CSCI 59000
Big Data Analytics Project Report

Classification & Prediction Analysis of Machine Learning Algorithms

Keyur Kirti Mehta
Shreyansh Mohnot
Love Modi
Darsh Sanghavi

Department of Computer Information and Science
Indiana University Purdue University Indianapolis, USA

Table of Contents

1 INTRODUCTION

Our aim through this project is to perform data classification and prediction on an Image Dataset as part 1 of this project. Clustering and Linear classification techniques (k-Nearest Neighbor and Support Vector Machine) have been used to generate classification results. The results from the classification have been compared and discussed. For part 2, we have picked Machine Learning Prediction techniques (Naïve Bayesian and Stochastic Gradient Decent with Logistic Loss Function) to analyze and predict results. A diabetes dataset has been used to do our analysis. Using previous medical history of patients, we have implemented a naïve Bayesian model and an Amazon Machine Learning prediction model and compared the test results for both. Hence predicting if a person for a given set of medical conditions has diabetes or not. Our motivation behind this project was to understand clustering, linear classification, and prediction models clearly, get to know about the different hyperparameters associated to further improve our results and finally compare the results to find which amongst them a better algorithm is.

2 RELATED WORK

On researching upon several Machine Learning articles for better understanding and implementation of clustering, classification and prediction algorithms, we came across several interesting and helpful papers. The following papers were most helpful 'The Distance-Weighted k-Nearest-Neighbor Rule' [1]. 'Deep Learning using Linear Support Vector Machines' [2], 'Beyond independence: Conditions for the optimality of the simple Bayesian classifier' [3]. Using information from these papers and some articles on the internet [4], we went ahead to develop an image recognition system based on k-Nearest Neighbor, Support Vector Machine, and Naïve Bayesian. We choose to implement SVM as a linear classifier model, as it forms the base for developing complex and state-of-the-art classification algorithm [5]. Other algorithms (kNN, Naïve Bayesian) also form fundamental concepts in the machine learning area of study.

3 DATASET

We have used the following datasets:

- CIFAR-10
- Pima Indian Diabetes Dataset

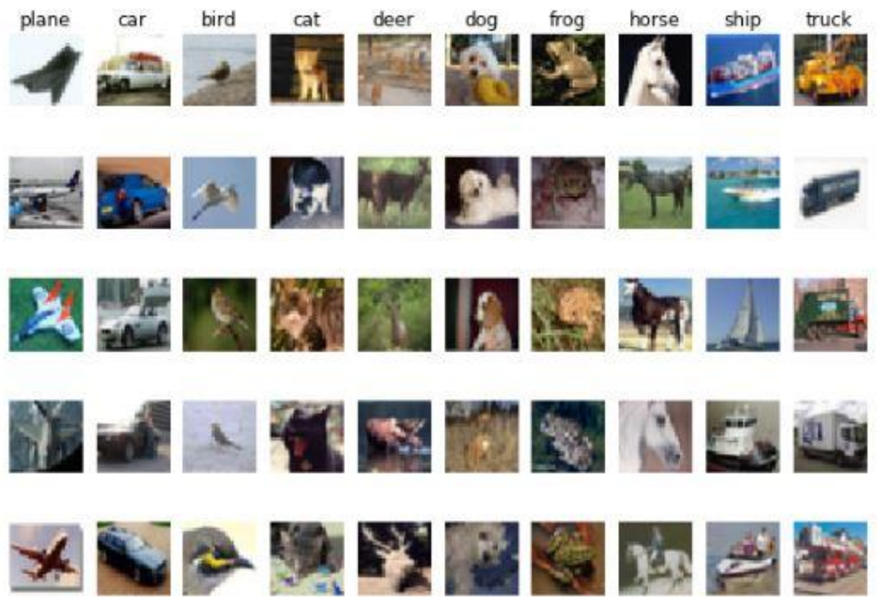


Figure 1 CIFAR 10 Image Dataset

The CIFAR-10 dataset is a widely available dataset. We found this dataset at the source. [6] It consists of 60000 images (resolution - 32 x 32 x 3, where 3 is the RGB value for each pixel) which are divided into 6 batches of 10000 images. Out of these 5 batches have been used to train our model and the 6th batch has been used as the testing set. We have used cross-validation to avoid overfitting of the training set while training our model.
The data source used for this project is called 'Pima Indians Diabetes' [7]. It is an open source data set available on the University of California, Irvine machine learning repository. It includes details regarding 768 females of at least 21 years of age belonging to the Pima Indian Heritage. Following are the attributes used in the dataset:

1. Number of times pregnant
2. Plasma glucose concentration in 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skinfold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/ (height in m) ^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

The attribute 9 is a binary attribute used as the target parameter. It states whether the subject has diabetes or not. The above data has been divided into two sets (A) Train and Test set (669 rows) (B) Test set for batch prediction (99). Dataset A has been used generate the machine learning model and study its quality (such as the Area under Curve, true positives, false positives etc.). Dataset B is used as a live data for batch predictions. Dataset B has been used to test how accurate the machine learning model is when subject to real-time data that it has not seen earlier. The results of these experiments are outlined in the next sections.

4 METHODOLOGY

4.1 Initial Set up and Approach

4.1.1 gCloud Setup

The dataset used for the prediction analysis contains 60000 images of resolution 32 * 32. It requires good processing speed and enough memory to create a prediction model which further can be tested. So, we have setup Google cloud to run and execute the same. The virtual instance is created on Google cloud having 8 vCPUs and 30 GB of memory i.e. 8 core are used. Ubuntu operating system is installed for the virtual machine. Jupyter instance is installed on this machine which is used to develop the code required for the scope of the project. This instance can be connected through gCloud SDK launcher using the following command.

```
gcloud compute ssh --zone=us-west1-b <INSTANCE-NAME>
```

To launch the jupyter hub, use the following command.

```
jupyter-notebook --no-browser --port=<defined-port>
```

4.1.2 Hyper Parameters

In any machine learning algorithm, the data set is divided into training set and testing set. A model is created using the training set and then the created model is verified on the test dataset to verify its accuracy. To create an ML model, different parameters are used related to the applied algorithm. These parameters are tuned with different values to get the best training model. For each algorithm, first needs to identify some parameters which can be tuned. The second step is to find different values in which can be used to get the better accuracy. For each different value, we train the model and find the accuracy of each model. Then the value of the parameter is finalized where we achieved the best accuracy. [8]

We have used different hyperparameters which are tuned to get the best accuracy in the training the model. To calculate the distance between the pixel values of different images we started with L1 Manhattan distance, then we tried the L2 Euclidean distance. A split ratio of training and testing the model is another hyper-parameter which gives better accuracy. For the specific algorithm, like k- Nearest Neighbor classifier we have tried using different k values to achieve the best accuracy.

4.1.3 Cross Validation

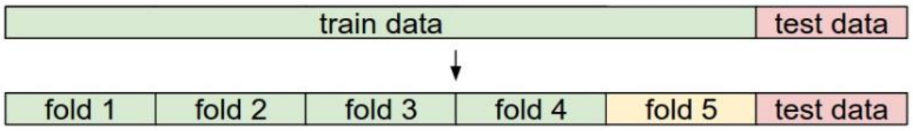


Figure 2 Cross Validation

It's not a good practice to use test data to get the accuracy until the model is finalized. So, to test the same, we use cross-validation. Cross-validation is a technique in which the training data is divided into different folds. And then for each value of hyperparameter, the model is run independently on each fold. One-fold out of n folds is used as validation fold and other is used as training fold. Model is trained on n-1 training folds and then tested on the validation fold to get the accuracy. This process is repeated n times using different validation fold each time. All the folds are used exactly once as validation fold. [9] For e.g. training data is divided into 5 folds then, 4-fold is used as training the model and 1-fold is used as validation fold. Accuracy is tested on the validation fold. This process is repeated 5 times using one-fold as validation fold exactly once. Then the average of all validation accuracy is calculated to get the best one. Cross-validation is one measure to increase the accuracy of the system, but it's not a good measure every time as it has high computational cost and requires a huge amount of time to get the accuracy.

4.2 k Nearest Neighbor (kNN)

k- Nearest Neighbor (kNN) algorithm is used for feature similarities. It is used to predict the class of the object. An object is classified for the class which has the most number of the neighborhood around it. In our image dataset, we have summed up the all the pixel values of the image. Each pixel contains the 3 values R, G, B data. Once all the value is added together, the image will be represented by 1 * n matrix. Now, this value is associated with a label which describes the class of the image. We have used different k values and Euclidean distance as hyper-parameter to increase the accuracy. Also, we have divided the training data using cross-validation in 5 folds to get the best accuracy. [10]

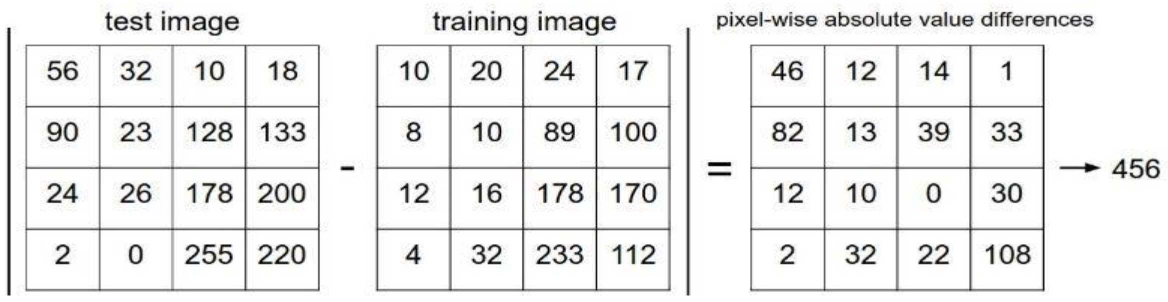


Figure 3 Calculation of image value for kNN

The distance between each image and the training images is calculated. Then nearest neighbor images for 1st class is being identified. This process is repeated for all the classes and for all the k values. The class whose images having the least distance and satisfy the k value i.e. finding k neighbors is being tagged to the test image. The accuracy of the model is then verified by the label predicted and the actual label which is tagged to the image. The best accuracy achieved by the model for hyperparameter value is considered as the accurate model and then being used for the testing dataset.

4.3 Support Vector Machine (SVM)

Support Vector Machine classifier uses the below-mentioned score function, where xi is the image flattened out to a single column vector of the shape [D x 1].

$$f(x_i; W; b) = W(x_i) + b$$

The matrix W (of size [K x D]) is the weight parameter, and the vector b (of size [K x 1]) is called the bias. These parameters get trained during the training process and guide the output during testing. They affect the output scores of images and need to be tuned over time to get the efficient classification. The function output is the class score which indicates which class image might belong to. Higher the class score higher is the probability that the image belongs to that class. We need to guide this score to the correct path using our training model. Using equation 1, a function from the pixel values to class scores was defined, parameterized by a set of weights W. This function needs a feedback mechanism which will act as a guiding factor to inform the system how wrong is it going with its predictions. This feedback function is called the loss function or more specifically the SVM loss in case of Support Vector Machine Classifier. Support Vector Machine function which gives the SVM classifier its name.

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda \sum_k \sum_l W_{k,l}^2$$

This function is used to reduce the raw class scores into normalized positive numerals that aggregate to one, on to which the SVM loss is applied to give feedback to the score results. The feedback is in terms of improving the loss value for the correct class of the input image. This is done using a mathematical procedure called the gradient descent and backpropagation (described in the next section). These help in configuring the weights and biases in such a way that the class scores are higher for the class to which the input image belongs to.

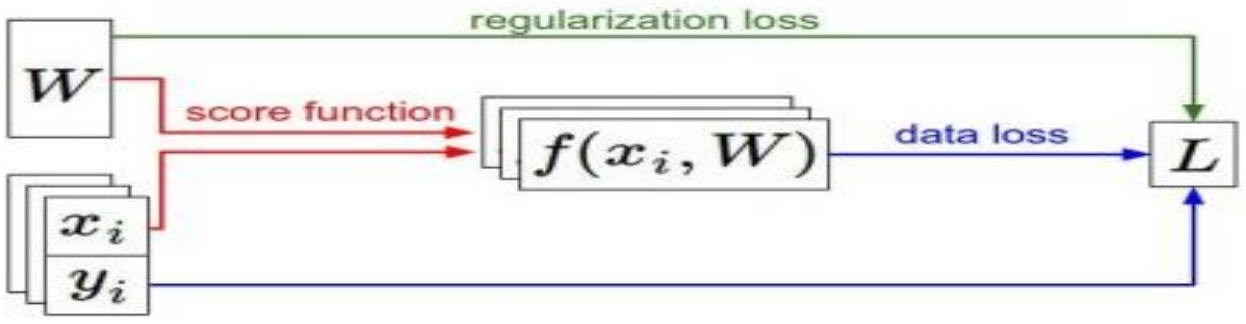


Figure 4 Loss Function calculation SVM

4.4 Naive Bayesian

It is a classifier technique based on the Bayes' theorem. In the naïve Bayesian algorithm, parameters are considered as independent to each other. I.e. if all the different parameters are true for 1st set of data value then that data will belong to the class. It does not correlate with different parameters present in the dataset. Therefore, the theorem is known as 'naïve'. [11]

$$\boxed{P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C)P(C)}{P(A_1 A_2 \ldots A_n)}}$$

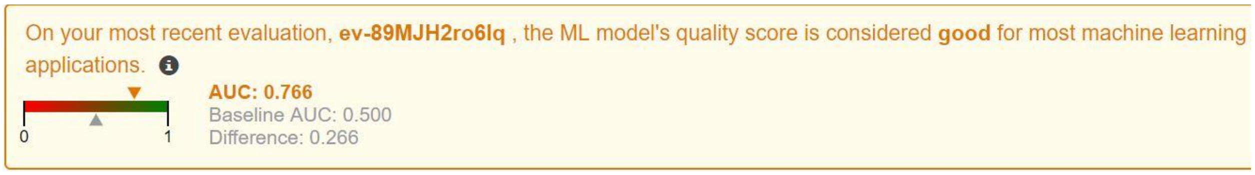
Bayes theorem provides a way of calculating posterior probability which is calculated using the formula given above. We have used binary classification to find the class of the record. The class which has higher probability is the winner for that record. There are different variants of this algorithm, which can be used to find the probability. To calculate the probability, we used Gaussian formula for each class.

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

4.5 Amazon Machine Learning

Steps were followed as outlined in the given project document. As soon as the data set A was fed to the Amazon Machine Learning system, it divided the data (internally) into a training set (70% of original data) and test set (30% of the original data). Following results were obtained after the model was generated and tests were performed on the data.

The Area under the curve (which is a how good a dataset is for machine learning application) was found out to be 0.766 as shown below:


On your most recent evaluation, **ev-89MJH2ro6lq** , the ML model's quality score is considered **good** for most machine learning applications.

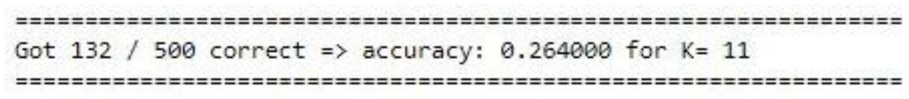
AUC: 0.766
Baseline AUC: 0.500
Difference: 0.266

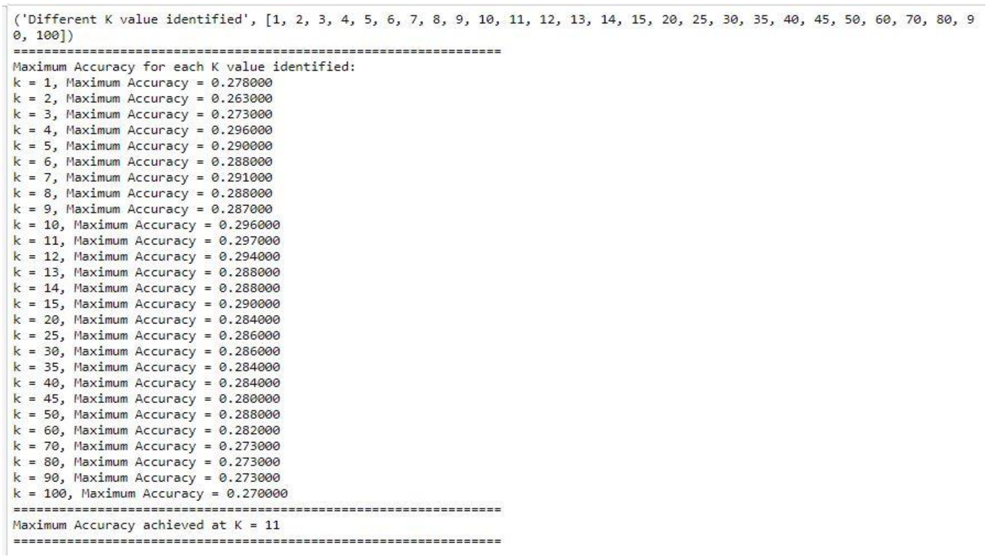
5 RESULTS

5.1 k Nearest Neighbor (kNN)

We have used k value as hyperparameter. So, we calculated accuracies for different k values. And got the best result when k = 11 which is 29.70 %

This k value of 11 is used in the testing dataset to verify the model created earlier. We achieved the accuracy of 26.40 % in the test dataset. This is due to the training and testing datasets are different.


```
====================================================================
Got 132 / 500 correct => accuracy: 0.264000 for K= 11
====================================================================
```



```
('Different K value identified', [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 9
0, 100])
================================================================
Maximum Accuracy for each K value identified:
k = 1, Maximum Accuracy = 0.278000
k = 2, Maximum Accuracy = 0.263000
k = 3, Maximum Accuracy = 0.273000
k = 4, Maximum Accuracy = 0.296000
k = 5, Maximum Accuracy = 0.290000
k = 6, Maximum Accuracy = 0.288000
k = 7, Maximum Accuracy = 0.291000
k = 8, Maximum Accuracy = 0.288000
k = 9, Maximum Accuracy = 0.287000
k = 10, Maximum Accuracy = 0.296000
k = 11, Maximum Accuracy = 0.297000
k = 12, Maximum Accuracy = 0.294000
k = 13, Maximum Accuracy = 0.288000
k = 14, Maximum Accuracy = 0.288000
k = 15, Maximum Accuracy = 0.290000
k = 20, Maximum Accuracy = 0.284000
k = 25, Maximum Accuracy = 0.286000
k = 30, Maximum Accuracy = 0.286000
k = 35, Maximum Accuracy = 0.284000
k = 40, Maximum Accuracy = 0.284000
k = 45, Maximum Accuracy = 0.280000
k = 50, Maximum Accuracy = 0.288000
k = 60, Maximum Accuracy = 0.282000
k = 70, Maximum Accuracy = 0.273000
k = 80, Maximum Accuracy = 0.273000
k = 90, Maximum Accuracy = 0.273000
k = 100, Maximum Accuracy = 0.270000
================================================================
Maximum Accuracy achieved at K = 11
================================================================
```

Figure 5 Best Accuracy for each k

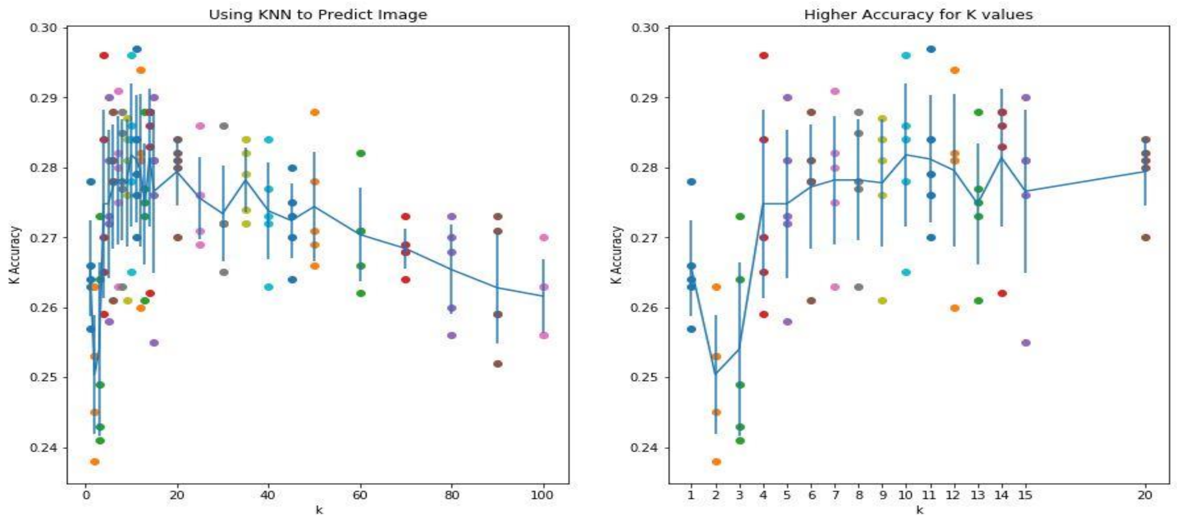
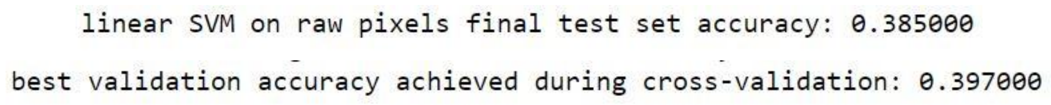


Figure 6 Accuracies for k value

5.2 Support Vector Machine

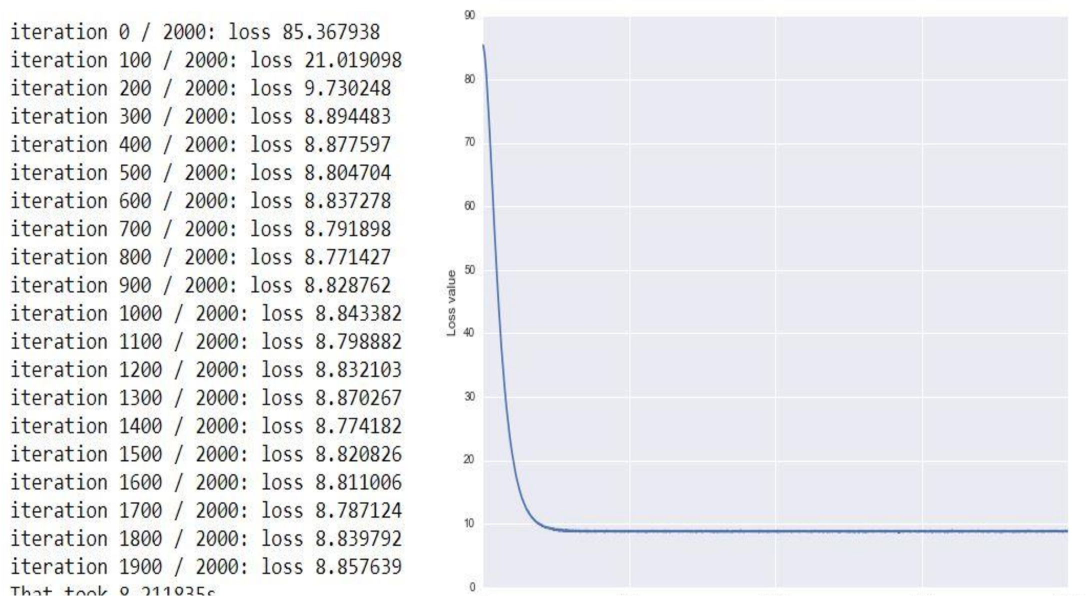
As we see in the below results, for 2000 rounds of training the loss goes on decreasing. We have captured results after every 100 rounds iteration. As the loss decreases, the accuracy goes on increasing. The interesting thing to note is that the loss becomes stable after certain rounds of iteration (~300) and does not go down more even if we go on until 2000 rounds.

The SVM Classifier model gave an accuracy of 38.5% on the training set and 39.7% on the test dataset. Considering the task of modeling and training a complex dataset, the results were quite satisfactory. It further emboldened our motivation to build a nonlinear classifier to see how far we would be able to improve the task of image classification.



```
linear SVM on raw pixels final test set accuracy: 0.385000

best validation accuracy achieved during cross-validation: 0.397000
```



```
iteration 0 / 2000: loss 85.367938
iteration 100 / 2000: loss 21.019098
iteration 200 / 2000: loss 9.730248
iteration 300 / 2000: loss 8.894483
iteration 400 / 2000: loss 8.877597
iteration 500 / 2000: loss 8.804704
iteration 600 / 2000: loss 8.837278
iteration 700 / 2000: loss 8.791898
iteration 800 / 2000: loss 8.771427
iteration 900 / 2000: loss 8.828762
iteration 1000 / 2000: loss 8.843382
iteration 1100 / 2000: loss 8.798882
iteration 1200 / 2000: loss 8.832103
iteration 1300 / 2000: loss 8.870267
iteration 1400 / 2000: loss 8.774182
iteration 1500 / 2000: loss 8.820826
iteration 1600 / 2000: loss 8.811006
iteration 1700 / 2000: loss 8.787124
iteration 1800 / 2000: loss 8.839792
iteration 1900 / 2000: loss 8.857639
That took 8.211835s
```

Figure 7 Loss Function Graph SVM

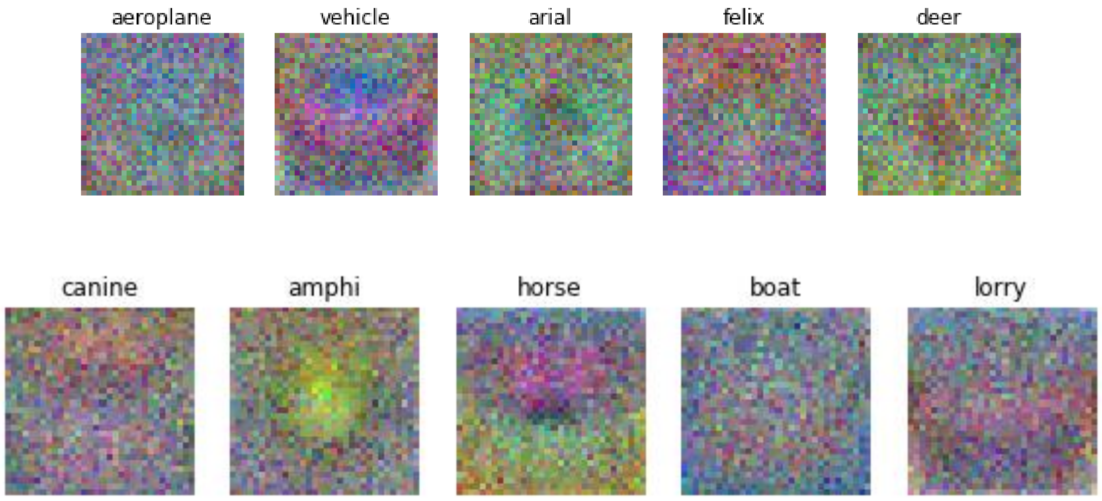
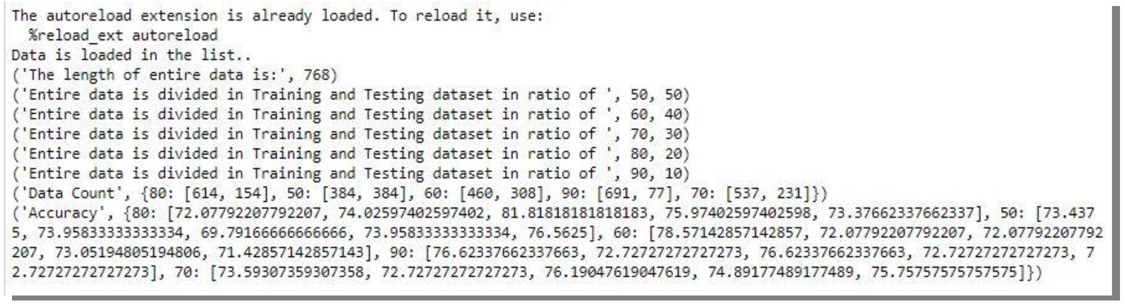


Figure 8 SVM Image Classifier

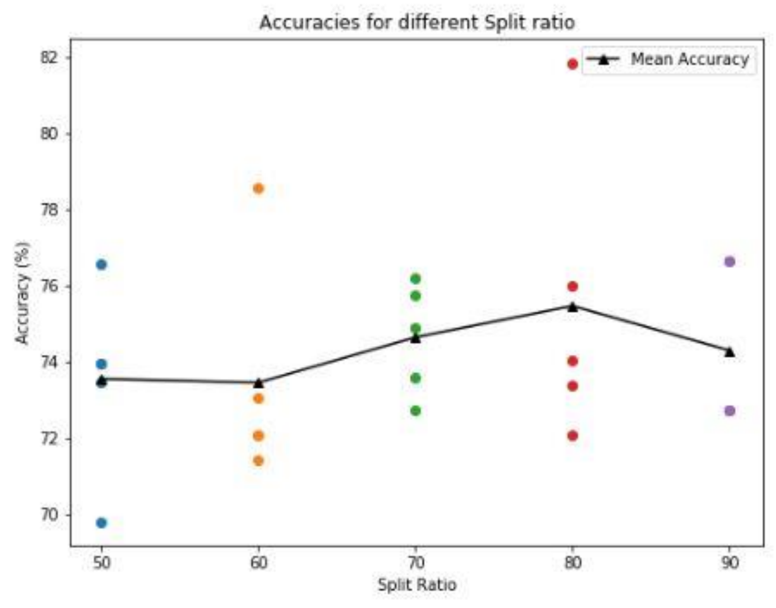
5.3 Naive Bayesian

Split ratio is used as the hyper-parameter for the naïve Bayesian algorithm. The training data is divided into a different split ratio and the accuracy is calculated for each of them. The process is repeated 5 times and the mean accuracy is then found and used as the best result for the algorithm.



```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
Data is loaded in the list..
('The length of entire data is:', 768)
('Entire data is divided in Training and Testing dataset in ratio of ', 50, 50)
('Entire data is divided in Training and Testing dataset in ratio of ', 60, 40)
('Entire data is divided in Training and Testing dataset in ratio of ', 70, 30)
('Entire data is divided in Training and Testing dataset in ratio of ', 80, 20)
('Entire data is divided in Training and Testing dataset in ratio of ', 90, 10)
('Data Count', {80: [614, 154], 50: [384, 384], 60: [460, 308], 90: [691, 77], 70: [537, 231]})
('Accuracy', {80: [72.07792207792207, 74.02597402597402, 81.81818181818183, 75.97402597402598, 73.37662337662337], 50: [73.437
5, 73.95833333333334, 69.79166666666666, 73.95833333333334, 76.5625], 60: [78.57142857142857, 72.07792207792207, 72.07792207792
207, 73.05194805194806, 71.42857142857143], 90: [76.62337662337663, 72.72727272727273, 76.62337662337663, 72.72727272727273, 7
2.72727272727273], 70: [73.59307359307358, 72.72727272727273, 76.19047619047619, 74.89177489177489, 75.75757575757575]})
```

Figure 9 Split Ratio Naive Bayesian



```
================================================================
The best accuracy 75.454545 is achieved at split ratio 80-20
================================================================
Precision is 62.701359
Recall is 70.981301
False Positive Rate is 21.919006
================================================================
```

Figure 10 Naive Bayesian Accuracy

5.4 Amazon Machine Learning

The outcome shows that 73% results are correct and 27% were detected wrong. There is an option to change the trade-based threshold if our requirement wants to find a less false positive with a trade-off with the false negative. So, depending on the tolerance level with respect to the false negatives/positives that the system can tolerate we can set the threshold. For this project, the threshold has been set to 50% which evaluates to the above result.

The figure below shows the performance evaluation for the given data set.

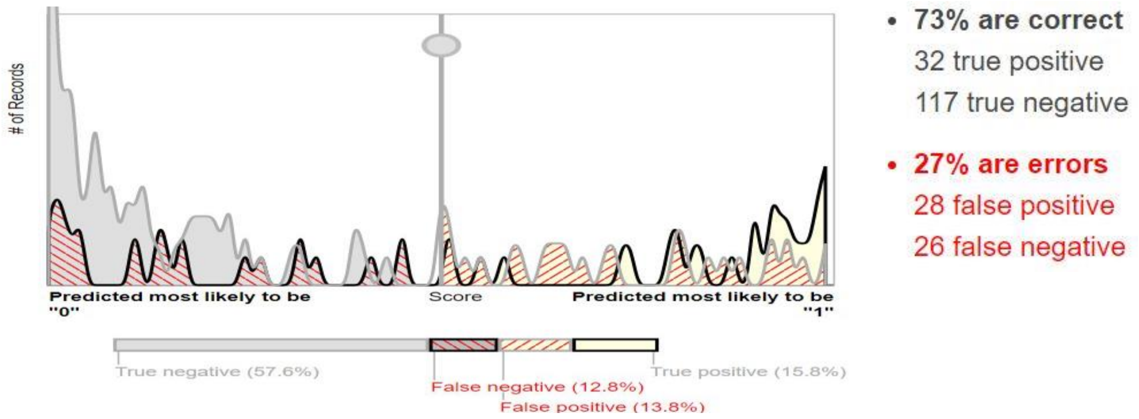


Figure 101 Accuracy for AWS ML

Once the results were obtained with dataset A, the dataset B was used for doing batch prediction.

Amazon machine learning has the feature to test real-time data new data on the model that we generated. Dataset B has 99 rows of such data on which the prediction was done to find how accurate the model can be. This dataset was not used in prior training and testing hence the dataset was suitable for such batch predictions. The results obtained from batch predictions were as follows:

- True Positive – 23.23 %
- True Negative – 50.50%
- False Positive – 14.14%
- False Negative – 12.12%

The above results are close to training and testing results found for Dataset A which revalidates that the model gives predictions as per its claims.

5.5 Result Comparison

Part 1	k Nearest Neighbor	SVM
Validation Accuracy	29.70%	40.08%
Test Accuracy	26.40%	38.06%

Table 1 KNN SVM Accuracy

Part 2	Naïve Bayesian	AML
Accuracy	74.61%	70.99%
Recall	69.19%	62.03%
Precision	64.07%	57.65%
False Positive Rate	22.33%	23.84%

Table 2 Naive Bayesian and AML Accuracy

6 INDIVIDUAL CONTRIBUTIONS

The table below highlights the individual contributions of each member throughout this project.

Team Member	Role
Keyur Mehta	1. Algorithm findings and analysis 2. Implemented KNN & Naïve Bayesian 3. Presentation and Report Writing
Shreyansh Mohnot	1. Research about image classification 2. Implemented KNN & Naïve Bayesian 3. Presentation and Report Writing
Love Modi	1. Finding Datasets 2. Implemented of SVM & AWS Machine Learning 3. Presentation and Report Writing
Darsh Sanghavi	1. Research about image reading 2. Implemented of SVM & AWS Machine Learning 3. Presentation and Report Writing

7 CONCLUSION

This project has been a very interesting and useful exercise. It exposed us to fundamentals of Machine Learning (ML) algorithms, get started with building state-of-the-art algorithms for clustering, classification and prediction, and different ways to improve results. The immense power and application range of machine learning was practically experienced through this project. We got exposed to different machine learning terms, definitions and concepts.

Our experiments revealed that if the correct domain related data can be used to create a machine learning model, it proves to be immensely useful for further predictions in real time as well as in batch prediction mode. Results were detected with around 74% accuracy for Naïve Bayesian and Amazon ML. For clustering and linear classification algorithms, the best accuracy we could obtain is around 40%. Even batch prediction gave similar results. The scope of machine learning extends to so many applications other than detecting possible patients with a certain disease. It can also be used for various classifications and predictions in sales, political, financial departments, etc. AML also informs us regarding the algorithm used and allows us to tweak many parameters to mold our model as per our requirements. It helps to understand how sound a machine learning algorithm is and if compare its result with other algorithms.

REFERENCES

[1] S. A. Dudani, "The Distance-Weighted k-Nearest-Neighbor Rule," *IEEE Transactions on Systems, Man, and Cybernetics SMC-6,* pp. 325-327, 1976.

[2] E. &. K. R. Bauer, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants.," *Machine Learning,* 1999.

[3] P. &. P. M. Domingos, "Beyond independence: Conditions for the optimality of the simple Bayesian classifier," *Thirteenth International Conference on Machine Learning,* pp. 105-112, 199.

[4] X. Z. S. R. J. S. K. He, "Deep residual learning for image recognition," *CVPR,* pp. 770-778, 2016.

[5] I. a. H. G. E. Alex Krizhevsky and Sutskever, "ImageNet Classification with Deep Convolutional Neural Networks".

[6] A. Krizhevsky, 2017. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html.

[7] [Online]. Available: https://www.kaggle.com/uciml/pima-indians-diabetes-database/data.

[8] "Hyper Parameter," [Online]. Available: https://www.quora.com/What-are-hyperparameters-in-machine-learning.

[9] "Cross Validation," [Online]. Available: https://www.openml.org/a/estimation-procedures/1.

[10] "knn," [Online]. Available: https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7.

[11] "Naive bayesian," [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/.