Drug Consumption

Data Mining Project

Professor: Lianwen Wang

By

Dinesh Botta – 700697014

Nikesh sai sriram bachina – 700699778

Areefulla Shaik – 700688813

Durganadh Ginjupalli – 700691594

TABLE OF CONTENTS

INDIVIDUAL CONTRIBUTIONS OF THE TEAM

Dinesh Botta - Preprocessing of the data, Naïve Bayes Classification.

Nikesh sai Sriram Bachina - SVM with Linear, Radial and Polynomial.

Areefulla Shaik – Cross Validation, Random Forest and Bagging.

Durganadh Ginjupalli – Pruning**,** Decision Tree and Holdout Method.

Abstract:

This report presents the Data Mining case study of the Letter Recognition Dataset available in UCI Machine learning repository. The objective is to predict Safe and Unsafe condition of a respondent based on seven classes: "Never Used(C0)", "Used over a Decade Ago(C1)", "Used in Last Decade(C2)", "Used in Last Year(C3)", "Used in Last Month(C4)", "Used in Last Week(C5)", and "Used in Last Day(C6)" . Three different classifiers namely Decision tree, Naïve Bayes and SVM (Support Vector Machine) are used to classify the data. An open source for statistical computation and graphics – R version 3.6.1 is used for the preprocessing and mining purposes. We present the experimental results in terms of Confusion matrix for each of the classes. Finally, we conclude with comparison and certain understanding of the data based on different classifiers used.

Introduction:

Data mining, an interdisciplinary field of computer science is the process of discovering new patterns from large data sets involving methods at the intersection of artificial intelligence, statistics and database systems. We can train the machine by mining into a training dataset and by extracting some rules for the output prediction of the test samples. Data mining consists of basically five kinds of tasks namely Anomaly detection, Association rule learning, clustering, Classification and regression. Among the above tasks, clustering and classification and regression plays a great role in the machine learning models.

Classification:

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target **class** for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.

The Dataset and the aim:

Drug Consumption(quantified)

Database contains records for 1885 respondents. For each respondent 12 attributes are known: Personality measurements which include NEO-FFI-R (neuroticism, extraversion, openness to experience, agreeableness, and conscientiousness), BIS-11 (impulsivity), and ImpSS (sensation seeking), level of education, age, gender, country of residence and ethnicity. All input attributes are originally categorical and are quantified. After quantification values of all input features can be considered as real-valued. In addition, participants were questioned concerning their use of 18 legal and illegal drugs (alcohol, amphetamines, amyl nitrite, benzodiazepine, cannabis, chocolate, cocaine, caffeine, crack, ecstasy, heroin, ketamine, legal highs, LSD, methadone, mushrooms, nicotine and volatile substance abuse and one fictitious drug (Semeron) which was introduced to identify over-claimers. For each drug they have to select one of the answers: never used the drug, used it over a decade ago, or in the last decade, year, month, week, or day.

Database contains 18 classification problems. Each of independent label variables contains seven classes: "Never Used", "Used over a Decade Ago", "Used in Last Decade", "Used in Last Year", "Used in Last Month", "Used in Last Week", and "Used in Last Day".

Problem which can be solved:

* Seven class classifications for each drug separately.

* Problem can be transformed to binary classification by union of part of classes into one new class. For example, "Never Used", "Used over a Decade Ago" form class "SAFE " and all other classes form class "UNSAFE".

* The best binarization of classes for each attribute.

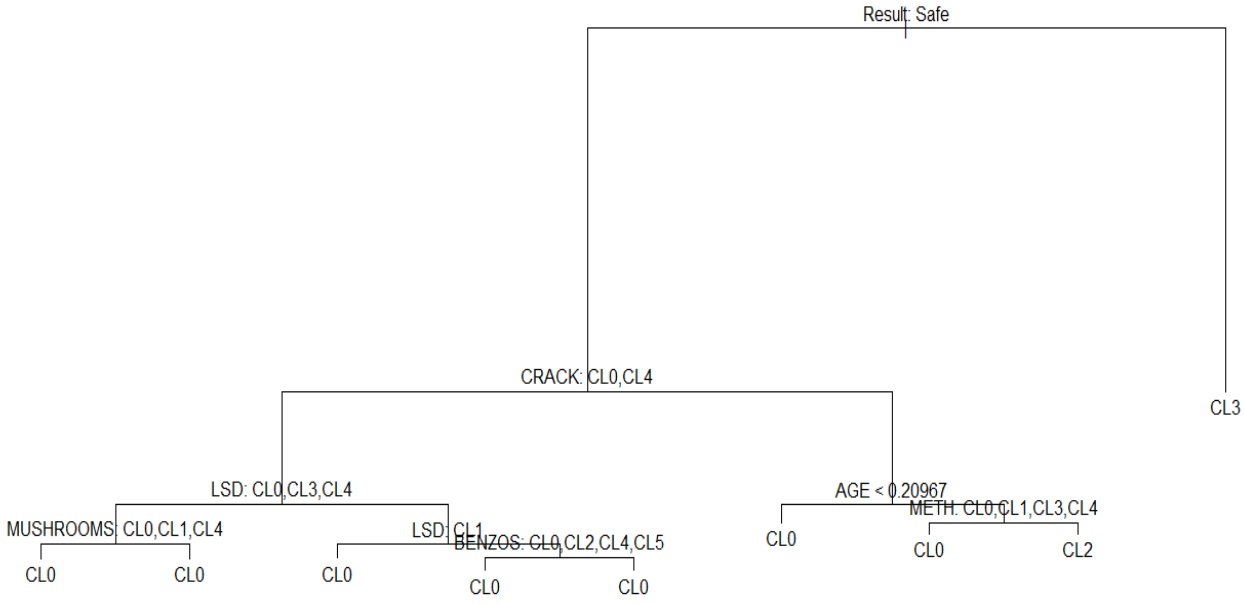
* Evaluation of risk to be drug consumer for each drug.

ID	AGE	GENDER	EDUCATION	COUNTRY	ETHNICITY	NSCORE	ESCORE	OSCORE	ASCORE	CSCORE	IMPULSIVE	SS	ALCOHOL	AMPHET	AMLY	BENZOS
1	0.49788	0.48246	-0.05921	0.96082	0.126	0.31287	-0.57545	-0.58331	-0.91699	-0.00665	-0.21712	-1.18084	CL5	CL2	CL0	CL2
2	-0.07854	-0.48246	1.98437	0.96082	-0.31685	-0.67825	1.93886	1.43533	0.76096	-0.14277	-0.71126	-0.21575	CL5	CL2	CL2	CL0
3	0.49788	-0.48246	-0.05921	0.96082	-0.31685	-0.46725	0.80523	-0.84732	-1.6209	-1.0145	-1.37983	0.40148	CL6	CL0	CL0	CL0
4	-0.95197	0.48246	1.16365	0.96082	-0.31685	-0.14882	-0.80615	-0.01928	0.59042	0.58489	-1.37983	-1.18084	CL4	CL0	CL0	CL3
5	0.49788	0.48246	1.98437	0.96082	-0.31685	0.73545	-1.6334	-0.45174	-0.30172	1.30612	-0.21712	-0.21575	CL4	CL1	CL1	CL0
6	2.59171	0.48246	-1.22751	0.24923	-0.31685	-0.67825	-0.30033	-1.55521	2.03972	1.63088	-1.37983	-1.54858	CL2	CL0	CL0	CL0
7	1.09449	-0.48246	1.16365	-0.57009	-0.31685	-0.46725	-1.09207	-0.45174	-0.30172	0.93949	-0.21712	0.07987	CL6	CL0	CL0	CL0
8	0.49788	-0.48246	-1.7379	0.96082	-0.31685	-1.32828	1.93886	-0.84732	-0.30172	1.63088	0.19268	-0.52593	CL5	CL0	CL0	CL0
9	0.49788	0.48246	-0.05921	0.24923	-0.31685	0.62967	2.57309	-0.97631	0.76096	1.13407	-1.37983	-1.54858	CL4	CL0	CL0	CL0
10	1.82213	-0.48246	1.16365	0.96082	-0.31685	-0.24649	0.00332	-1.42424	0.59042	0.12331	-1.37983	-0.84637	CL6	CL1	CL0	CL1
11	-0.07854	0.48246	0.45468	0.96082	-0.31685	-1.05308	0.80523	-1.11902	-0.76096	1.81175	0.19268	0.07987	CL5	CL0	CL1	CL0
12	1.09449	-0.48246	-0.61113	-0.28519	-0.31685	-1.32828	0.00332	0.14143	-1.92595	-0.52745	0.52975	1.2247	CL5	CL1	CL0	CL0
13	1.82213	0.48246	0.45468	0.96082	-0.31685	2.28554	0.16767	0.44585	-1.6209	-0.78155	1.29221	0.07987	CL5	CL1	CL0	CL4
14	1.82213	0.48246	-0.05921	0.24923	-0.31685	-0.79151	0.80523	-0.01928	0.94156	3.46436	-0.71126	-0.84637	CL1	CL0	CL0	CL0
15	1.82213	0.48246	-0.05921	0.96082	-0.31685	-0.92104	1.45421	0.44585	-0.60633	1.63088	1.29221	0.7654	CL6	CL0	CL0	CL0
16	1.82213	-0.48246	0.45468	0.96082	-0.31685	-2.05048	-1.50796	-1.55521	-1.07533	1.13407	-0.71126	-0.52593	CL5	CL2	CL2	CL0
17	0.49788	0.48246	-0.61113	0.96082	-0.31685	-1.55078	-0.80615	-1.68062	0.28783	0.7583	-0.21712	-2.07848	CL6	CL0	CL0	CL1
18	1.09449	-0.48246	-1.7379	0.96082	-0.31685	0.52135	-1.23177	-0.31776	-0.45321	-1.38502	-1.37983	-0.84637	CL6	CL1	CL1	CL0
19	1.82213	-0.48246	0.45468	-0.09765	-0.31685	1.37297	-0.15487	-0.17779	-1.92595	-1.5184	-0.71126	-0.21575	CL6	CL2	CL0	CL2
20	0.49788	-0.48246	-0.05921	0.96082	-0.31685	-0.34799	-1.7625	-2.39883	-1.92595	0.7583	-1.37983	-2.07848	CL4	CL1	CL0	CL0
21	1.09449	-0.48246	-0.05921	0.96082	-0.31685	-0.79151	0.80523	0.7233	1.61108	-1.13788	0.19268	-0.21575	CL6	CL1	CL1	CL0
22	2.59171	-0.48246	-2.43591	0.96082	-0.31685	-1.1943	0.47617	-1.11902	-0.60633	1.81175	-0.21712	-1.18084	CL5	CL0	CL0	CL0
23	1.09449	-0.48246	0.45468	0.96082	-0.31685	0.41667	-0.94779	-0.84732	1.11406	-0.89891	-0.71126	0.07987	CL4	CL0	CL0	CL0
24	1.09449	-0.48246	-1.7379	0.96082	-0.31685	1.60383	-3.27393	-1.27553	0.28783	-1.0145	-1.37983	-1.54858	CL6	CL2	CL1	CL5
25	1.82213	-0.48246	0.45468	0.96082	-0.31685	-0.14882	0.63779	1.24033	0.76096	1.46191	-0.21712	-0.52593	CL5	CL1	CL1	CL1
26	1.09449	-0.48246	-0.61113	0.96082	-0.31685	-0.79151	-0.43999	-1.27553	0.94156	-0.00665	-0.21712	0.40148	CL5	CL1	CL2	CL0
27	1.82213	0.48246	-1.22751	0.24923	-0.31685	-0.05188	-1.6334	-3.27393	-0.76096	0.58489	0.19268	-1.54858	CL6	CL0	CL0	CL2

BENZOS	CAFF	CANNABIS	CHOC	COKE	CRACK	ECSTASY	HEROIN	KETAMINE	LEGALH	LSD	METH	MUSHROOMS	NICOTINE	SEMER	VSA
CL2	CL6	CL0	CL5	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL2	CL0	CL0
CL0	CL6	CL4	CL6	CL3	CL0	CL4	CL0	CL2	CL0	CL2	CL3	CL0	CL4	CL0	CL0
CL0	CL6	CL3	CL4	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL1	CL0	CL0	CL0
CL3	CL5	CL2	CL4	CL2	CL0	CL0	CL0	CL2	CL0	CL0	CL0	CL0	CL2	CL0	CL0
CL0	CL6	CL3	CL6	CL0	CL0	CL1	CL0	CL0	CL1	CL0	CL0	CL2	CL2	CL0	CL0
CL0	CL6	CL0	CL4	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL6	CL0	CL0
CL0	CL6	CL1	CL5	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL6	CL0	CL0
CL0	CL6	CL0	CL4	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0
CL0	CL6	CL0	CL6	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL6	CL0	CL0
CL1	CL6	CL1	CL6	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL6	CL0	CL0
CL0	CL6	CL2	CL5	CL2	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL2	CL0	CL1
CL0	CL6	CL4	CL5	CL2	CL0	CL3	CL0	CL0	CL0	CL1	CL0	CL2	CL6	CL0	CL0
CL4	CL6	CL3	CL5	CL1	CL0	CL0	CL0	CL0	CL0	CL1	CL1	CL1	CL6	CL0	CL0
CL0	CL5	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL1	CL0	CL0
CL0	CL6	CL0	CL6	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL6	CL0	CL0
CL0	CL6	CL1	CL5	CL2	CL0	CL1	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0
CL1	CL6	CL3	CL5	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL6	CL0	CL0
CL0	CL6	CL6	CL4	CL1	CL0	CL1	CL0	CL2	CL0	CL1	CL0	CL1	CL6	CL0	CL0
CL2	CL6	CL3	CL6	CL2	CL0	CL2	CL0	CL0	CL2	CL1	CL0	CL1	CL0	CL0	CL0
CL0	CL6	CL1	CL6	CL0	CL0	CL1	CL0	CL0	CL0	CL0	CL6	CL0	CL1	CL0	CL0
CL0	CL6	CL2	CL6	CL0	CL0	CL0	CL0	CL0	CL0	CL1	CL0	CL1	CL3	CL0	CL0
CL0	CL6	CL1	CL6	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL1	CL1	CL0	CL0
CL0	CL5	CL0	CL6	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0
CL5	CL6	CL2	CL6	CL2	CL0	CL0	CL1	CL0	CL0	CL1	CL0	CL0	CL6	CL0	CL1
CL1	CL5	CL1	CL5	CL1	CL0	CL1	CL0	CL0	CL0	CL1	CL0	CL1	CL6	CL0	CL0
CL0	CL6	CL1	CL5	CL2	CL0	CL2	CL0	CL0	CL3	CL2	CL0	CL2	CL1	CL0	CL0
CL2	CL6	CL1	CL6	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL0	CL6	CL0	CL0

Decision Tree

A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.It is one of the predictive modeling approaches used in statistics, data miningand machine learning Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunction of features that lead to those class labels. Decision trees where the target variable can take continuous values (typicallyreal numbers) are called regression trees.



CROSS VALIDATION

Cross validation is a model evaluation method that is better than residuals. The problem with residual evaluations is that they do not give an indication of how well the learner will do when it is asked to make new predictions for data it has not already seen. One way to overcome this problem is to not use the entire data set when training a learner. Some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model on ``new'' data. This is the basic idea for a whole class of model evaluation methods called cross validation.

HOLDOUT METHOD

The holdout method is the simplest kind of cross validation. The data set is separated into two sets, called the training set and the testing set. The function approximator fits a function using the training set only. Then the function approximator is asked to predict the output values for the data in the testing set (it has never seen these output values before). The errors it makes are accumulated as before to give the mean absolute test set error, which is used to evaluate the model. The advantage of this method is that it is usually preferable to the residual method and takes no longer to compute however its evaluation can have a high variance. The evaluation may depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made.

Result Of Decision Tree:

Misclassification Error Rate

Classification tree:
tree(formula = HEROIN ~ ., data = drug2)
Variables actually used in tree construction:
[1] "Result" "CRACK" "LSD" "MUSHROOMS" "BENZOS" "AGE"
[7] "METH"
Number of terminal nodes: 9
Residual mean deviance: 0.5534 = 1038 / 1876
Misclassification error rate: 0.1088 = 205 / 1885

0.8912467, 0.1087533
0.858811, 0.141189

RANDOM FOREST

Random forests are based on a simple idea Aggregate of the results of multiple predictors gives a better prediction than the best individual predictor. A group of predictors is called an ensemble. Thus, this technique is called Ensemble Learning. Random decision forests correct for decision trees' habit of overfitting to their training set.

ntree = 500 & mtry = sqrt(7) Test Error Rate: 0.1295117 and Accuracy: 0.8704883

ntree = 800 & mtry = sqrt(7) Test Error Rate: 0.1284501 and Accuracy: 0.8715499

Bagging

Bootstrap aggregating, also called bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach.

Outputs:

ntree =200 & mtry=7 Test Error Rate: 0.1019108 and Accuracy: 0.8980892

ntree=500 & mtry = 7 Test Error Rate: 0.1061571 and Accuracy: 0.8938429

ntree = 800 & mtry = 7 Test Error Rate: 0.1061271 and Accuracy: 0.8938429

BOOSTING

Boosting means that each tree is dependent on prior trees. The algorithm learns by fitting the residual of the trees that preceded it. Thus, boosting in a decision tree ensemble tends to improve accuracy with some small risk of less coverage.

Outputs:

n.tree=5000 & interception.depth=4 Test Error Rate: 0.9989063

NAÏVE BAYES CLASSIFIER

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

Naive Bayes with 943 items in training set

- **Training set:**

  Correction rate and error rate: **0.8313892 0.1686108**

- **Test set:**

  Correction rate and error rate: **0.7845011 0.2154989**

Naive Bayes with 600 items in training set

- **Training set:**

  Correction rate and error rate: **0.865 0.135**

- **Test set:**

  Correction rate and error rate: **0.814786 0.185214**

Support Vector Machine(SVM)

Support vector machines are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. However, they are mostly used in classification problems.

Kernel trick: The kernel defines the similarity or a distance measure between new data and the support vectors.The dot product is the similarity measure used for linear SVM or a linear kernel because the distance is a linear combination of the inputs.Other kernels can be used that transform the input space into higher dimensions such as a Polynomial Kernel and a Radial Kernel. This is called the Kernel Trick.It is desirable to use more complex kernels as it allows lines to separate the classes that are curved or even more complex. This in turn can lead to more accurate classifiers.

Cost: – The Cost parameter tells the SVM optimization how much you want to avoid misclassifying each training. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

Gamma:-

Gamma is a parameter for nonlinear hyperplane. The higher the gamma value it tries to exactly fit the training data setThe gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The gamma parameters be the inverse of the radius of influence of samples selected by the model as support vectors. If gamma is too large, the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C will be able to prevent overfitting.

When gamma is very small, the model is too constrained and cannot capture the complexity or "shape" of the data. The region of influence of any selected support vector would include the whole training set. The resulting model will behave similarly to a linear model with a set of hyperplanes that separate the centres of high density of any pair of two classes.

Degree:

The degree parameter in used only in polynomial kernel which shows the degree of polynomial used to find the hyperplane to split the data.

SVM Method:-

We must take two data sets i.e. training data set and test data set

1. from the data set we use some data for the test data and remaining for the train data.

2. Build a predictive model using only the training set with the best cost and gamma.

3. Use the test set to compare predicted answers an actual answer.

4. Validate the performance of your predictive model by comparing predictions on test rows.

Outputs:

SVM with 943 items in training set:
Kernel = Linear & Cost= 0.01

Training Error Rate: 0.1410392 & Testing Error Rate: 0.156051

Kernel = Linear & Cost = 0.1

Training Error Rate: 0.09862142 & Testing Error Rate: 0.1464968
Kernel= Radial & Cost = 0.1 & Gamma = 0.2

Training Error Rate: 0.1410392 & Testing Error Rate: 0.156051
Kernel= Radial & Cost = 0.1 & Gamma = 0.1

Training Error Rate: 0.1410392 & Testing Error Rate: 0.156051
Kernel= Radial & Cost = 0.01 & Gamma = 0.1

Training Error Rate: 0.1410392 & Testing Error Rate: 0.156051
Kernel= Polynomial & Cost = 0.1 & Gamma = 3

Training Error Rate: 0 & Testing Error Rate: 0.1507431

Kernel= Polynomial & Cost = 0.1 & Gamma = 1

Training Error Rate: 0 & Testing Error Rate: 0.1507431

Kernel= Polynomial & Cost = 0.01 & Gamma = 1

Training Error Rate: 0 & Testing Error Rate: 0.1507431

Kernel= Polynomial & Cost = 1 & Gamma = 0.1

Training Error Rate: 0 & Testing Error Rate: 0.1507431

COMPARISON OF CLASSIFICATION METHODS

- SVM linear test error rate is 14%
- SVM radial test error rate is 15%
- SVM polynomial test error rate is 17%
- Naive Bayes test error rate is 18%
- Boosting test error rate is 99%
- Random Forest test error rate is 12%
- Bagging test error rate is 10%
- Holdout Method error rate is 10%

Potential Performance Issues and Future Study

The Scope and Research of this project is only limited in finding out the accuracy and error rates of each classifier and according to the Accuracy measurement, Bagging and Hold-out method after pruning demonstrated the best fit for this chess data set. There are many other comparison techniques and tools to analyse the Performance of the Classification which not only measures the Accuracy, but also measurement on Precision, F_score, time-complexity, ROC curve (measuring true-positives and false-positives) which opens the gate for the future study.

CONCLUSION

In this project, we have performed many classification techniques such as Decision tree hold out, Random forest, Bagging, Naïve Bayes and Support vector machine. We have analyzed the accuracy and error rates obtained from the techniques. Among all the techniques Holdout and Bagging methods are the best classification for our data set which provide less error rate of 10% and accuracy of 90%, When compared to other classifiers.

REFERENCES

[1] https://archive.ics.uci.edu/ml/datasets/Drug+consumption+%28quantified%29.

[2]https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms934a444fca47

[3]https://towardsdatascience.com/decision-trees-and-random-forestdf0c3123f991

[4]https://towardsdatascience.com/introduction-to-naive-bayes-classification-4cffabb1ae5

[5]https://www.cs.cmu.edu/~schneide/tut5/node42.html

[6]https://www.youtube.com/watch?v=jtGVYnv91ps&t=17s

APPENDIX

```
drug=read.table('C:/Users/dines/OneDrive/Desktop/drug_consumption.data',sep=',',header=F,stringsAsFactors = TRUE, na.strings ="?")

fix(drug)

colnames(drug) <-
c("ID","AGE","GENDER","EDUCATION","COUNTRY","ETHNICITY","NSCORE","ESCORE","OSCORE","ASCORE","CSCORE","IMPULSIVE","SS","ALCOHOL","AMPHET","AMLY","BENZOS","CAFF","CANNABIS","CHOC","COKE","CRACK","ECSTASY","HEROIN","KETAMINE","LEGALH","LSD","METH","MUSHROOMS","NICOTINE","SEMER","VSA")

fix(drug)

drug=na.omit(drug)

library(tree)

attach(drug)

Result=ifelse((HEROIN)=="CL0","Safe",ifelse((HEROIN)=="CL1","Safe",ifelse((HEROIN)=="CL2","Safe","Unsafe")))

drug=data.frame(drug,Result)

dim(drug)

fix(drug)

attach(drug)
```

#Hold-out Method

```
library(tree)

tree.drug =  tree(HEROIN~.,drug)

summary(tree.drug)
```

```
plot(tree.drug)

text(tree.drug,pretty=0)

tree.drug

tree.pred=predict(tree.drug,type="class")

table(tree.pred,HEROIN)

mean(tree.pred==HEROIN)

mean(tree.pred!=HEROIN)

set.seed(123)

training = sample(1:nrow(drug), 943)

drug.test = drug[-training,]

HEROIN.test=HEROIN[-training]

tree.drug=tree(HEROIN~.,drug,subset=training)

tree.pred=predict(tree.drug,drug.test,type="class")

table(tree.pred,HEROIN.test)

mean(tree.pred==HEROIN.test)

mean(tree.pred!=HEROIN.test)
```

#Random Forest and Bagging

```
library(randomForest)

set.seed(123)

tree.drug=randomForest(HEROIN~.,drug,subset=training,ntree=200,mtry=32)

tree.drug
```

```
tree.pred=predict(tree.drug,drug.test,type="class")

table(tree.pred,HEROIN.test)

mean(tree.pred==HEROIN.test)

mean(tree.pred!=HEROIN.test)

set.seed(123)

tree.drug=randomForest(HEROIN~.,drug,subset=training,ntree=500,mtry=7)

tree.drug

tree.pred=predict(tree.drug,drug.test,type="class")

table(tree.pred,HEROIN.test)

mean(tree.pred==HEROIN.test)

mean(tree.pred!=HEROIN.test)

set.seed(123)

tree.drug=randomForest(HEROIN~.,drug,subset=training,ntree=800,mtry=7)

tree.drug

tree.pred=predict(tree.drug,drug.test,type="class")

table(tree.pred,HEROIN.test)

mean(tree.pred==HEROIN.test)

mean(tree.pred!=HEROIN.test)

set.seed(123)

tree.drug=randomForest(HEROIN~.,drug,subset=training,ntree=500,mtry=sqrt(7))

tree.drug
```

```
tree.pred=predict(tree.drug,drug.test,type="class")

table(tree.pred,HEROIN.test)

mean(tree.pred==HEROIN.test)

mean(tree.pred!=HEROIN.test)

set.seed(123)

tree.drug=randomForest(HEROIN~.,drug,subset=training,ntree=800,mtry=sqrt(7))

tree.drug

tree.pred=predict(tree.drug,drug.test,type="class")

table(tree.pred,HEROIN.test)

mean(tree.pred==HEROIN.test)

mean(tree.pred!=HEROIN.test)

importance(tree.drug)

varImpPlot(tree.drug)
```

#Boosting

```
library(gbm)

drug$HEROIN=ifelse(drug$Result=="Safe",1,0)

set.seed(123)

training = sample(1:nrow(drug), 943)

testing = drug[-training,]

HEROIN.test=HEROIN[-training]
```

```
drug.train = drug[training,]

drug.test = drug[-training,]

boost.drug=gbm(HEROIN~.,drug[training, ],distribution="gaussian",n.tree=5000,interaction.depth=4)

yhat.boost=predict(boost.drug, drug[-training,], n.tree=5000, type = "response", interception.depth=4)

mean(yhat.boost !=drug.test)
```

#Naive Bayes

```
drug=read.table('C:/Users/dines/OneDrive/Desktop/drug_consumption.data',sep=',',header=F,stringsAsFactors = TRUE, na.strings ="?")

fix(drug)

colnames(drug) <-
c("ID","AGE","GENDER","EDUCATION","COUNTRY","ETHNICITY","NSCORE","ESCORE","OSCORE","ASCORE","CSCORE","IMPULSIVE","SS","ALCOHOL","AMPHET","AMLY","BENZOS","CAFF","CANNABIS","CHOC","COKE","CRACK","ECSTASY","HEROIN","KETAMINE","LEGALH","LSD","METH","MUSHROOMS","NICOTINE","SEMER","VSA")

fix(drug)

drug=na.omit(drug)

library(tree)

attach(drug)

Result=ifelse((HEROIN)=="CL0",
"Safe",ifelse((HEROIN)=="CL1","Safe",ifelse((HEROIN)=="CL2","Safe","Unsafe")))

drug=data.frame(drug,Result)

dim(drug)

fix(drug)
```

```
attach(drug)

set.seed(123)

train=sample(1:nrow(drug), 943)

training = drug[train,]

testing = drug[-train,]

fix(training)

fix(testing)

library(e1071)

Naive_Bayes_Model=naiveBayes(HEROIN~., data=training)

summary(Naive_Bayes_Model)

tr.pred = predict(Naive_Bayes_Model,training)

tr.table = table(tr.pred,Truth = training$HEROIN)

mean(tr.pred== training$HEROIN)

mean(tr.pred!= training$HEROIN)

ts.pred = predict(Naive_Bayes_Model,testing)

ts.table = table(ts.pred,Truth = testing$HEROIN)

mean(ts.pred== testing$HEROIN)

mean(ts.pred!= testing$HEROIN)
```

#SVM

```
library(tree)

library(e1071)

library(caret)

drug=read.table('C:/Users/dines/OneDrive/Desktop/drug_consumption.data',sep=',',header=F,stringsAsFactors = TRUE, na.strings ="?")

fix(drug)

colnames(drug) <-
c("ID","AGE","GENDER","EDUCATION","COUNTRY","ETHNICITY","NSCORE","ESCORE","OSCORE","ASCORE","CSCORE","IMPULSIVE","SS","ALCOHOL","AMPHET","AMLY","BENZOS","CAFF","CANNABIS","CHOC","COKE","CRACK","ECSTASY","HEROIN","KETAMINE","LEGALH","LSD","METH","MUSHROOMS","NICOTINE","SEMER","VSA")

drug= na.omit(drug)

attach(drug)

training = sample(1:nrow(drug), 943)

HEROIN.train=HEROIN[training]

HEROIN.test=HEROIN[-training]

drug.train = drug[training,]

drug.test = drug[-training,]

svmfit=svm(HEROIN~., data=drug.train, kernel="linear", cost=0.01)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)
```

```
tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="linear", cost=0.1)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)

tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="linear", cost=1)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)

tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)
```

```
mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="linear", cost=10)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)

tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="linear", cost=100)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)

tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="radial", cost=0.1,gamma=0.2)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)
```

```
tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="radial", cost=0.1,gamma=0.1)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)

tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="radial", cost=0.01,gamma=0.1)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)

tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)
```

```
mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="polynomial", cost=0.01,gamma=3)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)

tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="polynomial", cost=0.01,gamma=2)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)

tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="polynomial", cost=0.01,gamma=1)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)
```

```
tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="polynomial", cost=0.1,gamma=1)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)

tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="polynomial", cost=0.1,gamma=2)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)

tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)
```

```
mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="polynomial", cost=0.1,gamma=3)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)

tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="polynomial", cost=1,gamma=1)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)

tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="polynomial", cost=1,gamma=2)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)
```

```
tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)

svmfit=svm(HEROIN~., data=drug.train, kernel="polynomial", cost=1,gamma=3)

summary(svmfit)

tr.pred = predict(svmfit,drug.train)

tr.table = table(tr.pred, truth=drug.train$HEROIN)

mean(tr.pred!=  drug.train$HEROIN)

ts.pred = predict(svmfit,drug.test)

ts.table = table(ts.pred, truth = drug.test$HEROIN)

mean(ts.pred!= drug.test$HEROIN)
```