SMAI Mini Project 2
Report

Introduction

Dataset

For the purpose of this mini project, the CIFAR10 dataset has been used, which consists of 60,000 images each of dimensions 32x32. Due to technical and time constraints, a reduced version of the dataset has been used - only 20% images from the dataset have been used.

Representations

The following three methods have been used:-

1. PCA - Principal Component Analysis
2. LDA - Linear Discriminant Analysis
3. KPCA - Kernel Principal Component Analysis

Each of the above methods has been used to generate a reduced representation of the dataset in order to improve efficiency of the classifier.

Classifiers

The following four classifiers have been used:-

1. Linear SVM - Linear Support Vector Machine - varying C and gamma
2. Kernel SVM - with RBF Kernel - varying C and Gamma
3. MLP - Multi Layer Perceptron - varying all of its parameters
4. Logistic Regression - varying learning rate

Each of the above classifiers have been used with varied parameters along with multiple combinations of the aforementioned representations in order to maximise the accuracy of the classification.

Results

As mentioned in the question, the training data has been split into train and validation sets in the ratio 80:20. The accuracies for all the classifiers with all the representations mentioned above can be found in the table below:-

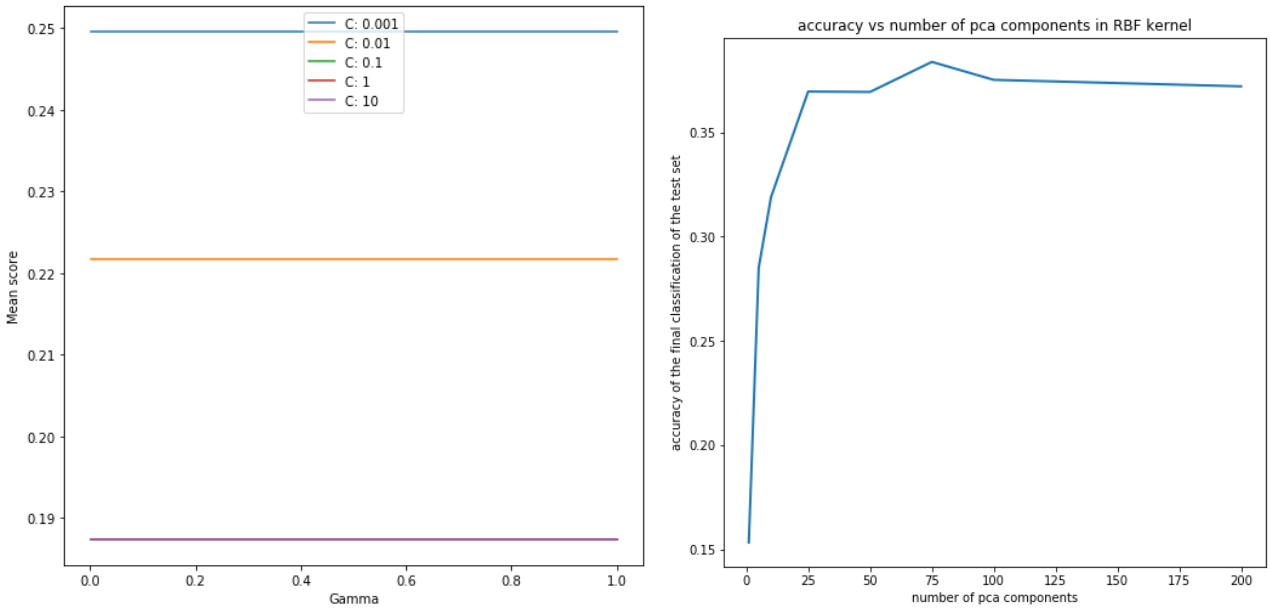
Classifier	Features	Accuracy	F1-Score
Linear SVM	PCA	0.379	0.3789
Linear SVM	LDA	0.2435	0.2435
Linear SVM	KPCA	0.3718	0.3718
RBF Kernel SVM	PCA	0.3602	0.3602
RBF Kernel SVM	LDA	0.2425	0.2425
RBF Kernel SVM	KPCA	0.3477	0.3477
Logistic Regression	PCA	0.3755	0.3755

Logistic Regression	LDA	0.2347	0.2347
Logistic Regression	KPCA	0.3674	0.3674
MLP	PCA	0.402	0.402
MLP	LDA	0.2347	0.2347
MLP	KPCA	0.42789	0.4279

1. Linear SVM

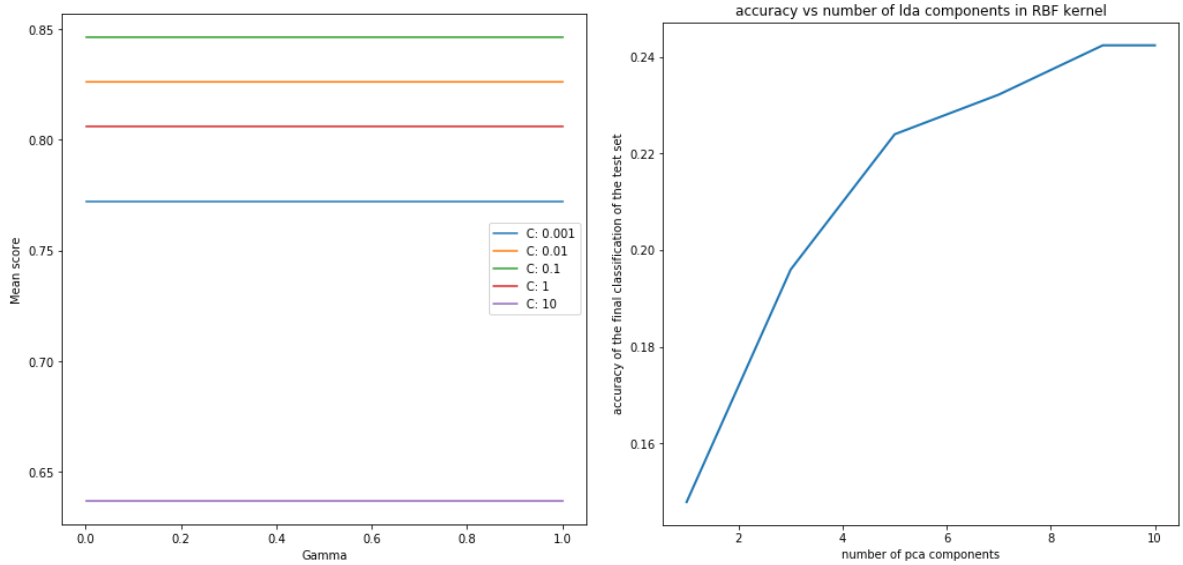
For a linear SVM, values C and gamma need to be selected.

1.1 PCA



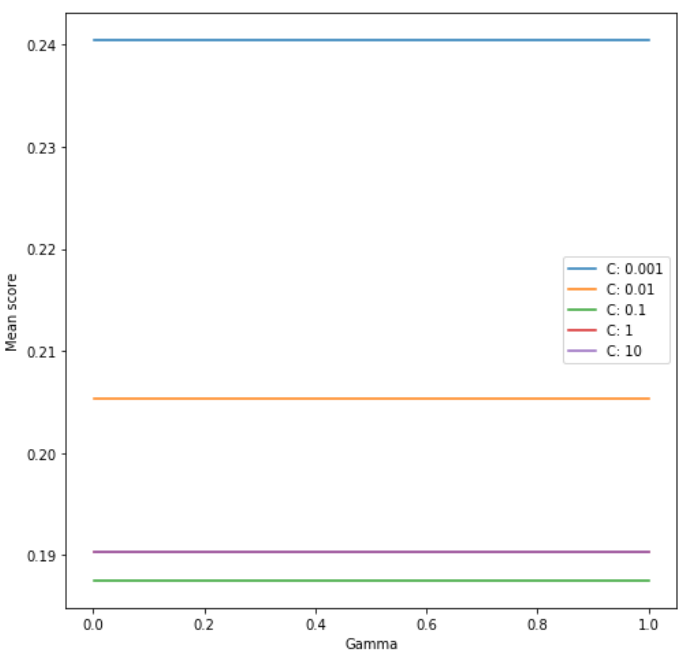
It is clear that the optimal parameters for C and gamma from the first graph are 0.001 and 0.001 respectively. From the second graph, it seems like we hit peak accuracy when the number of components is around 70.

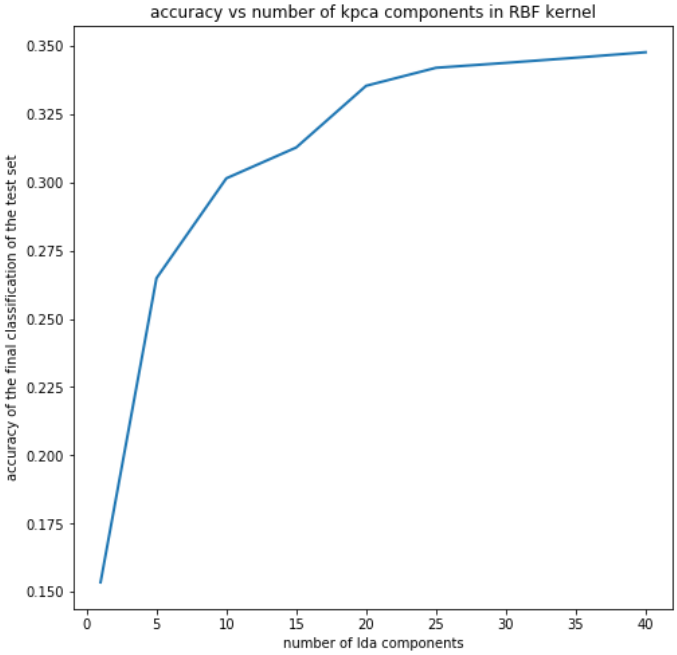
1.2 LDA



It is clear that the optimal parameters for C and gamma from the first graph are 0.1 and 0.001 respectively. From the second graph, it seems like we hit peak accuracy when the number of components is around 9.

1.3 KPCA

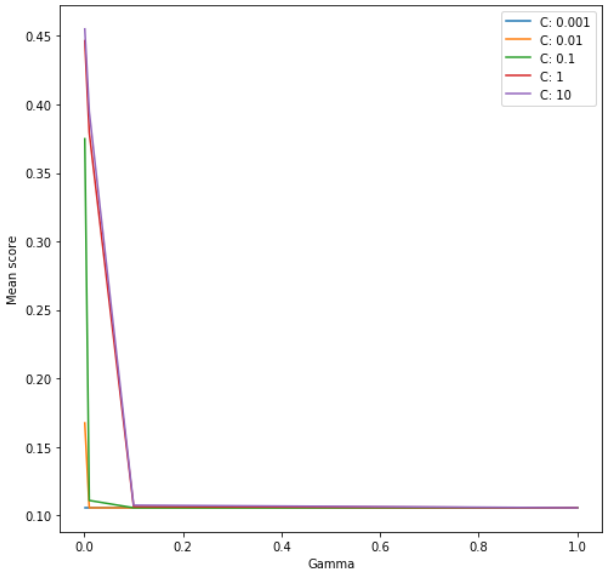


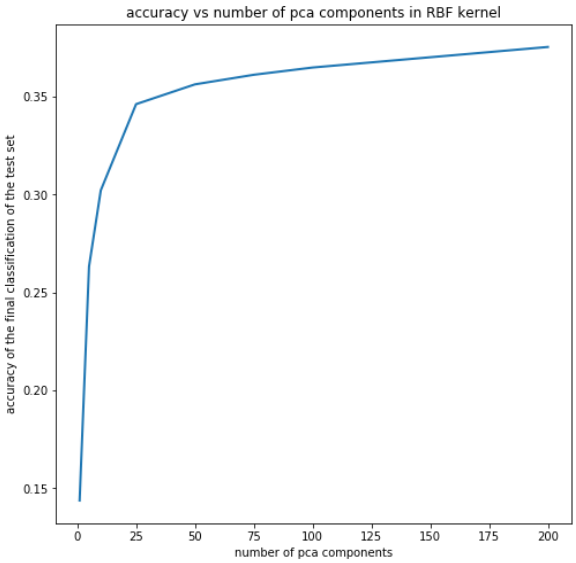


It is clear that the optimal parameters for C and gamma from the first graph are 0.001 and 0.001 respectively. From the second graph, it seems like we hit peak accuracy when the number of components is around 30.

2. RBF Kernel SVM

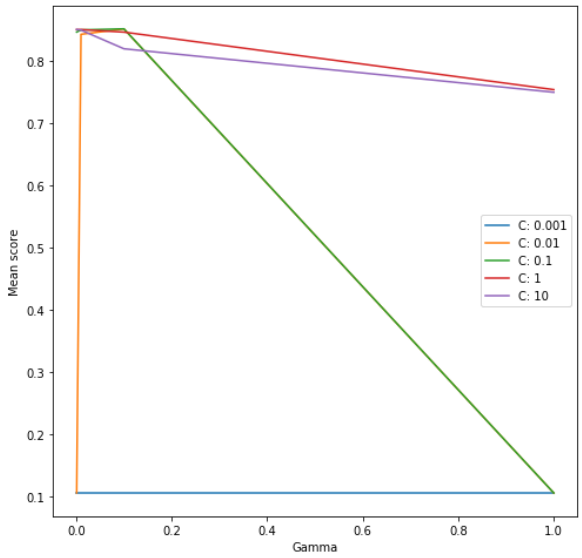
2.1 PCA

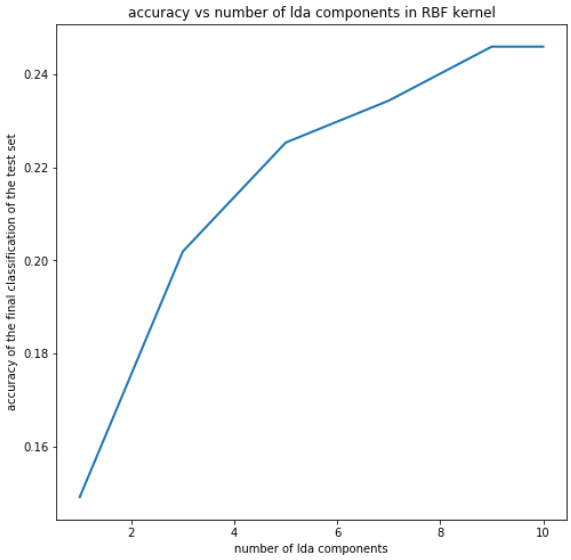




It is clear that the optimal parameters for C and gamma from the first graph are 10 and 0.001 respectively. From the second graph, it seems like we hit peak accuracy when the number of components is around 50.

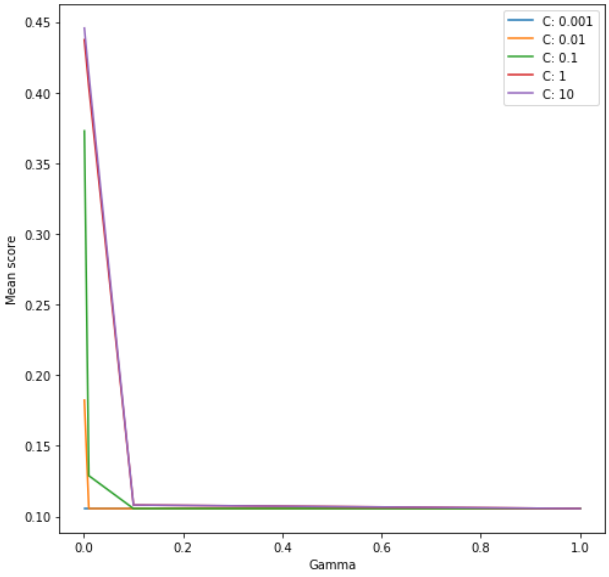
2.2 LDA

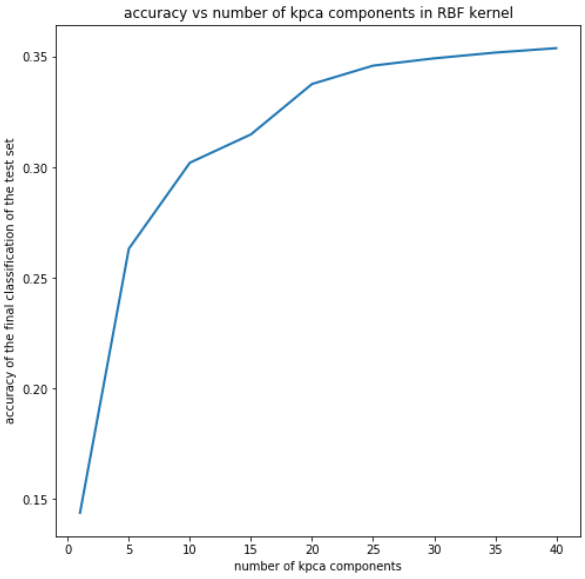




It is clear that the optimal parameters for C and gamma from the first graph are 0.1 and 0.1 respectively. From the second graph, it seems like we hit peak accuracy when the number of components is around 9.

2.3 KPCA

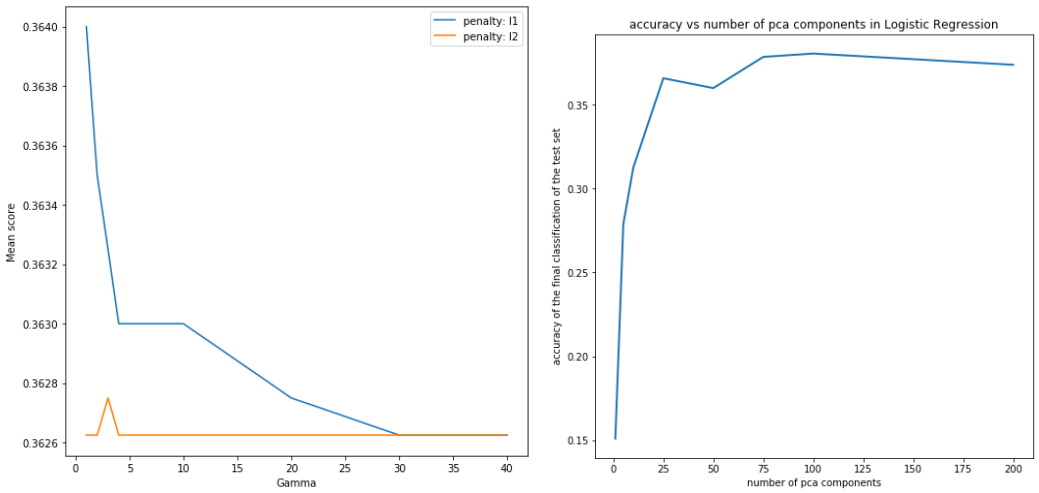




It is clear that the optimal parameters for C and gamma from the first graph are 10 and 0.001 respectively. From the second graph, it seems like we hit peak accuracy when the number of components is around 30.

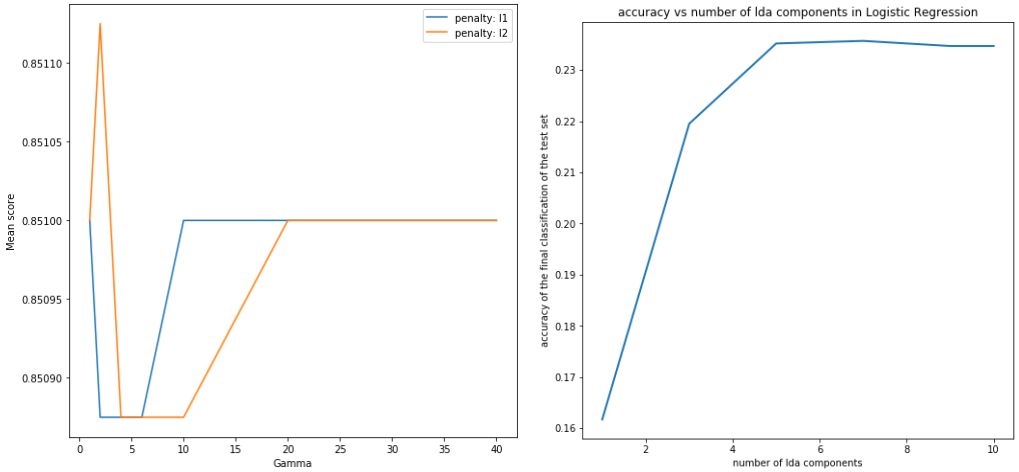
3. Logistic Regression

3.1 PCA



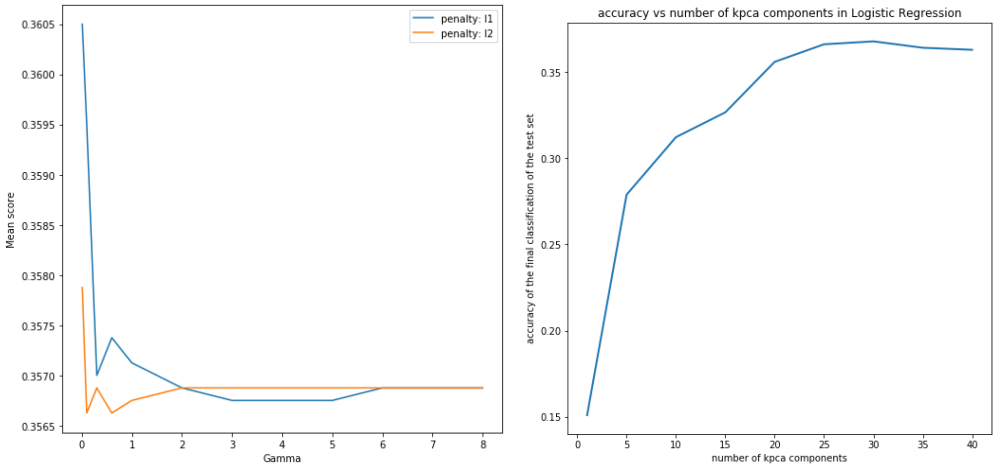
It is clear that the optimal parameters for C and penalty from the first graph are 1 and 11 respectively. From the second graph, it seems like we hit peak accuracy when the number of components is around 100.

3.2 LDA



It is clear that the optimal parameters for C and penalty from the first graph are 2 and 12 respectively. From the second graph, it seems like we hit peak accuracy when the number of components is around 16.

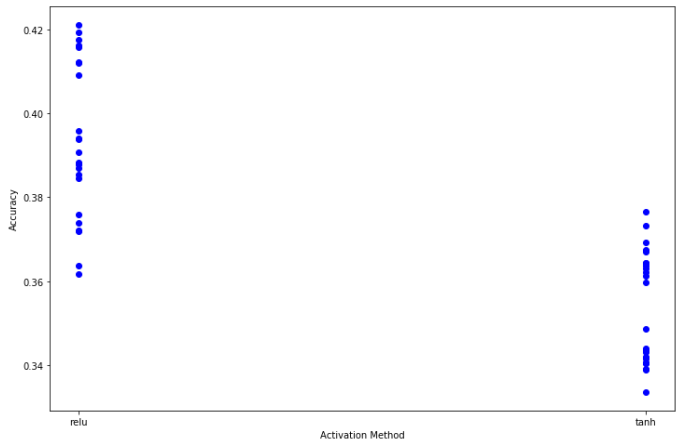
3.3 KPCA

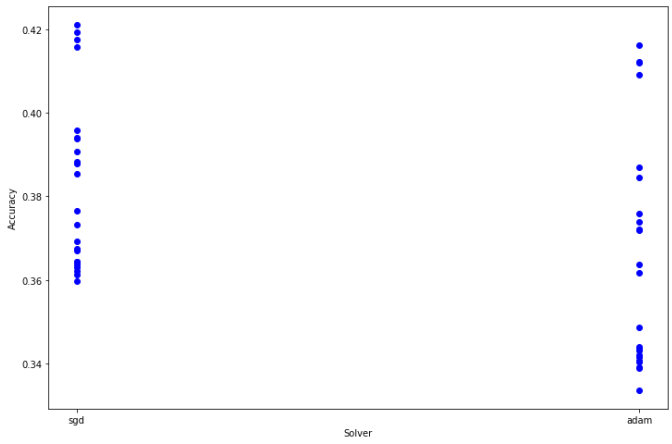


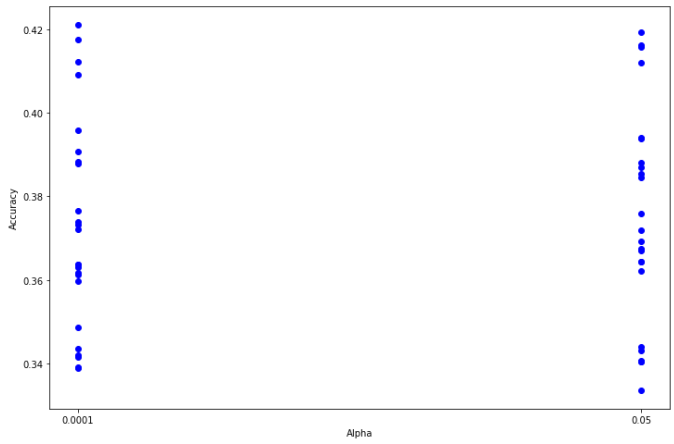
It is clear that the optimal parameters for C and penalty from the first graph are 0.01 and 11 respectively. From the second graph, it seems like we hit peak accuracy when the number of components is around 30.

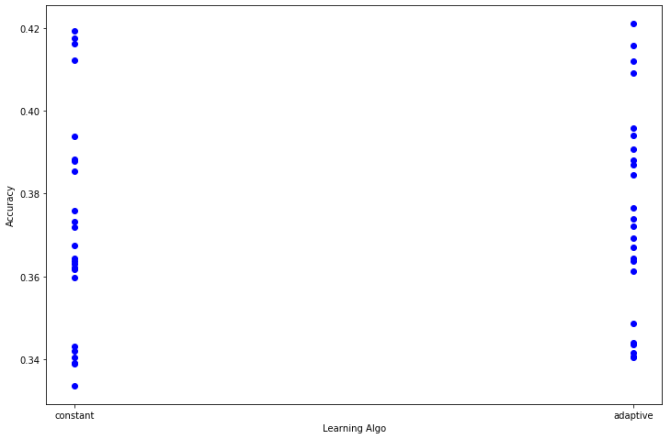
4. Multi Layer Perceptron

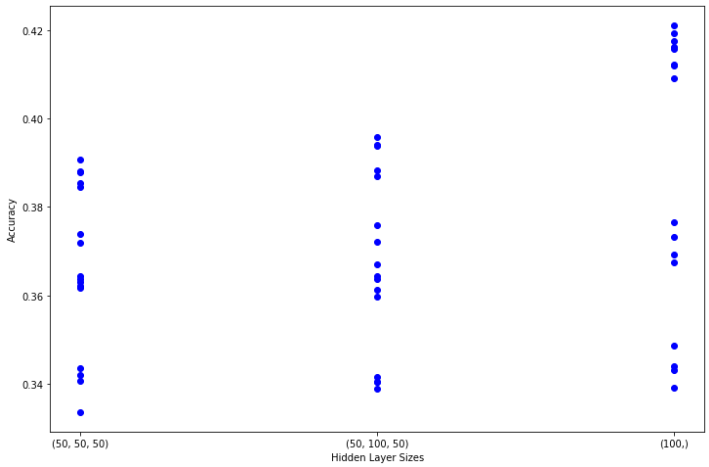
4.1 PCA





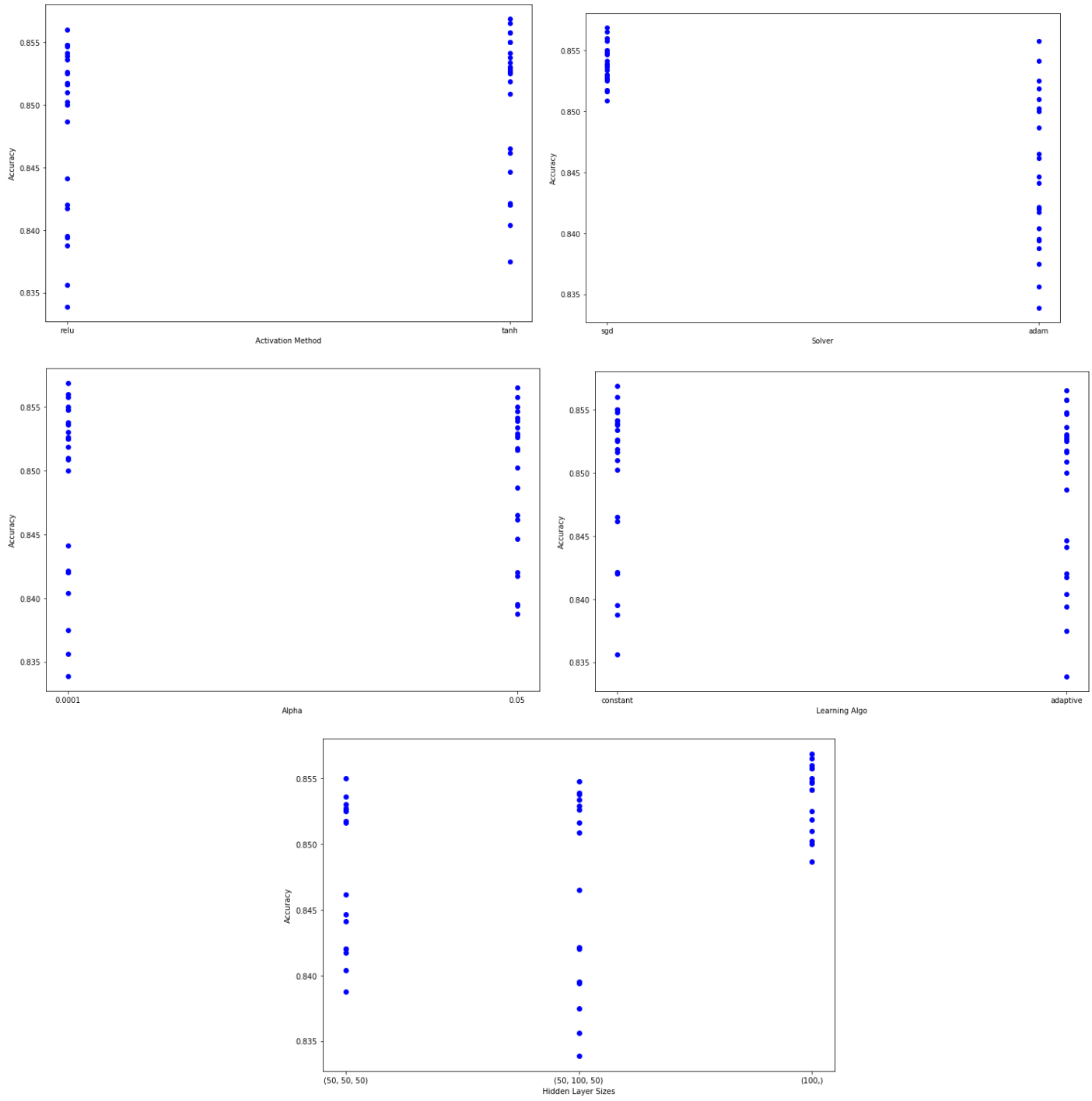






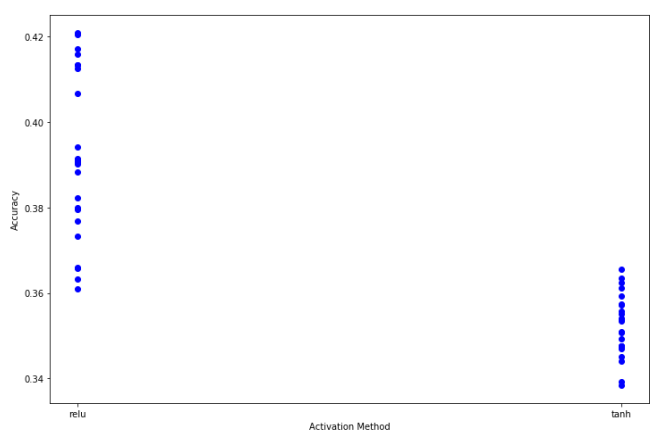
From left to right and top to bottom, first graph shows **relu gives better accuracy** than tanh as an activation function. Second graph shows **sgd is a better solver** than adam, third graph shows **alpha as 0.0001** is better than 0.05, fourth graph shows an **adaptive learning algorithm** is better than a constant one, and fifth graph shows **100 is the optimal number of hidden layer size**.

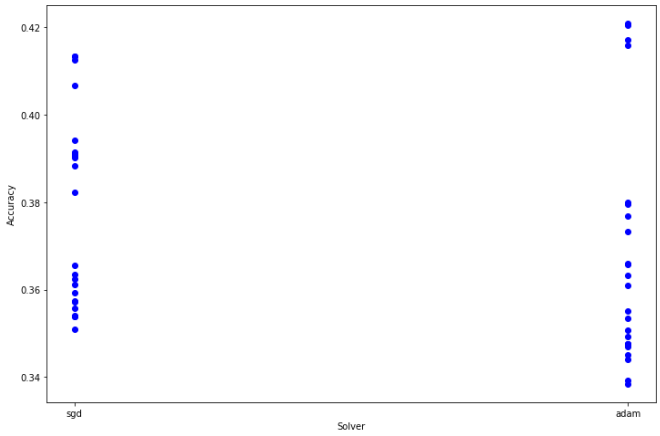
4.2 LDA

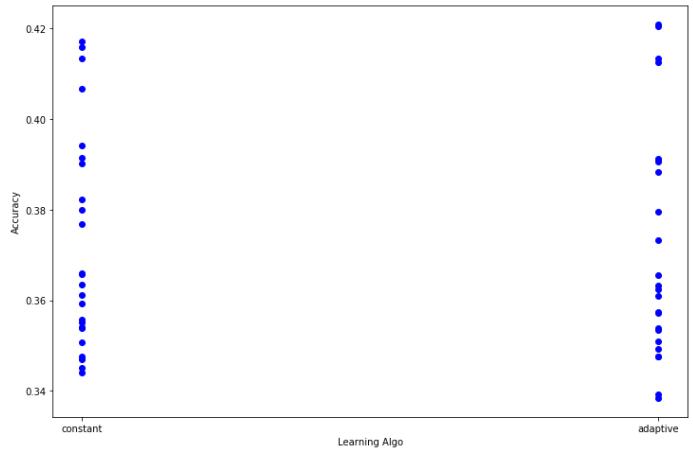


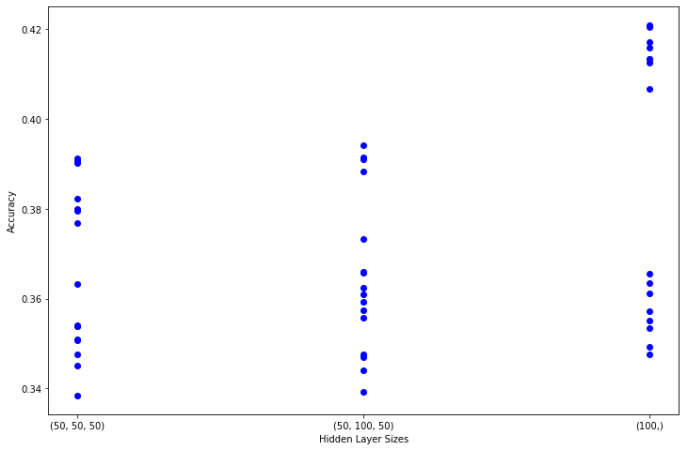
From left to right and top to bottom, first graph shows **tanh gives better accuracy** than relu as an activation function. Second graph shows **sgd is a better solver** than adam, third graph shows **alpha as 0.0001** is better than 0.05, fourth graph shows a **constant learning algorithm** is better than an adaptive one, and fifth graph shows **100 is the optimal number of hidden layer size**.

4.3 KPCA



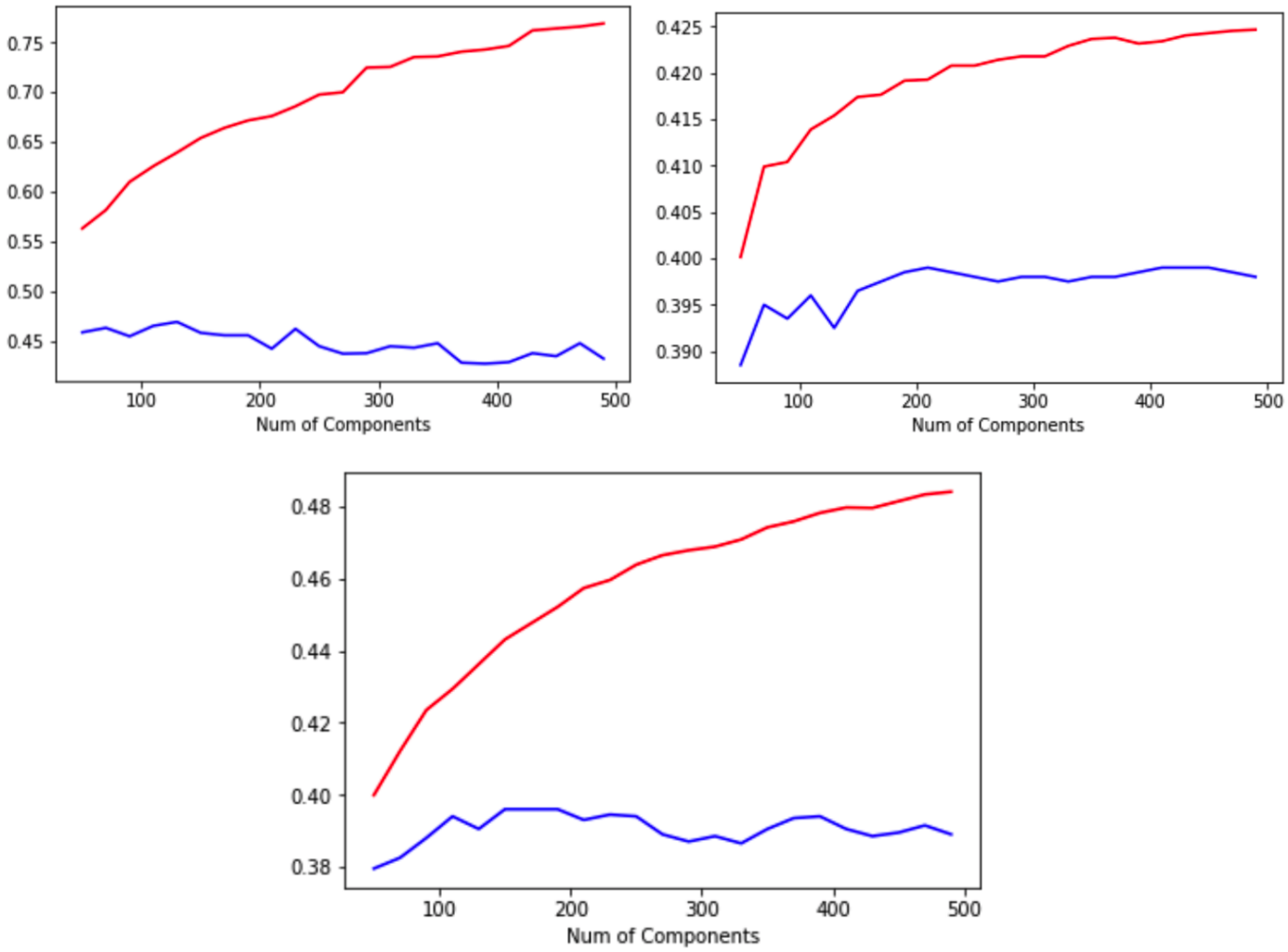






From left to right and top to bottom, first graph shows **relu gives better accuracy** than tanh as an activation function. Second graph shows **sgd is a better solver** than adam, third graph shows **alpha as 0.05** is better than 0.0001, fourth graph shows an **adaptive learning algorithm** is better than a constant one, and fifth graph shows **100 is the optimal number of hidden layer size**.

Overfitting



From left to right and top to bottom, the first graph is for an SVM, second for logistic regression, and third for multi layer perceptron. Across all three graphs, the red line indicates accuracy of the respective classifier for training data while the blue line indicates accuracy for the testing data. In theory, as we take a higher number of components the accuracies should increase. However, it is clear that the blue line in all the graphs, i.e., accuracy for the testing data reaches a peak and then decreases as we increase the number of components. This occurs due to the overfitting of the respective classifier model.

Summary

For all of the above results, inbuilt functions in sklearn have been used. The relevant code has been attached as a Jupyter Notebook along with this report.

Some practical issues:-

1. Size of dataset - The CIFAR-10 dataset consists of 10 classes and a total of 60,000 images. Due to the high number of images, it becomes very difficult to apply classical machine learning techniques optimally.
2. Training time - The time taken to train each model is exorbitantly high, and it becomes very difficult to practically use it without reducing the dataset.
3. Reduction of dataset - By reducing the dataset used for the training, we risk reducing the amount of information available to each classifier and heavily affect its performance.
4. Accuracy - The accuracy of these models rarely crossed 0.5 during my testing across all combinations of models and representations.
5. GPU support - Even with access to GPUs, some inbuilt functions don't offer GPU support in order to increase efficiency. For example, most of sklearn doesn't support GPU optimization.
6. Hyperparameter selection - While selecting parameters, one was varied while the rest remained constant. However, the parameters may not be linearly additive.