Data description and research question

The purpose of this report is to describe, analyze and implement a comprehensive data analysis as well as to try and find patterns and trends on the chosen datasets and research question. The research questions if the demographics, social class, and political biases influenced the result of the Brexit referendum. The datasets that will be used for this analysis is the census, the referendum results that took place in 2016 when the United Kingdom voted in favor of leaving the European Union, the Income and Tax per Constituency as well as the general elections results of 2015 in the UK. The first two datasets were acquired from Kaggle which is an online community/platform for data scientists. The dataset with the results of the general elections was acquired from the electoral commission website and finally the income and tax per parliamentary constituency was acquired from the UK's government website.

Data Preparation and cleaning

The main dataset has been formed by 4 subsets, Income and Tax, Population, referendum results, general election results. The inner join command was used to join the subsets together by constituency.

Census Dataset

From the census dataset have been kept only the area and the population variables for further analysis.

Referendum Dataset

From the referendum spreadsheet has been excluded the variables ID, Region Code, Area Code, Region, No.official.mark, multiple, marks, writing marks, unmarked or void as they were irrelevant with the subject and the question that this report will analyze. Furthermore, the numerical variables of leave and remain have been excluded since for better analysis purposes it has been decided that the variables with the percentages are more useful. Also, the variable percentage of remain will be removed as the subject question is what drove people to vote leave. Finally, the rejected ballots variable has been removed since the variable with the percentage has been kept instead.

Result of 2015 general elections Dataset

From this dataset, the variables: press, association id number, region, country, constituency ID, constituency type, election year, electorate and total number of valid notes counted have been excluded as irrelevant with the subject question as well as all the small political parties in order to avoid the "noise" in the dataset. From the political parties those kept are only the five main ones which are: Labour, Conservatives, Green, UKIP and Liberal Democrats. There are some occasions where constituencies appear to have N/A, meaning people did not vote for this party in the particular constituency. In these cases, the N/As have been replaced with the number 1 in order not to remove the whole constituency from the dataset and at the same time to manipulate the dataset as least as possible.

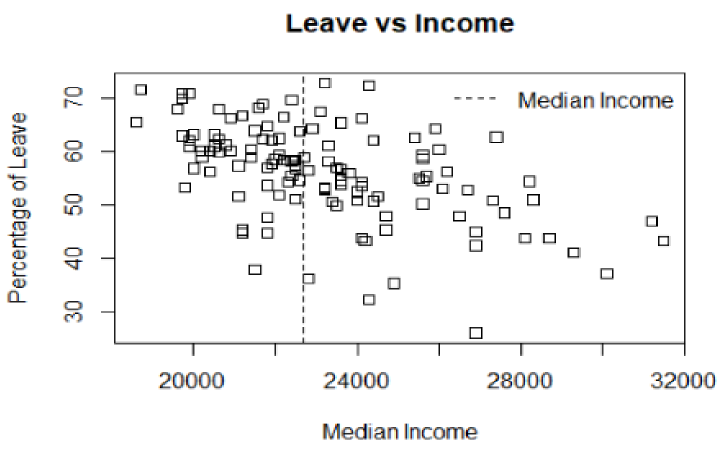
Income and Tax per Constituency Dataset

From this spreadsheet only the variables median income and tax and mean income and tax have been selected. The dataset also includes a breakdown of the median and mean income and tax, for self-employed and pensioners. The total income and tax have been chosen both for the mean and the median in order to avoid the noise. All the NAs have been removed from the dataset. The structure of the data has been formed: as factor the variable Area and all the other variables as numeric. The library dplyr was used to inner join all the datasets.

Exploratory data analysis

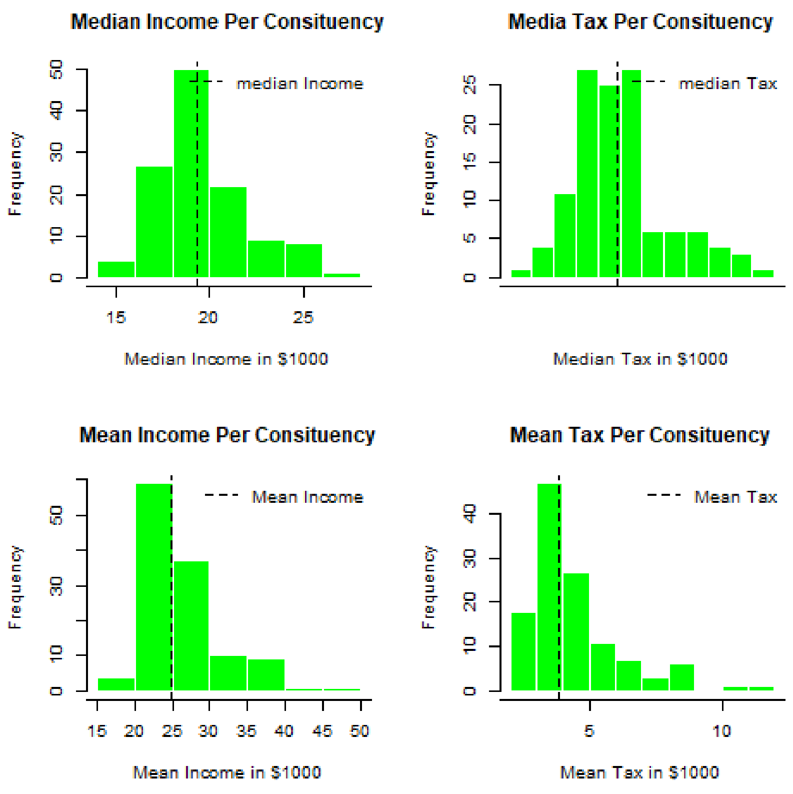
Graph 1

The graph 1 gives a first impression and an early result on how the social class played a role in forming the referendum result. It is visible that the percentage of leave is much higher at the lower income constituencies as the values have a downward trajectory.



Graph 2

The graph 2 shows the distribution of the variable's median income, median tax, mean income and mean tax. Exploratory histograms have been used to give a first impression of how the data look like. It is observed normal distribution in all the histograms. The dashed line presents the median and mean values of the data accordingly. The y axis presents the frequency and the x axis presents the value per £1000.



Principal Component Analysis (PCA)

Leave dataset before dimensionality reduction

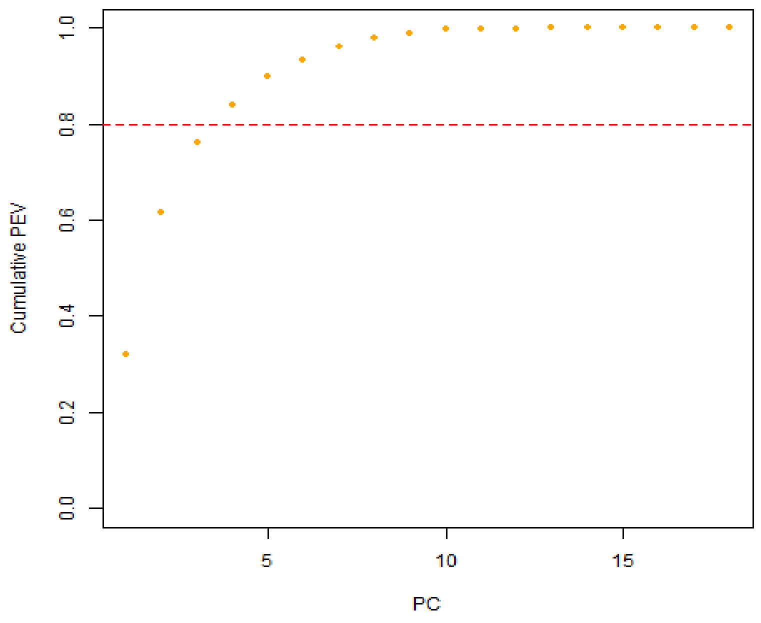
The PCA provides the correlation among variables into a 2D or 3D graph and in this case, it has been used for dimensionality reduction. The PCA is an unsupervised technique which helps to identify if subjects can separate into distinct groups. It is important that in order to proceed with PC Analysis the dataset must include only continuous/numerical variables. Additionally, the variables must be centered and scaled before the analysis because in the dataset there are different measurements, as for example the percentage of leave is from 0 to 100 and the median income is from £18000 to £45000. Therefore, if the dataset is not scaled and centered then the system will find the differences among these numbers and will form a result based on these differences and subsequently the result will be inaccurate/biased. Most of the times, the first PC has the biggest impact on the dataset, the second PC the second biggest impact and so on and so forth. This is also visible from the summary of the table 1 where the first PC covers 32% of the variance and the second PC covers the 30% of it. The cumulative variance below shows how the PCs add up to reach the 100%. It is observed that up to PC5 it has reached almost 90% as the rest of the PCAs do not contribute a lot. The standard deviation is the measure of the variability across the principal components and in this case PC1 and PC2 have the biggest variability.

Table 1

Importance of components	PC1	PC2	PC3	PC4	PC5
Standard Deviation	2.3983	2.3135	1.6124	1.19495	1.02987
Proportion of Variance	0.3196	0.2973	0.1444	0.07933	0.05982
Cumulative Proportion	0.3196	0.6169	0.84065	0.84065	0.89958
PEV	3.195539e-01	2.973393e-01	1.444322e-01	7.932793e-02	5.892361e-02

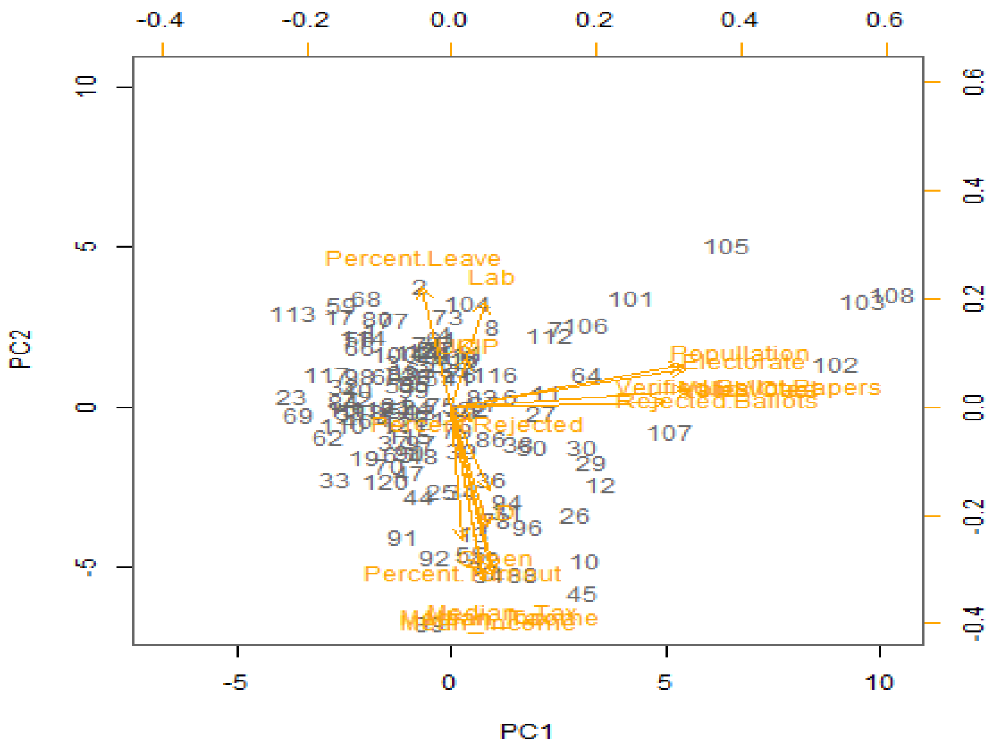
Graph 3

In the graph 3 a dashed line has been set at 80% of Proportion of Variance Explained (PVE) in order to be easily understood which PCs contribute more significantly. The first three PCs provide the more information. As a result, the dimensionality reduction will be based on these three PCs. The report will not take into consideration the PCs that are above the red dashed line.



The graph 4 presents the most important PCs in the form of biplots. A biplot is a graph used in exploratory data analysis to give indications on how variables react between one another. The main biplot is the combination of PC1 and PC2 which according to the loadings and variance plot explain most of the data. It is important to interpret the bitplot in accordance with the subject question, in this case, which variables affect the percentage of leave. The main biplot shows that in PC2 when the percentage of leave increases, the percentage of median income, median tax, Percent Turnout and Mean Income decrease. Therefore, that shows that people with lower income have voted for leave. PC1 gives the same results but with less significance since the variables are close to zero. The percent rejected is located exactly at the point zero both in PC1 and PC2, meaning that this variable can be removed from the dataset since it does not contribute at all. Furthermore, in the PC1 there is an increase in Population, Electorate, Rejected Ballots, Verified Ballots but the impact that all these variables have on percentage of leave, which is the subject question, is limited. A decision has been made for the variables Population, Electorate, Verified Ballot Papers, Vote Cast, Valid Noted, Rejected Ballots and Percentage of Rejected to be removed. This decision is crucial as after this point these variables will not be able to be recovered. The boxplot of the merged dataset indicates that there are outliers in the dataset especially in the mean income, although a decision has been made not to remove the outliers since it may have a significant impact on the actual result.

Graph 4 – Biplot Before Dimensionality Reduction



PCA after dimensionality reduction

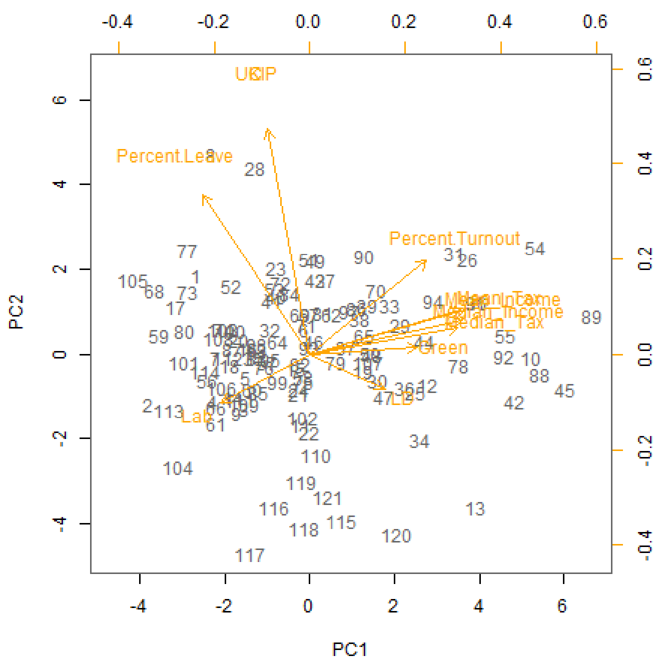
After the removal of the irrelevant variables the first PC explains most of the data. Similarly, the Variance plot indicates the same, the cumulative value of PEV indicates the first two PCs cover the 72% of it.

The new biplot in the graph 5 clearly shows the association between the percentage of leave and the income and tax as well as some association of political biases. For example, it shows that Green and Liberal Democrats voters tend to vote against leave in the referendum and on the other side UKIP and Conservatives(C) voters tend to vote at favor of leave. Furthermore, in the PC2 it seems that Labour voters have voted against leave as there is an opposing increase for these two variables. After the dimensionality reduction the dataset begin to give some initial messages on how and why people voted in favor of leave.

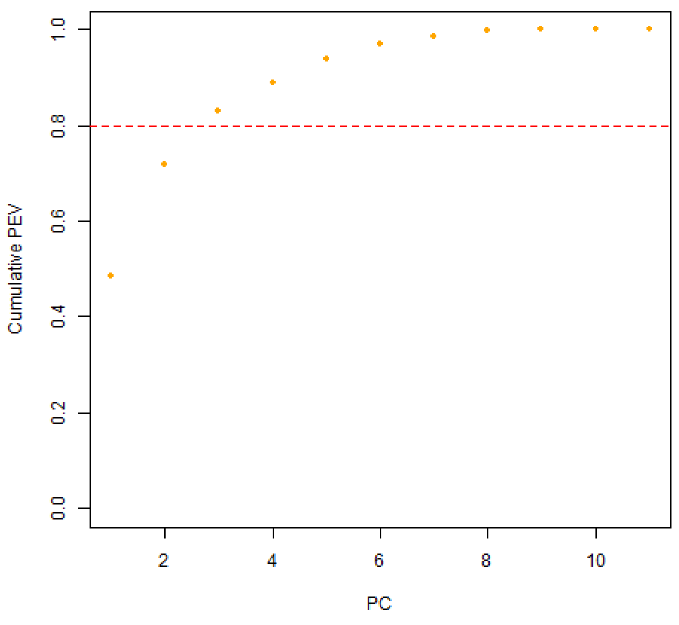
Table 2

Importance of components	PC1	PC2	PC3
Standard Deviation	2.3078	1.6040	1.1037
Proportion of Variance	0.4842	0.2339	0.1107
Cumulative Proportion	0.4842	0.7181	0.8288
PEV	4.84867e-01	2.33893e-01	1.107321e-01
Variance	5.32605	2.57282	1.21805

Graph 5



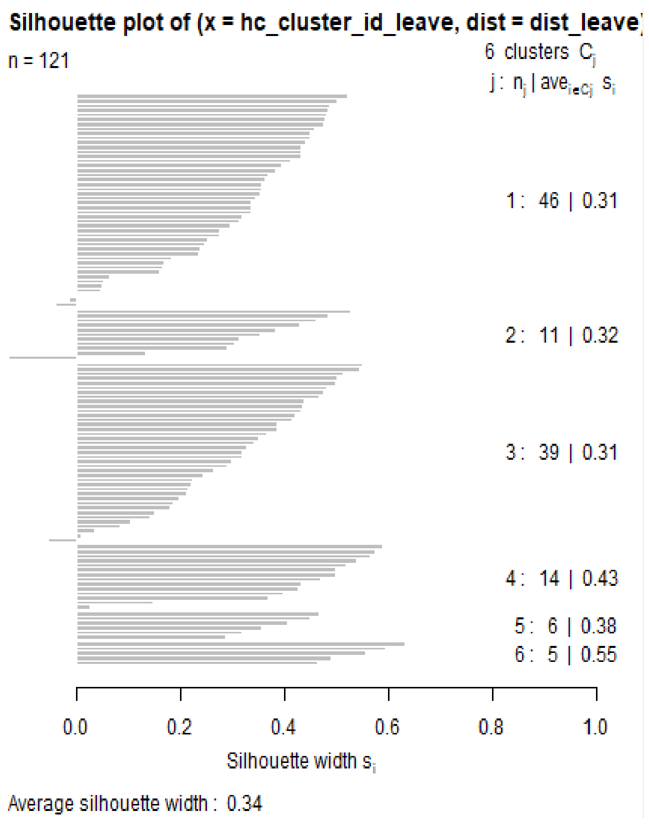
Graph 6



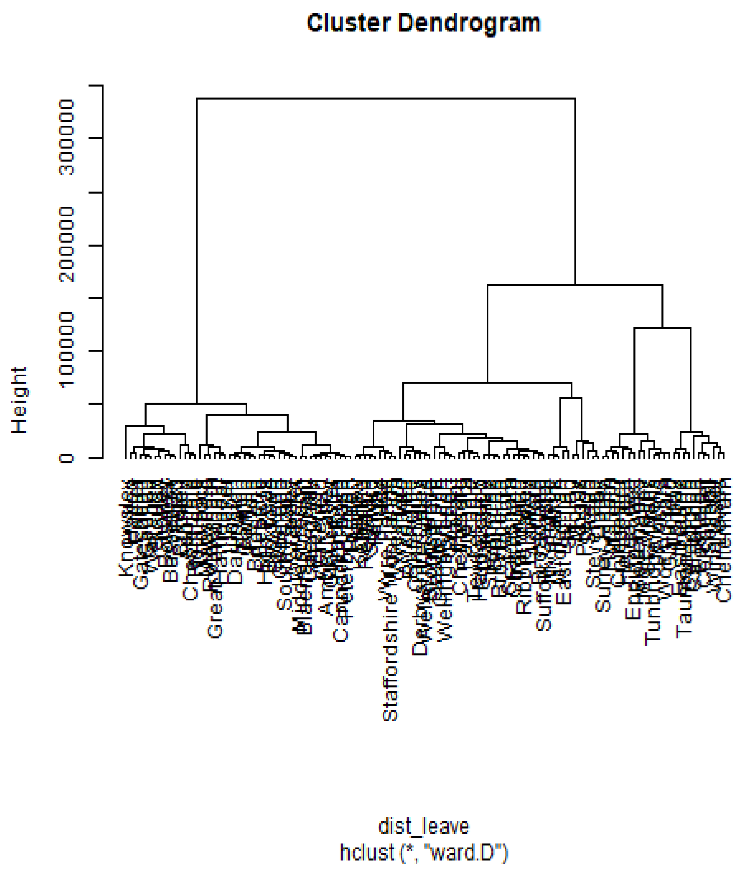
Cluster Analysis

Clustering is an unsupervised technique which groups objects that are similar to each other and separate them against the objects where there is a dissimilarity. The analysis has been based on the hierarchical technique which puts the objects into a hierarchical order. There are two ways to interpret the dendrogram in graph 8, the agglomerative and the divisive method. The agglomerative method begins to explore the data from the bottom to the top, in contrast the divisive method explores the data from the top to the bottom. In the agglomerative method every constituency begins to be treated as single cluster and as the clusters bind together, bigger clusters are formed to reach one big cluster at the end. The exact opposite procedure happens in the divisive method where it begins from a big cluster and ends up in many individual clusters. It is visible at the bottom of the dendrogram that when two constituencies are at the same small cluster, it means that these two constituencies are significantly similar. The agglomerative method has been used for analysis purposes. More specifically the two methods that have been used in the silhouette plot are the Ward's minimum variance method and the complete method. Ward's is a geometric method that indicates the distance between two clusters, A and B, is how much the sum of squares will increase when they are merged. On the other hand, complete-link clustering is graph method when the distance between clusters is the maximum distance between their members(Szekely and Rizzo, 2005)._The silhouette plot below shows how the two methods separate the objects/constituencies into clusters. The lines that are going to the left, minus direction, are the outliers while the average silhouette width is how much the separation between the clusters is, the higher the number the higher the separation. The first silhouette plot with the ward method seems to separate the cluster more efficiently and creates less outliers.

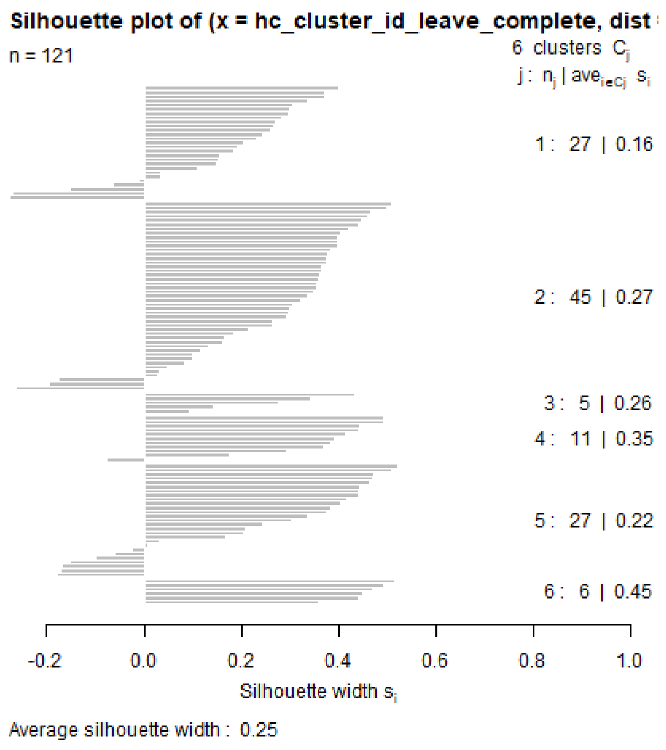
Graph 7



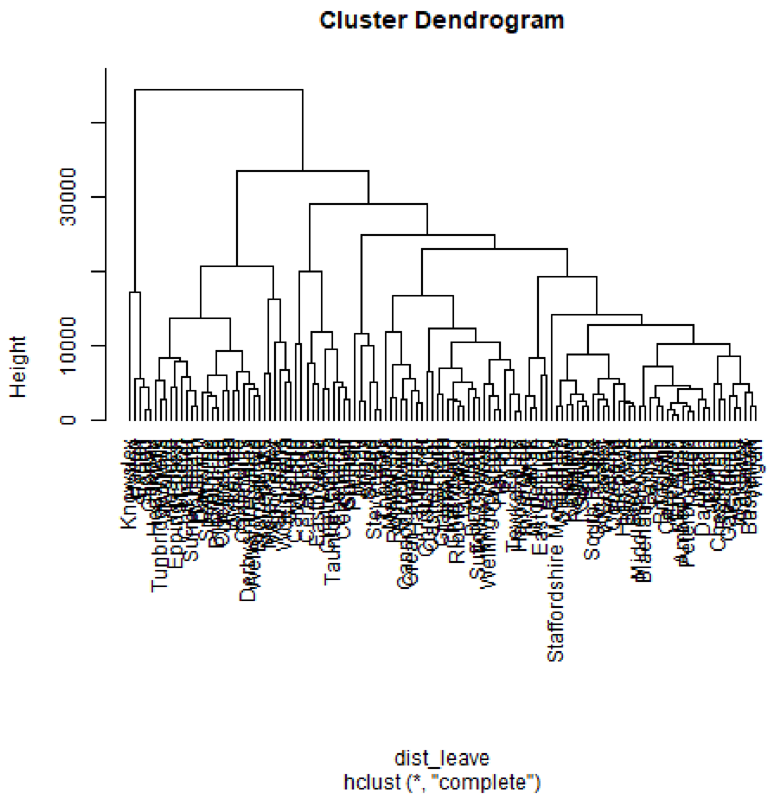
Graph 8



Graph 9



Graph 10



Machine learning prediction

Decision Tree

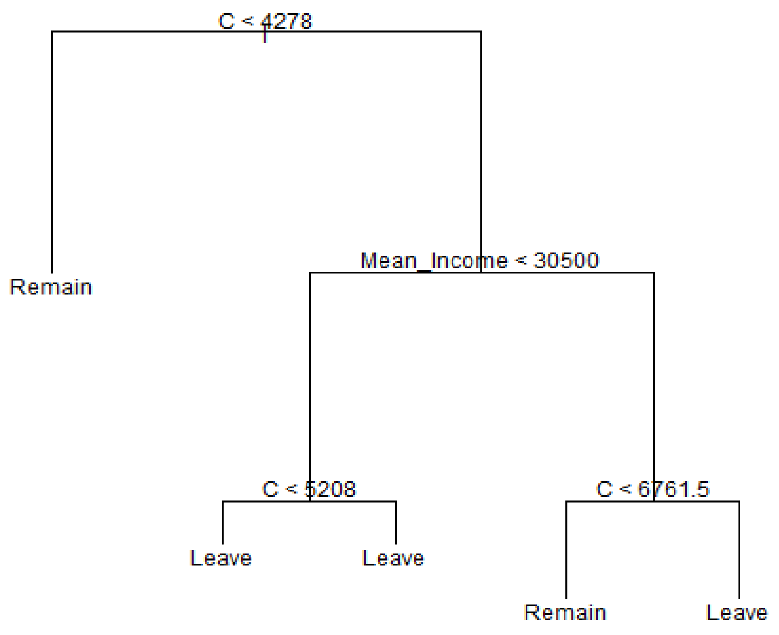
The decision tree is a tree based predictive method which explains the correlation between variables from top to bottom. The decision tree has been applied in this report to predict through classification the variables that affected the percentage of leave. In order to proceed with classification, it was essential that the percentage of leave was transformed in a new column as an outcome having two classes, leave and remain. The split was made for the constituencies with less than 50% of percentage of leave were marked as remain and similarly the constituencies with more than 50% percentage of leave were marked as leave. The result of the below decision tree shows that the most important variable in the prediction is the conservative party(C) which helps to classify the observations. The first variable in the decision tree is called the root node.

Furthermore, if the Conservative vote's numbers are less than 4278 then remain is decided showing political bias. If it is more than 4278 votes, then it goes to the right where it has the next most important variable in the prediction which is the mean income. This node is called the decision node. Similarly, if the mean income is less than 30500 automatically it goes to the left side where the only option is leave showing social bias. Finally, if it goes to the right side and it is less than 6761 votes it goes to remain and in case it is greater than 6761votes it goes to the leave side. The last nodes are called the terminal nodes. The summary shows the two variables that play significant role in the formation of the decision tree and these two variables are the conservative political party and the mean income. The nodes are the final predictions that the model can make according to the paths that the decision tree has created. The misclassification error is low, and this also can be confirmed from the fact that the model predicted wrong only 2 out of 84 classes. If the classification error was high, then we should prune the tree to minimize the size of the decision trees. Pruning was tried for the below tree, but the outcome was the same since we had a very accurate result since the beginning. The decision tree approach has many advantages on this dataset. First of all, it is intuitive and easily interpretable. Secondly, since the dataset is not high-dimensional, where the decision tree seems to have difficulty performing, can provide accurate results. The table 3 shows the two variables that affected the most, the percentage of leave, the number of terminal nodes as well as the misclassification error which is low and the accuracy which is high. The library used in R is tree.

Table 3

Variables used in the tree construction	"C" and "Mean Income"
Number of terminal nodes	5
Residual mean deviance	0.1318 = 10.41 / 79
Misclassification error rate	0.02381 = 2 / 84
Decision Tree's accuracy	91.89%

Graph 11 - Decision Tree



Random Forest

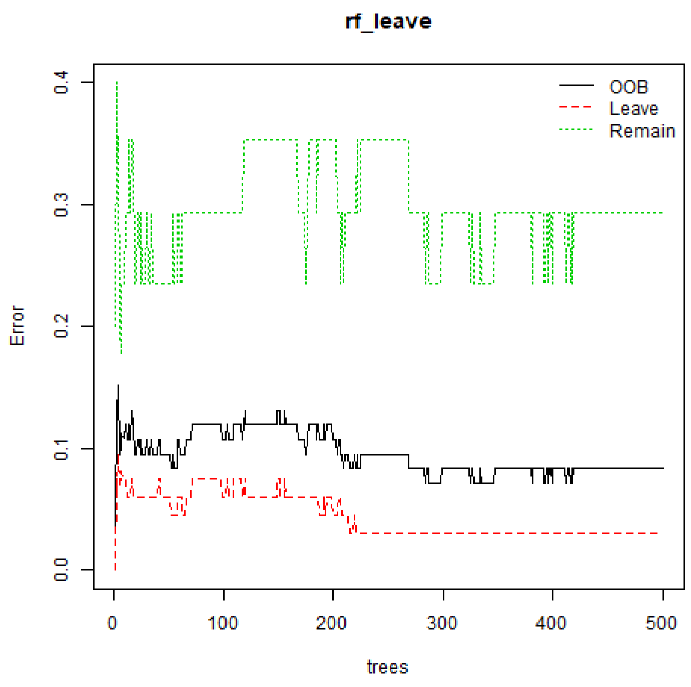
The random forest is a supervised predictive method which uses an ensemble technique and combines several decision trees together to get the optimal model. For the data partition the random seed has been set to 2018 and the dataset was split to 70% training and 30% test subsets. The outcome column will be used, as the classification has been chosen for prediction in this case. The formula of random forest has been created and the summary is shown below. The type of random forest is classification since the percent of leave has been split into two categories, leave, and remain. The number of trees that have been used is the default 500 and the out of bag estimate error is 8.33 which means almost 92% accuracy. From the confusion matrix it is observed that the leave classification error is very low and the remain is relatively low. This can be explained from the fact that in the dataset most of the inputs have been marked as leave so the system have limited inputs to analyze for remain creating with this way occasionally an error of class. A random sample that has been selected from the test and the training subsets correctly predicts the outcome.

The confusion matrix shows that the accuracy is almost 93% with its confidence interval tight between 85% and 97%. The p-value is very close to 0 which indicates that the model is accurate. The graph 12 shows how the random forest out of bag errors is distributed through the trees. The leave line has minor errors up to the 210 trees and then normalizes to reach the error of 0.029 up to 500 trees while the remain line has more ups and downs in the first 400 tree and the normalizes to reach the error of 0.28. Finally, the out of bag error follows almost the same trajectory as the leave line. The graph 13 shows the variables that play the most significant role in determining the prediction of the model. In more detail it shows that the conservative party, the UKIP party, and the Green party are the most significant. This shows a strong political bias behind the decision. The library used in R is Random forest.

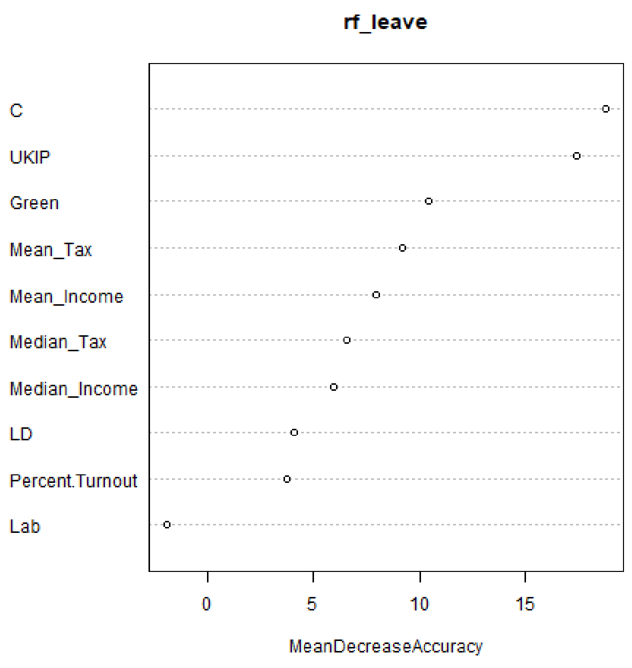
Table 4

Number of variables tried at each split	3		
Type of random forest	Classification		
Number of trees	500		
OBB estimate of error rate	8.33		
Confusion matrix training set	Leave	Remain	Class. Error
Leave	65	2	0.02985
Remain	5	12	0.29411

Graph 12



Graph 13



Support Vector Machines

Another machine learning technique for prediction is the method of support vector machines(svm) where the model will predict the future outcome of a constituency with set characteristics. Support vector machine is a supervised learning method for prediction of classification and regression. The dataset will use the column Outcome to predict through classification. This method uses kernel function for computational efficiency in order to find the perfect boundaries between the data points and separate them. The line that separates the data points is called hyperplane. In general, support vectors are the data points that seem to be closer to the hyperplane and help it separate from the rest of the data points. In most cases data points are not on the single line and then the hyperplane intervenes in order to expand the line and fit more data points on its borders. There are several kernel functions that can be used to extract the prediction, such as functions are, binomial, linear and sigmoid. The table 5 shows that the type of svm is classification and the kernel functions that was used is linear. The number of vectors is 16(8 for each class). The outcome of the prediction as shown below is fairly accurate while the system has misclassified only 2 leave observations and 1 remain. Similarly, the table 6 shows the same model used with a different kernel function. The function used this time is the polynomial which gives 29 vectors, 16 vectors for leave and 13 vectors for remain. The prediction as shown from the below table it shows that all the leave objects were predicted correctly while 4 of remain were predicted wrong. Also, the prediction is 97% accurate with 3% misclassification error._The third method used in the table 7 is the kernel sigmoid function. The results showing from the summary show that 34 vectors have been created 18 for leave and 19 for remain. The accuracy of this model is 82%. One main advantage of the svm technique is its high accuracy, on the other hand svm models are difficult to be interpreted and therefore not so intuitive. In this case, the svm is ideal for classification as well as is not sensitive to overfitting due to the small dataset available. The library used in R for svm is e1071.

Table 5 – Kernel function Linear

SVM - Type	C-Classification	
SVM - Kernel	Linear	
Number of Support Vectors	16 (8 8)	
Number of Classes	Leave and Remain	
Prediction	Leave	Remain
Leave	96	2
Remain	1	22
Classification Error	0.025%	
Accuracy	0.98%	

Table 6 – Kernel function Polynomial

SVM - Type	C-Classification	
SVM - Kernel	Polynomial	
Number of Support Vectors	29 (16 13)	
Number of Classes	Leave and Remain	
Prediction	Leave	Remain
Leave	97	4
Remain	0	20
Classification Error	0.03%	
Accuracy	0.97%	

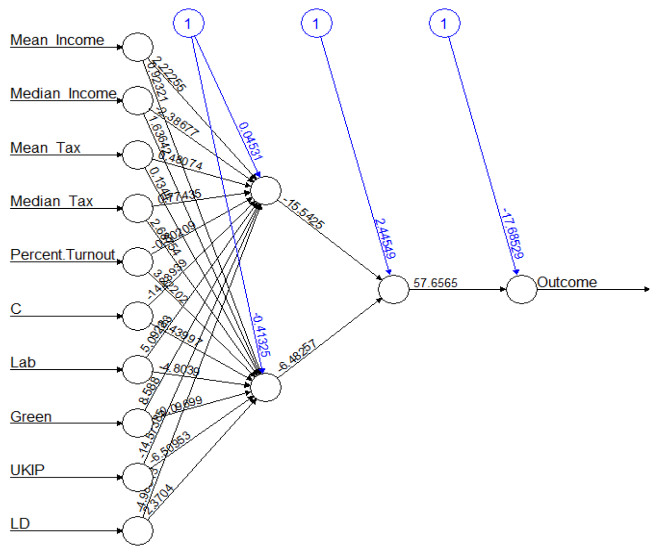
Table 7 – Kernel function Sigmoid

SVM - Type	C-Classification	
SVM - Kernel	Sigmoid	
Number of Support Vectors	34 (18 16)	
Number of Classes	Leave and Remain	
Prediction	Leave	Remain
Leave	88	13
Remain	9	11
Classification Error	0.18%	
Accuracy	0.82%	

<u>Neural Network</u>

To make a prediction through a neural network the data must be normalized again with the min max function in order to take values from 0 to 1 and make it easier for the network to find patterns between the data. The model has been used to predict the leave outcome and which factors affected this outcome. The neural network is consisted of the input layers which are the variables on the left, the first hidden layers, the second hidden layer and the output layer. In the neural network below 2 nodes have been used in the first hidden layer and 1 in the second hidden layer. The hidden layer tries to find the connections between the neurons so they can form a bold decision. The neural network can be described as a "black box", since the calculations it makes between its nodes and layers are unknown. The best way to evaluate the model's performance is to measure its accuracy and errors. However, the more nodes and layers it has, does not necessarily mean better performance and there are cases where a simpler model and a more complex model can give very similar results. In such cases, the simpler model is preferred as more time and resource effective. Extra hidden layers can cause overfitting or give very low accuracy since the machine will start to create relationships between the neurons that practically do not exist for a small dataset. There are six neural networks that have been selected to be analyzed. The neural network 65 is the most accurate one. This result can be obtaining from the cor command which gives the table at the performance evaluation chapter below. The nn65 has an accuracy of 91%, although further checks must be conducted to evaluate the performance of this prediction. The library used in R is neuralnet.

Graph 15 – Neural Network 65



Performance evaluation

Decision Tree's performance Evaluation, k-fold cross validation.

K-fold cross validation is a resampling procedure that split the dataset into specified numbers of subsets and tests the results on each subset. In this case for the decision tree the number of subsets k has been set to 10. The results of the cross validation below show that the model reached perfect accuracy. The graph 9 shows when the model reached the perfect accuracy which is at 0.5 cp. The table 8 shows the actual and predicted values form the test and the train dataset and all the objects have been predicted correctly apart from one. The library used in R is caret, rpart and performanceAnalytics.

Table 8 – Decision Tree Confusion Matrix

Accuracy	97.3%	
95% Confidence Intervals	0.8584, 0.9993	
P-Value	0.00411	
Prediction	Leave	Remain
Leave	30	1
Remain	0	6
Classification Error	0.07%	
Positive Class	Leave	

Table 9 – Decision Tree's Performance

Samples	84		
Predictors	12		
Classes	Leave and Remain		
Resampling	Cross-Validated (10 fold)		
Results	CP	Accuracy	Kappa
	0.0	1	1
	0.5	1	1
	1	0.7986	0

Random Forest's performance Evaluation

The result below shows the comparison between the training and test set. A prediction has returned accurate results for both sets. The training set reached 100% accuracy and the test misclassified three classes.

For the random forest prediction initially the bootstrapping method was used. Bootstrapping is a method that instead of 10 folds uses 1000 folds and it is ideal for big datasets where the system can find associations. Since the dataset in this report is small, bootstrapping may cause overfitting so a simpler method will be used. Also, the random forest was tuned but it is observed from the table 11 that when the tree was tuned the model reached almost 80% accuracy with only 2 mtry. Mtry is the number of variables randomly sampled as candidates at each split (Brodsky and Pippenger, 2005). The dataset extracts result easily, and advanced methods are not necessary to be utilized in this case.

Table 10 – Training and Test accuracy of Random Forest

Test set		
	Leave	Remain
Leave	29	2
Remain	1	5
Accuracy	91.8%	
Training set		
Leave	Leave	Remain
Remain	67	0
	0	17
Accuracy	100%	

Table 11 - Tuning of Random forest

Samples	84		
Predictors	12		
Classes	Leave and Remain		
Resampling	Cross-Validated (10 fold)		
Results	mtry	Accuracy	Kappa
	2	0.7896	0
	66	1	1
	131	1	1

Neural Network's performance Evaluation

As mentioned above, there is not a specific method in order to choose the optimal numbers of nodes and hidden layers. Therefore, several combinations and architectures have been tried for the neural network prediction. The best performing neural network is the number 65 where the correlation score of 91% is the highest among the others. Also, the Root Mean Square Error and the Mean Absolute Error is the lowest among the other models. Furthermore, the architecture used is quite simple that makes the model more convenient. The neural network has been used for classification with binary classifier. The model gives us a prediction on how people could vote in case that there is a second referendum. It is observed that people with lower incomes and from a center/right political background are in a favor of voting for leave and if these political parties have to base a political strategy on the result of leave, they should focus on how people will benefit financially from a potential leave of the European Union as well as what an impact this will have to the country's sovereignty. The library used in R is metrics.

Table 12

NNs	RSME	MAE	Correlation	Hidden Layers
Neural Network 64	0.1725	0.1721	0.8995	1,2
Neural Network 65	0.1645	0.1634	0.9102	2,1
Neural Network 66	0.1785	0.1776	0.8980	1,1
Neural Network 67	0.1705	0.1779	0.9006	2,2
Neural Network 68	0.1756	0.1728	0.8958	2,3
Neural Network 69	0.1801	0.1753	0.8902	4,4

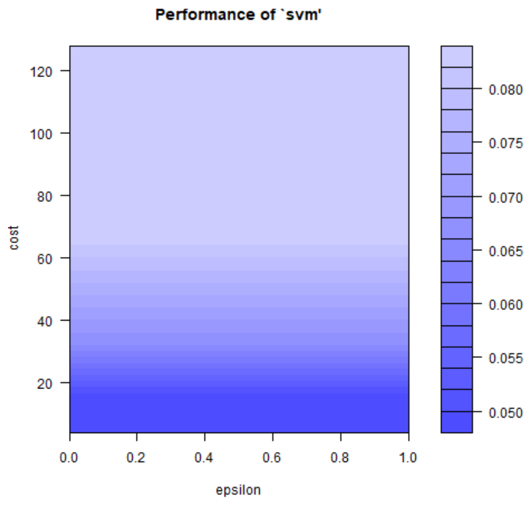
Performance Evaluation Support Vector Machines

Three kernel functions have been applied to extract a result from the support vector machine classification, all of them seem to have high accuracy. Although, in order to decide which one is the best method to use, the data will be tuned. Tuning is also known as hyper parameter optimization and provides a result of which is the best mode. The random seed was set to 2018, the parameter epsilon has been set with a sequence that goes from 0 to 1 with increment of 0.1. Also, the other parameter is cost. The cost parameter decides how much an svm should be allowed to "bend"(WANG, 2008). The cost has the default values which is 1, if the value of cost is high that means that the machine may have many vectors something that will lead to overfitting, on the other hand if the value is small there is the threat of under fitting which will have an impact on the accuracy of the model. The graphs 17 and 18 show how the result changes after the tuning of the cost parameter. The dark blue area shows that this area provides better results, as for example lower value for cost and different values for epsilon. The table 13 shows the best Kerner method out of three.

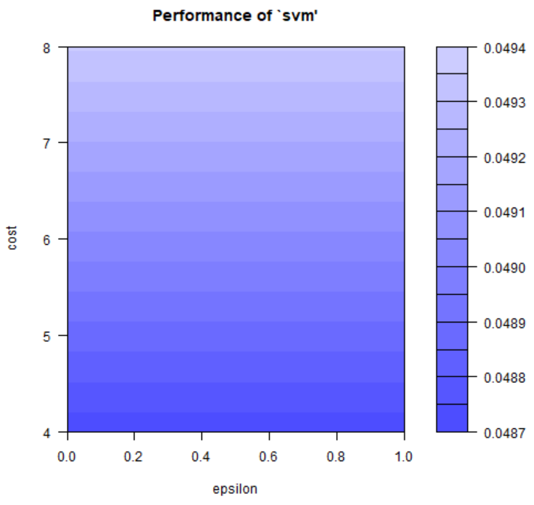
Table 13

SVM - Type	C-Classification	
SVM - Kernel	Linear	
Number of Support Vectors	16 (8 8)	
Number of Classes	Leave and Remain	
Prediction	Leave	Remain
Leave	96	3
Remain	1	21
Classification Error	0.025%	
Accuracy	0.98%	

Graph 17 – Performance of SVM with high cost value



Graph 18 – Performance of SVM with low cost value



Discussion of Findings

The report concludes that there are significant indications that the result at the EU Referendum in 2016 was formed by political and social biases. The decision tree reported that the most significant variable was the political party of Conservatives and the second most important the Mean Income. All the supervised techniques reported association between these two variables and the Outcome. Political parties would have based their campaign targeting specific aspects of everyday life to persuade people to vote for leave. In the scenario that there is a second referendum, the political parties that will support remain, should build a campaign on trying to persuade people on the financial benefits of remaining in the EU or that a leave decision will not change their financial status as the opposition states. It is observed that the political parties that backed the EU referendum persuaded their voters to vote against remain.

The fact that the center/right political parties had more votes where the percentage of leave was higher shows a key point of sovereignty might played a significant role on how to persuade these voters. Similarly, the percentage of leave was higher in constituencies with lower median and mean income, indicating that political parties could take advantage of this situation and point that the European Union was responsible for this. In the unsupervised method of biplot, there was an indication between the votes of the political parties and the outcome of the referendum. For example, UKIP and the Conservative parties it is known that backed the leave campaign(Hobolt, 2016) and it is observed from the bitplot that when the UKIP and C votes go up the percentage of leave goes up too.

On the other hand, it is known that the Green and Liberal Democrats backed the remain campaign (Hobolt, 2016) and this can be shown in the biplot where while the votes for these parties go up the percentage of leave goes down. The above observations draw the conclusion that voters remain faith to their political parties. The political party that seems to be the “game-changer” is the Labour party which is the second biggest party in the UK and backed the remain campaign(Hobolt, 2016). The biplot before the dimensionality reduction shows that when the votes for Labour go up then the percentage of leave goes up too, something irregular given the political theses of the party. In the second biplot, when the percentage of leave goes up the Labour votes slightly go down but not significantly. The table below shows the 6 constituencies with the most votes for Labour, the percentage of leave on these constituencies, as well as the median income.

Table 14 – Constituencies with the most Labour Votes

Constituency	Median Income	Percentage of Leave
Knowsley	17400	51.56
Halton	18300	57.42
Exeter	19100	44.72
Bassetlaw	18000	67.83
Wigan	19500	63.90
Slough	19200	54.33

The Labour voters did not follow their party’s direction. Only one constituency(Exeter), with the most Labour votes, out of 6 voted to remain. It is also worth mentioning that all these constituencies were below the median income threshold. It seems that voters put the social bias above the political. The supervised methods reported very accurate results and it could be used in case there is a second referendum to try and predict what the future result would be as well as how their political parties should form and organize their campaigns. All the models give high value of prediction because the dataset can extract results easily and it does not require advanced and more in thorough methods.

Limitations

- The social and political biases have been based only on the votes from the 2015 general elections and the income per constituency. There are several other factors that can influence how people voted.
- Only one year of general elections was collected and analyzed and the result could be different if more general elections were combined.
- Only a third of the total constituencies has been analyzed for this report.
- There are several speculations on how the political parties based their campaign.

References

Szekely, G. and Rizzo, M., 2005. Hierarchical Clustering via Joint Between-Within Distances: Extending Ward's Minimum Variance Method. Journal of Classification, 22(2), pp.151-183.

Brodsky, A. and Pippenger, N., 2005. The Boolean functions computed by random Boolean formulas or how to grow the right function. Random Structures and Algorithms, 27(4), pp.500-519.

WANG, D., 2008. Utilizing particle swarm optimization to optimize hyper-parameters of SVM classifier. Journal of Computer Applications, 28(1), pp.134-135.

Hobolt, S., 2016. The Brexit vote: a divided nation, a divided continent. Journal of European Public Policy, [online] pp. 1-35. Available at: <http://eprints.lse.ac.uk/67546/7/Hobolt_The%20Brexit%20vote%20a%20divided%20.pdf> [Accessed 10 April 2020].

Appendix

R Script

#Data Preparation and Cleaning

#Census

```
census <- census[,-c(1,5:23)]
popullation <- census[,-1]
colnames(popullation)[2] <- "Popullation"
```

#Referendum Results

```
referendum <- EU.referendum.result.data[,-c(1:4,12,13,15:19,7)]
```

#General elections results 2015

```
results1 <- RESULTS.FOR.ANALYSIS[,-c(1,3,4,5,6,7)]
results2 <- results1[,-c(4:20,22:40,42:46)]
results3 <- results2[,-c(7:21,23,25:50)]
results4 <- results3[,-c(9:41,43:50)]
results <- results4[-c(10:17,2,3)]
results <- results[,-3]
```

#NAs

```
colnames(results)[1] <- "Area"
Results_NA_count <- apply(is.na(results), 2, sum)
Results_NA_count
Results_NA_perc <- Results_NA_count / dim(results)[1] * 100
Results_NA_perc
results[is.na(results$C), 'C'] = '1'
results[is.na(results$Green), 'Green'] = '1'
results[is.na(results$Lab), 'Lab'] = '1'
results[is.na(results$LD), 'LD'] = '1'
results[is.na(results$UKIP), 'UKIP'] = '1'
colnames(results)[1] <- "Area"
```

#Income and Tax

```
Income_and_Tax <- NS_Table_3_15_1516[,-c(2:15,19,20,23,16)]
colnames(Income_and_Tax)[1] <- "Area"
colnames(Income_and_Tax)[2] <- "Mean_Income"
colnames(Income_and_Tax)[3] <- "Median_Income"
colnames(Income_and_Tax)[4] <- "Mean_Tax"
colnames(Income_and_Tax)[5] <- "Median_Tax"
Income_and_Tax <- na.omit(Income_and_Tax)
Income_and_Tax$Area <- factor(Income_and_Tax$Area)
Income_and_Tax$Median_Tax <- as.numeric(as.character(Income_and_Tax$Median_Tax))
Income_and_Tax$Median_Income <- as.numeric(as.character(Income_and_Tax$Median_Income))
Income_and_Tax$Mean_Tax <- as.numeric(as.character(Income_and_Tax$Mean_Tax))
Income_and_Tax$Mean_Income <- as.numeric(as.character(Income_and_Tax$Mean_Income))
```

#Library dplyr

```
leave <- inner_join(referendum,results)
leave <- inner_join(Income_and_Tax,leave)
leave <- inner_join(popullation,leave)

leave$Area <- factor(leave$Area)
leave$C <- as.numeric(as.character(leave$C))
leave$Green <- as.numeric(as.character(leave$Green))
leave$UKIP <- as.numeric(as.character(leave$UKIP))
leave$Lab <- as.numeric(as.character(leave$Lab))
leave$LD <- as.numeric(as.character(leave$LD))
```

#Exploratory Data Analysis

```
summary(leave)
png(file = "leave_clear")
par(mfrow = c(2,2))

hist(
  leave$Median_Income,
  xlab = 'Median Income in $1000',
  ylab = 'Frequency',
  main = 'Median Income Per Consituency',
  col = rgb(0, 1, 0),
  border = 'white',
  xaxt = 'n'
)
axis(1, at = seq(0, 65000, 5000), seq(0, 65, 5))
abline(v = median(leave$Median_Income,), lty = 2)
legend('topright', 'median Income', lty = 2, bty = 'n')


hist(
  leave$Median_Tax,
  xlab = 'Median Tax in $1000',
  ylab = 'Frequency',
  main = 'Media Tax Per Consituency',
  col = rgb(0, 1, 0),
  border = 'white',
  xaxt = 'n'
)
axis(1, at = seq(0, 65000, 5000), seq(0, 65, 5))
```

```
abline(v = median(leave$Median_Tax,), lty = 2)
legend('topright', 'median Tax', lty = 2, bty = 'n')

hist(
  leave$Mean_Income,
  xlab = 'Mean Income in $1000',
  ylab = 'Frequency',
  main = 'Mean Income Per Consituency',
  col = rgb(0, 1, 0),
  border = 'white',
  xaxt = 'n'
)
axis(1, at = seq(0, 65000, 5000), seq(0, 65, 5))
abline(v = median(leave$Mean_Income,), lty = 2)
legend('topright', 'Mean Income', lty = 2, bty = 'n')

hist(
  leave$Mean_Tax,
  xlab = 'Mean Tax in $1000',
  ylab = 'Frequency',
  main = 'Mean Tax Per Consituency',
  col = rgb(0, 1, 0),
  border = 'white',
  xaxt = 'n'
)
axis(1, at = seq(0, 65000, 5000), seq(0, 65, 5))
abline(v = median(leave$Mean_Tax,), lty = 2)
legend('topright', 'Mean Tax', lty = 2, bty = 'n')




png(file = "percent.leave")
plot(
  leave$Pct_Leave ~ leave$Median_Income,
  ylab = 'Percentage of Leave',
  xlab = 'Median Income',
  main = 'Leave vs Income', pch = 0.15
)
abline(v = median(leave$Median_Income,), lty = 2)
legend('topright', 'Median Income', lty = 2, bty = 'n')
dev.off()
plot(
  leave$Pct_Leave ~ leave$Median_Tax,
  ylab = 'Percentage of Leave',
  xlab = 'Median Tax',
  main = 'Leave vs Tax', pch = 0.15
)
abline(v = median(leave$Median_Tax,), lty = 2)
legend('topright', 'Median Tax', lty = 2, bty = 'n')

par(mfrow = c(2,2))

plot(density(leave[, 3]), main = names(leave)[3], xlab = names(leave)[3])
plot(density(leave[, 4]), main = names(leave)[4], xlab = names(leave)[4])
plot(density(leave[, 5]), main = names(leave)[5], xlab = names(leave)[5])
plot(density(leave[, 6]), main = names(leave)[6], xlab = names(leave)[6])
```

```
pc_leave <- prcomp(leave[,-1], center = T, scale. = T)
pc_leave_var <- pc_leave$sdev^2
pc_leave_var
pc_leave_PEV <- pc_leave_var / sum(pc_leave_var)
pc_leave_PEV
png(file = "percent")
plot(pc_leave)
dev.off()
plot(
  cumsum(pc_leave_PEV),
  ylim = c(0,1),
  xlab = 'PC',
  ylab = 'Cumulative PEV',
  pch = 20,
  col = 'orange'
)
abline(h = 0.8, col = 'red', lty = 'dashed')
pc_leave_loadings <- pc_leave$rotation
pc_leave_loadings

colvector = c('red', 'orange', 'yellow', 'green', 'cyan', 'blue')
labvector = c('PC1', 'PC2', 'PC3')
barplot(
  pc_leave_loadings[,c(1:3)],
  beside = T,
  yaxt = 'n',
  names.arg = labvector,
  col = colvector,
  ylim = c(-1,1),
  border = 'white',
  ylab = 'loadings'
)
axis(2, seq(-1,1,0.1))
legend(
  'bottomright',
  bty = 'n',
  col = colvector,
  pch = 15,
  row.names(pc_leave_loadings)
)
par(mfrow = c(3,1))
biplot(
  pc_leave,
  scale = 0,
  col = c('grey40','orange')
)
biplot(
  pc_leave,
  choices = c(1,3),
  scale = 0,
  col = c('grey40','orange')
)
biplot(
  pc_leave,
  choices = c(2,3),
  scale = 0,
  col = c('grey40','orange')
)

leave_clear <- leave[,-c(2,8,10,11,12,14,7)]
summary(pc_leave_clear)
pc_leave_clear <- prcomp(leave_clear[,-1], center = T, scale. = T)
pc_leave_var <- pc_leave_clear$sdev^2
pc_leave_var
pc_leave_PEV <- pc_leave_var / sum(pc_leave_var)
pc_leave_PEV
plot(pc_leave_clear)
png(file = "percent")
plot(
  cumsum(pc_leave_PEV),
  ylim = c(0,1),
  xlab = 'PC',
  ylab = 'Cumulative PEV',
  pch = 20,
  col = 'orange'
)
abline(h = 0.8, col = 'red', lty = 'dashed')
pc_leave_loadings <- pc_leave_clear$rotation
pc_leave_loadings
opar <- par()
colvector = c('red', 'orange', 'yellow', 'green', 'cyan', 'blue')
labvector = c('PC1', 'PC2', 'PC3')
barplot(
  pc_leave_loadings[,c(1:3)],
  beside = T,
  yaxt = 'n',
  names.arg = labvector,
  col = colvector,
  ylim = c(-1,1),
  border = 'white',
  ylab = 'loadings'
)
axis(2, seq(-1,1,0.1))
legend(
  'bottomright',
  bty = 'n',
  col = colvector,
  pch = 15,
  row.names(pc_leave_loadings)
)
par(mfrow = c(2,2))
png(file = "percent")
biplot(
  pc_leave_clear,
  scale = 0,
  col = c('grey40','orange')
)
biplot(
  pc_leave_clear,
  choices = c(1,3),
  scale = 0,
  col = c('grey40','orange')
)
```

```
biplot(
  pc_leave_clear,
  choices = c(2,3),
  scale = 0,
  col = c('grey40','orange')
)
pca3d::pca3d(pc_leave_clear, show.labels = T)

summary(pc_leave)
summary(pc_leave_clear)

plot(pc_leave, type = "l")


#Clustering

dist_leave <- dist(leave_clear[,-1], method = 'euclidian')
#   then apply complete linkage
png(file = "percent")
hc_leave <- hclust(dist_leave, method = 'ward.D')
hc_leave

dist_leave <- dist(leave_clear[,-1], method = 'euclidian')
hc_leave_complete <- hclust(dist_leave, method = 'complete')
# plot the associated dendrogram

plot(hc_leave, hang = -0.1, labels = leave_clear$Area)
png(file = "percent")
plot(hc_leave_complete, hang = -0.1, labels = leave_clear$Area)


# Evaluation of cluster results

sil_hc_leave <- cluster::silhouette(hc_cluster_id_leave, dist_leave)
sil_hc_leave_complete <- cluster::silhouette(hc_cluster_id_leave_complete, dist_leave)

opar <- par()
par(mfrow = c(2,1))
png(file = "percent")
plot(sil_hc_leave)
png(file = "percent")
plot(sil_hc_leave_complete)
par(opar)

leave_clear["Outcome"] <- 0
leave_clear$Outcome[ leave_clear$Pct_Leave > 50] <- "1"
leave_clear$Outcome[ leave_clear$Pct_Leave < 50] <- "0"
leave_clear$Outcome <- as.numeric(leave_clear$Outcome)

#Neural network

set.seed(2018)
MinMax <- function(x){
  tx <- (x - min(x)) / (max(x) - min(x))
  return(tx)
}
leave_minmax <- apply(leave_clear[,-1], 2, MinMax)
leave_minmax <- as.data.frame(leave_minmax)
n_rows <- nrow(leave_minmax)
training_idx <- sample(n_rows, n_rows * 0.7)
training_leave_minmax <- leave_minmax[training_idx,]
test_leave <- leave_minmax[-training_idx,]
leave_formula = Outcome ~ Mean_Income + Median_Income + Mean_Tax + Median_Tax + Pct_Turnout + C + Lab + G
reen + UKIP + LD
png(file = "p")
```

#Library neural net

```
leave_nn_64 <- neuralnet(leave_formula, hidden = c(1,2), data = training_leave_minmax,linear.output = FALSE)
leave_nn_65 <- neuralnet(leave_formula, hidden = c(2,1), data = training_leave_minmax,linear.output = FALSE)
leave_nn_66 <- neuralnet(leave_formula, hidden = c(1,1), data = training_leave_minmax, linear.output = FALSE)
leave_nn_67 <- neuralnet(leave_formula, hidden = c(2,2), data = training_leave_minmax,linear.output = FALSE)
leave_nn_68 <- neuralnet(leave_formula, hidden = c(2,3), data = training_leave_minmax,linear.output = FALSE)
leave_nn_69 <- neuralnet(leave_formula, hidden = c(4,4), data = training_leave_minmax, linear.output = FALSE)
pred_leave_nn_64 <- compute(leave_nn_64, test_leave[,-13])
pred_leave_nn_65 <- compute(leave_nn_65, test_leave[,-13])
pred_leave_nn_66 <- compute(leave_nn_66, test_leave[,-13])
pred_leave_nn_67 <- compute(leave_nn_67, test_leave[,-13])
pred_leave_nn_68 <- compute(leave_nn_68, test_leave[,-13])
pred_leave_nn_69 <- compute(leave_nn_69, test_leave[,-13])

leave_results <- data.frame(
  actual = test_leave$Outcome,
  nn_64 = pred_leave_nn_64$net.result,
  nn_65 = pred_leave_nn_65$net.result,
  nn_66 = pred_leave_nn_66$net.result
)
leave_results <- data.frame(
  actual = test_leave$Outcome,
  nn_67 = pred_leave_nn_67$net.result,
  nn_68 = pred_leave_nn_68$net.result,
  nn_69 = pred_leave_nn_69$net.result
)

cor(leave_results[,'actual'], leave_results[,c("nn_64","nn_65", "nn_66")])
cor(leave_results[,'actual'], leave_results[,c("nn_67","nn_68", "nn_69")])
leave_results <- data.frame(
  actual = test_leave$Outcome,
  nn_65 = pred_leave_nn_65$net.result
)
leave_results <- data.frame(
  actual = test_leave$Outcome,
  nn_65 = pred_leave_nn_65$net.result
)



temptest <- subset(test_leave, select = c("Mean_Income", "Median_Income", "Mean_Tax", "Median_Tax", "Pct_Turn
out", "C", "Lab", "Green", "UKIP", "LD"))
head(temptest)
nn_results <- compute(leave_nn_65,temptest)
nn_results <- compute(leave_nn_69,temptest)
nn_results
results <- data.frame(actual = test_leave$Outcome, prediction = nn_results$net.result)
results
```

```
roundedresults <- sapply(results, round,digits=0)
roundedresultsdf = data.frame(roundedresults)
table(leave_results)

predict_testNN = (pred_leave_nn_65$net.result * (max(test_leave$Outcome) - min(test_leave$Outcome))) + min(tes
t_leave$Outcome)
predict_testNN = (pred_leave_nn_64$net.result * (max(test_leave$Outcome) - min(test_leave$Outcome))) + min(tes
t_leave$Outcome)
predict_testNN = (pred_leave_nn_66$net.result * (max(test_leave$Outcome) - min(test_leave$Outcome))) + min(tes
t_leave$Outcome)
predict_testNN = (pred_leave_nn_67$net.result * (max(test_leave$Outcome) - min(test_leave$Outcome))) + min(tes
t_leave$Outcome)
predict_testNN = (pred_leave_nn_68$net.result * (max(test_leave$Outcome) - min(test_leave$Outcome))) + min(tes
t_leave$Outcome)
predict_testNN = (pred_leave_nn_69$net.result * (max(test_leave$Outcome) - min(test_leave$Outcome))) + min(tes
t_leave$Outcome)

plot(test_leave$Outcome, predict_testNN, col='blue', pch=16, ylab = "Predicted Outcome NN", xlab = "Real Outcome
")

abline(0,1)
RMSE.NN = (sum((test_leave$Outcome - predict_testNN)^2) / nrow(test_leave)) ^ 0.5
RMSE.NN

1-mae(pred_leave_nn_64$net.result * (max(test_leave$Outcome) - min(test_leave$Outcome))) + min(test_leave$Ou
tcome)
1-mae(pred_concrete_nn_65$net.result * (max(test_concrete_minmax$Outcome) - min(test_concrete_minmax$Out
come))) + min(test_concrete_minmax$Outcome)
1-mae(pred_concrete_nn_66$net.result * (max(test_concrete_minmax$Outcome) - min(test_concrete_minmax$Out
come))) + min(test_concrete_minmax$Outcome)
1-mae(pred_concrete_nn_67$net.result * (max(test_concrete_minmax$Outcome) - min(test_concrete_minmax$Out
come))) + min(test_concrete_minmax$Outcome)
1-mae(pred_concrete_nn_68$net.result * (max(test_concrete_minmax$Outcome) - min(test_concrete_minmax$Out
come))) + min(test_concrete_minmax$Outcome)
1-mae(pred_concrete_nn_69$net.result * (max(test_concrete_minmax$Outcome) - min(test_concrete_minmax$Out
come))) + min(test_concrete_minmax$Outcome)
```

#Create an extra column for classification

```
leave_clear["Outcome"] <- 0
leave_clear$Outcome[ leave_clear$Pct_Leave > 50] <- "Leave"
leave_clear$Outcome[ leave_clear$Pct_Leave < 50] <- "Remain"
leave_clear$Outcome <- as.factor(leave_clear$Outcome)
```

#Decision Tree
# Library tree

```
set.seed(2018)
n_rows <- nrow(leave_clear)
training_idx <- sample(n_rows, n_rows * 0.7)
training_leave <- leave_clear[training_idx,]
test_leave <- leave_clear[-training_idx,]
leave_formula = Outcome ~ Mean_Income + Median_Income + Mean_Tax + Median_Tax + Pct_Turnout + C + Lab + G
reen + UKIP + LD
tree_leave <- tree(leave_formula, data = training_leave)
summary(tree_leave)
```

```
plot(tree_leave)
text(tree_leave, pretty = 0)
cv_leave <- cv.tree(tree_leave, FUN=prune.misclass)
cv_leave_table <- data.frame(
  size = cv_leave$size,
  error = cv_leave$dev
)

plot(
  cv_leave,
  xaxt = 'n',
  yaxt = 'n'
)
axis(1, seq(1,max(cv_leave_table$size)))
axis(2, seq(50,150,5))

pruned_tree_size <- cv_leave_table[which.min(cv_leave_table$error), 'size']
pruned_tree_leave <- prune.misclass(tree_leave, best = pruned_tree_size)
summary(pruned_tree_leave)
tree_leave_pred <- predict(tree_leave, test_leave[,-13], type= "class")
pruned_tree_leave_pred <- predict(pruned_tree_leave, test_leave[,-13], type= "class")
leave_results <- data.frame(
  actual = test_leave$Outcome,
  unpruned = tree_leave_pred,
  pruned = pruned_tree_leave_pred)

unpruned_results_table <- table(leave_results[,c('actual', 'unpruned')])
unpruned_results_table
pruned_results_table <- table(leave_results[,c('actual', 'pruned')])
pruned_results_table

acc_unpruned <- sum(diag(unpruned_results_table)) / sum(unpruned_results_table)
acc_unpruned
acc_pruned <- sum(diag(pruned_results_table)) / sum(pruned_results_table)
acc_pruned
```

#Performance evaluation Decision Tree and Random Forest

#Library caret

```
set.seed(2018)

leave_formula <- reformulate(names(training_leave[, -13]), response = 'Outcome')

ctrl_parameters <- trainControl(method = 'CV', number = 10)
ctrl_parameters <- trainControl(method = 'CV', number = 30)
modelLookup('rpart')
leave_tree_perf <- train(leave_formula, data = training_leave, method = "rpart", trControl = ctrl_parameters)
leave_tree_perf
png(file = "p")
plot(leave_tree_perf)

modelLookup('rf')
leave_rf <- train(leave_formula, data = training_leave, method = "rf", trControl = ctrl_parameters)
leave_rf
plot(leave_rf)
leave_tree_predict <-  cbind(
  actual = test_leave$Outcome,
  predicted = predict(leave_tree_perf, test_leave[, -13], type = 'raw'),
  predict(leave_tree_perf, test_leave[, -13], type = 'prob')
)


leave_rf_predict <-  cbind(
  actual = test_leave$Outcome,
  predicted = predict(leave_rf, test_leave[, -13], type = 'raw'),
  predict(leave_rf, test_leave[, -13], type = 'prob')
)
leave_rf_predict
plot(leave_rf_predict)

tree_confmat <- confusionMatrix(data = leave_tree_predict$predicted, reference = leave_tree_predict$actual, positi
ve = "Leave")
rf_confmat <- confusionMatrix(data = leave_rf_predict$predicted, reference = leave_rf_predict$actual, positive = "Le
ave")

tree_confmat
rf_confmat

leave_models_prob <- data.frame(
  tree = leave_tree_predict$Leave,
  rf = leave_rf_predict$Leave
)
leave_label <- data.frame(
  tree = leave_tree_predict$actual,
  rf = leave_rf_predict$actual
)

opar <- par()
par(pty = 's')
png(file = "p")
```

#Random Forest
#Library Random Forest

```
print(leave_rf)

rf_leave <- randomForest(leave_formula, ntree = 500, importance = T, data = training_leave)
png(file = "p")
plot(rf_leave)
rf_leave_pred <- predict(rf_leave, test_leave[,-1], type= "class")
rf_leave_pred <- predict(rf_leave, training_leave[,-1], type= "class")
rf_results_table <- table(rf = rf_leave_pred,  actual = test_leave$Outcome)
rf_results_table <- table(rf = rf_leave_pred,  actual = training_leave$Outcome)
rf_results_table
acc_rf <- sum(diag(rf_results_table)) / sum(rf_results_table)
acc_rf
```

#Performance Evalution Random Forest

```
leave_rf_predict <-  cbind(
  actual = test_leave$Outcome,
  predicted = predict(rf_leave, test_leave[, -13], type = 'response'),
  predict(rf_leave, test_leave[, -13], type = 'response')
)
plot(leave_rf_predict)


#Support vector machines
qplot(Percent.Leave, Median_Income, data = leave_clear, colour = Outcome)
qplot(C, Lab, data = leave_clear, colour = Outcome)
```

#Kernel linear

#Library e1071

```
mymodel <- svm(Outcome ~ Mean_Income + Median_Income + Mean_Tax + Median_Tax + Pct_Turnout + C + Lab + G
reen + UKIP + LD, data = leave_clear, kernel = "linear")
mymodel <- svm(Outcome ~ Mean_Income + Median_Income + Mean_Tax + Median_Tax + Pct_Turnout + C + Lab + G
reen + UKIP + LD, data = leave_clear, kernel = "polynomial")
mymodel <- svm(Outcome ~ Mean_Income + Median_Income + Mean_Tax + Median_Tax + Pct_Turnout + C + Lab + G
reen + UKIP + LD, data = leave_clear, kernel = "radial")

summary(mymodel)
plot(mymodel , data = leave_clear,
    Median_Income~LD,
   )
pred <- predict(mymodel, leave_clear)
tab <- table(Predicted = pred, Actual = leave_clear$Outcome)
tab
1-sum(diag(tab)/sum(tab))
sum(diag(tab)/sum(tab))
```

#Kernel Stigmoid

```
mymodel <- svm(Outcome ~ Mean_Income + Median_Income + Mean_Tax + Median_Tax + Percent.Turnout + C + La
b + Green + UKIP + LD, data = leave_clear, kernel = "sigmoid")
summary(mymodel)
plot(mymodel , data = leave_clear,
    Median_Income~LD,
)
pred <- predict(mymodel, leave_clear)
tab <- table(Predicted = pred, Actual = leave_clear$Outcome)
tab
sum(diag(tab)/sum(tab))
1-sum(diag(tab)/sum(tab))
```

#Tuning

```
set.seed(2018)
tmodel <- tune(svm,Outcome ~ Mean_Income + Median_Income + Mean_Tax + Median_Tax + Pct_Turnout + C + Lab
+ Green + UKIP + LD, data = leave_clear, ranges = list(epsilon = seq(0,1,0.1), cost = 2^(2:5)))

plot(tmodel)
summary(tmodel)
bestmodel <- tmodel$best.model
summary(bestmodel)
png(file = "p")
plot(bestmodel , data = leave_clear,
    C~Pct_Leave,
)
pred <- predict(mymodel, leave_clear)
tab <- table(Predicted = pred, Actual = leave_clear$Outcome)
tab
1-sum(diag(tab)/sum(tab))
sum(diag(tab)/sum(tab))


p1 <- predict(rf_leave, data = training_leave)
head(p1)
head(training_leave$Outcome)
confusionMatrix(p1, training_leave$Outcome)
p2 <- predict(rf_leave, test_leave)
confusionMatrix(p2, test_leave$Outcome)
png(file = "p")
plot(rf_leave)
legend('topright', colnames(rf_leave$err.rate), bty = 'n', lty = c(1,2,3), col = c(1:3))
png(file = "p")
varImpPlot(rf_leave, type = 1)
```