economics variables.

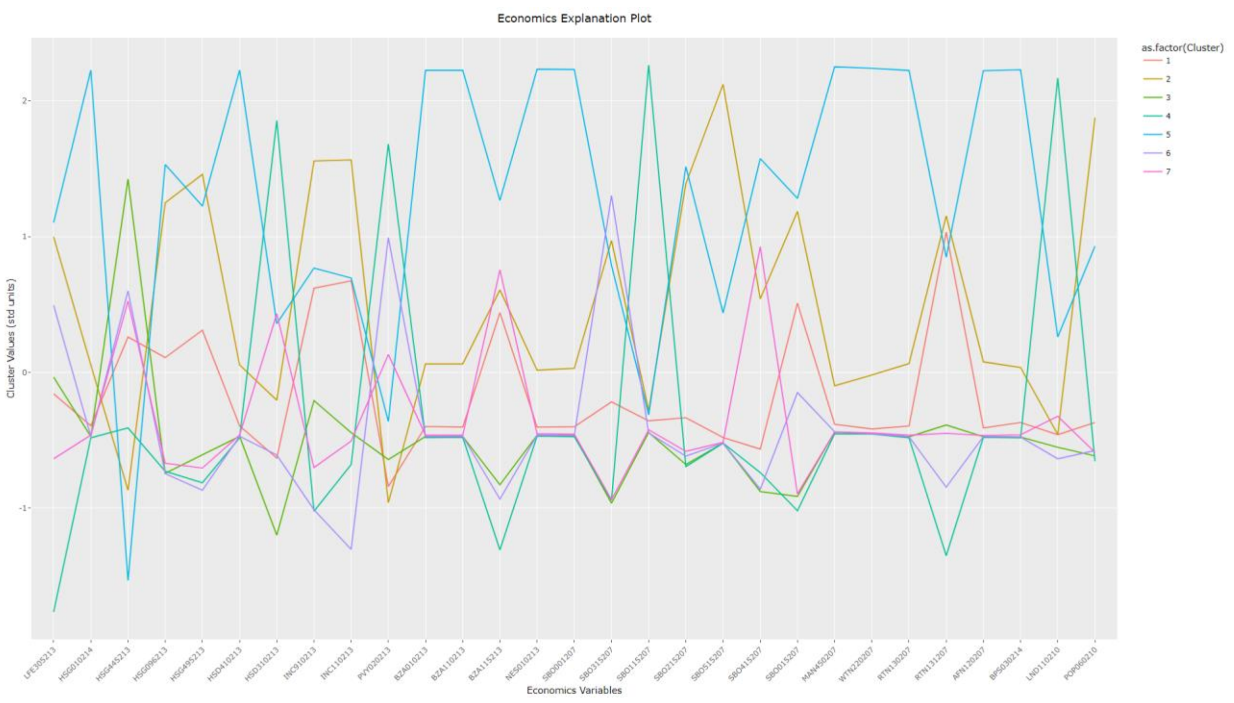


Figure 38: All seven clusters economic explanation plot.

To compare the clusters based on the economic variables, we could look at the mean values of each variable for each cluster, as shown in the table that we presented earlier. Here are some observations about the observed insights between the clusters:

Cluster 5 has either the highest or the lowest values to the most of the economics characteristics, something we can expect since it is the "outliers", cluster with only **5 observations** out of the total 3143. Thus we can expect to have extreme values especially in an economics characteristics case like this one. By excluding it from the plot we have the following:

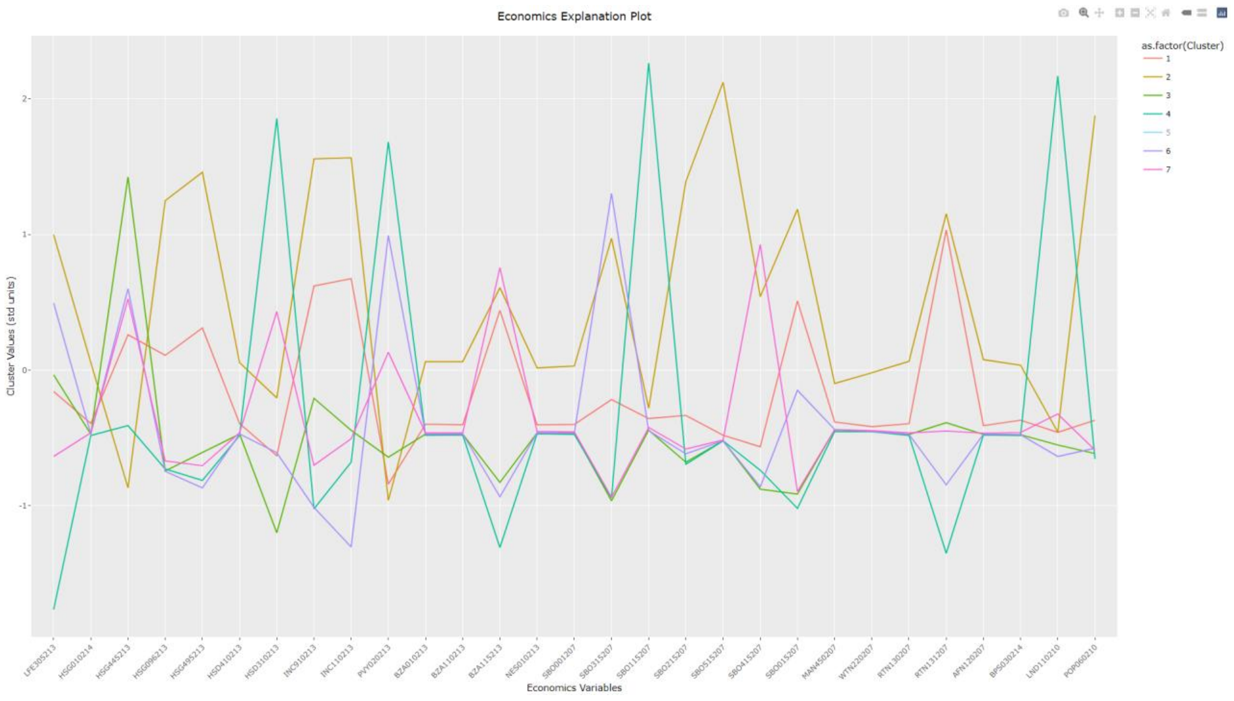


Figure 39: : The six clusters economic explanation plot - without the outliers' cluster (5th cluster).

Cluster 1 contains **551 observations** (about the 1/6 of the total observations) and is the cluster with the smaller silhouette width, while presents along with the Cluster 2 the biggest misclassification rate compared with the other 4 clusters. If we can notice something from the its economcs characteristics is that this cluster it is not present any highest or lowest values, but we can summarize that when it has the **second highest values** in some variables like the Private nonfarm employment, percent change (BZA115213), Retail sales per capita (RTN131207), Retail sales (RTN130207), Population per square mile (POP060210), Women-owned firms (SBO015207), Hispanic-owned firms (SBO415207), Manufacturers shipments (MAN450207), Median value of owner-occupied housing units (HSG495213), Housing units in multi-unit structures (HSG096213), the two incomes variables Median household income (INC110213), Per capita money income in past 12 months (INC910213), Merchant wholesaler sales (WTN220207), Asian-owned firms (SBO215207) and Native Hawaiian- and Other Pacific Islander-owned firms, percent (SBO515207) is always only behind the Cluster 2. However, it is important to note that Cluster 2 consistently exhibits the highest values in these variables.

Cluster 2 is composed of only **103 observations**, which is the smallest cluster in terms of size. However, it has the highest values in the majority of the variables examined, as previously mentioned. Specifically, Cluster 2 has the highest values in Private nonfarm employment, percent change (BZA115213), Retail sales per capita (RTN131207), Retail sales (RTN130207), Population per square mile (POP060210), Women-owned firms (SBO015207), Hispanic-owned firms (SBO415207), Manufacturers shipments (MAN450207), Median value of owner-occupied housing units (HSG495213), Housing units in multi-unit structures (HSG096213), Median household income (INC110213), Per capita money income in the past 12 months (INC910213), Merchant wholesaler sales (WTN220207), Asian-owned firms (SBO215207), and Native Hawaiian- and Other Pacific Islander-owned firms, percent (SBO515207). Nevertheless, as previously mentioned, Cluster 2 presents a smaller number of observations and has the highest misclassification rate along with Cluster 1 compared to the other clusters.

If we wanted to compare these two clusters we couldsummarize that, while Cluster 1 and Cluster 2 share several economic characteristics, including similar values in many of the variables examined, they differ significantly in terms of their size, silhouette width, and misclassification rate. Specifically, Cluster 1 exhibits the second-highest values in many of the variables examined, but also presents the highest misclassification rate compared to the other clusters. In contrast, Cluster 2 has the highest values in many of the variables examined but has a smaller number of observations and, along with Cluster 1, has the **highest misclassification rate compared to the other clusters**. Despite that if we wanted to entitle them we could say that **Cluster 1 as the "Mid-Range Economic Characteristics Cluster"** and **Cluster 2 as the "High Economic Performance Cluster"**.

Cluster 4 is composed of only **44 observations**, but it has the **second highest silhouette width of 0.37**, indicating that the observations in this cluster are relatively **well-separated from the observations in other clusters**. When analyzing the economic characteristics of this cluster, we can observe that it has the **highest percentage of persons below poverty level (PVY020213)** among all the clusters. Furthermore, Cluster 4 has the highest percentage of American Indian- and Alaska Native-owned firms (SBO115207) compared to the other clusters. The land area (LND110210) in this cluster is also the highest among all the clusters, and the number of persons per household (HSD310213) is also the highest compared to other clusters.

On the other hand, the mean travel time to work (LFE305213) is the lowest among all the clusters, indicating that people in Cluster 4 spend the least amount of time traveling to work. The population per square mile (POP060210) and the retail sales per capita (RTN131207) in this cluster are also the lowest compared to other clusters. The retail sales (RTN130207) and the total number of firms (SBO001207) are also among the lowest compared to other clusters. In addition, the percentage of women-owned firms (SBO015207), black-owned firms (SBO315207), and merchant wholesaler sales (WTN220207) are also relatively low compared to other clusters. The accommodation and food services sales (AFN120207) in Cluster 4 are also relatively low compared to other clusters.

Based on these characteristics, we **can title Cluster 4 as the "Under Poverty Level Cluster"** as it has the highest percentage of persons below poverty level compared to other clusters. An important note here is that the Cluster 4 despote the percentage of persons under the poverty level, it has relatively low values regarding the **two income related variables but the lowest belong to Cluster 6**.

In addition to having the highest percentage of black-owned firms (SBO315207), **Cluster 6**, as the "Low Income Cluster" also has the second highest mean travel time to work (LFE305213) among all clusters, only behind Cluster 2. Furthermore, this cluster has a relatively high percentage of housing units in multi-unit structures (HSG096213) and a relatively low median value of owner-occupied housing units

(HSG495213) compared to other clusters. However, despite having the lowest income level, Cluster 6 is not significantly different from other clusters in some variables such as the number of households (HSD410213), which is comparable to the number of households in other clusters. Moreover, the percentage of housing units in multi-unit structures (HSG096213) is only slightly lower compared to other clusters.

Overall, Cluster 6 is characterized by its low income level, high percentage of black-owned firms, and relatively long mean travel time to work. The housing characteristics of this cluster indicate a relatively low value of owner-occupied housing units and a relatively high percentage of housing units in multi-unit structures. Hence, we can title **Cluster 6 as the "Low Income Cluster"** highlighting its distinctive characteristics related to income and housing. It is worth noting that Cluster 6 has a relatively large number of observations (**390**) compared to some other clusters, and it also has a relatively good average silhouette width of **0.27**, indicating that the observations within this cluster are well separated from those in other clusters.

Cluster 7 is a relatively moderate-sized cluster containing **227 observations**, and it has a respectable average silhouette width of **0.27**, indicating that the observations within this cluster are well separated from those in other clusters. In terms of economic variables, Cluster 7 has the **highest percentage of Hispanic-owned firms (SBO415207) and the highest percent change in private nonfarm employment (BZA115213)**. On the other hand, this cluster has some of the lowest values in certain variables such as women-owned firms (SBO015207), black-owned firms (SBO315207), American Indian- and Alaska Native-owned firms (SBO115207), Asian-owned firms (SBO215207), total number of firms (SBO001207), Native Hawaiian- and Other Pacific Islander-owned firms (SBO515207), and mean travel time to work (LFE305213).

We can entitle Cluster 7 based on the economic characteristics as **"Private nonfarm employment and Hispanic-owned businesses leaders with low diversity"**. We can do so since Cluster 7 has the highest values in Private nonfarm employment, percent change (BZA115213) and Hispanic-owned firms, percent (SBO415207). Additionally, the cluster presents the lowest values in women-owned, black-owned, American Indian- and Alaska Native-owned, Asian-owned, and Native Hawaiian- and Other Pacific Islander-owned firms, indicating a lower level of diversity in the business landscape compared to other clusters. Overall, the title accurately captures the main economic features that distinguish Cluster 7 from other clusters in the dataset.

Cluster 3 is the **largest cluster** in the dataset, with **1824 observations**, and it has a high silhouette value of **0.4**, indicating that the observations within this cluster are tightly grouped and well separated from those in other clusters. This cluster is characterized by having the **highest homeownership rate** (HSG445213) among all clusters, indicating that residents in this cluster tend to own their homes. It also has a relatively high mean travel time to work (LFE305213), indicating that residents in this cluster may have longer commutes to work. On the other hand, it has the second lowest land area in square miles (LND110210), indicating that this cluster tends to be more urban. In terms of economic variables, this cluster has relatively low private nonfarm employment growth (BZA115213), manufacturers shipments (MAN450207), and Hispanic-owned firms (SBO415207), but it has a relatively high homeownership rate (HSG445213). It also has the lowest percentage of black-owned firms (SBO315207) and the lowest number of persons per household (HSD310213) among all clusters, indicating that this cluster may be more diverse and have smaller households.

Cluster 3 can be titled as "Homeownership and Manufacturing cluster" based on its high homeownership rate (HSG445213) and its relatively low value in manufacturing shipments (MAN450207) compared to other clusters. It also has a relatively low percentage of black-owned firms (SBO315207) and a moderate percentage of Hispanic-owned firms (SBO415207).

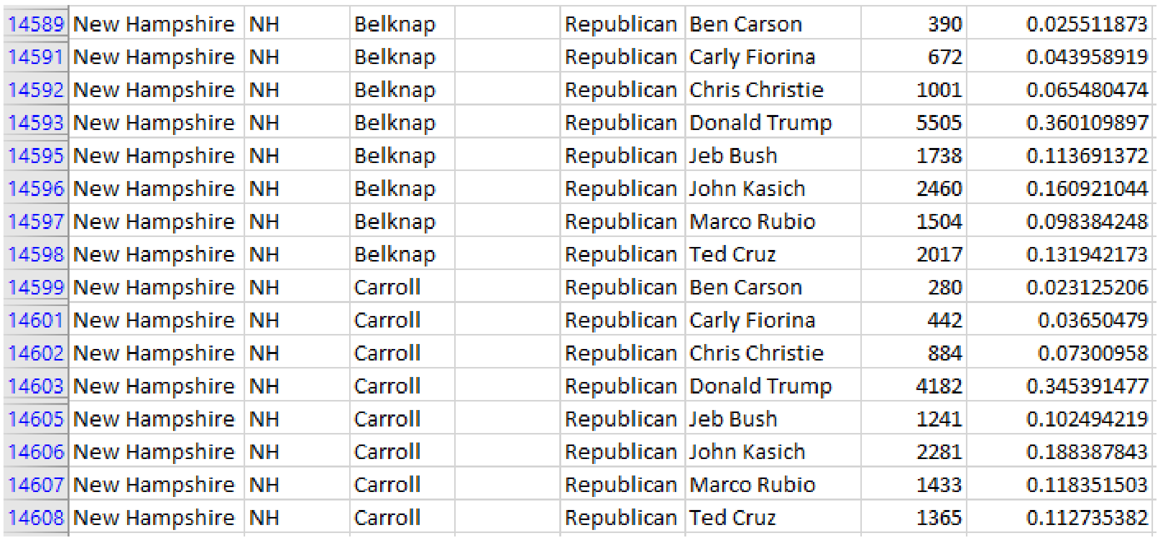
Datasets Archive

All the datasets created during this project (and the ones to relate the county_facts with the votes Excel sheets), as well as the provided one, can be found on the links provided below:

- The given dataset that contains three Excel sheets, stat BA II project I.xlsx
- Massachusetts (MA) assignment of county to town/city CSV file, MA.csv
- Connecticut (CT) assignment of county to town/city CSV file, CT.csv
- Rhode Island (RI) assignment of county to town/city CSV file, RI.csv
- Vermont (VT) assignment of county to town/city CSV file, VT.csv
- The votes dataset with the proper FIPS codes, filtered only for the Republicans votes, and containing only the response variable CSV file, us_elections_2016.csv
- The latest dataset after the merge with the 'county_facts' Excel sheet of the stat BA II project I.xlsx dataset. This is the final dataset of the data cleaning phase, us_elections_2016FINAL.csv
- The clustering dataset based on the stat BA II project I.xlsx without the response variable, the states and the fips codes but with the addition of the county variable (data points based on counties), us_elections_2016_clustering.csv
- Economics explanation/profiling clusters plot (Economics_explanation_clusters.html).

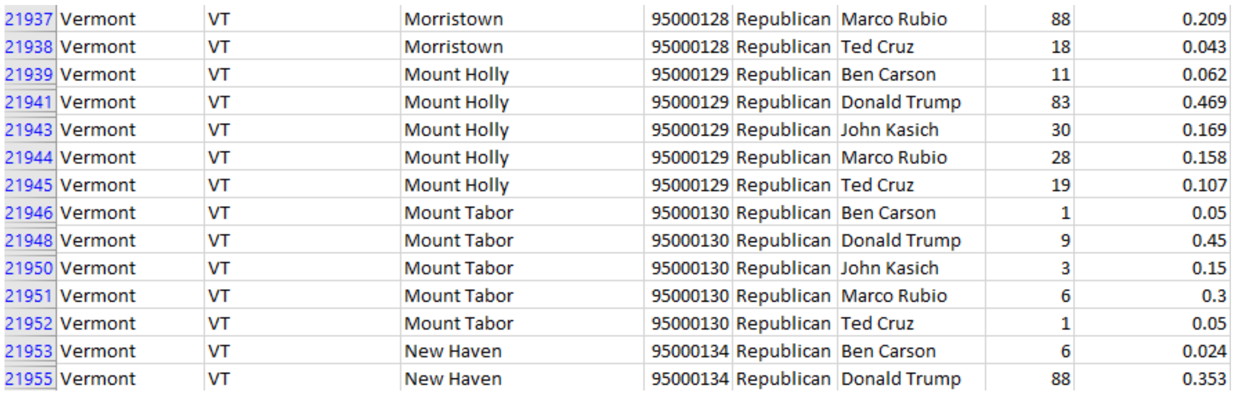
Appendix

The *Appendix* section provides additional information and data to support the analysis presented in the main body of the report.



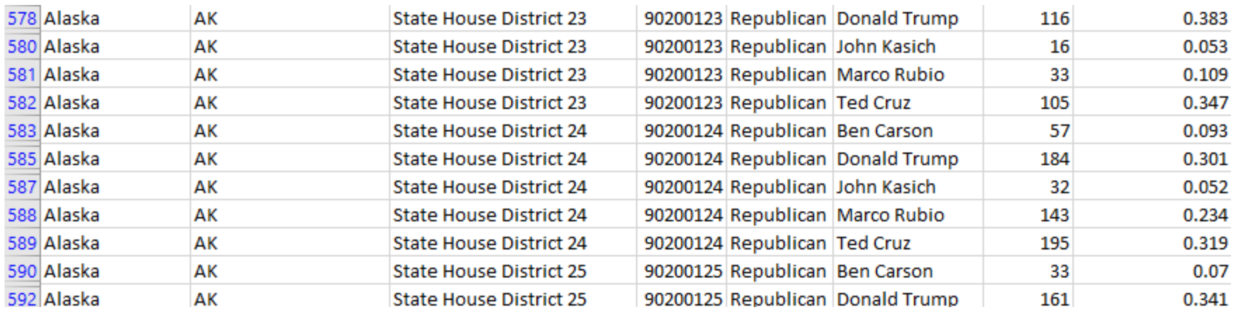
14589	New Hampshire	NH	Belknap		Republican	Ben Carson	390	0.025511873
14591	New Hampshire	NH	Belknap		Republican	Carly Fiorina	672	0.043958919
14592	New Hampshire	NH	Belknap		Republican	Chris Christie	1001	0.065480474
14593	New Hampshire	NH	Belknap		Republican	Donald Trump	5505	0.360109897
14595	New Hampshire	NH	Belknap		Republican	Jeb Bush	1738	0.113691372
14596	New Hampshire	NH	Belknap		Republican	John Kasich	2460	0.160921044
14597	New Hampshire	NH	Belknap		Republican	Marco Rubio	1504	0.098384248
14598	New Hampshire	NH	Belknap		Republican	Ted Cruz	2017	0.131942173
14599	New Hampshire	NH	Carroll		Republican	Ben Carson	280	0.023125206
14601	New Hampshire	NH	Carroll		Republican	Carly Fiorina	442	0.03650479
14602	New Hampshire	NH	Carroll		Republican	Chris Christie	884	0.07300958
14603	New Hampshire	NH	Carroll		Republican	Donald Trump	4182	0.345391477
14605	New Hampshire	NH	Carroll		Republican	Jeb Bush	1241	0.102494219
14606	New Hampshire	NH	Carroll		Republican	John Kasich	2281	0.188387843
14607	New Hampshire	NH	Carroll		Republican	Marco Rubio	1433	0.118351503
14608	New Hampshire	NH	Carroll		Republican	Ted Cruz	1365	0.112735382

Appendix-Figure 1: Votes dataset missing completely the New Hampshire (NH) FIPS codes.



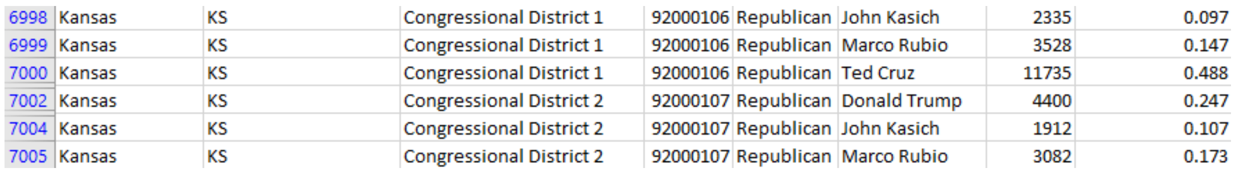
21937	Vermont	VT	Morristown	95000128	Republican	Marco Rubio	88	0.209
21938	Vermont	VT	Morristown	95000128	Republican	Ted Cruz	18	0.043
21939	Vermont	VT	Mount Holly	95000129	Republican	Ben Carson	11	0.062
21941	Vermont	VT	Mount Holly	95000129	Republican	Donald Trump	83	0.469
21943	Vermont	VT	Mount Holly	95000129	Republican	John Kasich	30	0.169
21944	Vermont	VT	Mount Holly	95000129	Republican	Marco Rubio	28	0.158
21945	Vermont	VT	Mount Holly	95000129	Republican	Ted Cruz	19	0.107
21946	Vermont	VT	Mount Tabor	95000130	Republican	Ben Carson	1	0.05
21948	Vermont	VT	Mount Tabor	95000130	Republican	Donald Trump	9	0.45
21950	Vermont	VT	Mount Tabor	95000130	Republican	John Kasich	3	0.15
21951	Vermont	VT	Mount Tabor	95000130	Republican	Marco Rubio	6	0.3
21952	Vermont	VT	Mount Tabor	95000130	Republican	Ted Cruz	1	0.05
21953	Vermont	VT	New Haven	95000134	Republican	Ben Carson	6	0.024
21955	Vermont	VT	New Haven	95000134	Republican	Donald Trump	88	0.353

Appendix-Figure 2: Vermont (VT) different codes format - not the proper 5-digit FIPS codes.



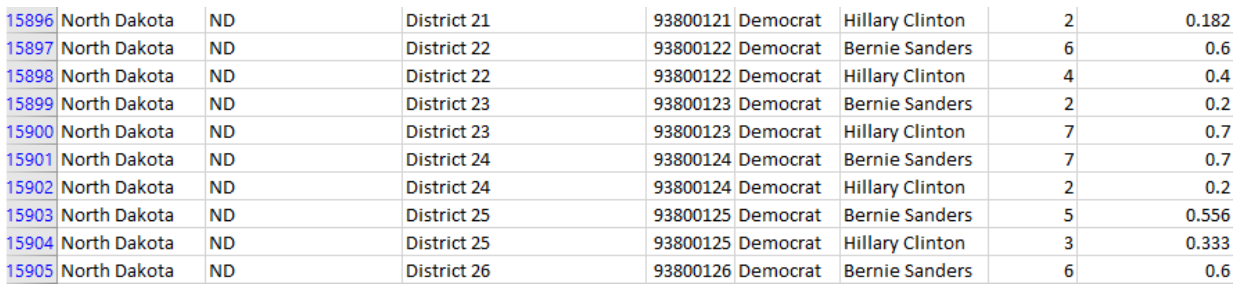
578	Alaska	AK	State House District 23	90200123	Republican	Donald Trump	116	0.383
580	Alaska	AK	State House District 23	90200123	Republican	John Kasich	16	0.053
581	Alaska	AK	State House District 23	90200123	Republican	Marco Rubio	33	0.109
582	Alaska	AK	State House District 23	90200123	Republican	Ted Cruz	105	0.347
583	Alaska	AK	State House District 24	90200124	Republican	Ben Carson	57	0.093
585	Alaska	AK	State House District 24	90200124	Republican	Donald Trump	184	0.301
587	Alaska	AK	State House District 24	90200124	Republican	John Kasich	32	0.052
588	Alaska	AK	State House District 24	90200124	Republican	Marco Rubio	143	0.234
589	Alaska	AK	State House District 24	90200124	Republican	Ted Cruz	195	0.319
590	Alaska	AK	State House District 25	90200125	Republican	Ben Carson	33	0.07
592	Alaska	AK	State House District 25	90200125	Republican	Donald Trump	161	0.341

Appendix-Figure 3: Alaska's (AK) State House District case, with not the proper 5-digit FIPS codes.



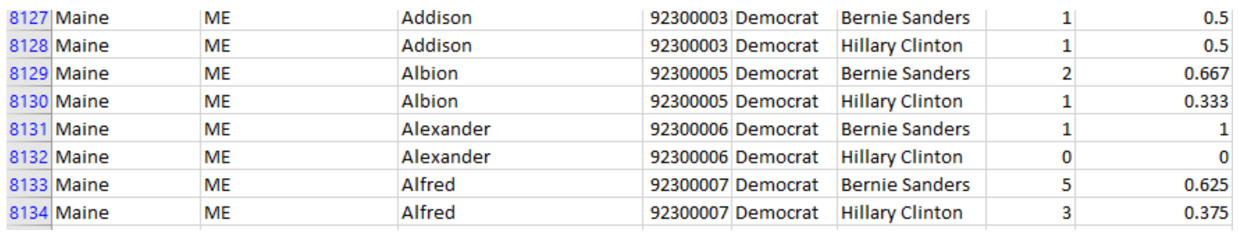
6998	Kansas	KS	Congressional District 1	92000106	Republican	John Kasich	2335	0.097
6999	Kansas	KS	Congressional District 1	92000106	Republican	Marco Rubio	3528	0.147
7000	Kansas	KS	Congressional District 1	92000106	Republican	Ted Cruz	11735	0.488
7002	Kansas	KS	Congressional District 2	92000107	Republican	Donald Trump	4400	0.247
7004	Kansas	KS	Congressional District 2	92000107	Republican	John Kasich	1912	0.107
7005	Kansas	KS	Congressional District 2	92000107	Republican	Marco Rubio	3082	0.173

Appendix-Figure 4: Kansas' (KS) Congressional District case, with not the proper 5-digit FIPS codes.



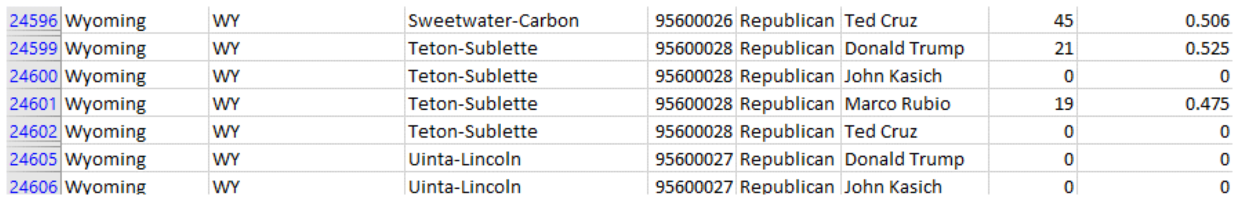
15896	North Dakota	ND	District 21	93800121	Democrat	Hillary Clinton	2	0.182
15897	North Dakota	ND	District 22	93800122	Democrat	Bernie Sanders	6	0.6
15898	North Dakota	ND	District 22	93800122	Democrat	Hillary Clinton	4	0.4
15899	North Dakota	ND	District 23	93800123	Democrat	Bernie Sanders	2	0.2
15900	North Dakota	ND	District 23	93800123	Democrat	Hillary Clinton	7	0.7
15901	North Dakota	ND	District 24	93800124	Democrat	Bernie Sanders	7	0.7
15902	North Dakota	ND	District 24	93800124	Democrat	Hillary Clinton	2	0.2
15903	North Dakota	ND	District 25	93800125	Democrat	Bernie Sanders	5	0.556
15904	North Dakota	ND	District 25	93800125	Democrat	Hillary Clinton	3	0.333
15905	North Dakota	ND	District 26	93800126	Democrat	Bernie Sanders	6	0.6

Appendix-Figure 5: North Dakota (ND)'s case contain District instead of counties and not the 5-digit FIPS codes but contains only data for Democrats candidates.



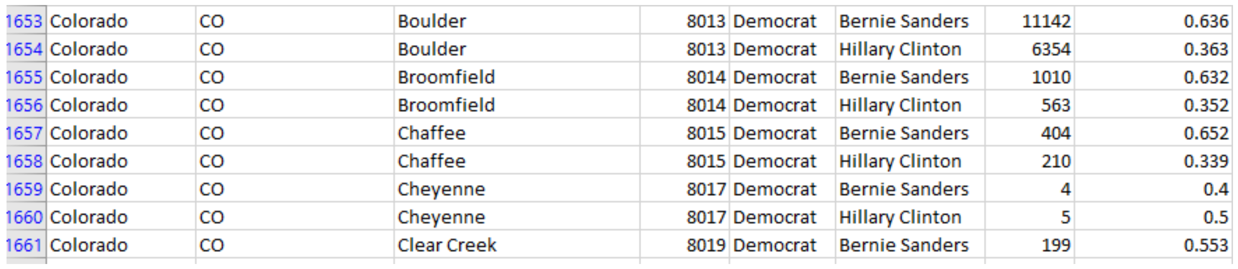
8127	Maine	ME	Addison	92300003	Democrat	Bernie Sanders	1	0.5
8128	Maine	ME	Addison	92300003	Democrat	Hillary Clinton	1	0.5
8129	Maine	ME	Albion	92300005	Democrat	Bernie Sanders	2	0.667
8130	Maine	ME	Albion	92300005	Democrat	Hillary Clinton	1	0.333
8131	Maine	ME	Alexander	92300006	Democrat	Bernie Sanders	1	1
8132	Maine	ME	Alexander	92300006	Democrat	Hillary Clinton	0	0
8133	Maine	ME	Alfred	92300007	Democrat	Bernie Sanders	5	0.625
8134	Maine	ME	Alfred	92300007	Democrat	Hillary Clinton	3	0.375

Appendix-Figure 6: Maine (ME) contains incorrect FIPS codes, but only data for Democrats candidates.



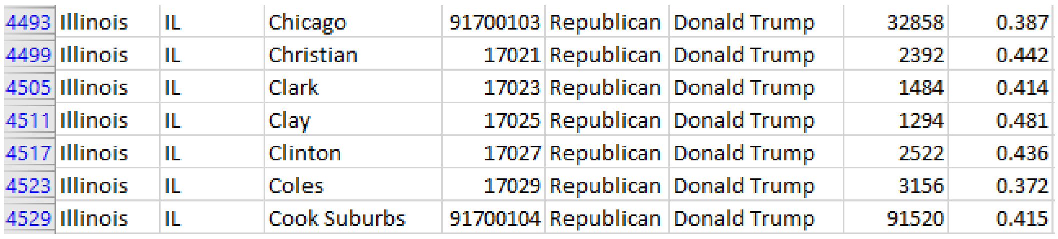
24596	Wyoming	WY	Sweetwater-Carbon	95600026	Republican	Ted Cruz	45	0.506
24599	Wyoming	WY	Teton-Sublette	95600028	Republican	Donald Trump	21	0.525
24600	Wyoming	WY	Teton-Sublette	95600028	Republican	John Kasich	0	0
24601	Wyoming	WY	Teton-Sublette	95600028	Republican	Marco Rubio	19	0.475
24602	Wyoming	WY	Teton-Sublette	95600028	Republican	Ted Cruz	0	0
24605	Wyoming	WY	Uinta-Lincoln	95600027	Republican	Donald Trump	0	0
24606	Wyoming	WY	Uinta-Lincoln	95600027	Republican	John Kasich	0	0

Appendix-Figure 7: Wyoming's (WY) data are providing in pairs with not the proper 5-digit FIPS codes.



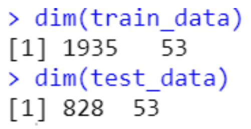
1653	Colorado	CO	Boulder	8013	Democrat	Bernie Sanders	11142	0.636
1654	Colorado	CO	Boulder	8013	Democrat	Hillary Clinton	6354	0.363
1655	Colorado	CO	Broomfield	8014	Democrat	Bernie Sanders	1010	0.632
1656	Colorado	CO	Broomfield	8014	Democrat	Hillary Clinton	563	0.352
1657	Colorado	CO	Chaffee	8015	Democrat	Bernie Sanders	404	0.652
1658	Colorado	CO	Chaffee	8015	Democrat	Hillary Clinton	210	0.339
1659	Colorado	CO	Cheyenne	8017	Democrat	Bernie Sanders	4	0.4
1660	Colorado	CO	Cheyenne	8017	Democrat	Hillary Clinton	5	0.5
1661	Colorado	CO	Clear Creek	8019	Democrat	Bernie Sanders	199	0.553

Appendix-Figure 8: Colorado (CO) contains correct formatted FIPS codes but only data for the Democrats candidates.



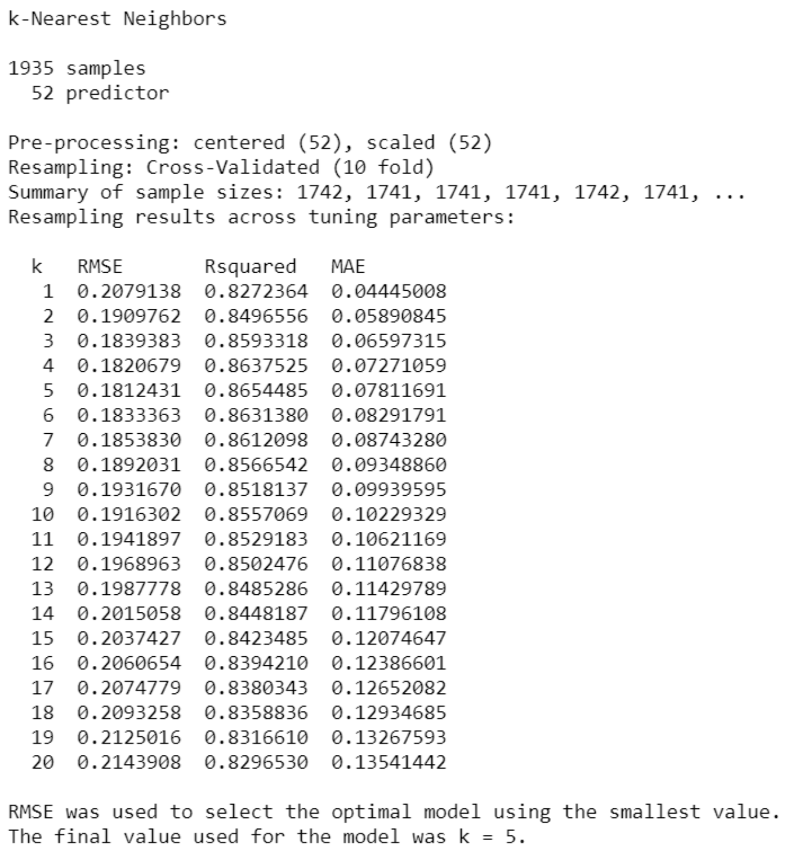
4493	Illinois	IL	Chicago	91700103	Republican	Donald Trump	32858	0.387
4499	Illinois	IL	Christian	17021	Republican	Donald Trump	2392	0.442
4505	Illinois	IL	Clark	17023	Republican	Donald Trump	1484	0.414
4511	Illinois	IL	Clay	17025	Republican	Donald Trump	1294	0.481
4517	Illinois	IL	Clinton	17027	Republican	Donald Trump	2522	0.436
4523	Illinois	IL	Coles	17029	Republican	Donald Trump	3156	0.372
4529	Illinois	IL	Cook Suburbs	91700104	Republican	Donald Trump	91520	0.415

Appendix-Figure 9: Illinois (IL) case with only the Cook County's cities have non-FIPS codes.



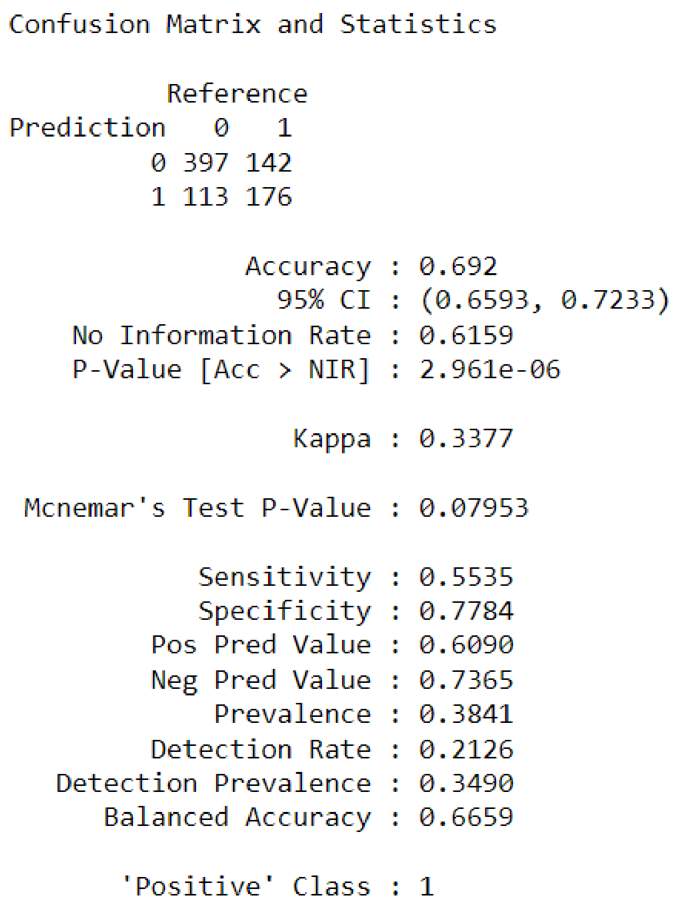
```
> dim(train_data)
[1] 1935   53
> dim(test_data)
[1] 828  53
```

Appendix-Figure 10: Check of the dimensions of the training and testing datasets to ensure they are equal.



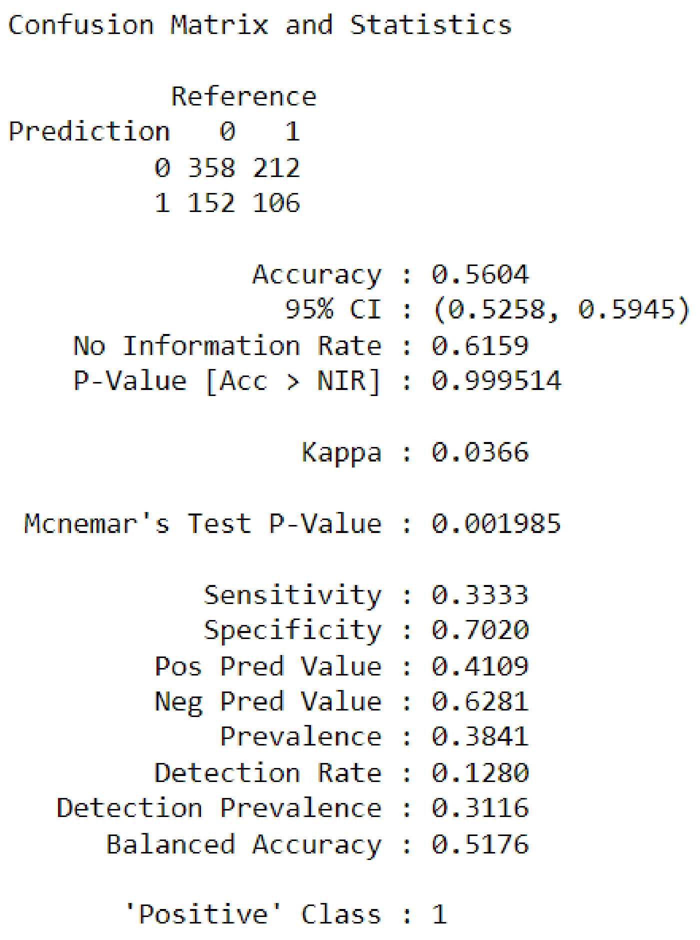
```
k-Nearest Neighbors

1935 samples
  52 predictor

Pre-processing: centered (52), scaled (52)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1742, 1741, 1741, 1741, 1742, 1741, ...
Resampling results across tuning parameters:

  k   RMSE       Rsquared   MAE
   1  0.2079138  0.8272364  0.04445008
   2  0.1909762  0.8496556  0.05890845
   3  0.1839383  0.8593318  0.06597315
   4  0.1820679  0.8637525  0.07271059
   5  0.1812431  0.8654485  0.07811691
   6  0.1833363  0.8631380  0.08291791
   7  0.1853830  0.8612098  0.08743280
   8  0.1892031  0.8566542  0.09348860
   9  0.1931670  0.8518137  0.09939595
  10  0.1916302  0.8557069  0.10229329
  11  0.1941897  0.8529183  0.10621169
  12  0.1968963  0.8502476  0.11076838
  13  0.1987778  0.8485286  0.11429789
  14  0.2015058  0.8448187  0.11796108
  15  0.2037427  0.8423485  0.12074647
  16  0.2060654  0.8394210  0.12386601
  17  0.2074779  0.8380343  0.12652082
  18  0.2093258  0.8358836  0.12934685
  19  0.2125016  0.8316610  0.13267593
  20  0.2143908  0.8296530  0.13541442

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 5.
```

Appendix-Figure 11: Find the optimal number of NN results table.



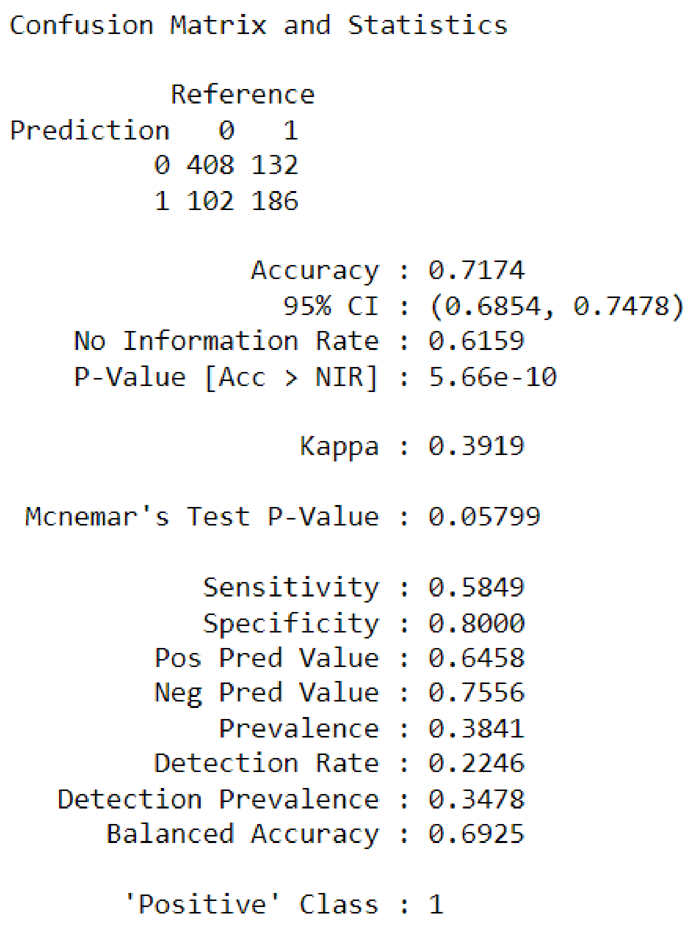
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 397 142
         1 113 176

               Accuracy : 0.692
                 95% CI : (0.6593, 0.7233)
    No Information Rate : 0.6159
    P-Value [Acc > NIR] : 2.961e-06

                  Kappa : 0.3377

 Mcnemar's Test P-Value : 0.07953

            Sensitivity : 0.5535
            Specificity : 0.7784
         Pos Pred Value : 0.6090
         Neg Pred Value : 0.7365
             Prevalence : 0.3841
         Detection Rate : 0.2126
   Detection Prevalence : 0.3490
      Balanced Accuracy : 0.6659

       'Positive' Class : 1
```

Appendix-Figure 12: Confusion Matrix and Statistics of KNN with K = 5 and scaled data.



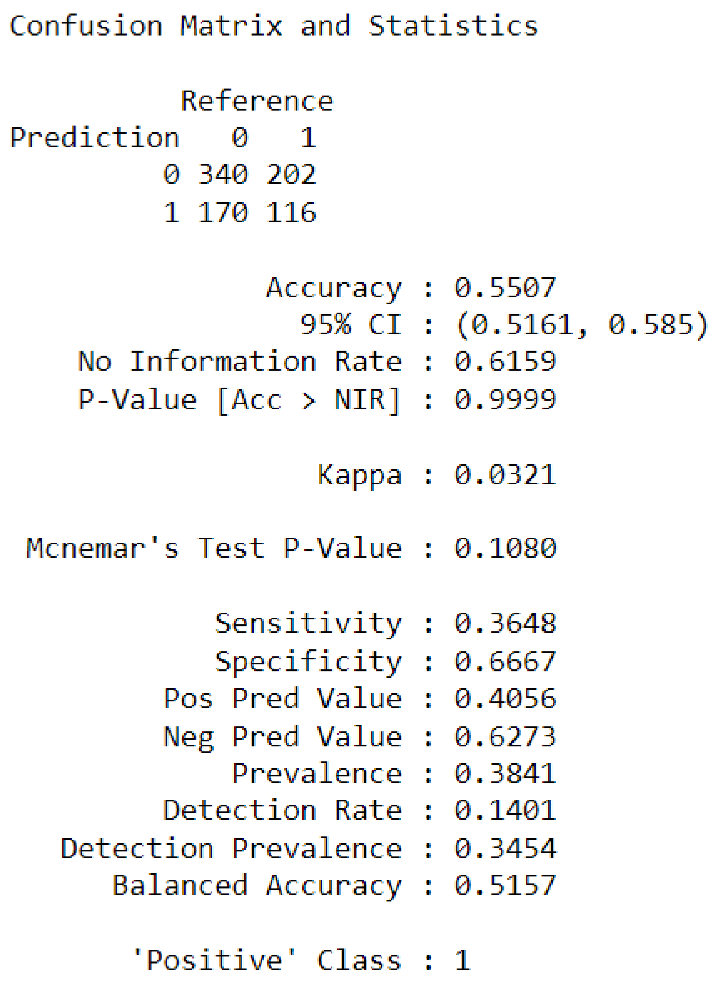
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 358 212
         1 152 106
                                         
               Accuracy : 0.5604         
                 95% CI : (0.5258, 0.5945)
    No Information Rate : 0.6159         
    P-Value [Acc > NIR] : 0.999514       
                                         
                  Kappa : 0.0366         
                                         
 Mcnemar's Test P-Value : 0.001985       
                                         
            Sensitivity : 0.3333         
            Specificity : 0.7020         
         Pos Pred Value : 0.4109         
         Neg Pred Value : 0.6281         
             Prevalence : 0.3841         
         Detection Rate : 0.1280         
   Detection Prevalence : 0.3116         
      Balanced Accuracy : 0.5176         
                                         
       'Positive' Class : 1              
```

Appendix-Figure 13: Confusion Matrix and Statistics of KNN with K = 5 and not scaled data.



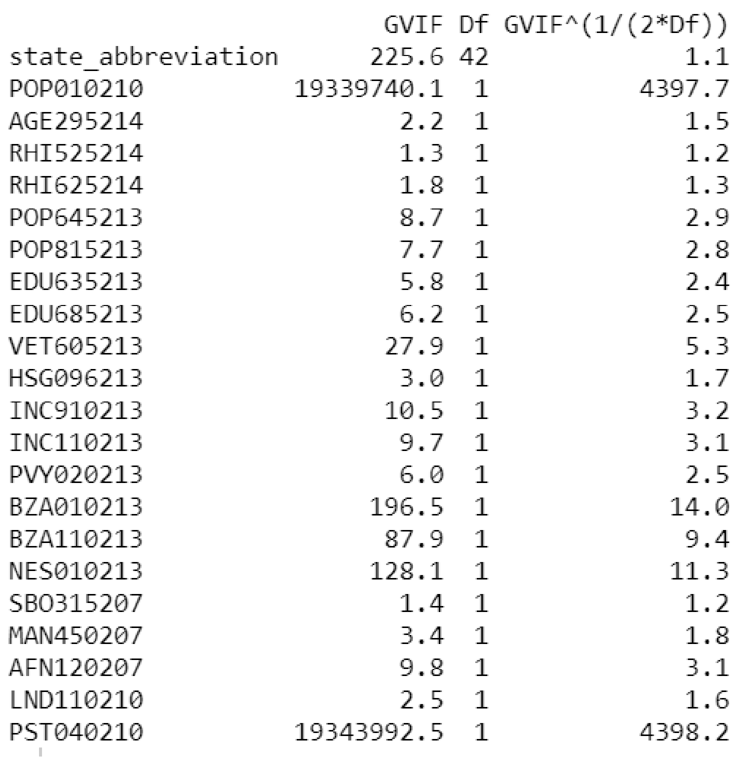
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 408 132
         1 102 186
                                         
               Accuracy : 0.7174         
                 95% CI : (0.6854, 0.7478)
    No Information Rate : 0.6159         
    P-Value [Acc > NIR] : 5.66e-10       
                                         
                  Kappa : 0.3919         
                                         
 Mcnemar's Test P-Value : 0.05799        
                                         
            Sensitivity : 0.5849         
            Specificity : 0.8000         
         Pos Pred Value : 0.6458         
         Neg Pred Value : 0.7556         
             Prevalence : 0.3841         
         Detection Rate : 0.2246         
   Detection Prevalence : 0.3478         
      Balanced Accuracy : 0.6925         
                                         
       'Positive' Class : 1              
```

Appendix-Figure 14: Confusion Matrix and Statistics of KNN with K = 4 and scaled data.



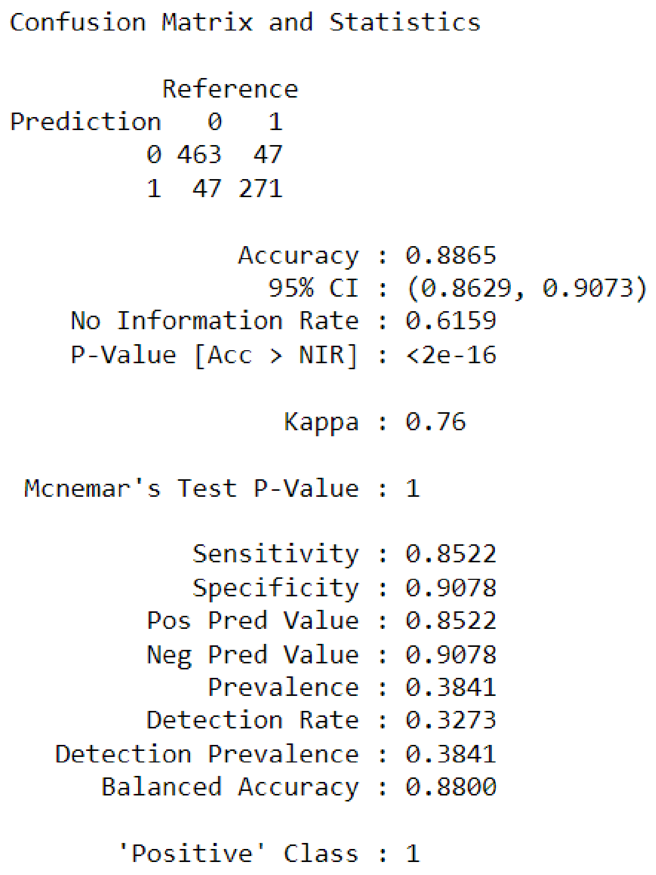
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 340 202
         1 170 116
                                         
               Accuracy : 0.5507         
                 95% CI : (0.5161, 0.585)
    No Information Rate : 0.6159         
    P-Value [Acc > NIR] : 0.9999         
                                         
                  Kappa : 0.0321         
                                         
 Mcnemar's Test P-Value : 0.1080         
                                         
            Sensitivity : 0.3648         
            Specificity : 0.6667         
         Pos Pred Value : 0.4056         
         Neg Pred Value : 0.6273         
             Prevalence : 0.3841         
         Detection Rate : 0.1401         
   Detection Prevalence : 0.3454         
      Balanced Accuracy : 0.5157         
                                         
       'Positive' Class : 1              
```

Appendix-Figure 15: Confusion Matrix and Statistics of KNN with K = 4 and not scaled data.



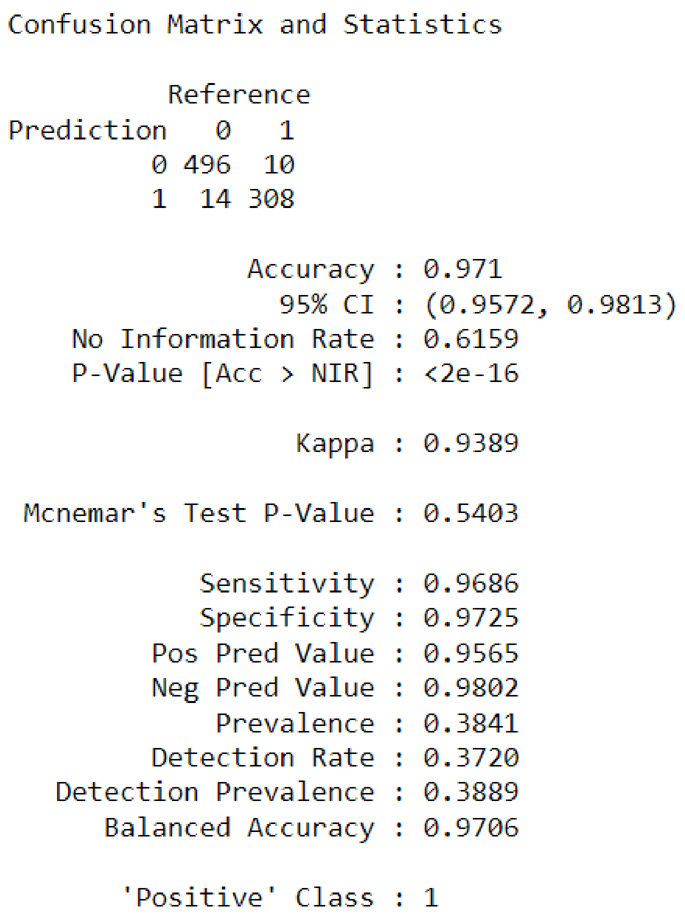
```
                         GVIF Df GVIF^(1/(2*Df))
state_abbreviation      225.6 42             1.1
POP010210          19339740.1  1          4397.7
AGE295214                 2.2  1             1.5
RHI525214                 1.3  1             1.2
RHI625214                 1.8  1             1.3
POP645213                 8.7  1             2.9
POP815213                 7.7  1             2.8
EDU635213                 5.8  1             2.4
EDU685213                 6.2  1             2.5
VET605213                27.9  1             5.3
HSG096213                 3.0  1             1.7
INC910213                10.5  1             3.2
INC110213                 9.7  1             3.1
PVY020213                 6.0  1             2.5
BZA010213               196.5  1            14.0
BZA110213                87.9  1             9.4
NES010213               128.1  1            11.3
SBO315207                 1.4  1             1.2
MAN450207                 3.4  1             1.8
AFN120207                 9.8  1             3.1
LND110210                 2.5  1             1.6
PST040210          19343992.5  1          4398.2
```

Appendix-Figure 16: Variance Inflation Factor (VIF) metrics of logistic regression.



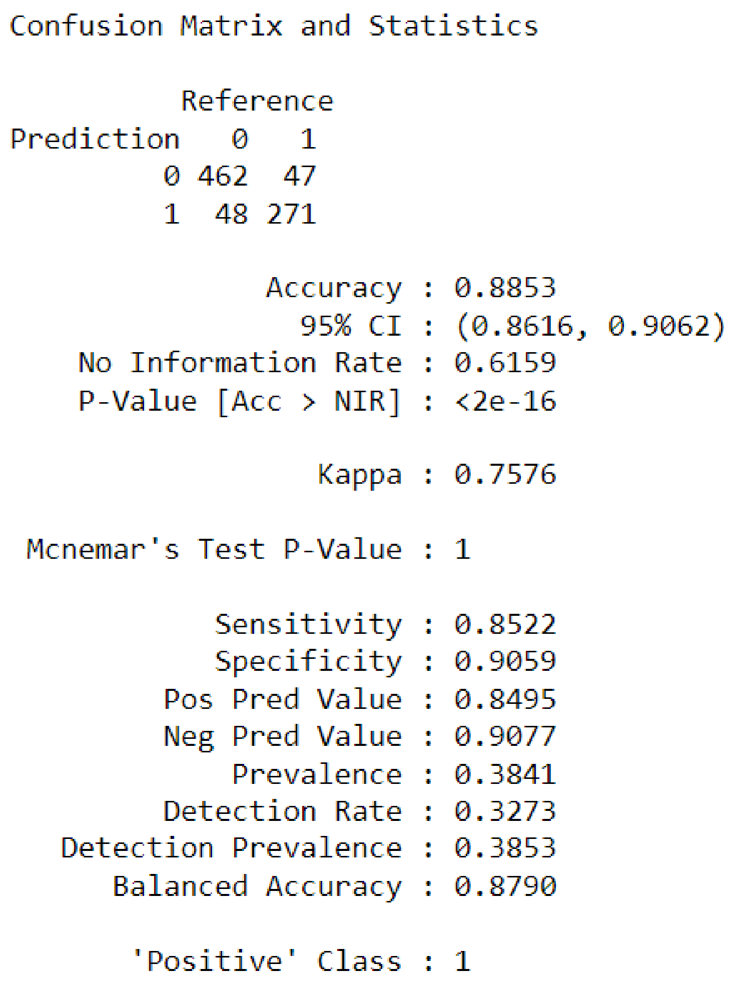
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 463  47
         1  47 271

               Accuracy : 0.8865
                 95% CI : (0.8629, 0.9073)
    No Information Rate : 0.6159
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.76

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8522
            Specificity : 0.9078
         Pos Pred Value : 0.8522
         Neg Pred Value : 0.9078
             Prevalence : 0.3841
         Detection Rate : 0.3273
   Detection Prevalence : 0.3841
      Balanced Accuracy : 0.8800

       'Positive' Class : 1
```

Appendix-Figure 17: Confusion Matrix and Statistics of SVM model.



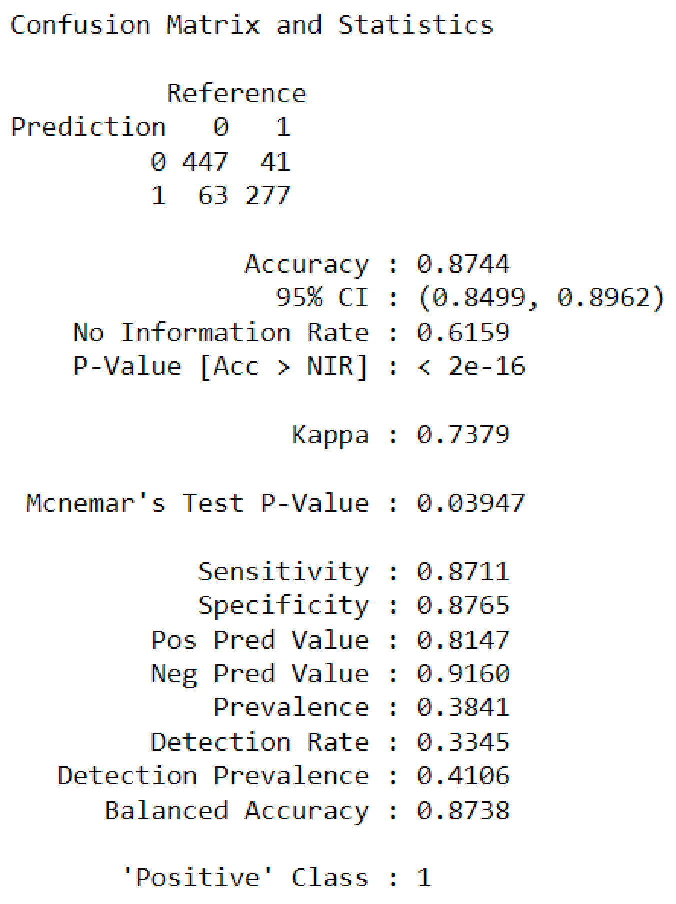
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 496  10
         1  14 308

               Accuracy : 0.971
                 95% CI : (0.9572, 0.9813)
    No Information Rate : 0.6159
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9389

 Mcnemar's Test P-Value : 0.5403

            Sensitivity : 0.9686
            Specificity : 0.9725
         Pos Pred Value : 0.9565
         Neg Pred Value : 0.9802
             Prevalence : 0.3841
         Detection Rate : 0.3720
   Detection Prevalence : 0.3889
      Balanced Accuracy : 0.9706

       'Positive' Class : 1
```

Appendix-Figure 18: Confusion Matrix and Statistics of Naïve Bayes model.



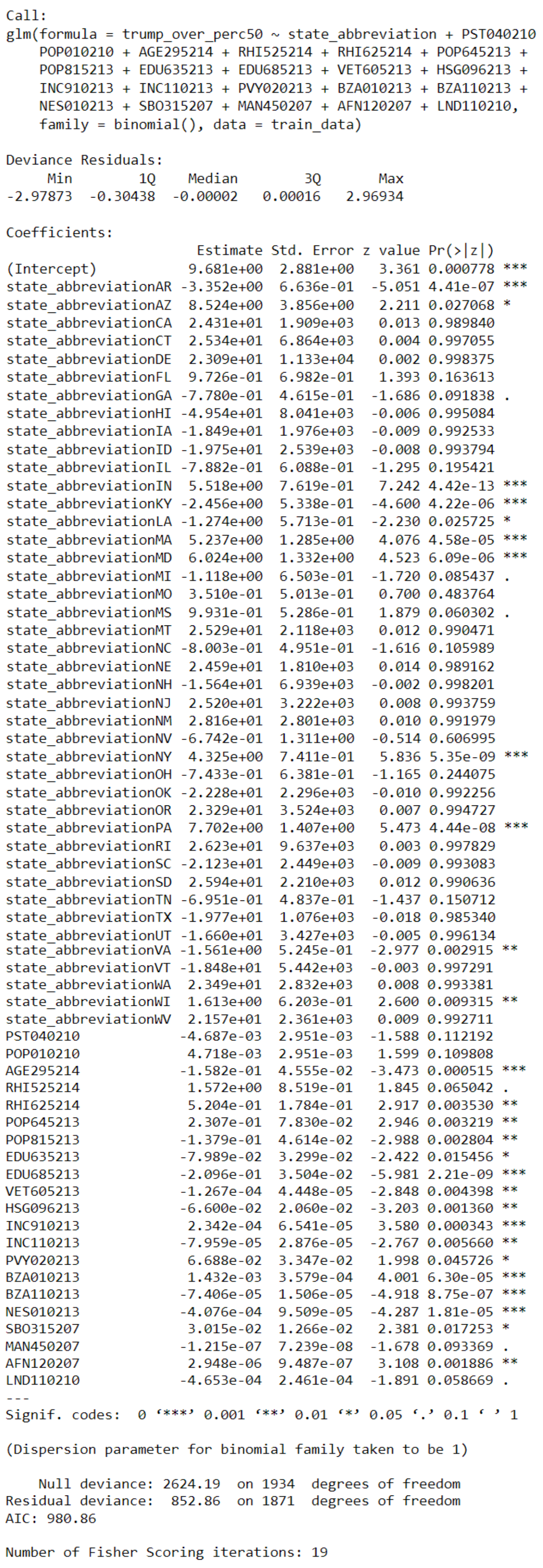
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 462  47
         1  48 271
                                          
               Accuracy : 0.8853          
                 95% CI : (0.8616, 0.9062)
    No Information Rate : 0.6159          
    P-Value [Acc > NIR] : <2e-16          
                                          
                  Kappa : 0.7576          
                                          
 Mcnemar's Test P-Value : 1               
                                          
            Sensitivity : 0.8522          
            Specificity : 0.9059          
         Pos Pred Value : 0.8495          
         Neg Pred Value : 0.9077          
             Prevalence : 0.3841          
         Detection Rate : 0.3273          
   Detection Prevalence : 0.3853          
      Balanced Accuracy : 0.8790          
                                          
       'Positive' Class : 1               
```

Appendix-Figure 19: Confusion Matrix and Statistics of Logistic Regression model.



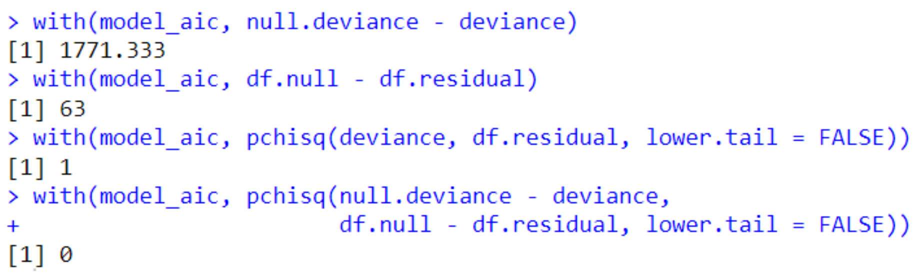
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 447  41
         1  63 277
                                          
               Accuracy : 0.8744          
                 95% CI : (0.8499, 0.8962)
    No Information Rate : 0.6159          
    P-Value [Acc > NIR] : < 2e-16         
                                          
                  Kappa : 0.7379          
                                          
 Mcnemar's Test P-Value : 0.03947         
                                          
            Sensitivity : 0.8711          
            Specificity : 0.8765          
         Pos Pred Value : 0.8147          
         Neg Pred Value : 0.9160          
             Prevalence : 0.3841          
         Detection Rate : 0.3345          
   Detection Prevalence : 0.4106          
      Balanced Accuracy : 0.8738          
                                          
       'Positive' Class : 1               
```

Appendix-Figure 20: Confusion Matrix and Statistics of Decision Tree model.



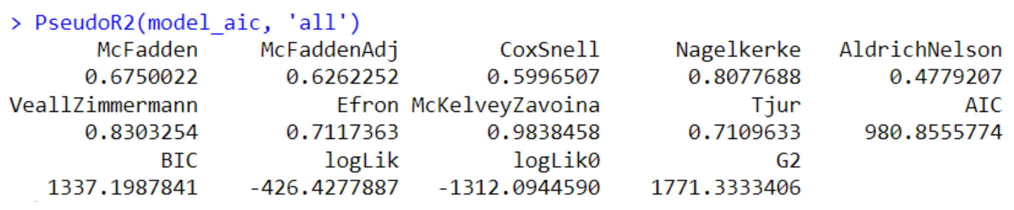
```
Call:
glm(formula = trump_over_perc50 ~ state_abbreviation + PST040210
    POP010210 + AGE295214 + RHI525214 + RHI625214 + POP645213 +
    POP815213 + EDU635213 + EDU685213 + VET605213 + HSG096213 +
    INC910213 + INC110213 + PVY020213 + BZA010213 + BZA110213 +
    NES010213 + SBO315207 + MAN450207 + AFN120207 + LND110210,
    family = binomial(), data = train_data)

Deviance Residuals:
     Min        1Q    Median       3Q      Max
-2.97873  -0.30438  -0.00002  0.00016  2.96934

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)           9.681e+00  2.881e+00   3.361 0.000778 ***
state_abbreviationAR -3.352e+00  6.636e-01  -5.051 4.41e-07 ***
state_abbreviationAZ  8.524e+00  3.856e+00   2.211 0.027068 *
state_abbreviationCA  2.431e+01  1.909e+03   0.013 0.989840
state_abbreviationCT  2.534e+01  6.864e+03   0.004 0.997055
state_abbreviationDE  2.309e+01  1.133e+04   0.002 0.998375
state_abbreviationFL  9.726e-01  6.982e-01   1.393 0.163613
state_abbreviationGA -7.780e-01  4.615e-01  -1.686 0.091838 .
state_abbreviationHI -4.954e+01  8.041e+03  -0.006 0.995084
state_abbreviationIA -1.849e+01  1.976e+03  -0.009 0.992533
state_abbreviationID -1.975e+01  2.539e+03  -0.008 0.993794
state_abbreviationIL -7.882e-01  6.088e-01  -1.295 0.195421
state_abbreviationIN  5.518e+00  7.619e-01   7.242 4.42e-13 ***
state_abbreviationKY -2.456e+00  5.338e-01  -4.600 4.22e-06 ***
state_abbreviationLA -1.274e+00  5.713e-01  -2.230 0.025725 *
state_abbreviationMA  5.237e+00  1.285e+00   4.076 4.58e-05 ***
state_abbreviationMD  6.024e+00  1.332e+00   4.523 6.09e-06 ***
state_abbreviationMI -1.118e+00  6.503e-01  -1.720 0.085437 .
state_abbreviationMO  3.510e-01  5.013e-01   0.700 0.483764
state_abbreviationMS  9.931e-01  5.286e-01   1.879 0.060302 .
state_abbreviationMT  2.529e+01  2.118e+03   0.012 0.990471
state_abbreviationNC -8.003e-01  4.951e-01  -1.616 0.105989
state_abbreviationNE  2.459e+01  1.810e+03   0.014 0.989162
state_abbreviationNH -1.564e+01  6.939e+03  -0.002 0.998201
state_abbreviationNJ  2.520e+01  3.222e+03   0.008 0.993759
state_abbreviationNM  2.816e+01  2.801e+03   0.010 0.991979
state_abbreviationNV -6.742e-01  1.311e+00  -0.514 0.606995
state_abbreviationNY  4.325e+00  7.411e-01   5.836 5.35e-09 ***
state_abbreviationOH -7.433e-01  6.381e-01  -1.165 0.244075
state_abbreviationOK -2.228e+01  2.296e+03  -0.010 0.992256
state_abbreviationOR  2.329e+01  3.524e+03   0.007 0.994727
state_abbreviationPA  7.702e+00  1.407e+00   5.473 4.44e-08 ***
state_abbreviationRI  2.623e+01  9.637e+03   0.003 0.997829
state_abbreviationSC -2.123e+01  2.449e+03  -0.009 0.993083
state_abbreviationSD  2.594e+01  2.210e+03   0.012 0.990636
state_abbreviationTN -6.951e-01  4.837e-01  -1.437 0.150712
state_abbreviationTX -1.977e+01  1.076e+03  -0.018 0.985340
state_abbreviationUT -1.660e+01  3.427e+03  -0.005 0.996134
state_abbreviationVA -1.561e+00  5.245e-01  -2.977 0.002915 **
state_abbreviationVT -1.848e+01  5.442e+03  -0.003 0.997291
state_abbreviationWA  2.349e+01  2.832e+03   0.008 0.993381
state_abbreviationWI  1.613e+00  6.203e-01   2.600 0.009315 **
state_abbreviationWV  2.157e+01  2.361e+03   0.009 0.992711
PST040210            -4.687e-03  2.951e-03  -1.588 0.112192
POP010210             4.718e-03  2.951e-03   1.599 0.109808
AGE295214            -1.582e-01  4.555e-02  -3.473 0.000515 ***
RHI525214             1.572e+00  8.519e-01   1.845 0.065042 .
RHI625214             5.204e-01  1.784e-01   2.917 0.003530 **
POP645213             2.307e-01  7.830e-02   2.946 0.003219 **
POP815213            -1.379e-01  4.614e-02  -2.988 0.002804 **
EDU635213            -7.989e-02  3.299e-02  -2.422 0.015456 *
EDU685213            -2.096e-01  3.504e-02  -5.981 2.21e-09 ***
VET605213            -1.267e-04  4.448e-05  -2.848 0.004398 **
HSG096213            -6.600e-02  2.060e-02  -3.203 0.001360 **
INC910213             2.342e-04  6.541e-05   3.580 0.000343 ***
INC110213            -7.959e-05  2.876e-05  -2.767 0.005660 **
PVY020213             6.688e-02  3.347e-02   1.998 0.045726 *
BZA010213             1.432e-03  3.579e-04   4.001 6.30e-05 ***
BZA110213            -7.406e-05  1.506e-05  -4.918 8.75e-07 ***
NES010213            -4.076e-04  9.509e-05  -4.287 1.81e-05 ***
SBO315207             3.015e-02  1.266e-02   2.381 0.017253 *
MAN450207            -1.215e-07  7.239e-08  -1.678 0.093369 .
AFN120207             2.948e-06  9.487e-07   3.108 0.001886 **
LND110210            -4.653e-04  2.461e-04  -1.891 0.058669 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2624.19  on 1934  degrees of freedom
Residual deviance:  852.86  on 1871  degrees of freedom
AIC: 980.86

Number of Fisher Scoring iterations: 19
```

Appendix-Figure 21: model_aic summary and formula.



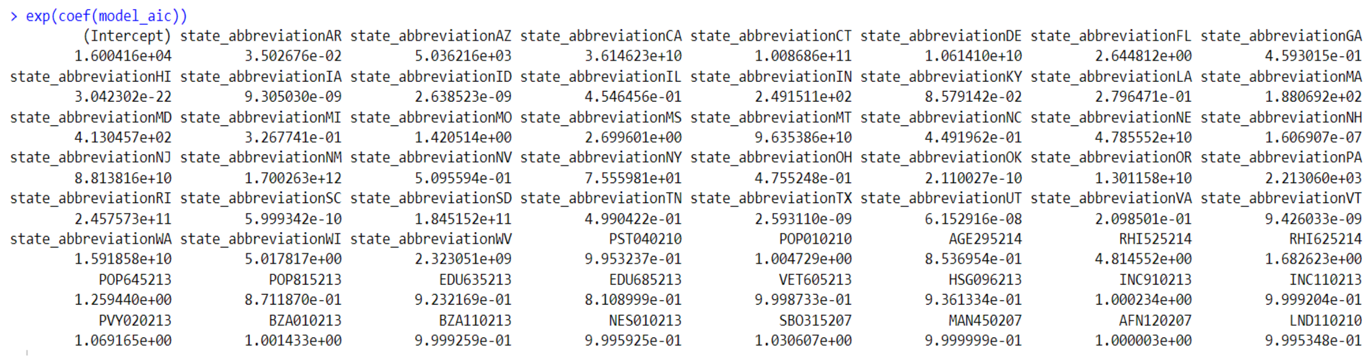
```
> with(model_aic, null.deviance - deviance)
[1] 1771.333
> with(model_aic, df.null - df.residual)
[1] 63
> with(model_aic, pchisq(deviance, df.residual, lower.tail = FALSE))
[1] 1
> with(model_aic, pchisq(null.deviance - deviance,
+                        df.null - df.residual, lower.tail = FALSE))
[1] 0
```

Appendix-Figure 22: Stepwise on the full model based on AIC criterion model (model_aic) LRT test's results.



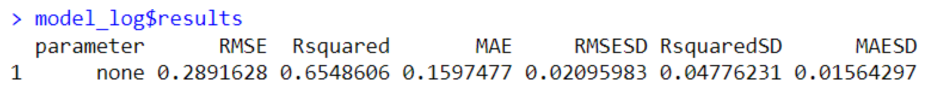
```
> PseudoR2(model_aic, 'all')
       McFadden     McFaddenAdj        CoxSnell      Nagelkerke   AldrichNelson
      0.6750022       0.6262252       0.5996507       0.8077688       0.4779207
VeallZimmermann           Efron McKelveyZavoina            Tjur             AIC
      0.8303254       0.7117363       0.9838458       0.7109633     980.8555774
            BIC          logLik         logLik0              G2
   1337.1987841    -426.4277887   -1312.0944590    1771.3333406
```

Appendix-Figure 23: Stepwise on the full model based on AIC criterion model (model_aic) pseudo R2.



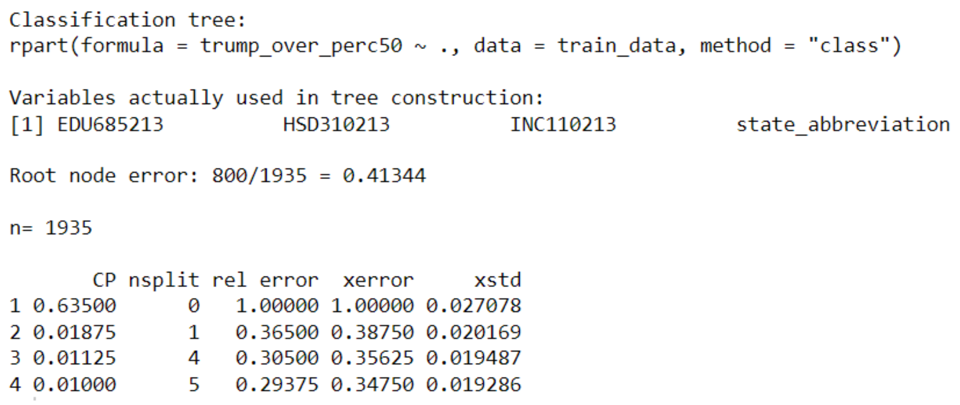
```
> exp(coef(model_aic))
        (Intercept) state_abbreviationAR state_abbreviationAZ state_abbreviationCA state_abbreviationCT state_abbreviationDE state_abbreviationFL state_abbreviationGA
       1.600416e+04         3.502676e-02         5.036216e+03         3.614623e+10         1.008686e+11         1.061410e+10         2.644812e+00         4.593015e-01
state_abbreviationHI state_abbreviationIA state_abbreviationID state_abbreviationIL state_abbreviationIN state_abbreviationKY state_abbreviationLA state_abbreviationMA
       3.042302e-22         9.305030e-09         2.638523e-09         4.546456e-01         2.491511e+02         8.579142e-02         2.796471e-01         1.880692e+02
state_abbreviationMD state_abbreviationMI state_abbreviationMO state_abbreviationMS state_abbreviationMT state_abbreviationNC state_abbreviationNE state_abbreviationNH
       4.130457e+02         3.267741e-01         1.420514e+00         2.699601e+00         9.635386e+10         4.491962e-01         4.785552e+10         1.606907e-07
state_abbreviationNJ state_abbreviationNM state_abbreviationNV state_abbreviationNY state_abbreviationOH state_abbreviationOK state_abbreviationOR state_abbreviationPA
       8.813816e+10         1.700263e+12         5.095594e-01         7.555981e+01         4.755248e-01         2.110027e-10         1.301158e+10         2.213060e+03
state_abbreviationRI state_abbreviationSC state_abbreviationSD state_abbreviationTN state_abbreviationTX state_abbreviationUT state_abbreviationVA state_abbreviationVT
       2.457573e+11         5.999342e-10         1.845152e+11         4.990422e-01         2.593110e-09         6.152916e-08         2.098501e-01         9.426033e-09
state_abbreviationWA state_abbreviationWI state_abbreviationWV             PST040210             POP010210             AGE295214             RHI525214             RHI625214
       1.591858e+10         5.017817e+00         2.323051e+09         9.953237e-01         1.004729e+00         8.536954e-01         4.814552e+00         1.682623e+00
          POP645213            POP815213            EDU635213            EDU685213            VET605213            HSG096213            INC910213            INC110213
       1.259440e+00         8.711870e-01         9.232169e-01         8.108999e-01         9.998733e-01         9.361334e-01         1.000234e+00         9.999204e-01
          PVY020213            BZA010213            BZA110213            NES010213            SBO315207            MAN450207            AFN120207            LND110210
       1.069165e+00         1.001433e+00         9.999259e-01         9.995925e-01         1.030607e+00         9.999999e-01         1.000003e+00         9.995348e-01
```

Appendix-Figure 24: Exponentiation of the coefficients of the stepwise on the full model based on AIC criterion model (model_aic).



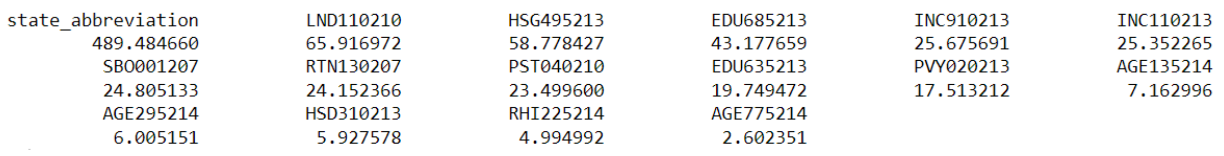
```
> model_log$results
  parameter      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
1      none 0.2891628 0.6548606 0.1597477 0.02095983 0.04776231 0.01564297
```

Appendix-Figure 25: Evaluation metrics of the trained logistic regression model (model_log).



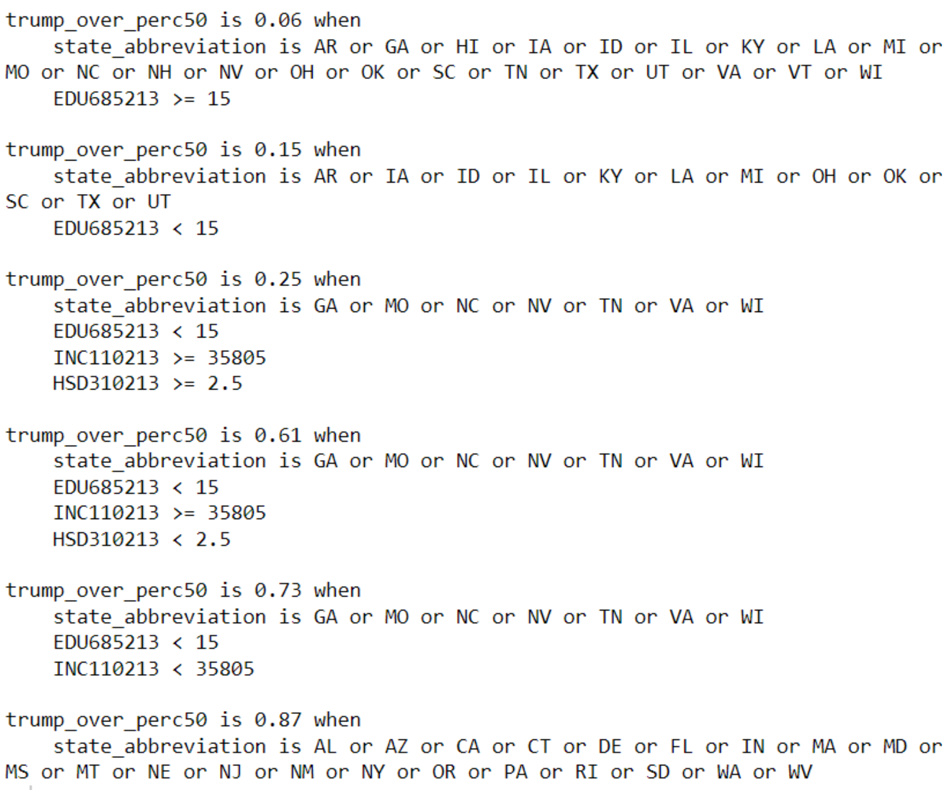
```
Classification tree:
rpart(formula = trump_over_perc50 ~ ., data = train_data, method = "class")

Variables actually used in tree construction:
[1] EDU685213          HSD310213          INC110213          state_abbreviation

Root node error: 800/1935 = 0.41344

n= 1935

       CP nsplit rel error  xerror     xstd
1 0.63500      0   1.00000 1.00000 0.027078
2 0.01875      1   0.36500 0.38750 0.020169
3 0.01125      4   0.30500 0.35625 0.019487
4 0.01000      5   0.29375 0.34750 0.019286
```

Appendix-Figure 26: Complexity parameter table of the Decision Tree model (dt_model).



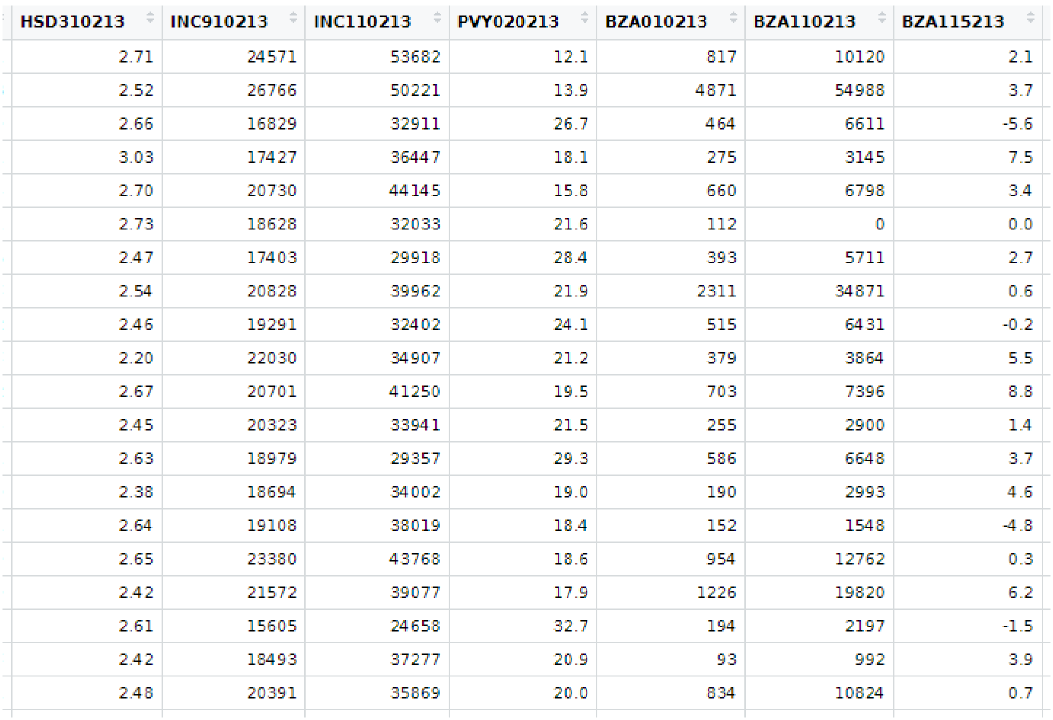
```
state_abbreviation          LND110210          HSG495213          EDU685213          INC910213          INC110213
        489.484660          65.916972          58.778427          43.177659          25.675691          25.352265
         SBO001207          RTN130207          PST040210          EDU635213          PVY020213          AGE135214
         24.805133          24.152366          23.499600          19.749472          17.513212           7.162996
         AGE295214          HSD310213          RHI225214          AGE775214
          6.005151           5.927578           4.994992           2.602351
```

Appendix-Figure 27: Decision Tree model (dt_model) variables' importance.



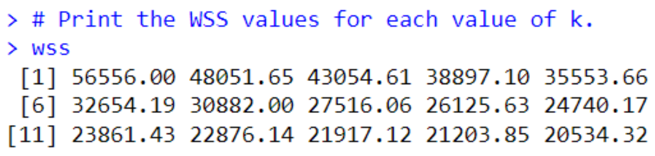
```
trump_over_perc50 is 0.06 when
    state_abbreviation is AR or GA or HI or IA or ID or IL or KY or LA or MI or
MO or NC or NH or NV or OH or OK or SC or TN or TX or UT or VA or VT or WI
    EDU685213 >= 15

trump_over_perc50 is 0.15 when
    state_abbreviation is AR or IA or ID or IL or KY or LA or MI or OH or OK or
SC or TX or UT
    EDU685213 < 15

trump_over_perc50 is 0.25 when
    state_abbreviation is GA or MO or NC or NV or TN or VA or WI
    EDU685213 < 15
    INC110213 >= 35805
    HSD310213 >= 2.5

trump_over_perc50 is 0.61 when
    state_abbreviation is GA or MO or NC or NV or TN or VA or WI
    EDU685213 < 15
    INC110213 >= 35805
    HSD310213 < 2.5

trump_over_perc50 is 0.73 when
    state_abbreviation is GA or MO or NC or NV or TN or VA or WI
    EDU685213 < 15
    INC110213 < 35805

trump_over_perc50 is 0.87 when
    state_abbreviation is AL or AZ or CA or CT or DE or FL or IN or MA or MD or
MS or MT or NE or NJ or NM or NY or OR or PA or RI or SD or WA or WV
```

Appendix-Figure 28: Decision Tree model (dt_model) rules (text format).



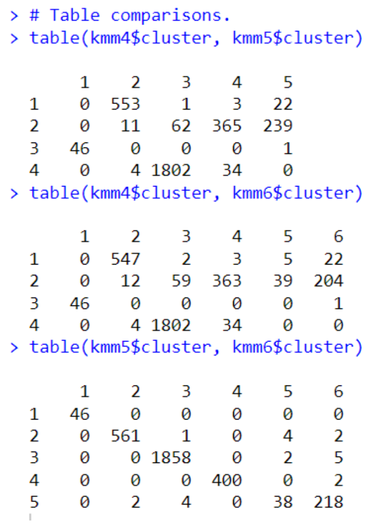
HSD310213	INC910213	INC110213	PVY020213	BZA010213	BZA110213	BZA115213
2.71	24571	53682	12.1	817	10120	2.1
2.52	26766	50221	13.9	4871	54988	3.7
2.66	16829	32911	26.7	464	6611	-5.6
3.03	17427	36447	18.1	275	3145	7.5
2.70	20730	44145	15.8	660	6798	3.4
2.73	18628	32033	21.6	112	0	0.0
2.47	17403	29918	28.4	393	5711	2.7
2.54	20828	39962	21.9	2311	34871	0.6
2.46	19291	32402	24.1	515	6431	-0.2
2.20	22030	34907	21.2	379	3864	5.5
2.67	20701	41250	19.5	703	7396	8.8
2.45	20323	33941	21.5	255	2900	1.4
2.63	18979	29357	29.3	586	6648	3.7
2.38	18694	34002	19.0	190	2993	4.6
2.64	19108	38019	18.4	152	1548	-4.8
2.65	23380	43768	18.6	954	12762	0.3
2.42	21572	39077	17.9	1226	19820	6.2
2.61	15605	24658	32.7	194	2197	-1.5
2.42	18493	37277	20.9	93	992	3.9
2.48	20391	35869	20.0	834	10824	0.7

Appendix-Figure 29: Different scales & different units variables.



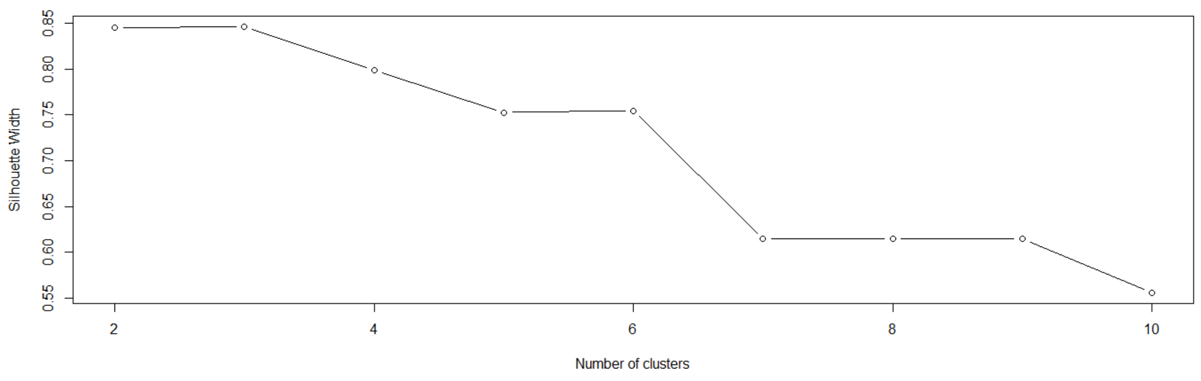
```
> # Print the WSS values for each value of k.
> wss
 [1] 56556.00 48051.65 43054.61 38897.10 35553.66
 [6] 32654.19 30882.00 27516.06 26125.63 24740.17
[11] 23861.43 22876.14 21917.12 21203.85 20534.32
```

Appendix-Figure 30: WSS Table.

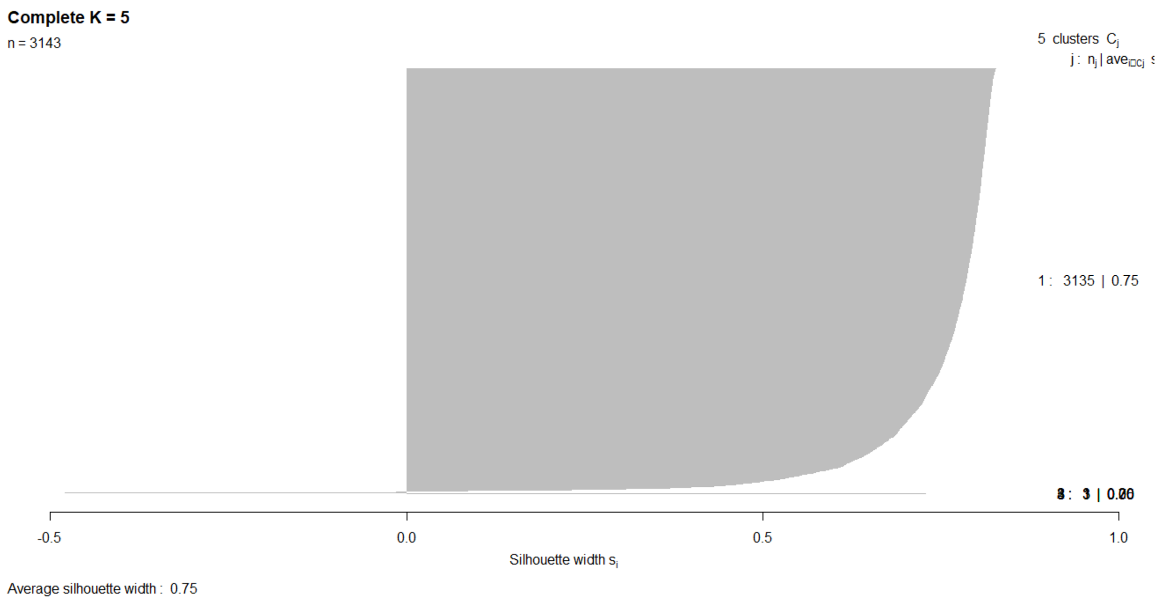


```
> # Table comparisons.
> table(kmm4$cluster, kmm5$cluster)
   
       1    2    3    4    5
  1    0  553    1    3   22
  2    0   11   62  365  239
  3   46    0    0    0    1
  4    0    4 1802   34    0
> table(kmm4$cluster, kmm6$cluster)
   
       1    2    3    4    5    6
  1    0  547    2    3    5   22
  2    0   12   59  363   39  204
  3   46    0    0    0    0    1
  4    0    4 1802   34    0    0
> table(kmm5$cluster, kmm6$cluster)
   
       1    2    3    4    5    6
  1   46    0    0    0    0    0
  2    0  561    1    0    4    2
  3    0    0 1858    0    2    5
  4    0    0    0  400    0    2
  5    0    2    4    0   38  218
```

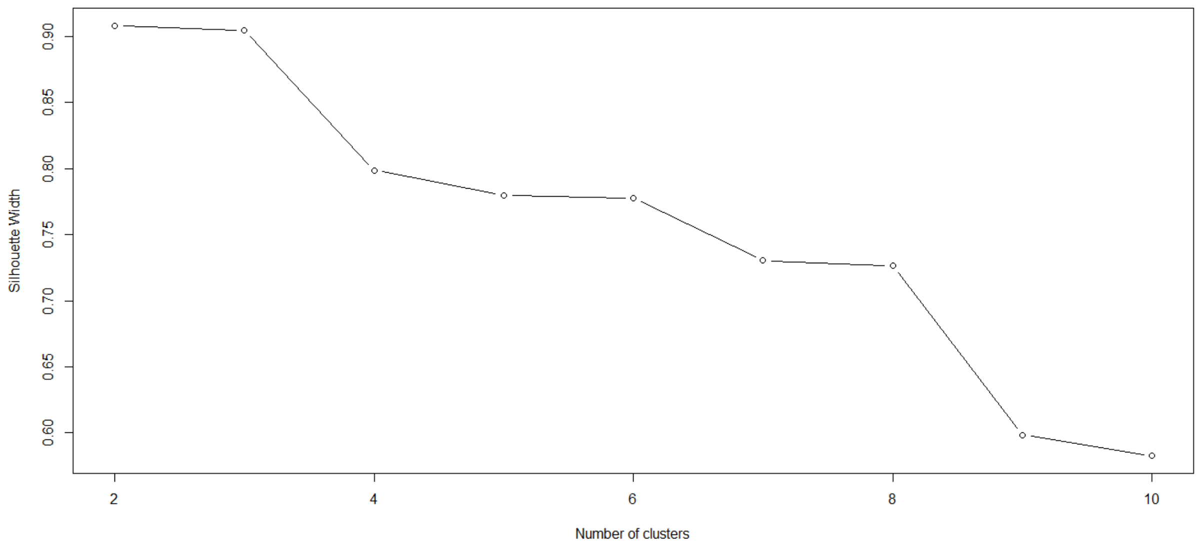
Appendix-Figure 31: Table Comparisons.



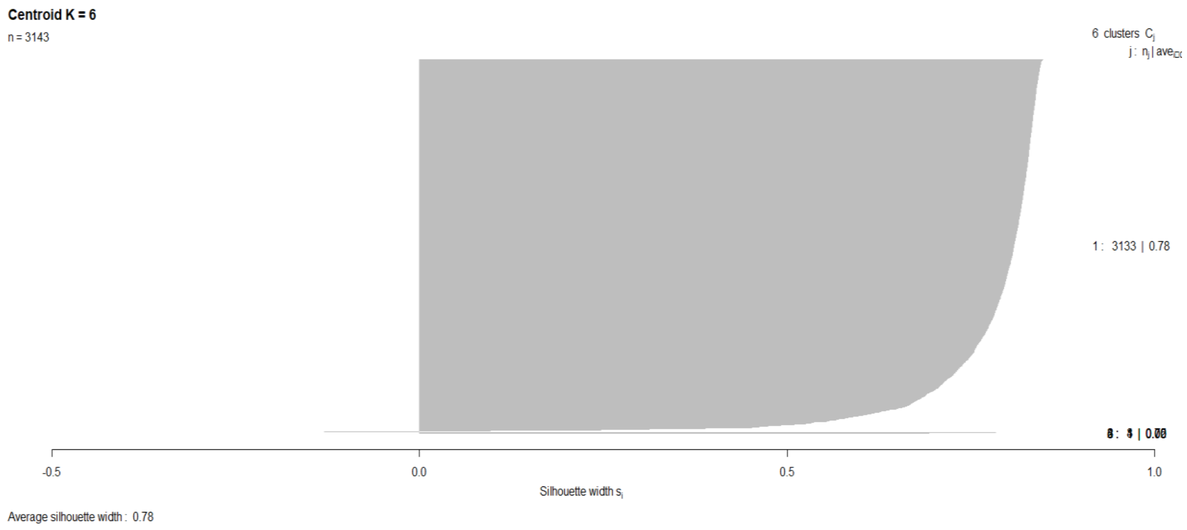
Appendix-Figure 32: Silhouette Coefficient of Complete Method with Euclidean Distance for various clusters.



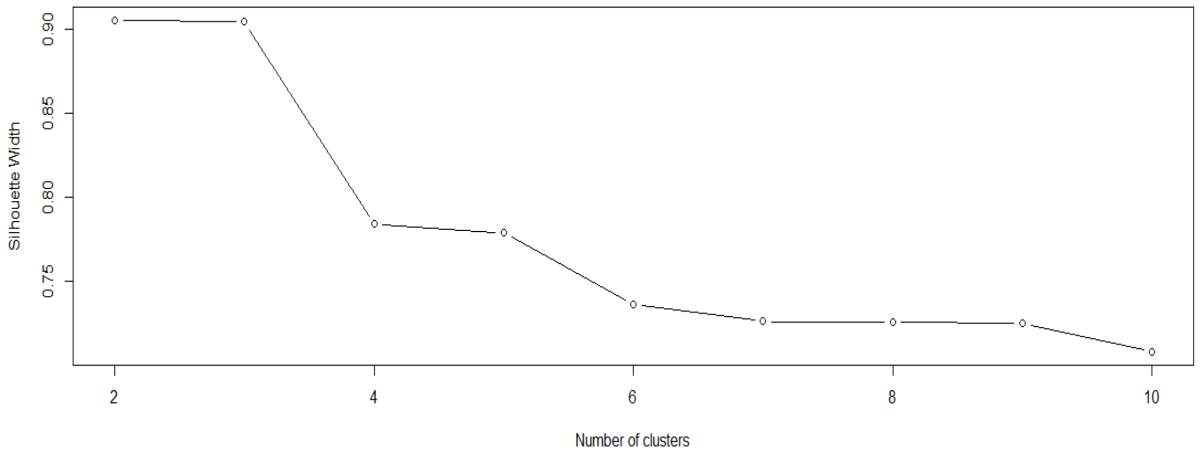
Appendix-Figure 33: Silhouette Plot of Complete Method with Euclidean Distance.



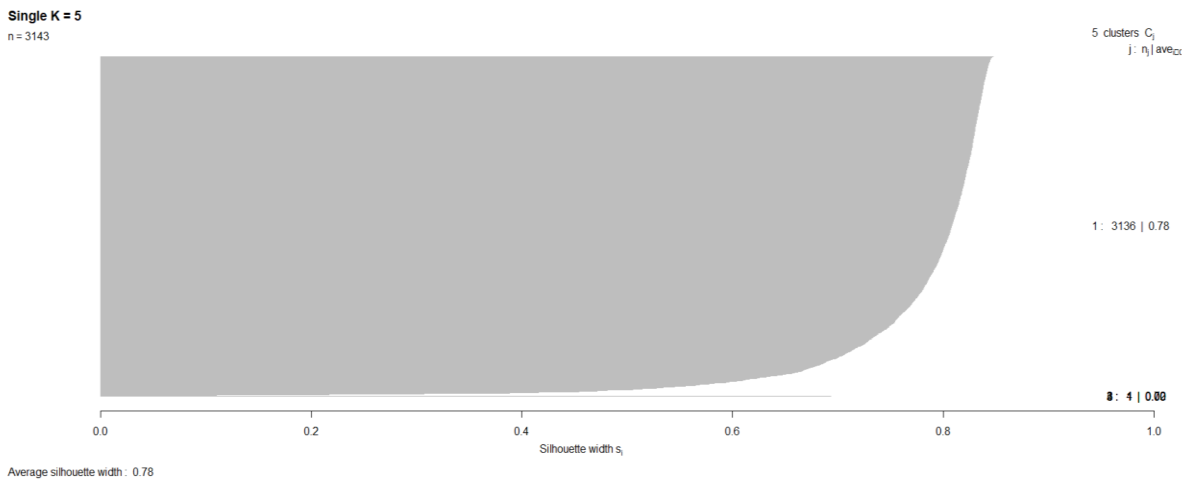
Appendix-Figure 34: Silhouette Coefficient of Centroid Method with Euclidean Distance for various clusters.



Appendix-Figure 35: Silhouette Plot of Centroid Method with Euclidean Distance.



Appendix-Figure 36: Silhouette Coefficient of Simple Method with Euclidean Distance for various clusters.



Appendix-Figure 37: Silhouette Plot of Single Method with Euclidean Distance.

```
> table(cutree(hcl_ward_man, k = 6), kmeans_6$cluster)

       1    2    3    4    5    6
  1    0   65 1569   27    0    1
  2    0  369   68   31    0    4
  3    0    0   38  310    0    0
  4    2   27  135   31    0  214
  5   44   93    0    1    0    8
  6    0    9   53    0   44    0
```

Appendix-Figure 38: K-Means and Hierarchical Clustering methods observations assignment to clusters table.