IMPLEMENTATION OF MACHINE LEARNING ALGORITHMS FOR CLASSIFICATION OF MNIST DATASET

KASI VISWANATH (18414)

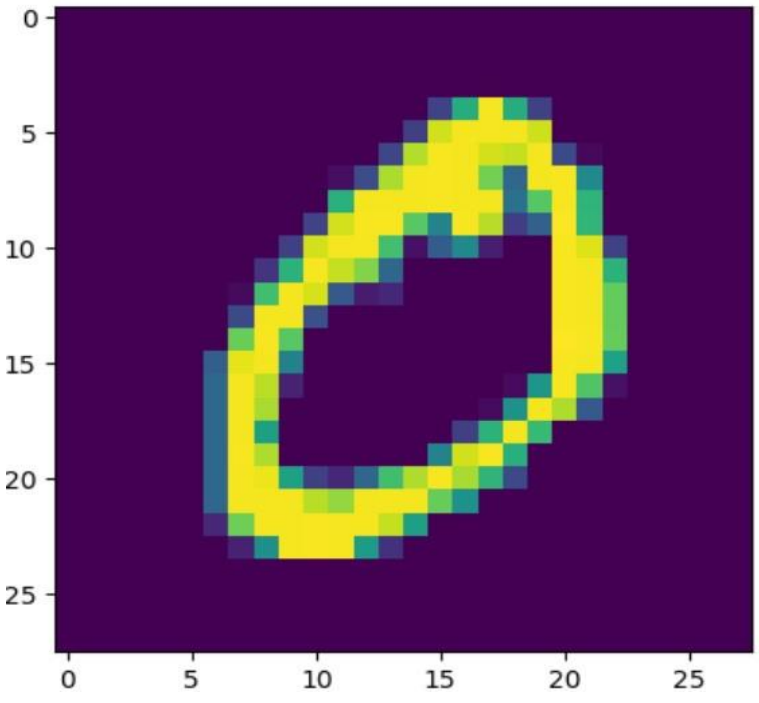
INTRODUCTION

Machine learning and Artificial Intelligence are the two names that has boomed both in the field of Industry and Academia. The tremendous advancement in Autonomous Driving and medical diagnostics has led to a huge demand in this field. Even though the surge in these field are recent but the traces dates back to WWII when Alan Turing developed Christopher (Machine to decode German encrypted code).

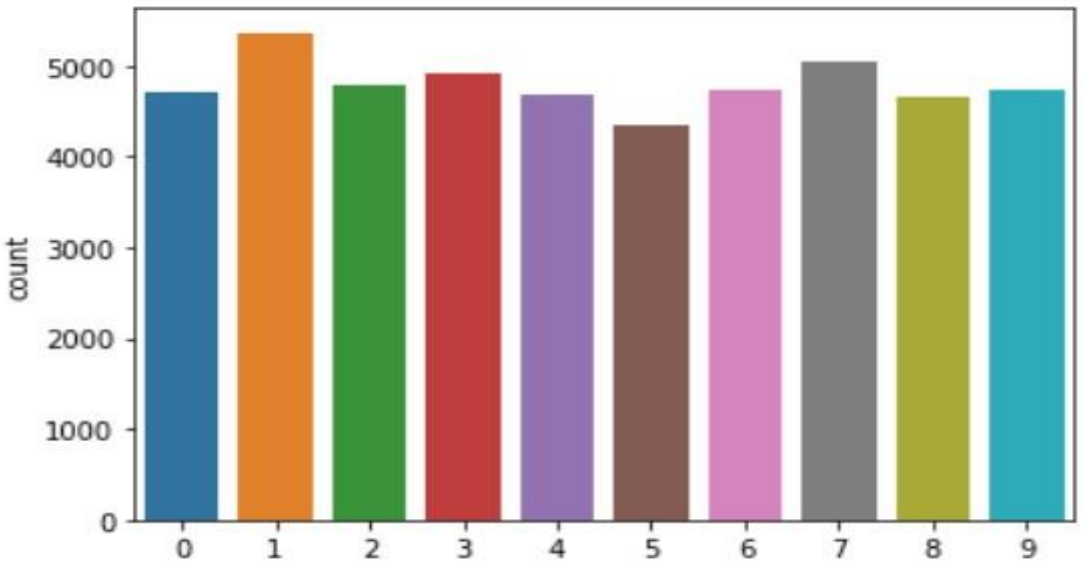
Data is the backbone of every field. It is the major component of any ML/AI algorithms. MNIST is a dataset which consists of 70,000 hand written digits represented in a 28*28 resolution image with pixel values in the range 0-255. This report consists of Observations and Inferences from implementing different Machine Learning techniques for classification of the dataset.

MNIST DATASET

MNIST dataset is a subset of NIST dataset consisting of 70,000 images of hand written digits represented in a 28*28 resolution image. The dataset is labelled from 0-9(10 classes) which makes it suitable for both supervised and un-supervised learning.



Here we split the dataset into three, training data, validation data and testing data. **The split of 3 is made because especially in ANN classification, there is a leakage of data from validation to training process as we choose the best validation accuracy model which may not be the best model and evaluating the same on test data will give us the accurate inference.** The training data consists of 48,000 images, validation consists of 7,000 images and test consists of 15,000 images.



The above histogram represents the datapoint distribution to different classes and we can see that the class imbalance is in the tolerance regime of <5% making the dataset suitable for training.

The following sections consists of results and inference obtained from implementing different ML algorithms on the MNIST dataset.

EXPERIMENTS & RESULTS

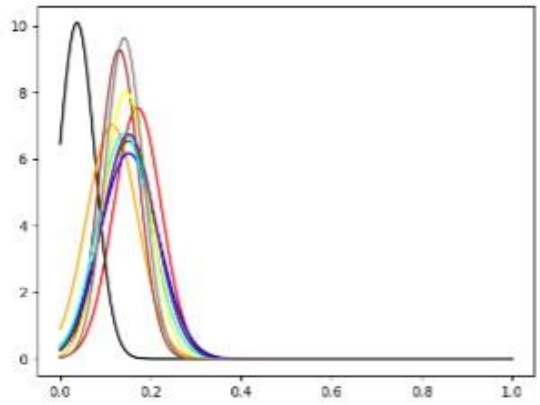
LINEAR REGRESSION

Linear Regression cannot be used here as we are trying the classification task on MNIST dataset containing 10 classes. As Linear Regression tries to find the best fit for the data points and even if we try to define the boundary condition after lots of trial and error, the **model will not be robust** and will be **overfitted to the training set** thus resulting is very low accuracy and misclassification.

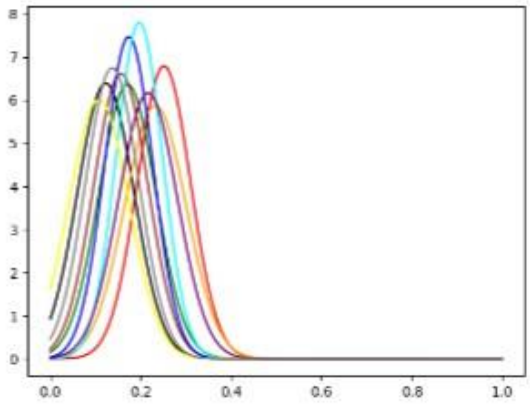
MAXIMUM LIKELIHOOD ESTIMATION (MLE)

MLE employs a method of estimating a likelihood function and hence determining the parameters of a model. The parameter values are such that the process described by the model produced the data that were actually observed. It assumes that the data generation can be described by a **Gaussian Distribution** ("most commonly found distribution in daily life"). The gaussian consists of **μ and σ** which are mean and standard deviation respectively, and these are the exact values which will be estimated resulting in the curve fit the data.

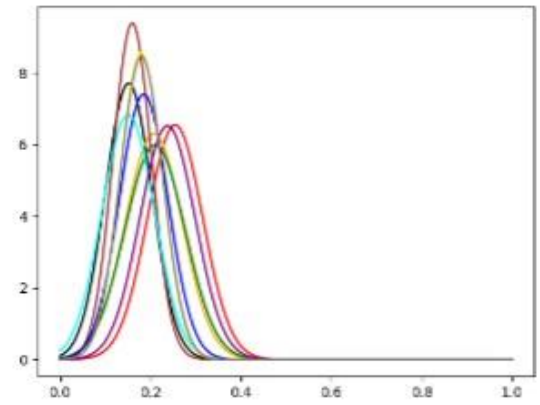
Talking about our scenario, we observed that a specific property of the dataset is quite interesting. We did a small analysis of splitting the datapoint consisting of **784 pixels into 4 quadrant and took their distribution for the 10 classes and there, it resulted to be gaussian distributions**.



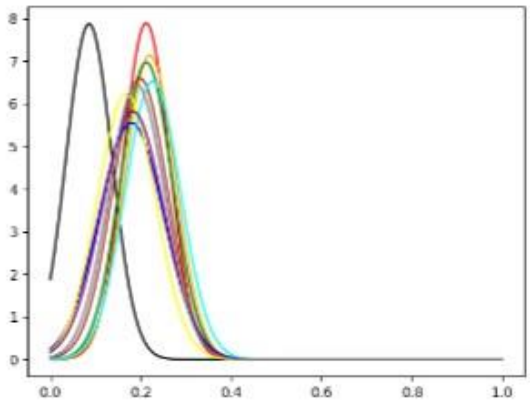
(a) Quadrant 1



(b) Quadrant 2



(c) Quadrant 3



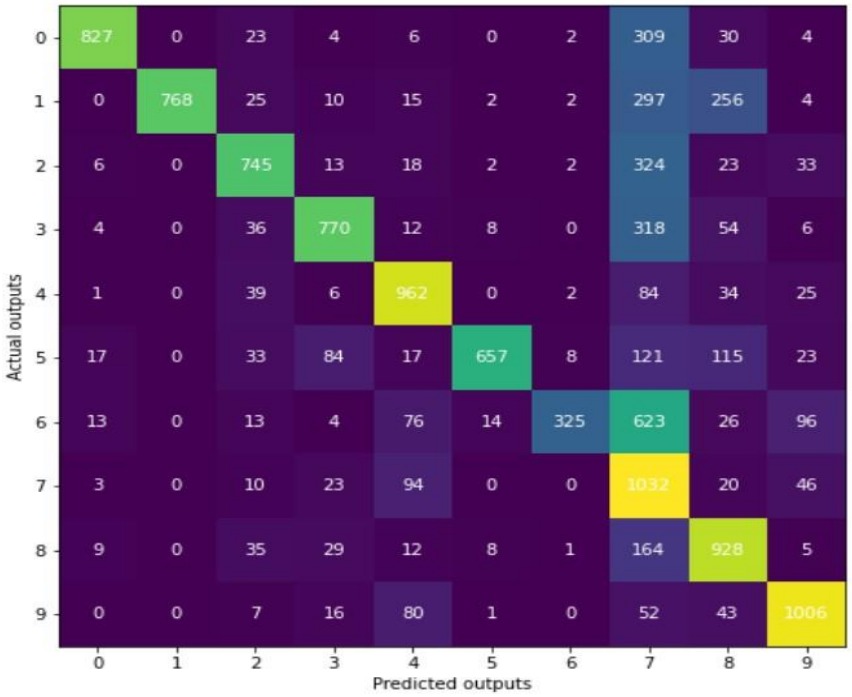
(d) Quadrant 4

CLASSIFICATION REPORT:

	precision	recall	f1-score	support
0	0.94	0.69	0.79	1205
1	1.00	0.56	0.72	1379
2	0.77	0.64	0.70	1166
3	0.80	0.64	0.71	1208
4	0.74	0.83	0.79	1153
5	0.95	0.61	0.74	1075
6	0.95	0.27	0.42	1190
7	0.31	0.84	0.45	1228
8	0.61	0.78	0.68	1191
9	0.81	0.83	0.82	1205
accuracy			0.67	12000
macro avg	0.79	0.67	0.68	12000
weighted avg	0.79	0.67	0.68	12000

MLE obtained an accuracy of 67.3% which is quite depressing when compared to our expectations. The parameters μ and σ for the 2-D Gaussian curve are fixed based on the training set. Thus, **as a result of overfitting on the training set/less generality, the accuracy is low**. If the training set would have been a better representative of the overall distribution of data samples, accuracy would have been better for test set also.

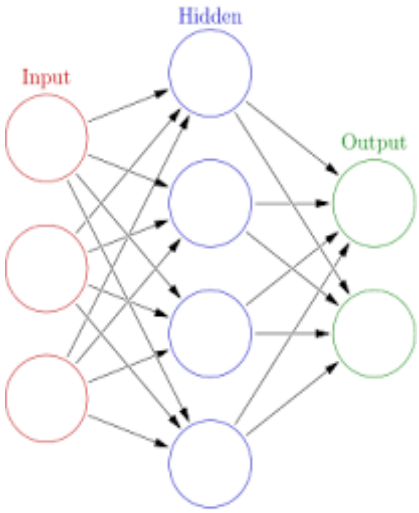
CONFUSION MATRIX:



The analysis of confusion matrix show that the **class 7 has many false positives** i.e., every classes is being misclassified as 7 in a huge ratio. The class 6 is being misclassified as 7 more than it has correct predictions. Another point to note is **the precision of 1 is 100% and for 0,5,6 they are above 95%.** This means there is no false positive for the class 1 but there are many false negative, i.e., misclassification.

ARTIFICIAL NEURAL NETWORK(ANN)

An artificial neural network (ANN) is the component of artificial intelligence that is **inspired from the functioning of a human brain**. Processing units make up ANNs, which in turn consist of inputs and outputs. The inputs are what the ANN learns from to produce the desired output. The input is fed into a series of layers stacked and interconnected called hidden layers. These hidden layers consist of neurons which is the main processing unit. These processing units are made up of input and output units. The input units receive various forms and structures of information based **on an internal weighting system**, and the neural network attempts to learn about the information presented to produce one output report.



For our classification problem, I have created an ANN model which consists of **4 layers with two hidden layers**. The number of nodes is assigned for the layers in a **bottleneck fashion** (The main reason for extraction of features). Final layer consists of 10 nodes (which is equal to the number of class) and is having **"SoftMax" activation**. The rest of the layers have**" Relu" activation** with **dropout layers (α=0.2)** in between them. The dropout layers obstruct the

model from overfitting. The loss function used is **"Categorical Crossentropy"** with optimizer as **"Adam".**

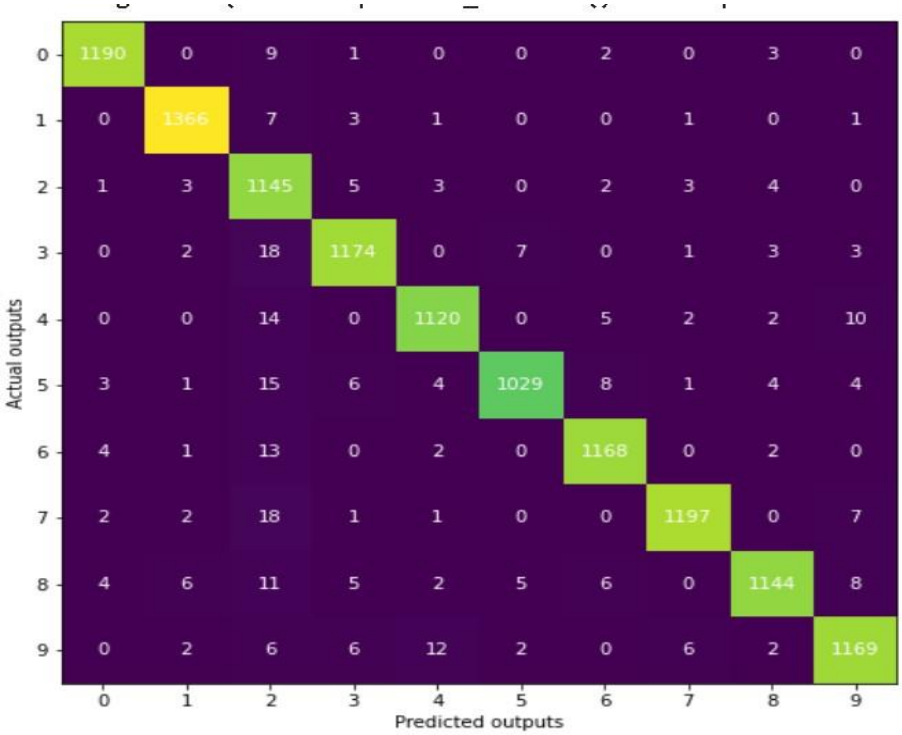
CLASSIFICATION REPORT:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	1205
1	0.99	0.99	0.99	1379
2	0.91	0.98	0.95	1166
3	0.98	0.97	0.97	1208
4	0.98	0.97	0.97	1153
5	0.99	0.96	0.97	1075
6	0.98	0.98	0.98	1190
7	0.99	0.97	0.98	1228
8	0.98	0.96	0.97	1191
9	0.97	0.97	0.97	1205
accuracy			0.98	12000
macro avg	0.98	0.97	0.97	12000
weighted avg	0.98	0.98	0.98	12000

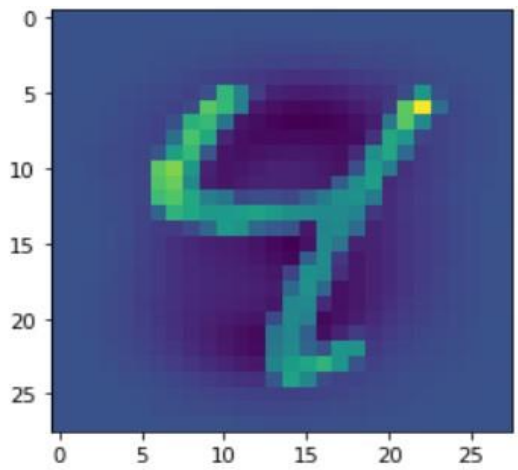
This is the **highest accuracy of 98%** which we achieved in the whole project. Although this result was obtained after lots of trial and error. I will point out observation recorded while tuning the parameters.

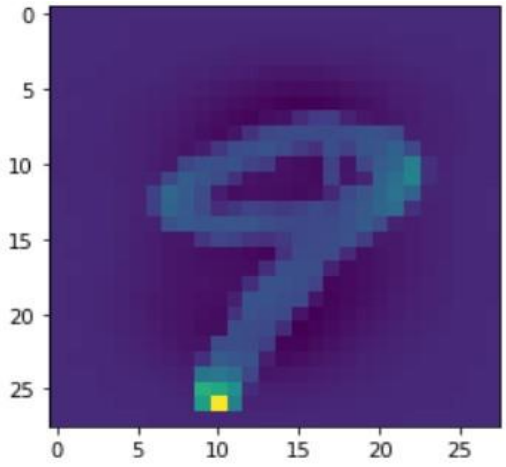
- Increasing the number of layers may speed up the accuracy convergence but lots of layers would decrease the accuracy as the model is too complex.
- Increasing the number of nodes of a layer may increase the accuracy achieved up to a certain point but decreases after that point.
- RELU activation gave the best accuracy on the model.
- The model learns up to a convergence point so the number of epochs should be selected such that the convergence point is achieved thereby reducing the computation time.
- Setting a callback function is necessary after every interval of epochs such that the model which achieves the best validation accuracy Is saved.
- Including dropout function with optimal α is necessary to avoid overfitting.
- An efficient/optimal model is achieved by tuning the above said points and selecting the right activation, loss functions.

CONFUSION MATRIX:



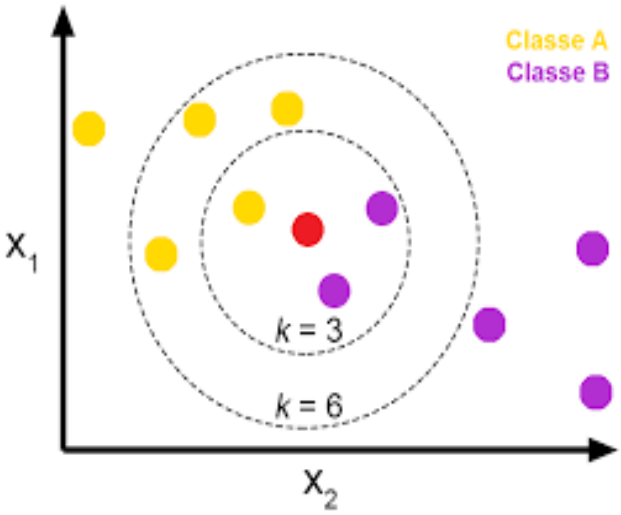
From the confusion matrix it is evident that **every class except 2 is being classified with <1% error**. The class 2 is being wrongly predicted to the rest of the classes. We observe **this phenomenon in results obtained from most of the algorithms tested. The class 4/9 (4 being predicted as 9 and vice versa)** predictions are as usual giving error but its minimal. The reason is off course **similarity in appearance**.





K NEAREST NEIGHBOUR (KNN)

Unlike other methods which we have discussed which basically learns parameters for the classification, knn is a non-parametric algorithm that can be used for classification as well as regression tasks. The principal of knn is by **finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label**. K is a user defined constant and the test data is classified but assigning the label which appears frequently among the k training samples. The commonly used method to find the distance is the **Euclidean distance**.



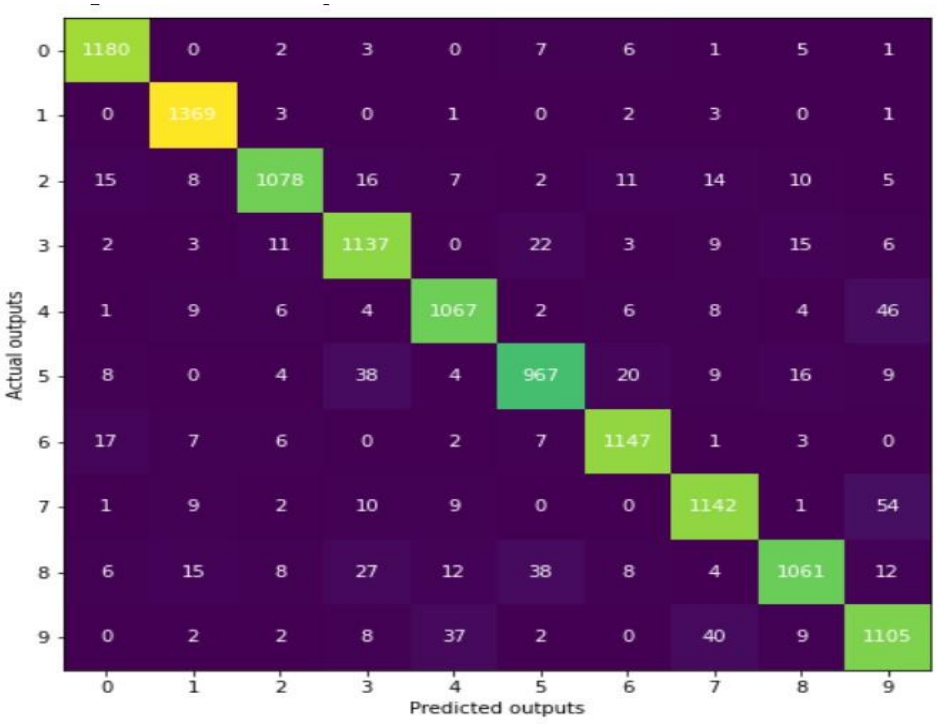
CLASSIFICATION REPORT:

	precision	recall	f1-score	support
0	0.96	0.98	0.97	1205
1	0.96	0.99	0.98	1379
2	0.96	0.92	0.94	1166
3	0.91	0.94	0.93	1208
4	0.94	0.93	0.93	1153
5	0.92	0.90	0.91	1075
6	0.95	0.96	0.96	1190
7	0.93	0.93	0.93	1228
8	0.94	0.89	0.92	1191
9	0.89	0.92	0.90	1205
accuracy			0.94	12000
macro avg	0.94	0.94	0.94	12000
weighted avg	0.94	0.94	0.94	12000

I classified the test data by training the knn algorithm for 3 different **k values (1, 3, 7)** so that we can get an idea as of which k value will give the best accuracy (93.77%). **Only 50% (30,000 datapoints) of the dataset was used as it took very high computation time (~55 mins for a single k value). The best result was achieved with k=1, but the accuracy with k= 3, 7 is in the range within 0.1% to the accuracy achieved with k=1.** This closeness in the range of accuracy is due to the following reasons:

- **Huge amount of data available** for training. We observed that with increase in data, the accuracy of the test data also increased for every value of k. The comparisons with the manipulation of data are provided in the python code.
- The **datapoints of a class is close with each other but widely spaced from other classes resulting in higher votes** for a class therein increasing the accuracy irrespective of the k value.

CONFUSION MATRIX:



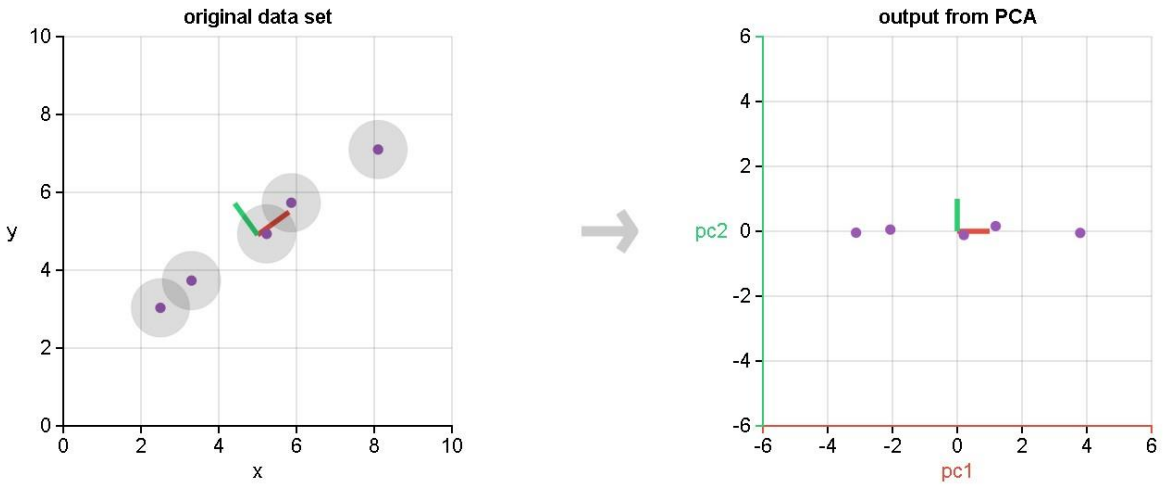
The usual scenario of **misclassification between 4/9 and 7/9** is observed here which occurs due to **similarity**. The rather interesting case is the **less error in class 2 predictions** compared to other classes. Most of the other

algorithm exhibited the issue with class 2. The confusion matrix is very similar for the all the values of k.

The major drawback of knn algorithm is the amount of time taken for computation. For 30000*784 datapoints, it took close to 56 mins for each value of k. While training with different amount of datapoint, I observed that the **time consumption is linearly proportional to the amount of data**.

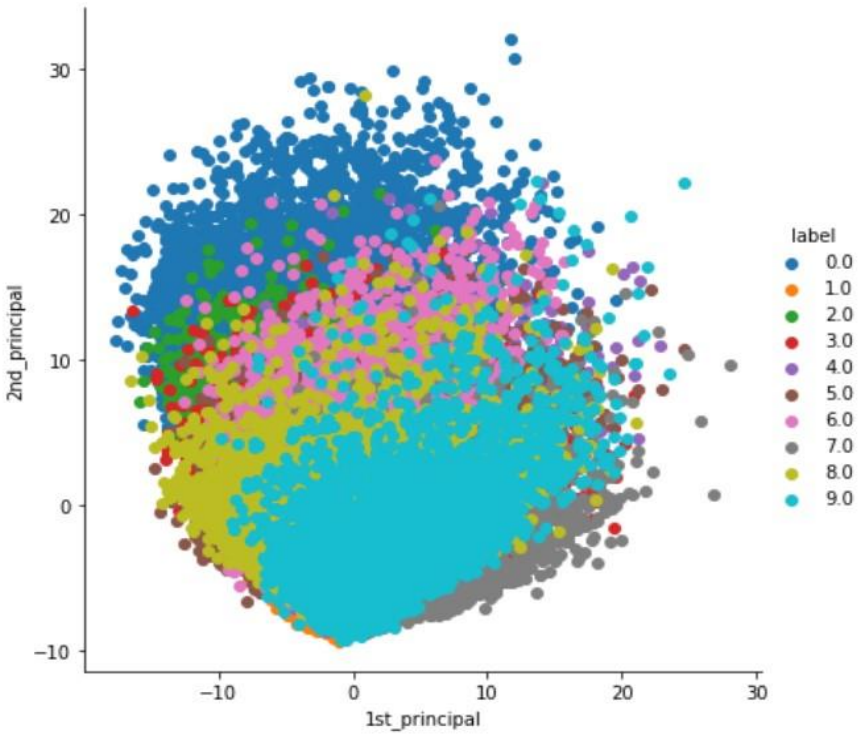
PRINCIPAL COMPONENT ANALYSIS + SUPPORT VECTOR MACHINE (PCA+SVM)

Recently there is a huge availability of datasets with very large amount of data which are often difficult to interpret. Principal component analysis **(PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss**. The technique involves generating new independent variables from the dataset which maximizes variance. It can also be considered as a **feature extractor** by dropping the least important feature hence **reducing the dimensionality of the dataset**.



Reducing a 2-Dimensional dataset into 1-Dimensional datapoints.

I have generated two principals (for visualization purpose) for the dataset by solving the eigen values/vectors. The same was plotted for all the classes. The generated **datapoints appears to be clustered together for each class but there is a huge overlapping among classes**.



To check if PCA helps in improvising the test results (which I don't expect), the outputs were used for training the SVM model (with which we obtained one among the best results and trained by my teammate Swadhin). We used the same kernel and C value for training.

CLASSIFICATION REPORT:

	precision	recall	f1-score	support
0	0.98	0.99	0.98	1205
1	0.99	0.99	0.99	1379
2	0.95	0.97	0.96	1166
3	0.96	0.96	0.96	1208
4	0.96	0.96	0.96	1153
5	0.96	0.95	0.95	1075
6	0.98	0.97	0.98	1190
7	0.98	0.96	0.97	1228
8	0.94	0.97	0.95	1191
9	0.96	0.94	0.95	1205
accuracy			0.97	12000
macro avg	0.97	0.97	0.97	12000
weighted avg	0.97	0.97	0.97	12000

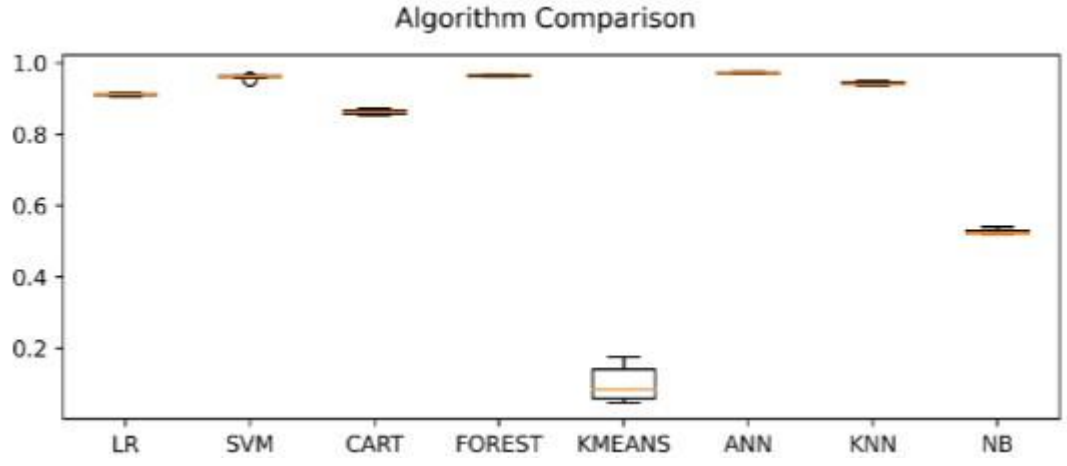
Unlike for the representation where we took only 2 principals, for training after a bit of trial and error**, PCA with 50 principals** achieved the highest accuracy (97%) and it remained constant ± 0.2. The accuracy obtained for **SVM trained on the scaled dataset is more than (98%) what we got employing PCA on the dataset**. We also did the training on other employed algorithms like random forest, decision trees etc., all of them yielded a lesser accuracy from previously obtained results. This is because as the **original images are reshaped into independent vectors, PCA doesn't respect the spatial relationship between the pixels** resulting in lesser prediction accuracy.

CONCLUSION

The accuracy obtained by algorithms discussed in this report is as given below:

Algorithm	Accuracy
MLE	67%
ANN	98%
KNN	94%
SVM	98%
PCA+SVM	97%

From the above table it is clear that **SVM and ANN gives the best training accuracy** of 98%. **MLE faces the problem of overfitting and KNN even though the accuracy is impressive but has high time complexity**. PCA doesn't improve the dataset representation as MNIST is already balanced and well distributed.



**Logistic Regression, SVM, K-means, Decision tree and Random forest are done by my teammate Swadhin Agarwal.*