

CSE 477(1) Project

Comparative analysis of different classification algorithms

Submitted by:Md. Ariful Islam (2016-1-60-019)

1. Introduction

In this project I have implemented five classification algorithms on the cars selection database found at <https://www.mldata.io/download-csv-weka/cars/> .All algorithms has been implemented using python.

The implemented algorithms are Naïve Bayesian Classifier, k Nearest-Neighbor Classifier, Decision Tree Classifier, Random Forest Classifier and Support Vector Machines. The Naïve Bayesian Classifier has been implemented from scratch without using any library directly related to the algorithm. Only **csv** and **random** these two libraries has been used to read CSV file and shuffling tuples respectively. All other algorithms has been implemented with appropriate python libraries. These five algorithms has been implemented in Weka too.

2. Classification Algorithms

**Naïve Bayesian classifier** classifies an object based on conditional and prior probabilities calculated during training phase.

For a class, **posterior probability or likelihood =conditional probability \* prior probability.**

A tuple belongs to the class that has highest likelihood according to the formula above.

**K Nearest Neighbor** is a lazy learning algorithm because all the computation happens in the testing phase during prediction. A given tuple is compared with most similar k tuples and the class that occurs most of the time among those k tuples is considered to be the class that the given tuple belongs to.

**Decision tree algorithm** defines a tree like structure in which the leafs are the destination classes and all intermediate nodes works as condition checkers that checks the value of different attributes and decides which node will be activated next to check other conditions until leaf node is reached. A leaf node defines the class a tuple belongs to.

**Random forest** is an improved version of decision tree .It uses ensemble methods to lower error and take better decision. In random forest multiple decision trees built from different samples of size N, created by bagging with replacement method is used where N is the size of dataset. The class predicted most of the time by the trees in the random forest is considered to be the final class.

**Support Vector Machine** classifies a tuple based on the hyperplane formulated using the support vectors found during training phase. It can classify both linearly separable and inseparable data. For linearly inseparable data different kernels like polynomial, Gaussian rbf , Sigmoid etc. are used which essentially maps the training objects to higher dimensional plane and finds hyperplane in that space. Only support vectors define the characteristic of dataset so SVM is less prone to overfitting.

3. Dataset

The dataset that I have worked on can be found at <https://www.mldata.io/download-csv-weka/cars/> .The dataset has 1728 tuples, 6 attributes column and a class label column. All attributes are of string type and there is no missing values.The attribute and class labels with their distinct values are given below-

Attribute columns –

**buying**- vhigh, high, med, low

**maint**- vhigh, high, med, low

**doors**- two,three,four,5more

**persons**- two, four, more

**lug\_boot**- small, med, big

**safety**- low, med, high

Class label columns-

**car**- unacc, acc, vgood, good

As no values are missing it was not necessary to preprocess the data during implementation of naïve Bayesian classifier, but for all other algorithms I encoded the attribute values into numerical form.

4. Implementation.

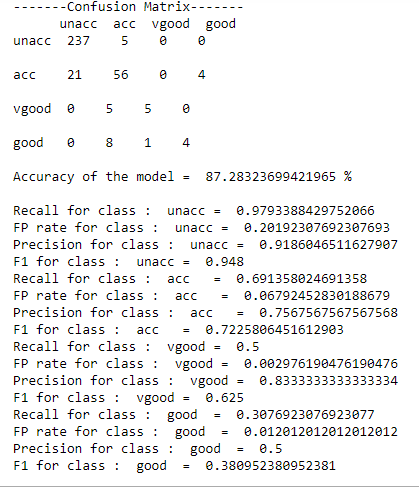
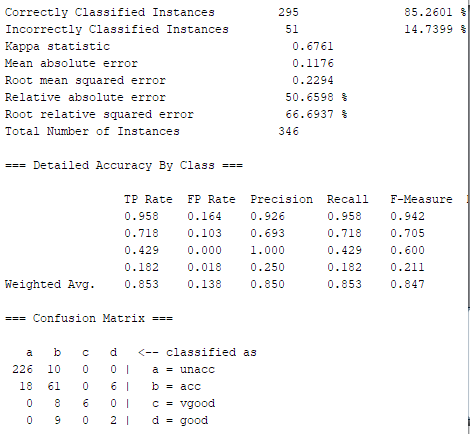
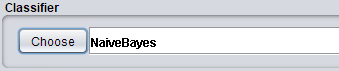
All experiments has been conducted using a PC with AMD A4-6210 processor running at 1.8GHz, 8GB RAM and 500 GB HDD.

Figure 2 Naive Bayesian Classifier in Python

Figure 1 Naive Bayesian Classifier in Weka

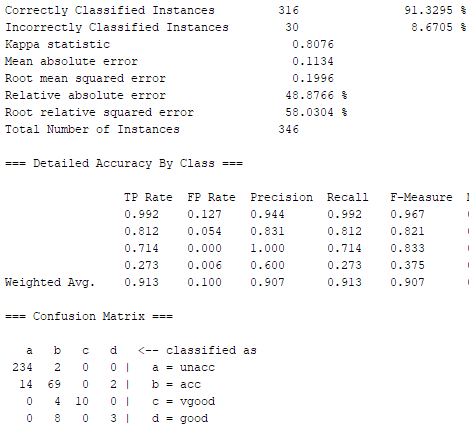
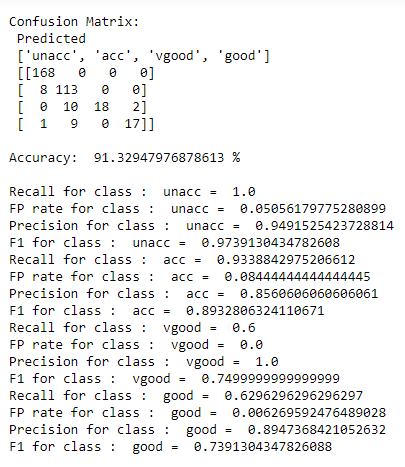
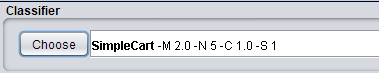
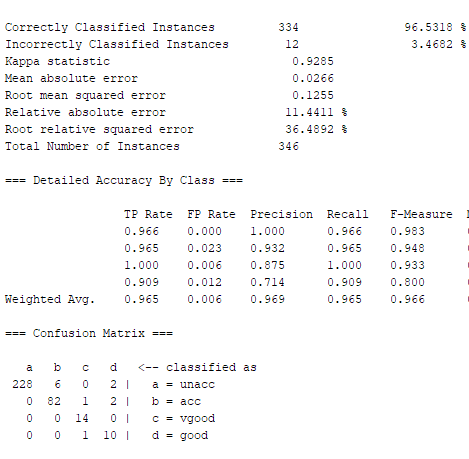
 

Figure 4 KNN (K=5) in Python

Figure 3 KNN( K=5) in Weka



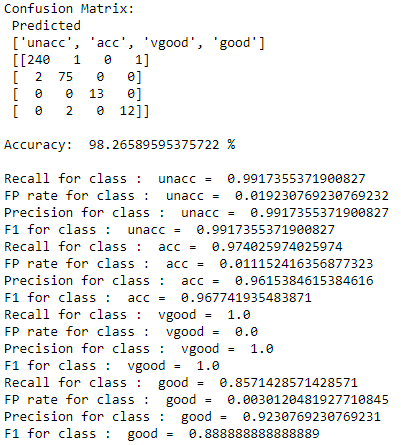


Figure 5 Decision Tree (CART) in Weka

Figure 6 Decision Tree (CART) in Python

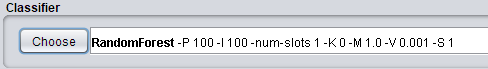
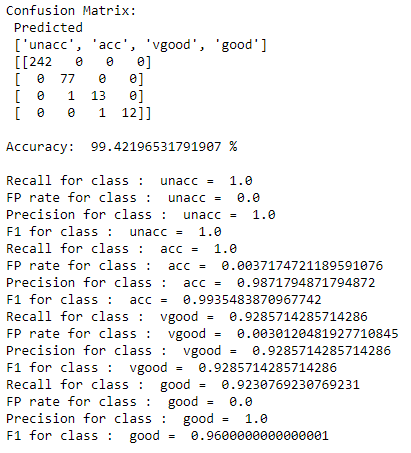
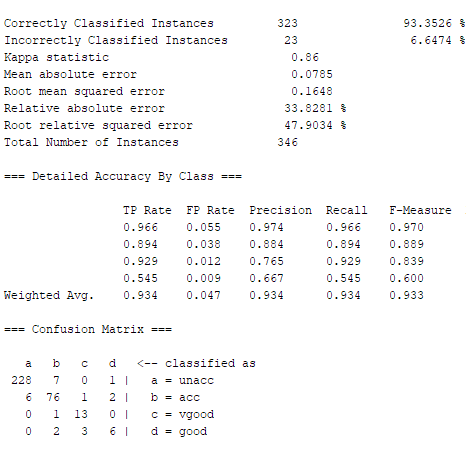
 

Figure 8 Random forest in Python

Figure 7 Random forest in Weka

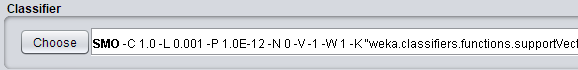
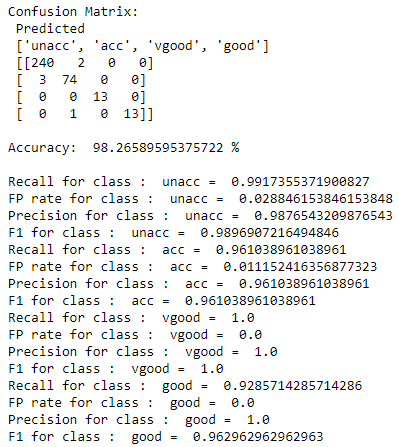
 

Figure 10 SVM in python

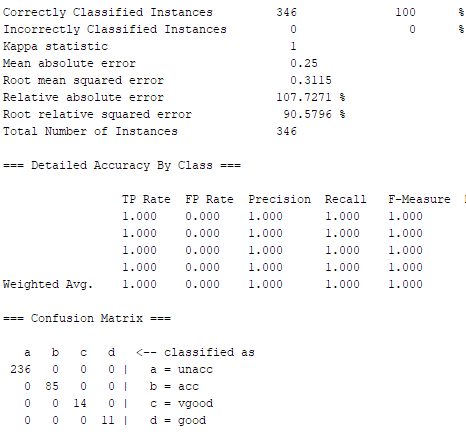


Figure 9 SVM in Weka

5. Performance Evaluation

**Naïve Bayesian Classifier:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Implemented using** | **Accuracy (%)** | **Training Time (s)** | **Testing Time (s)** |
| Weka | 85.26 | 0.01 | 0.01 |
| Python | 84.39 | 0.012 | 0.016 |

F1-Score:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Implemented using** | **Class Label** | | | |
| unacc | acc | vgood | good |
| Weka | 0.942 | 0.718 | 0.429 | 0.182 |
| Python | 0.921 | 0.662 | 0.761 | 0.375 |

**KNN Classifier:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Implemented using** | **Accuracy (%)** | **Training Time (s)** | **Testing Time (s)** |
| Weka | 91.32 | 0.01 | 0.11 |
| Python | 91.32 | 0.009 | 0.061 |

F1-Score:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Implemented using** | **Class Label** | | | |
| unacc | acc | vgood | good |
| Weka | 0.967 | 0.821 | 0833 | 0.375 |
| Python | 0.974 | 0.893 | 0.759 | 0.74 |

**Decision Tree Classifier:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Implemented using** | **Accuracy (%)** | **Training Time (s)** | **Testing Time (s)** |
| Weka | 96.53 | 0.42 | 0 |
| Python | 98.26 | 0.005 | 0.002 |

F1-Score:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Implemented using** | **Class Label** | | | |
| unacc | acc | vgood | good |
| Weka | 0.983 | 0.948 | 0.933 | 0.800 |
| Python | 0.992 | 0.968 | 1 | 0.88 |

**Random Forest Classifier:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Implemented using** | **Accuracy (%)** | **Training Time (s)** | **Testing Time (s)** |
| Weka | 93.35 | 0.24 | 0.04 |
| Python | 97.12 | 0.43 | 0.045 |

F1-Score:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Implemented using** | **Class Label** | | | |
| unacc | acc | vgood | good |
| Weka | 0.970 | 0.889 | 0.839 | 0.600 |
| Python | 0.992 | 0.95 | 0.87 | 0.83 |

**SVM:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Implemented using** | **Accuracy (%)** | **Training Time (s)** | **Testing Time (s)** |
| Weka | 100 | 5.46 | 0.07 |
| Python | 98.27 | 0.22 | 0.0063 |

F1-Score:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Implemented using** | **Class Label** | | | |
| unacc | acc | vgood | good |
| Weka | 1 | 1 | 1 | 1 |
| Python | 0.99 | 0.96 | 1 | 0.96 |

For all the tests here 80 % training and 20% testing set has been used and random train-test split has been used whenever possible. It has been observed that in different runs the algorithms take different amount of time for training and prediction. Best among three runs has been taken.

Among all the algorithms SVM has shown to be best in terms of accuracy with 100% accuracy in Weka and 98.27% accuracy in Python. The measurements included here has been taken for SVM with polynomial kernel with degree 3 which has shown to perform better than rbf and sigmoid kernel. But it’s training time is significantly high (5.46 s) in Weka even though it is much lower in python (0.22 s).It’s F1-scores are also among the best too , both in Weka and python.

Decision tree classifier with gini index has taken minimum time to train (0.005 s) with second highest accuracy (96.53% in Weka and 98.56 % in Python) after SVM. Testing times are lowest at 0.002 s in python.

Random forest has performed worse than decision tree in terms of accuracy, training and testing time. But that could be due to different parameter setting. Random forests are known to be better at reducing error by ensemble methods but it is not apparent here.

Even though KNNs accuracy is lower than all algorithms except the Naïve Bayesian classifier, it has a good accuracy of 91.32% with close to 0.01 s training time and 0.11 sec testing time in Weka. The F1 score for different classes are still good.

The results from Naïve Bayesian Classifier are consistent both in Weka and python with accuracy around 86% and training and testing time around 0.01 s.

The comparisons above are merely based on the measurements done. Variation in measurements has been observed for same algorithms in different runs. So above was analysis based on approximated measurments.

6. Discussion

Classification algorithms like the ones implemented here can be divided into two types of learners, the eager learners and the lazy learners. Eager learner algorithms, during training phase upon getting the dataset works on it to build a model that they can use for prediction during training phase. So the overall computation tasks are divided into training and testing phase. Among the implemented algorithms Naïve Bayesian Classifier, Decision Tree, Random forest and Support Vector Machine are eager learners.

The lazy learners on the other hand when given a dataset during training phase simply stores them without constructing any model. So all the related computations take place during the testing phase when the classifier has to predict something. This makes lazy learners less preferable for realtime predictions. KNN algorithm is a lazy learner.

The Naïve Bayesian Classifier is a probabilistic classifier. It predicts the class label based on the posterior probability or likelihood that is calculated using prior probability and conditional probability. The task in training phase is calculation of the frequency of every feature value of every class. This is what makes this algorithm so fast. This algorithm thus is more suitable for realtime predictions. This algorithm works better with categorical input variable compared to numeric. When assumption of independence holds this algorithm works better compared to other models like logistic regression. One of the problem of this algorithm is that if some feature value that was not observed during training phase, is found during testing phase then the model assigns 0 probability an fails to predict. This problem can be solved by using techniques like Laplace estimation. One major problem is that it is almost impossible to find dataset with completely independent features in real life. But if speed is the absolute necessity over anything else then this is the algorithm to go with.

Decision tree classifiers are easy to interpret and great for visual representation. Decision trees are capable of handling both continuous and categorical variables. In decision tree in each level the previously unselected feature that results in highest discrimination is selected automatically. One of the problem of this algorithm is that it tends to overfit, though pruning can be used to minimize this problem. Decision trees performs worse when dataset size is small and number of features is high. So if visual representation is of most importance than Decision tree is the choice.

One improvement over the simple decision tree is the Random Forest algorithm. Instead of using a single tree for classification the random forest uses multiple decision trees that are built from dataset generated using bagging with replacement method on the training set. Apart from that the random feature examination for selecting the splitting point brings further variation in the trees used in random forest. In this way the trees works like a committee in which the class that gets the maximum vote is selected as the final class. The good thing about this algorithm is that, the probability of error is reduced due to usage of ensemble method. When prediction is to be made on unexpected validation set random forest gives better accuracy than decision tree. Random forest can be computationally expensive. The number of trees in the forest is a user given parameter. Large value can make the algorithm computationally expensive whereas lower value can give worse result. When properly tuned random forest algorithm can outperform decision trees so random forest should be given more preference.

Support Vector Machine makes classification based on the hyperplane that is generated using the support vectors. A hyperplane is a separator that has the widest margin. It is a line for 2D dataset, a plane for 3D dataset. Support vectors are the data points that are closest to the margins of the hyperplane. When classes are is linearly separable Linear SVM is used. If classes are not linearly separable Kernel SVM is used. In Kernel SVM so called kernels are used that essentially maps the data points to a higher dimension in the hope that a linear separation is possible in that dimension. Inherently SVM is binary classifier but to work with multiclass classification problems OVA (One Versus All) or OVO (One Versus One) classifier model can be used. SVM is highly accurate. It figures out most important characteristic of a given dataset in the form of support vectors. It is less prone to overfitting because only the support vectors defines the prediction model instead of all data points. One of the problems of SVM is that it can be slow when working with large dataset. Selecting the perfect kernel is also a matter of experiment. SVM will be a great choice if accuracy is more important than efficiency.

KNN is a nonparametric lazy learner algorithm. It is classifier a testing datapoint based its distance from K training datapoints. KNN is simple, intuitive, has no assumptions, easy to implement for multi class problem. One problem of KNN is that the parameter K has to be given as input by the user and there is no straight forward rule for choosing best value of K. The efficiency of this algorithm decreases very fast with increasing size of dataset. It requires features to be on same scale, otherwise during distance calculation the effect of one feature in the measured distance can outweigh others. It is also highly sensitive to outliers.

According to the discussion above Naïve Bayesian Algorithm should have the lowest training time. But from the tables from performance evaluation section we can see that the training time of Naïve Bayesian classifier is higher than that of decision trees. The most probable reason is that the code that I have used is not efficient enough even though training time is same for Weka implementation.

SVM has proved its capability in terms of accuracy. It has also shown to be taking the highest training time.

Even though random forest is known to generalize better here it has shown lower performance than decision tree in all aspects. Reasons can be not big enough dataset, non-optimal parameter settings in code.

7. Conclusion  
There is no single best algorithm for classification problems. Different algorithms works better with different type of data. Size of the dataset also determines which algorithm will give best results. Attribute type, missing values, presence of noise also affects different algorithms in different ways. Only way to determine a good solution is to experiment.