CMPUT 566: Introduction to Machine Learning

Car Evaluation Classification: Project Report

Islam A. Ali
Student ID: 1633813
iaali@ualberta.ca

Abstract

Data classification is considered a very important corner stone in supervised learning and in machine learning in general. That's due to the wide spectrum of problems in which intelligent classification is needed, and also the available algorithms that proven proper performance when tested. In this report, we explore different ML classification algorithms in order to evaluate acceptability of cars using a publicly available car evaluation data set. The algorithms tested include: Naive Bayes, KNN, Decision Trees, Random Forest, and SVM. The implementation was done using Python3, numpy, pandas, and skLearn libraries. Finally, The algorithms are compared to each other based on the accuracy as the main measure of success.

1 Problem Formulation

The problem at hand is classification problem with labeled data set with an objective to correctly determine the level of acceptability of car based on a number of provided features. The following subsections show more information about the data set, validation mechanism, and also the performance metrics measured when executing classification with different algorithms.

1.1 Selected Dataset

Car Evaluation Data Set [1] was utilized in this project to train ML models and to test their performances. The data set consists of 1728 labeled samples with six different categorical features, as well as a categorical label. The following table illustrates of the available features, its semantic meaning and their values range.

Table 1: Car Evaluation Data set Details

Data Set Field	**Type**	**Definition**	**Range**
buying	Feature	Buying price	{ v-high, high, med, low }
maint	Feature	Maintenance price	{ v-high, high, med, low }
doors	Feature	Number of doors	{ 2, 3, 4, 5-more }
persons	Feature	Capacity in terms of persons to carry	{ 2, 4, more }
lug_boot	Feature	The size of the luggage boot	{ small, med, big }
safety	Feature	Estimated safety of the car	{ low, med, high }
car_class	Label	Car acceptability	{ unacc, acc, good, v-good }

1.2 Training-Validation-Testing Framework

The training-validation-testing framework is adopted in this work in order to have a concrete and realistic measure of algorithm's success. For that, k-fold with a choice of 5 folds approach was utilized due to the size of the data set which does not exceed ~1800 samples. Other approaches such as leave-out validation may not be suitable for small data sets and more applicable with larger and more generic data sets.

[1]The data set is available at: https://archive.ics.uci.edu/ml/datasets/Car+Evaluation

1.3 Measure of Success

The main measure of success is the classification accuracy given by:

$$Accuracy = \frac{\sum_{m=1}^{M} \mathbb{1}\{\hat{t} = t\}}{M} \tag{1}$$

Other performance metrics are also reported and used in the comparison such that, precision, recall, F1-score, and confusion matrix.

2 Data Set Representation

The categorical values provided in the original data set is not suitable for usage with different machine learning algorithms inside the sklearn library. For this reason, this categorical values require transformation into numerical data to become compatible with the library's algorithms. Two different representations of features are applicable: the first one is what we call decimal representation, and the second one is binary representation. In the following subsections, both representation are illustrated. Moreover, both representations were tested and the corresponding performance was reported with an aim to provide an analysis of the impact of features representation on the classification accuracy.

2.1 Decimal Features Representation

The first representation is converting each data feature field into a set of decimal values corresponding to the categorical values, which means that the resulting feature vector will have the same size as the data set attributes size which is 1×6 for each data sample.

Table 2: Decimal Features Mapping

Data Set Field	**Numerical Range**	**Categorical Range**
buying	{ v-high, high, med, low }	{ 5, 4, 3, 1 }
maint	{ v-high, high, med, low }	{ 5, 4, 3, 1 }
doors	{ 2, 3, 4, 5-more }	{ 2, 3, 4, 5 }
persons	{ 2, 4, more }	{ 2, 4, 5 }
lug_boot	{ small, med, big }	{ 1, 3, 5 }
safety	{ low, med, high }	{ 1, 3, 5 }

2.2 Binary Features Representation

The second representation is representing features by values indicators which increases the size of the input features vector. This service is provided by the $get_dummies$ function available in the *pandas* package in Python3. The size of the input feature vector will become 1×21 for each data sample. The increase in the size of the feature vector is beneficial for some classification algorithms such as the naive bayes classification.

2.3 Label Representation

The car label must be a single value for each data sample, which follows the definition of the decimal feature representation. The following is the mapping used for the output label from the categorical space to the numerical space. The following table shows how mapping is done for the car acceptability output labels.

Table 3: Decimal Labels Mapping

Data Set Field	**Categorical Range**	**Categorical Range**
car_class	{ unacc, acc, good, v-good }	{ 1, 2, 3, 4 }

3 Classification Baseline

The classification base line is defined to classifying all samples as being the dominating class. The following graph provide the frequency of each car label.

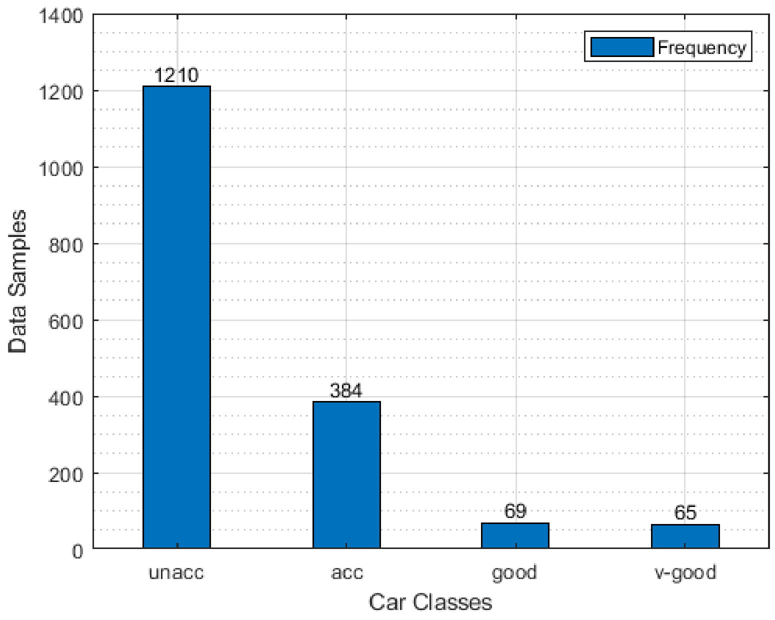


Figure 1: Frequency of different car classes in the data set

The graph shows that the dominating class is ”unacc” with a frequency of 70.023%, meaning that classifying all samples to belong to ”unacc” class will result in an accuracy of 70.023%, which is considered the base line accuracy that should be enhanced or surpassed by the usage of ML classification algorithms. It also shows that the data is highly skewed towards the ”unacc” class, which makes it more challenging to have a generalized model with high accuracy. It also adds another motivation for the usage of k-fold validation to ensure generalization.

4 Experimented Classification Algorithms

4.1 Naive Bayes

Naive Bayes is a simple probabilistic classifier, that depends on applying Bayes theorem with a naive features' independence assumption. This assumption may not hold in a wide range of cases as this kind of correlation may exist and may have a strong impact on the quality of classification. The classifier also makes benefit from the extended number of features and may have a performance improvement with larger feature vector size.

Hyper-parameters

Hyperparameter	**Definition**	**Tuning Technique**
alpha	Additive smoothing to categorical data	Exhaustive search from 0 to 1 with 0.1 step size

4.2 K-Nearest Neighbors (KNN)

KNN is a classification algorithm that depends on assign the class value of a certain data sample to be the result of a majority vote of it K-nearest neighbors. The value of K is determined by the training phase and picked to maximize the validation accuracy. In case of having k=1, the classification is determined based on the single nearest sample to the unknown sample.

Hyper-parameters

Hyperparameter	Definition	Tuning Technique
n_neighbors	Number of neighbors to use for classification	Exhaustive search from 1 to 30 with 1 step size

4.3 Decision Tree

The decision tree is a classification algorithms that depends on having a tree-like structure where nodes represent a certain feature and its outward edges represent decisions made based on the feature value. The leaf nodes (which has no outward edges) on the other hand, represent the outcome of the decision tree after passing through multiple checks on the way. The training procedure tries to find out the optimal structure for this tree, the split locations, and features choices that can maximize the validation accuracy.

Hyper-parameters

Hyperparameter	Definition	Tuning Technique
criterion	The function used to measure the quality of the split	Random search in the range {'gini', 'entropy'}
max_depth	The maximum depth of the tree	Random search in the range {None, 1, 5, 10, 30, 50, 100, 1000}
min_samples_split	The minimum number of samples needed to split a node	Random search in the range from 2 to 10 with an step = 1
min_samples_leaf	The minimum samples required to form a leaf node	Random search in the range from 1 to 10 with a step = 1

4.4 Random Forest

Random forest is another classification algorithm that depends heavily on the definition we made for the decision tree. That's due to the fact that it is an ensemble classifier that has multiple decision trees running at the same time, and the output should be the majority vote for these estimators. The parameters are very similar to that of the decision tree with few extra ones to control the forest structure. A good point to mention here, is that in order to fairly evaluate the random forest performance against the decision tree, one shall use the same set of common parameters in both to ensure that the only difference is the forest structure not the trees themselves.

Hyper-parameters

Hyperparameter	Definition	Tuning Technique
criterion	The function used to measure the quality of the split	Random search in the range {'gini', 'entropy'}
max_depth	The maximum depth of the tree	Random search in the range {None, 1, 5, 10, 30, 50, 100, 300, 500, 1000}
n_estimators	The number of trees to present in the forest	Random search in the range {10, 30, 50, 100, 500, 1000}
max_features	The number of features to consider when splitting	Random search in the range {'auto', 'sqrt', 'log2', None}
min_samples_split	The minimum number of samples needed to split a node	Random search in the range from 2 to 10 with an step = 1
min_samples_leaf	The minimum samples required to form a leaf node	Random search in the range from 1 to 10 with a step = 1

4.5 Support Vector Machines (SVM)

SVM is a classification algorithm that aims at finding the best hyperplane to separate two classes with the maximum margin between the classes available in the data set. In case of having more than two classes, SVM uses the strategy of one-vs-all in order to classify the correct class. Hyperplane is mainly the decision boundary between classes while the data points located at either sides of the decision boundary is called the support vectors due to the fact that the algorithm depends on these samples to decide on the best hyperplane.

Hyper-parameters

Hyperparameter	Definition	Tuning Technique
C	Regularization parameter, strictly positive	Random search in the range {0.001, 0.003, 0.005, 0.01, 0.03, 0.05, 0.1, 0.3, 0.5, 1, 3, 5, 10, 30, 50, 100}
kernel	The kernel to be used with SVM	Random search in the range {'linear', 'poly', 'rbf', 'sigmoid'}
degree	Polynomial degree for 'poly' kernel	Random search in the range from 1 to 10 with a step = 1
gamma	Kernel coefficient for 'poly', 'rbf', 'sigmoid'	Random search in the range {'scale', 'auto', 0.001, 0.003, 0.005, 0.01, 0.03, 0.05, 0.1, 0.3, 0.5, 1, 3, 5, 10, 30, 50, 100}

5 Results and Discussion

In this section, the experimental results of the 5 algorithms are provided with discussion of the findings. The results include reporting of the accuracy, precision, recall, F1-measure, and the confusion matrix.

5.1 Accuracy

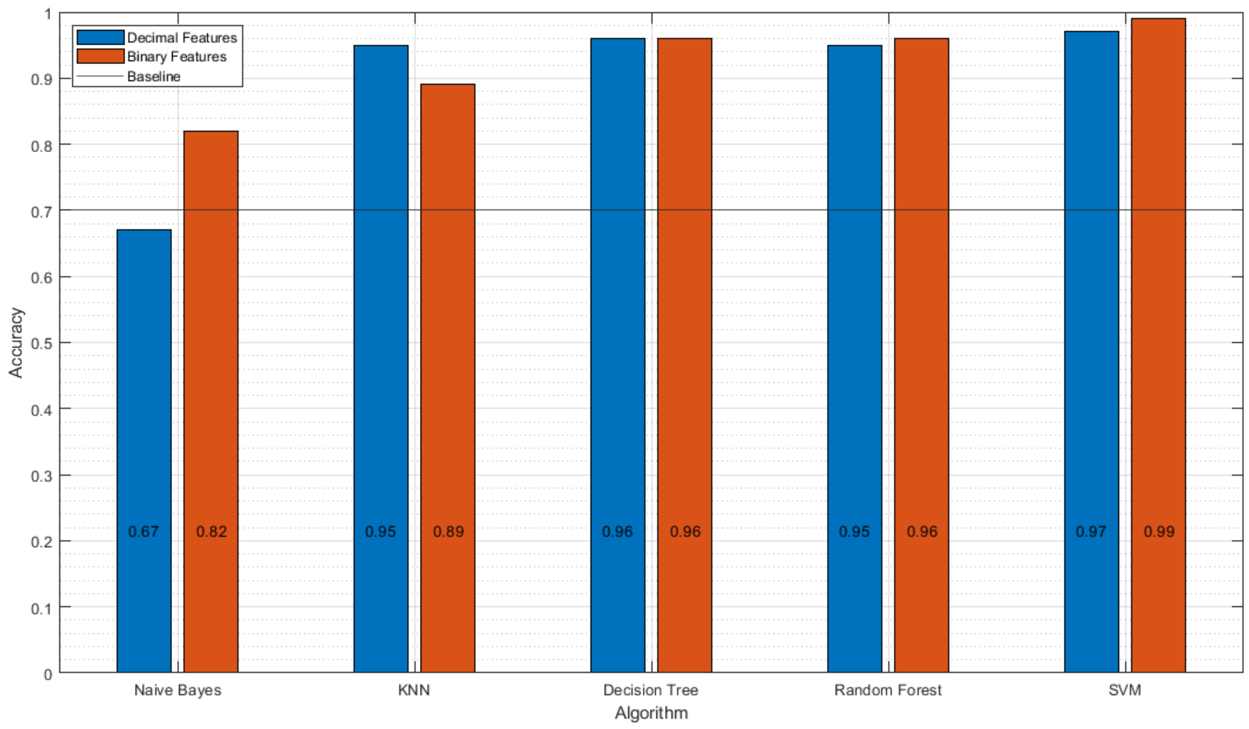


Figure 2: Classification Accuracy in case of Decimal and Binary Features

5.2 Precision

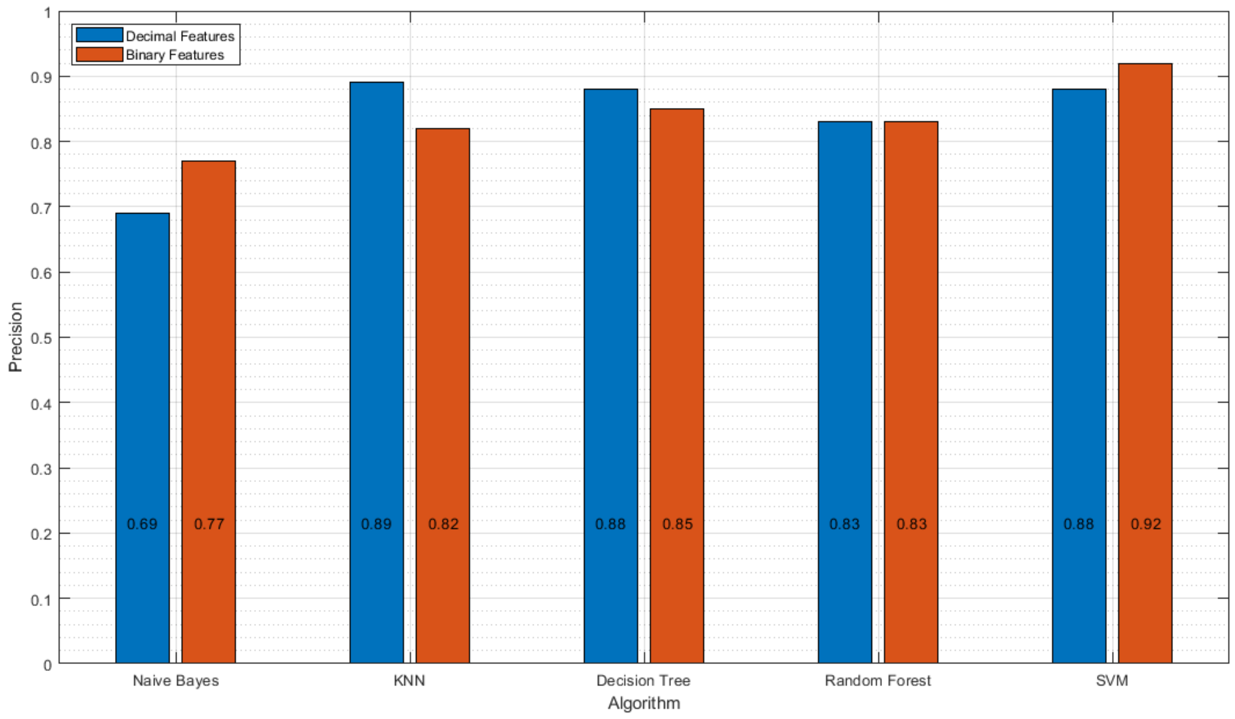


Figure 3: Classification Accuracy in case of Decimal and Binary Features

5.3 Recall

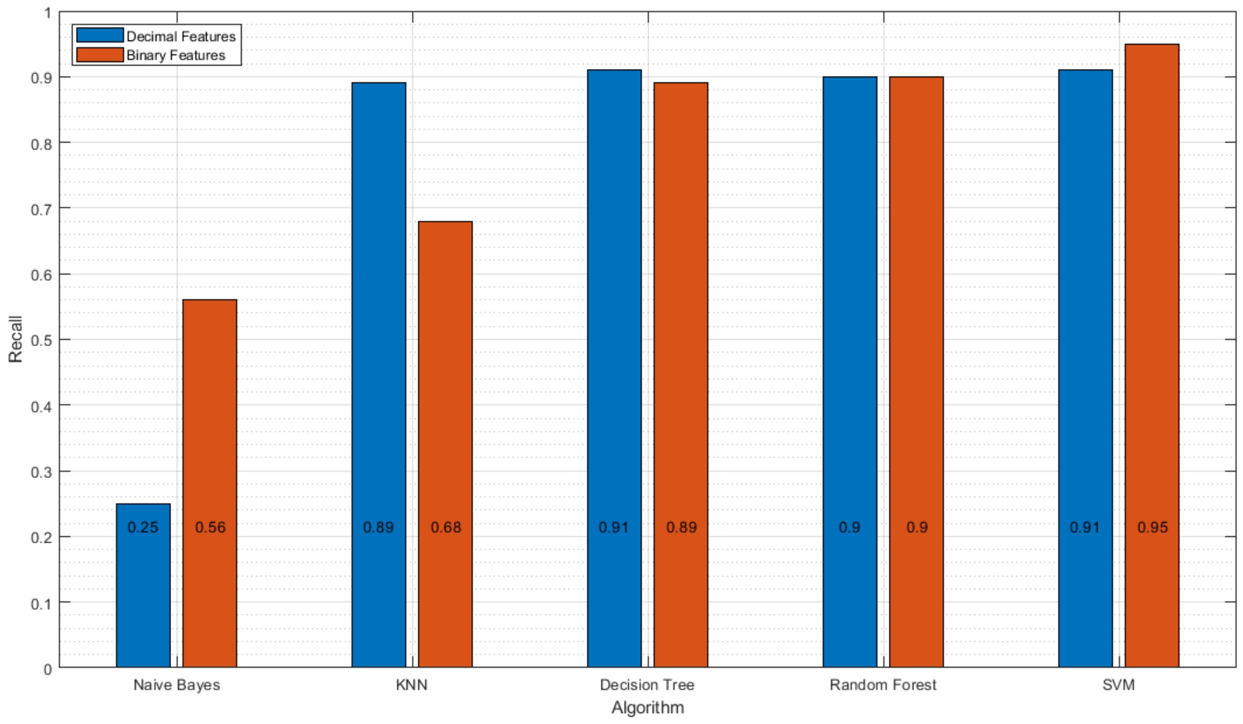


Figure 4: Classification Accuracy in case of Decimal and Binary Features

5.4 F1-Measure

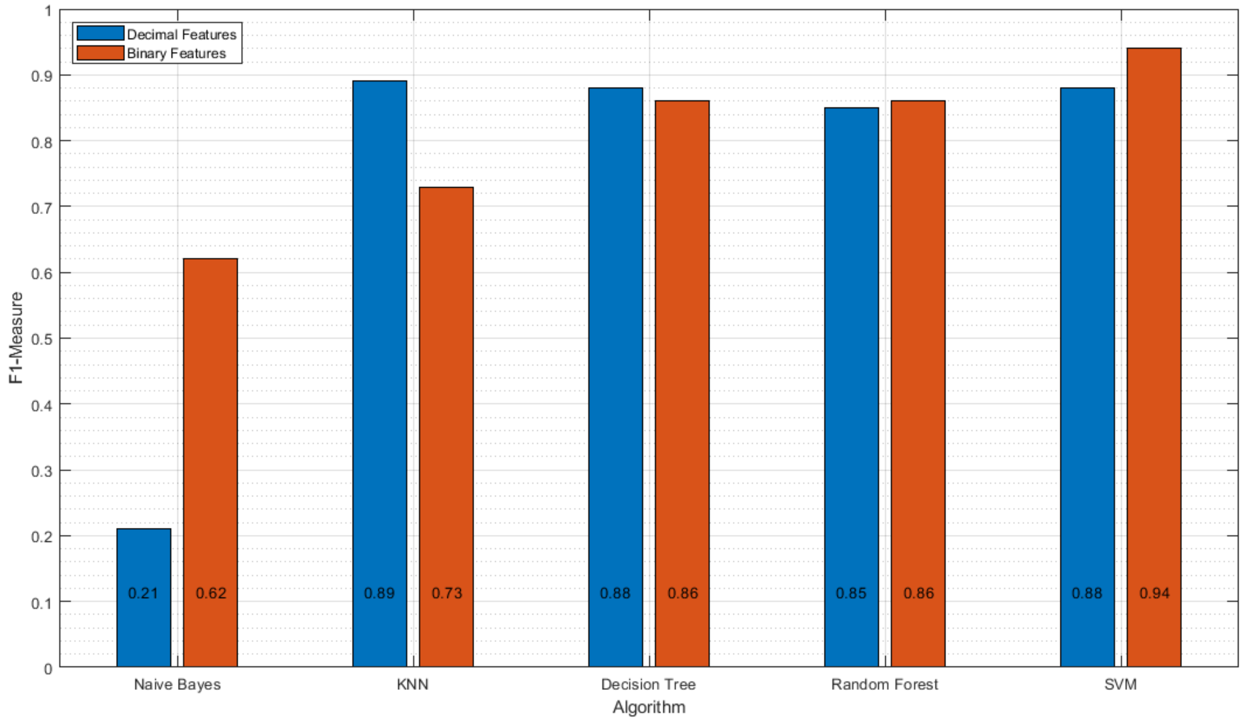
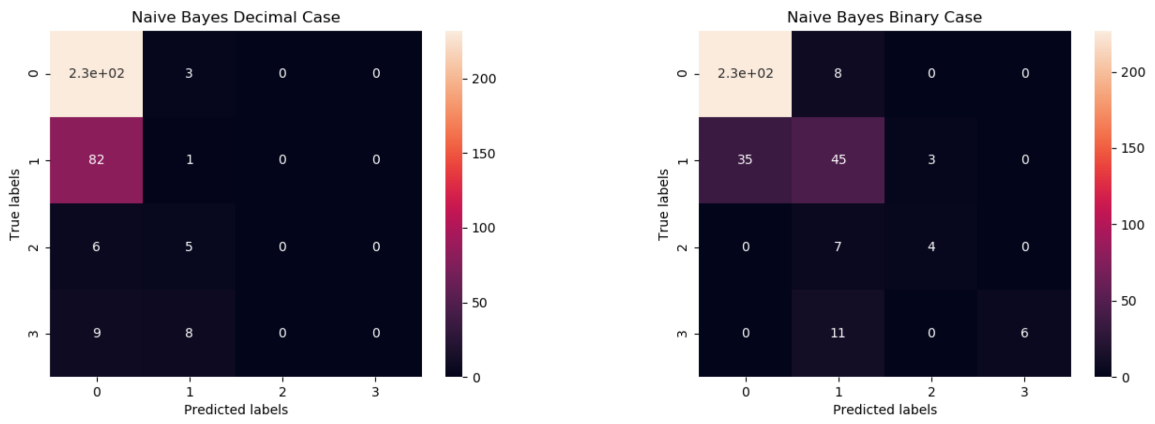


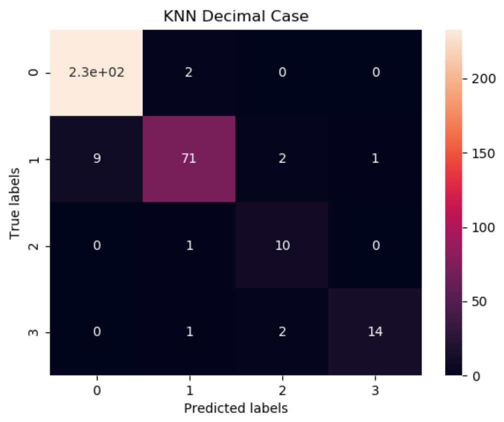
Figure 5: Classification Accuracy in case of Decimal and Binary Features

5.5 Confusion Matrix

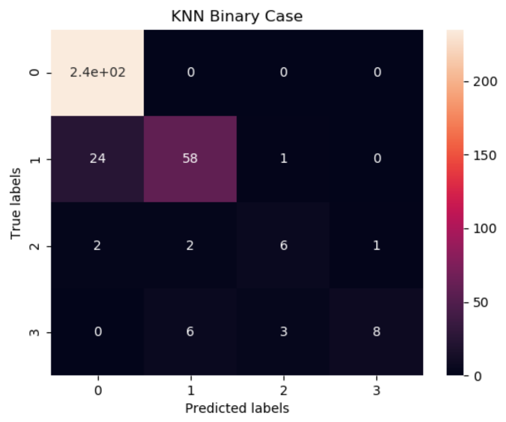


(a) Decimal Features (b) Binary Features

Figure 6: Naive Bayes Confusion Matrix

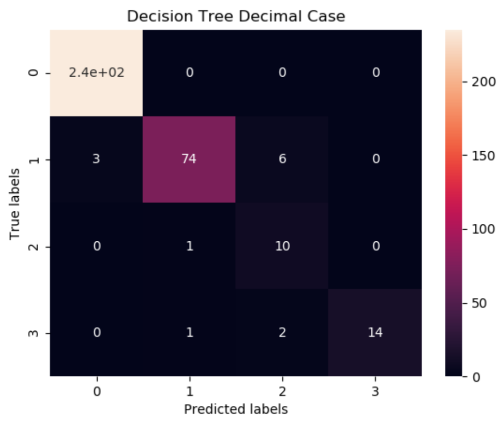


(a) Decimal Features

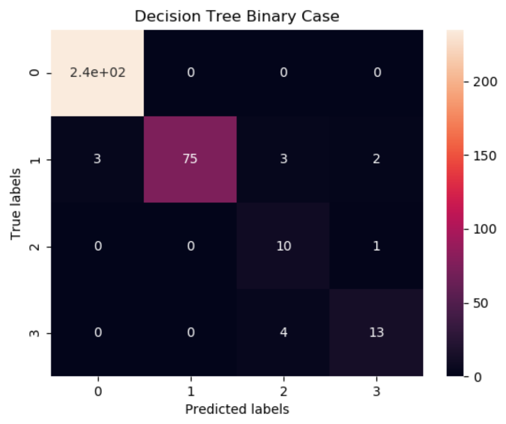


(b) Binary Features

Figure 7: KNN Confusion Matrix

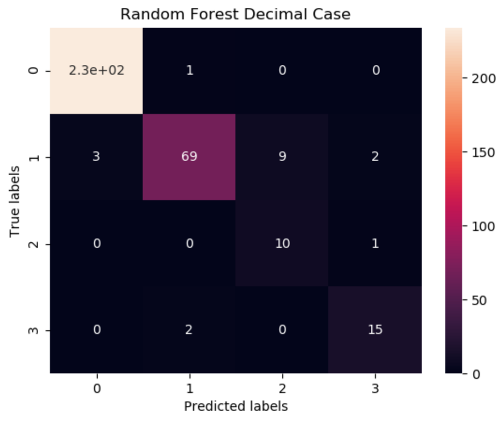


(a) Decimal Features

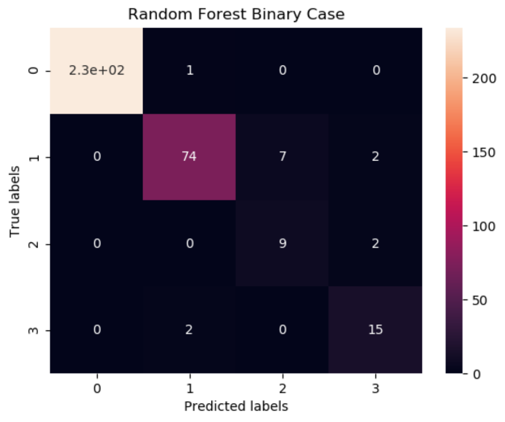


(b) Binary Features

Figure 8: Decision Tree Confusion Matrix

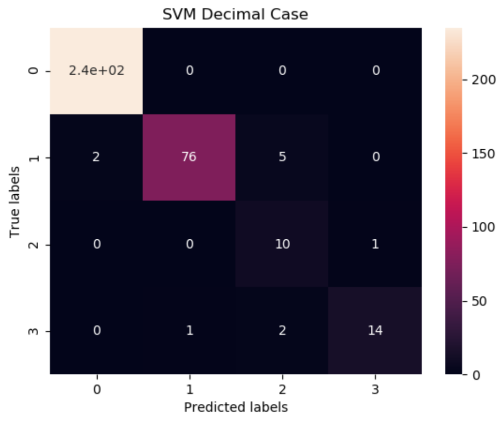


(a) Decimal Features

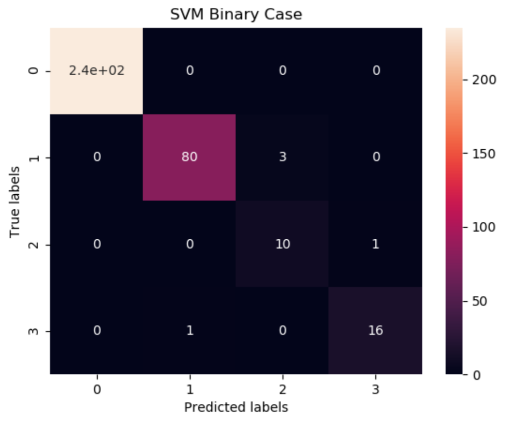


(b) Binary Features

Figure 9: Random Forest Confusion Matrix



(a) Decimal Features



(b) Binary Features

Figure 10: SVM Confusion Matrix

6 Conclusion