Survey Paper: Comparison of Classification Techniques in Supervised Learning

Vinit Nitin Karanje
School of Electrical Engineering and Computer Science
Washington State University
Pullman, Washington

Advisor : Professor Assefaw Gebremedhin
School of Electrical Engineering and Computer Science
Washington State University
Pullman, Washington
assefaw@eecs@wsu.edu

***Abstract*—The purpose of this survey paper is to study several classification techniques in supervised learning and provide a systematic comparison between them. The paper also presents the empirical study of the few of the classification techniques such as logistic regression, Naïve Bayes optimization, K-nearest neighbors, SVM and Decision Tree. The overall comparison of these techniques will help us understand which classification techniques is better for a specific dataset.**

I. Introduction

With the advent of technology, more and more data are being produced nowadays. This increase in storing the real-world data has given rise to the various fields such as Machine Learning, Data Science, Data Mining etc. Because of the availability of datasets, machine learning and data science applications have become a part of our day-to-day life. Now, most of the machine learning problems can be categorized as Supervised and Unsupervised learning. In Supervised learning, we are provided with labeled dataset which means for any input vector x there is an output label. So, the general supervised learning setting is to train a model based on the labeled datasets and test this trained classifier on a new dataset to predict the output. Few examples of the supervised learning methods are linear regression, logistic regression etc. In unsupervised learning, we are not provided with a labeled dataset and the goal here is to identify interesting patterns within the input data. Some of the unsupervised learning methods are Principal Component Analysis, K-means Clustering etc.

Next, we can further classify supervised learning problems into two types namely classification & regression. A regression problem can be defined as any problem predicting a qualitative or continuous value. So, output for any regression problem will be a continuous value. A few examples of regression problems could be predicting age of a person, predicting future stock prices of a company or predicting housing prices. In these examples, the output will always be a real-value. A classification task, on the other hand can be defined as building a function C(X) such that for given input feature vector X, it predicts the value for Y by taking values in the set C. In other words, classification tasks have a set of classes C from which the response Y takes the value. Therefore, we can say that the output for classification can always be between the given classes. For example, if a person arrives at a hospital with a set of symptoms and we need to identify if the person's medical condition is either *normal*, *moderate,* or *critical*. In this example, based on the symptoms, the person's condition will always be between the three output values i.e. *normal*, *moderate,* and *critical*.

II. Classification Methods

A. Logistic Regression:

In some classification problems, instead of finding what category an input vector X belongs to, we are concerned about the predicting the probabilities that input vector X belongs to some category in set C. In general, when the outcome is dependent on multiple independent variables in a dataset, we are concerned with finding the probability that the input vector belongs to particular category. This is known as logistic regression. In logistic regression, the response variable is dichotomous in nature i.e. binary. Logistic regression uses following logistic function so that the value of $p(X)$ will always be between 0 and 1.

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

Also, logistic regression maximum likelihood function to estimate the coefficients β_0 and β_1 . In mathematical terms, the likelihood function is defined as follows.

$$\ell(\beta_0, \beta) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1-p(x_i))$$

The goal of this likelihood function is select $\beta_0$and β_1 such that putting these values into the above logistic equation will maximize the likelihood of the observed data. The plot in Figure 1, shows relationship between a continuous predictor (x-axis) and probability of an event or outcome (Y-axis). As shown in the plot, the linear model does not fit the relationship between predictor x and probability. Therefore, to model this relationship directly, we need to use a nonlinear function i.e. Logistic model which is a sigmoid function. As we can see in the plot, the value of this function is always between 0 and 1.

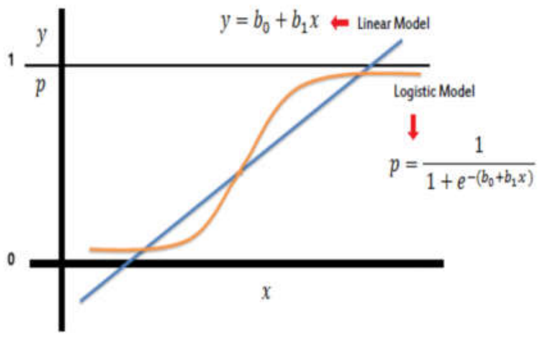


Figure 1: Logistic Regression model

Source: Google Images

One example that describes logistic regression better can be predicting whether a patient has lung cancer based on number of predictors such as age, weight and number of cigarettes smoked per day. In this case, first, we need to identify whether a patient has lung cancer or not, which means the response variable is dichotomous i.e yes or no. Second, there are three independent variables (age, weight and number of cigarettes smoked per day) are given. Therefore, we can say that logistic regression can be used in classification problem when we need to identify the relationship between the independent predictors and a dichotomous response variable. Some other examples of logistic regression include predicting if bank customers would default based on the previous transaction history or predicting absenteeism pattern of employees based on their individual characteristic.

B. K-Nearest Neighbors:

K-nearest neighbors is one of the simplest classification techniques in machine learning. K-NN falls under the category of lazy learning as there is no training phase in this technique before classification. The main idea here is to classify new data point same as the closest known point from the training data. In other words, consider we are given a training set of labeled examples and a test set of new data points. Now, in order to classify the new data points, we look at the training data points nearest to the test data point and classify the test data point based on the majority of the training labels from the nearest training data points. To decide the distance between the two data points, there are certain similarity (distance) metrics that can be used based of the data type. The commonly used similarity metrics are: Euclidean distance, Cosine similarity, Jaccard Distance, Hamming distance and Manhattan Distance.

In K-NN, K defines the number of nearest training data points to select in order to classify a test label. This means if the value of k is 3, then we check the labels for three nearest training examples and classify the test data point based on the maximum number of labels in the three training examples. To select a value of K, we need to understand data well and try few different values to see how the evaluation changes. The evaluation metrics can be used as one of the following for K-NN. The evaluation metric in machine learning is used to determine accuracy of the classifier. For K-NN some of the evaluation metrics are: Accuracy, Precision, Recall and F-score.

A stepwise K-NN process can be described as follows:

Step 1: Select one of the Similarity (distance) metrics mentioned in Figure 1.

Step 2: Split the input labeled dataset into training and test data.

Step 3: Select one of the evaluation metrics mentioned in Figure 2.

Step 4: Run K-NN several times for different values of k and check the evaluation measure for each value of k.

Step 5: Select the k-value that gives the best evaluation measure.

Step 6: Using the same training set, create a new test set with data item we need to predict.

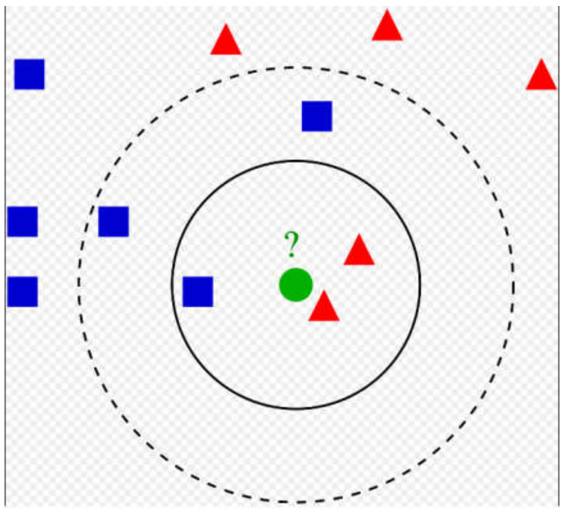


Figure 2: K-NN Example

Source: Wikipedia

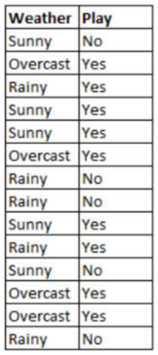
The K-NN can be better explained with the example mentioned in Figure 2 above. In the above example, the goal is to classify the green circle to either red triangles or blue squares. Now, if k = 3 in the above example (solid line circle), then we can see that there are total two red triangles and one blue square. Since the maximum number of the nearest labels are red triangles, the test point (green circle) will be classified as red triangle. However, if k = 5 (dashed line circle), then there are three blue squares and two red triangles. Therefore, in this case, the test point will be classified as blue squares. Some of the real-world applications where K-NN is used are recommender systems, Concept search, detecting patterns in credit card usage etc.

C. Naïve Bayes Classification:

Naïve Bayes is a classification technique based on Bayesian Theorem. Naïve Bayes works with an assumption of independence among predictors i.e. it assumes that presence of a particular feature is unrelated to the presence of any other feature. Therefore, it is called Naïve. The Bayesian theorem provides the posterior probability $P(A|B)$ which gives us probability of occurrence of A given B occurs. Following equation in figure 5 states the posterior probability formula i.e. Bayesian rule.

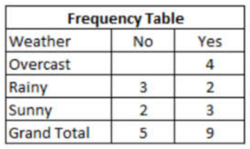
$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

To understand Naïve Bayes classification better, consider an example where we are given a labeled training set of variables Weather and Play (Figure 3: Table 1). The goal here is to classify whether the player play or not given the weather conditions.



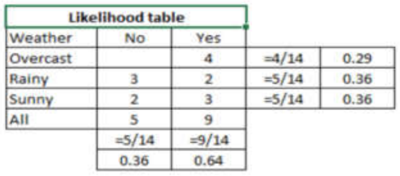
Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Table 1



Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Table 2



Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Table 3

Figure 3: Naïve Bayes Classification Example

Source: Google Images

As mentioned in the figure 3 above, to solve the problem using Naïve Bayes classification, we convert the given dataset into frequency table (Figure 3: Table 2) by obtaining number of times players play and number of times players do not play for all weather conditions (Sunny, Overcast and Rainy). Next, we create a likelihood table (Figure 3: Table 3) by finding individual probabilities. Finally, we use Naïve Bayes equation to calculate the posterior probability for each class. The highest posterior probability is selected as correct class.

For overcast weather,

$$P(Yes|Overcast) = \frac{P(Overcast|Yes) * P(Yes)}{P(Overcast)}$$

$$= \frac{4}{9} * \frac{9}{14} * \frac{14}{4} = 1$$

Since $P(Yes|Overcast) = 1$ is the maximum posterior probability, weather condition $Overcast$ is classified as Yes. This means players will play if weather is $Overcast$.

Similarly, $P(Yes|Sunny) = 0.64$ which is higher therefore, $Sunny$ too will be classified as Yes. However, $P(Yes|Rainy) = 0.40$ which is smaller than $P(No|Rainy)$, therefore, $Rainy$ will be classified as No. A few of the real-world examples of Naïve Bayes classification are: classification of news by sections, email spam detection, face recognition and Sentiment Analysis.

D. Support Vector Machines:

Support Vector Machine (SVM) is a supervised machine learning algorithm used mostly for classification however, in general, it can be used for both classification and regression. In SVM, for classification problem, we plot the data in n-dimensional space (n is the number of features). The main idea is to find a hyperplane that differentiates the two classes. Here, hyperplane is defined as the line that separates the data points in n-dimensional space. Now, the distance between a hyperplane and a nearest data point is known as margin. A good margin is the one where the distance is larger for both the classes whereas if the separation distance between the nearest training point and hyperplane is less, it is considered as bad margin. Therefore, in SVM, the hyperplane is selected such that the margin is maximum. This way, there is a better chance of data being classified correctly.

Consider a plot shown in figure 4. The data points are plotted in the n-dimensional space (n=2 here) and are separated by three hyperplanes A, B and C. Now, as mentioned above, the best hyperplane between these three would be the one which maximizes the margin i.e. distance between the hyperplane and the nearest point on the plane (on either side). Clearly, it's C which maximizes the margin.

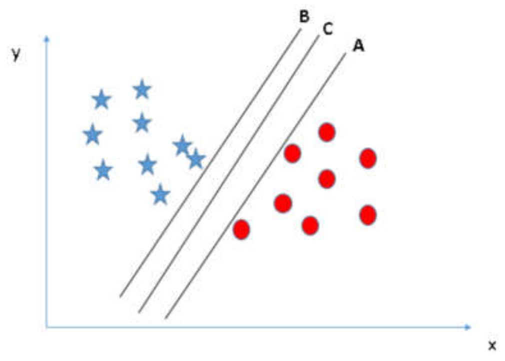


Figure 4: Linear SVM Binary Classification Example

Source: Google

Next, in case of binary classification, linear SVM works well as it is easier to find a linear hyperplane between two classes. However, when there is a problem where we cannot have linear hyperplane between the classes, then SVM uses a Kernel functions which basically takes the low dimensional input space and converts it into higher dimensional space i.e. a non-separable problem is converted into separable problem. Some of the real-world examples where SVM is used include image recognition tasks, handwritten digit recognition, text classification and many more.

E. Decision Tree:

Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal here is to predict value of a target variable by learning from simple decision rules inferred from the data features. Decision Trees learn from data to approximate a sine curve with a set of if-then-else decision rules. Depth of the tree decides the complexity of the rules and how fit the model is. The models are built in the form of a tree structure which breaks down the data into smaller subsets. Some of the key factors to understand the decision trees are Splitting, Pruning, Entropy and Information Gain. A detailed explanation of each of them is given below.

Splitting:

Splitting is the process of partitioning the dataset into subsets. Splits are formed on a particular attribute. The splitting rule in decision trees is such that we want to measure the purity of the split i.e. the attribute that reduces the uncertainty the most. The uncertainty is measured through information gain and entropy explained below.

Pruning:

Pruning is defined as the shortening of branches of tree. Pruning reduces size of the tree by converting some branch nodes into leaf nodes. Pruning is necessary in order to improve predictive accuracy of the tree by reducing overfitting.

Entropy:

The entropy is used in decision trees to calculate the homogeneity of the data samples. If the sample is completely homogenous, the entropy is zero and for equally divided sample, the entropy is one. The value of entropy is always between 0 and 1. For example, consider a set of N items divided into two classes A and B. Let n items belong to class A and m items belong to class B. In this case, the entropy function is defined as,

$$E = -p_1 log_2(p_1) - p_2 log_2(p_2)$$

where, $p_1 = \frac{n}{N}$ and $p_2 = \frac{m}{N} i.e. 1 - p_1$

Information Gain:

Information Gain is used in decision trees to measure the uncertainty of a random variable X defined by Entropy.

$$Information\ Gain(Z) = H(X) - H(X|Z)$$

where, $H(X|Z)$ is conditional entropy of X given Z and $H(X)$ is entropy of X

The example in figure 5 illustrates the decision tree structure. The text in black represent if-then-else conditions in decision tree. Based on these rules, the tree is further split into branches. The end of the branch that doesn't split anymore is known as leaf node. The decision tree in this example illustrates a classification problem which classifies whether a passenger died or survived (marked as red or green).

There are several approaches to implement decision trees such as ID3, C4.5, CART etc.

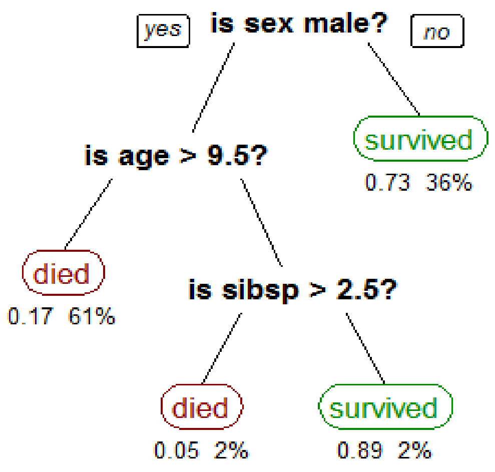


Figure 5: Decision Tree Example

Source: Wikipedia

III. Experiments

This section describes the experiments performed using several datasets to understand and compare the different classifiers mentioned in the previous section.

A. Environment:

The environment used for these experiments is a Windows 10 machine with 16GB RAM. Programming language used for coding is Python and coding is done in a Jupyter environment. Some of the python libraries used are: numpy, pandas and scikit-learn.

B. Datasets:

The scikit learn package in python used for implementation of different classifiers also has *datasets* package. This package basically contains small toy datasets. This function can be used to generate controlled synthetic datasets. Using this package, following datasets were generated to test the performance of the different classifiers.

MNIST Image Digits Recognition:

The *load_digits* function in scikit-learn *datasets* package loads and returns the digits dataset for classification. In this dataset, each datapoint is a 8 x 8 image of a digit. This dataset has 1797 samples in total and 10 classes. The goal for classification problem in this dataset is to identify a digit between 0 to 9 from the training images.

Iris Dataset:

The *load_iris* function in scikit-learn *datasets* package loads and returns the iris dataset for classification. This dataset has 50 samples per class and there are total 3 classes to predict. Therefore, the total samples in this dataset are 150. Based on the input attributes sepal length, sepal width, petal length and petal width, the goal is to predict a class of iris plant between Iris Setosa, Iris Versicolour and Iris Veiginica.

Wine Recognition Dataset:

This dataset is a result of chemical analysis of wine grown in the same region in Italy by three different cultivators. There are 178 samples in total and based on the 13 different attributes in the training data, the goal is to predict the three different classes i.e. class_0, class_1 and class_2.

Breast Cancer Wisconsin Dataset:

This dataset consists of 569 samples and 2 Classes to be predicted from the 30 attributes computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. These features describe characteristics of the cell nuclei present in the image.

C. Experimental Setting:

In order to compare between the different classifier, the datasets mentioned in the previous section were used. The given datasets were split into 75% training data and 25% testing data. Next, for each dataset, every classifier model was trained, and the output was predicted on the test data using scikit-learn. For better understanding of the results, training time, prediction time, accuracy and confusion matrix for each of the classifier were calculated. Since the synthetic data generated has small samples, the training time calculated did not differentiated much between the classifier and hence is not described in much detail. The results obtained are described in the next section.

IV. RESULTS

This section describes the results obtained from the experiments performed in the previous section. Figure 6 shows the plot of number of attributes and the accuracy of the classifier. The number of attributes in our case were 4, 13, 30 and 64 in our case for the datasets Iris, Wine Recognition, Breast Cancer Wisconsin and MNIST Image digits recognition respectively. As shown in the figure, we can see that Naïve Bayes classifier showed 100% accuracy when the number of attributes were less than 20 which is 4 and 13 in

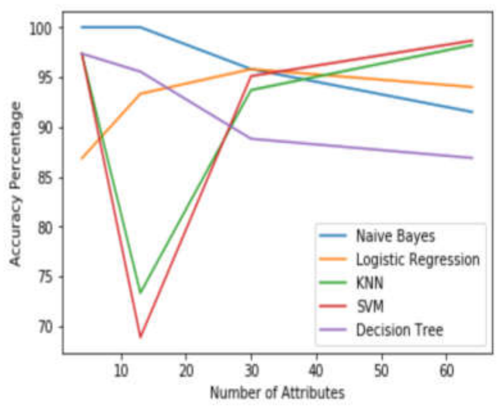


Figure 6: Accuracy vs Number of attributes

this case with Iris dataset and Wine recognition dataset respectively. The performance of Naïve Bayes classifier reduced for the remaining two datasets as the number of attributes increased to 30 and 64 for Breast Cancer Wisconsin and MNIST Image digits recognition respectively. Classifiers KNN and SVM showed significant decrease in the accuracy for classification of the dataset Wine recognition. This low accuracy can be improved in SVM by changing the parameter configuration. Moreover, the performance was further improved as the number of attributes were increased and SVM showed highest performance for maximum attributes 64 for MNIST Image digits recognition.

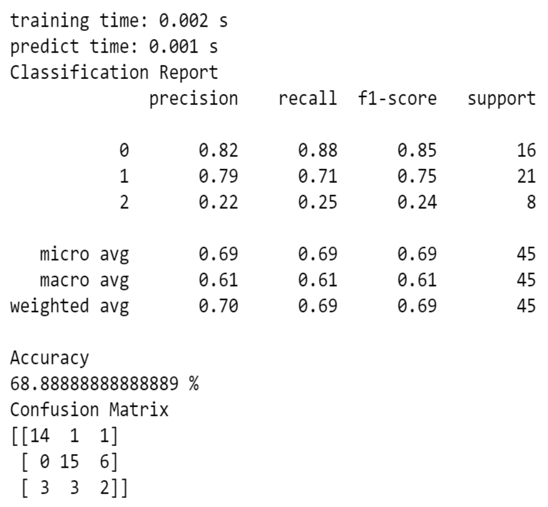
We can also see the performance of Decision Tree classifier constantly decreasing with the increase in number of attributes. Logistic regression, on the other hand, seems to give a moderate performance as the number of attribute increase. As we can see in figure 6, the accuracy increases with the number of attributes or the first three datasets and decreases for the MNIST dataset where the number of features is high.

Dataset\Classifiers	NB	LR	KNN	SVM	DT
Iris Dataset	100	86.84	97.36	97.36	97.36
Wine Recognition Dataset	100	93.33	73.33	68.88	95.55
Breast Cancer Dataset	95.80	95.80	93.70	95.10	88.81
MNIST Image Digits Recognition	91.50	94	98.22	98.66	86.88

Figure 7: Accuracy obtained for each dataset

Figure 7 shows the table mentioning the accuracies obtained for each of the dataset. From Figure 7, it seems that most of the classifier performed well on the datasets given except K-NN and SVM for a particular dataset. For Iris and Wine Recognition datasets, the highest accuracy obtained was 100% by using Naïve Bayes classifier while the other classifiers performed within the range 68% to 95%. For MNIST Image digits recognition dataset, the highest accuracy of 98.66% was achieved by SVM classifier and KNN also achieved approximately similar accuracy i.e. 98.22%.

Since classification accuracy does not provide all the details about the model, it is difficult to understand where a classifier is making mistake in classification. Therefore, apart from the accuracies of the classifiers, the confusion matrix and classification report were computed for different datasets. The confusion matrix displayed the counts of correct and incorrect classification for respective classes. These results helped understand the dataset better. Figure 8 shows the lowest result obtained for SVM classifier for the Wine Recognition Dataset.



```
training time: 0.002 s
predict time: 0.001 s
Classification Report
             precision    recall  f1-score   support

           0       0.82      0.88      0.85        16
           1       0.79      0.71      0.75        21
           2       0.22      0.25      0.24         8

   micro avg       0.69      0.69      0.69        45
   macro avg       0.61      0.61      0.61        45
weighted avg       0.70      0.69      0.69        45

Accuracy
68.88888888888889 %
Confusion Matrix
[[14  1  1]
 [ 0 15  6]
 [ 3  3  2]]
```

Figure 8: SVM Result

As we can see, figure 8 shows the confusion matrix and classification report. In classification report, the precision column shows the accuracy with which the samples were classified. Here, the classifier showed only 22% accuracy for classifying the test data to class_2 which reduced the overall accuracy to 68%. The test data was classified as class_0 and class_1 with 82% and 79% accuracy. Finally, the confusion matrix at the end display the number of mistakes made by classifier in classifying the test data. Out of the 45 testing samples, 14 were correctly classified as class_0, 15 were correctly classified as class_1 and 2 were correctly classified as class_2. The maximum number of mistakes were made in identifying the class_2 samples as 6 of the samples were incorrectly classified as class_1 and 1 sample was incorrectly classified as class_0.

V. Analysis

This section describes the overall comparison between the different classifiers mentioned in this paper. Based on the results of the experiments performed, we can say that Naïve Bayes classifier performed poorly on MNIST dataset whereas SVM and KNN performed better on the MNIST dataset. So, it seems that Naïve Bayes doesn't perform well when the attributes are repeated. Even though SVM performed better, the training time for SVM for MNIST dataset was 0.239 seconds which is far more than Naïve Bayes training time which is 0.025 seconds. The computed training time seems small because the synthetic datasets generated were small. If the dataset size is increased, the training time increases with it. Similar is the case between KNN and Naïve Bayes, even though the accuracy in KNN is better, the query time is higher in KNN and there is an extra overhead of finding the k value first. Also, the computation cost is high in case of KNN as we need to calculate the distance between each query instance to all the training samples. So, there is a trade-off between the accuracy and training time taken. Therefore, it makes sense to use Naïve Bayes model when there are limited resources available in terms of CPU, memory and training time.

Also, between logistic regression and decision tree, we see that logistic regression performed better in larger dataset, but the difference between these two is that logistic regression searches for a single linear decision boundary in the feature space whereas decision tree partitions the feature space into half-spaces which means there is a non-linear decision boundary in case of decision tree. So, it seems that decision trees are better when the dataset is not linearly separable while logistic regression can be used when dataset is linearly separable.

Another way to compare between the two classifiers Logistic regression and Naïve Bayes, is segregating them into generative and discriminative classifier categories. Generative classifiers learn the model that generates the data by estimating the assumptions and distributions of the model whereas discriminative classifiers try to model by depending on the observed data. In general, generative classifiers make their predictions by using Bayes rule while discriminative classifiers model the posterior probability directly. From these differences, we can say that the Naïve Bayes classifier is a generative classifier whereas logistic regression and KNN are discriminative models.

VI. Conclusion And Future Work

Finally, it seems that there is no clear winner when it comes to selecting the best classifier for classification problems in supervised machine learning. So we can say that, in order to choose the best classifier for a classification problem, the important thing is to understand the problem and the kind of data that we have. Several things that can help choosing classifier for a classification problem can be the dimensionality of the data, number of classes, size of the data, available resources etc.

Since there are some more classifiers which can perform better that the ones mentioned in this paper, future work involves understanding the other classifiers such as Random Forest, AdaBoost, LDA, QDA etc. Also, the datasets used in this study were quite small, so it would be interesting to see how these classifiers will perform when dataset is large.

References

[1] Entezari-Maleki, Reza, Arash Rezaei, and Behrouz Minaei-Bidgoli. "Comparison of classification methods based on the type of attributes and sample size." *Journal of Convergence Information Technology* 4.3 (2009): 94-102.

[2] Ng, Andrew Y., and Michael I. Jordan. "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes." *Advances in neural information processing systems*. 2002.

[3] Amancio, Diego Raphael, et al. "A systematic comparison of supervised classifiers." *PloS one* 9.4 (2014): e94137.

Appendix

The Python code for the above experiments and datasets is available on GitHub.

https://github.com/vinitkaranje/DataScienceSurvey_ClassificationTechniques.git