Name : Lakshmee Sravya
Roll Number : 20161084

REPORT

Results obtained :

Classifier	Features	Accuracy	F1-score
MLP	Principal Components (PCA)	43.6%	43.6
MLP	LDA	22.4%	22.4
MLP	Raw-data	42.24%	42.24
Logistic Regression	Principal Components (PCA)	38.95%	38.95
Logistic Regression	LDA	23.95%	23.95
Logistic Regression	Raw-data	26.86%	26.86
Random Forest	Principal Components (PCA)	46.8%	46.8
Random Forest	LDA	24.25%	24.25
Random Forest	Raw-data	44.01%	44.01
SVM	Principal Components (PCA)	47.61%	47.61
SVM	LDA	25.65%	25.65
SVM	Raw-data	46.14%	46.14

Classifier 1 : Multi-layer Perceptron (MLP)

MLP is a class of artificial neural network. An MLP consists of, at least, three layers of nodes: an input layer, a hidden layer and an output layer.
Each node is a neuron that uses a nonlinear activation function except for the input nodes. A supervised learning technique called back propagation is used for training the MLP.

I have implemented MLP classifier using the 'sklearn' library from python.

Hyperparameters optimised :

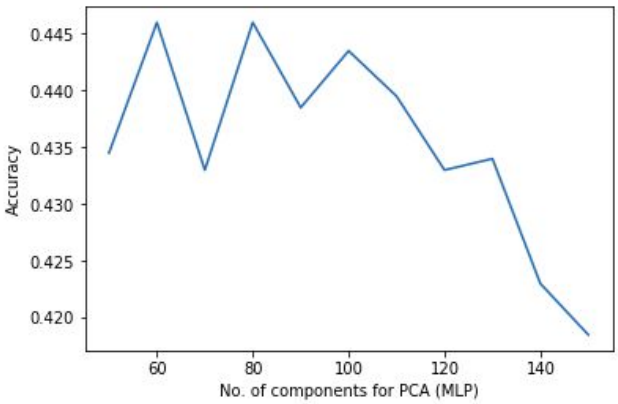
1) **Number of principal components (Both for PCA and LDA)**
2) **Number of hidden layers**
   - Can vary from 1 to any number
3) **Size of each of the hidden layers**
   - The ith element in a tuple represents the number of neurons in the ith hidden layer (its size)
4) **Maximum number of iterations**
   - This refers to the maximum number of iterations till convergence or this number of iterations.

The rest of the parameters take their default values.

Each of these hyperparameters have been optimised separately for when the data is reduced using PCA (or) LDA (or) rawdata.

Optimising MLP when the data is reduced using PCA :

- ***Optimising the number of PCA components :***



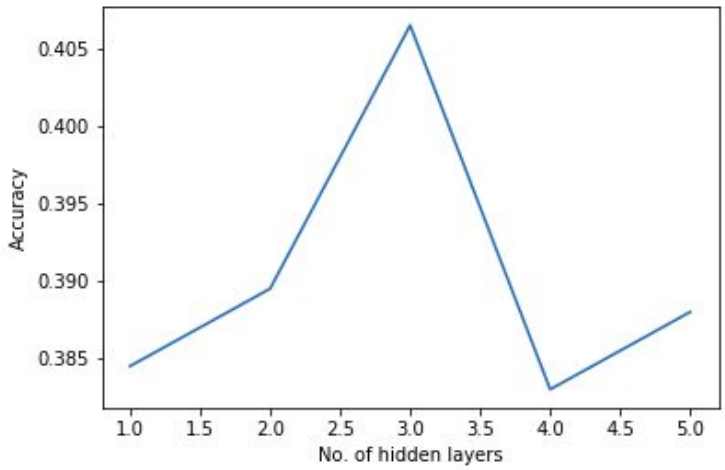
Keeping the other hyperparameters (number of hidden layers, size of each layer, and max number of iterations) constant, I have computed the accuracy of the MLP classifier each time, varying only the number of PCA components from 50 to 150.
The above graph (No. of PCA components vs Accuracy) is plotted.

From the graph, we can see that the optimal value of number of PCA components is **100.**

- ***Optimising the number of hidden layers :***

Reducing the data with PCA (number of components is kept 100 - optimal). Keeping the maximum number of iterations (5000) and the size of each hidden layer constant (100) , the number of hidden layers is varied between 1 to 5 and the graph is plotted between the number of hidden layers and accuracy.

Fig 2

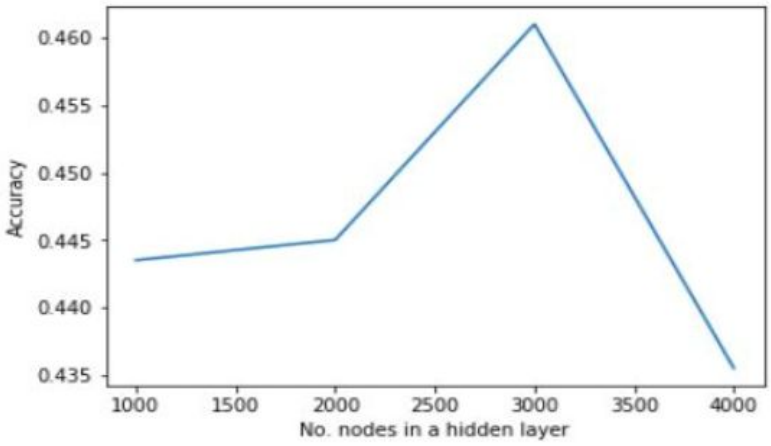


From the above graph, we can see that the optimal value of number of hidden layers is **3.**

- ***Optimising the size of each hidden layer (the number of neurons in each of the hidden layers) :***

Here we consider equal number of neurons in all hidden layers as there are infinite number of ways in which we can assign unequal neurons across the hidden layers and also we don't know how one hidden layer is different from the other one.

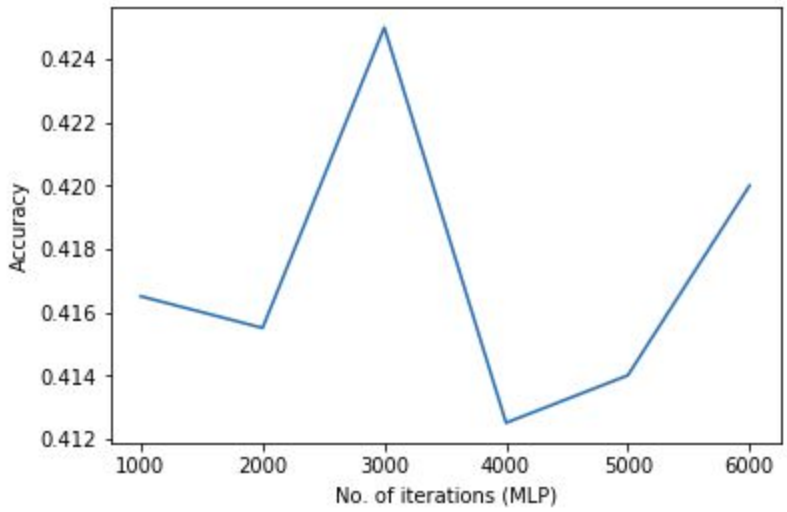
Reducing the data with PCA (number of components is kept 100 - optimal). Keeping the maximum number of iterations (5000) to be constant, the size of each of the **3** hidden layers (3 found to optimal above) is varied between 1000 and 5000 (all the hidden layers are given the same size) and the graph is plotted between the size of each of the hidden layer and accuracy.



From the above graph, we can see that the optimal value of size of each of the hidden layer is **3000.**

- ***Optimising the maximum number of iterations :***

Reducing the data with PCA (number of components is kept 100 - optimal). Keeping the size of each of the 3 hidden layers constant (200 each) the maximum of iterations is varied between 1000 and 6000 and the graph is plotted between the maximum number of iterations and accuracy.



From the above graph, we can see that the optimal value of maximum number of iterations is **3000.**

Now all the optimal values of the hyperparameters is obtained (when the data is reduced with PCA). The optimal values are as follows :

- Number of PCA components = 100
- Number of hidden layers = 3
- Size of each hidden layer = 3000
- Maximum number of iterations = 3000

Finally, tested the MLP classifier with above found hyperparameters giving best accuracy on a new test data of the given dataset.

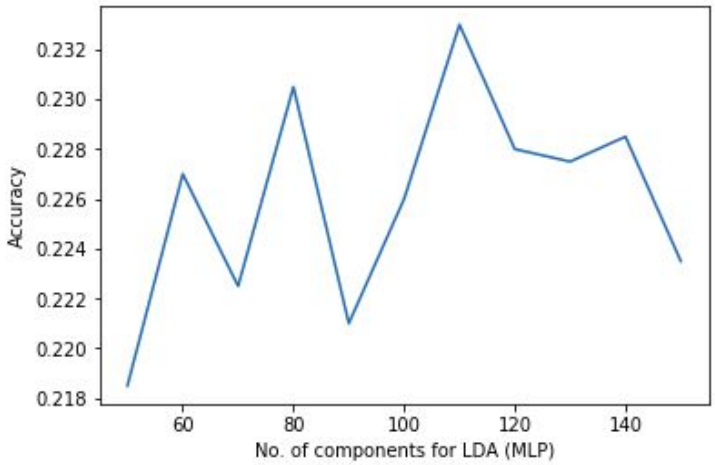
Accuracy obtained : 0.4359999999994 (43.6%)
F-score obtained : 0.436 (43.6)

- ***Optimising MLP when the data is reduced using LDA :***

The same approach is followed as above (One hyperparameter is varied while keeping the other hyperparameters constant) Then the optimal value of that hyperparameter is obtained through the graph plotted.

Results obtained when data is reduced using LDA :

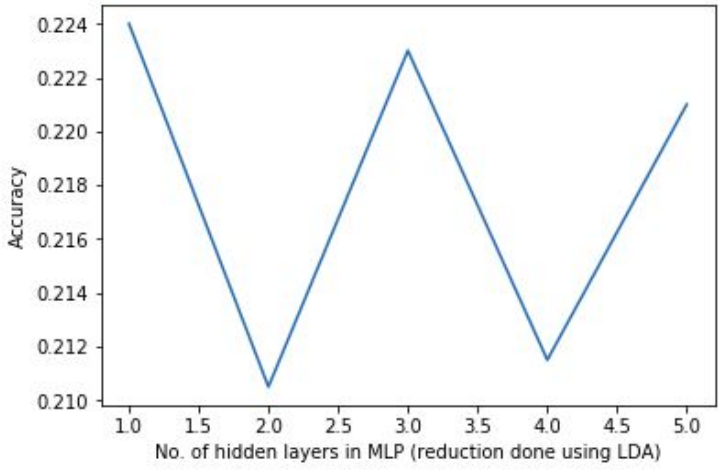
- ***Optimising the number of LDA components :***



From the graph, we can see that the optimal value of number of LDA components is **110.**

- ***Optimising the number of hidden layers :***

Reducing the data with LDA (number of components is kept 110 - optimal). Keeping the maximum number of iterations (5000) and the size of each hidden layer constant (100) , the number of hidden layers is varied between 1 to 5 and the graph is plotted between the number of hidden layers and accuracy.



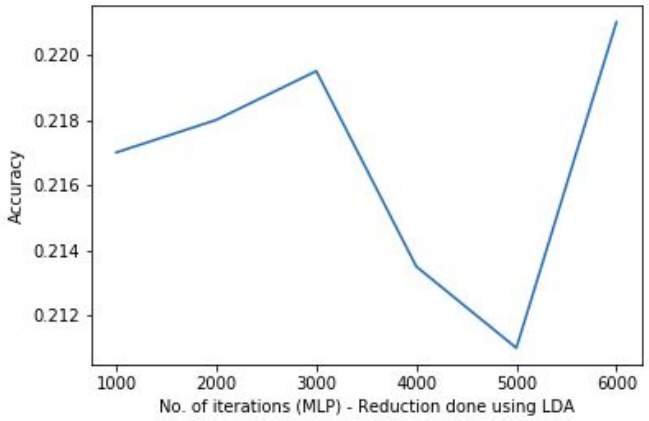
From the above graph, we can see that the optimal value of number of hidden layers is **3.**

- ***Optimising the size of each hidden layer :***

Reducing the data with LDA (number of components is kept 110 - optimal). Keeping the maximum number of iterations (5000) to be constant, the size of each of the **3** hidden layers (3 found to optimal above) is varied between 1000 and 5000 and the graph is plotted between the size of each of the hidden layer and accuracy.

- ***Optimising the maximum number of iterations :***

Reducing the data with LDA (number of components is kept 110 - optimal). Keeping the size of each of the 3 hidden layers constant (200 each) the maximum of iterations is varied between 1000 and 6000 and the graph is plotted between the maximum number of iterations and accuracy.



From the above graph, we can see that the optimal value of maximum number of iterations is **6000.**

Now all the optimal values of the hyperparameters is obtained (when the data is reduced with LDA). The optimal values are as follows :

- Number of LDA components = 110
- Number of hidden layers = 3
- Size of each hidden layer = 2000
- Maximum number of iterations = 6000

MLP classifier is run with the above mentioned parameters.

Accuracy obtained : 0.224 (22.4%)

F-score obtained : 0.224 (22.4)

Analysis : The accuracy obtained using MLP classifier almost becomes half when the data is reduced using LDA instead of PCA.

Similarly when we run the MLP classifier with the above found optimal hyperparameters the results obtained are as follows :

Accuracy obtained : 0.4224 (42.24%)
F-score obtained : 0.4224 (42.24)

Classifier 2 : Decision-Tree

Hyperparameters optimised :

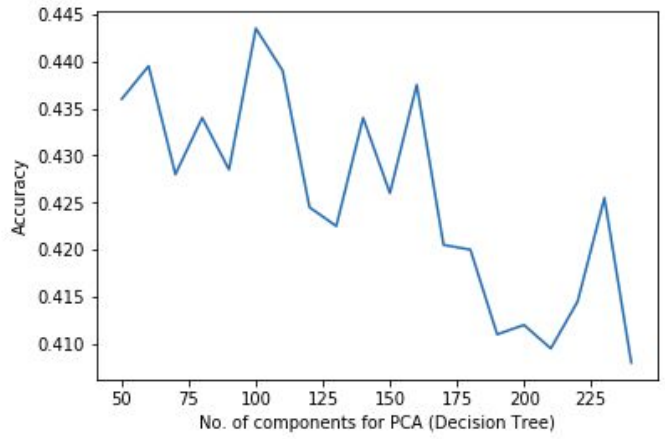
1) Number of principal components (Both for PCA and LDA)
2) 'min_samples_split' (The minimum number of nodes in a node such that we stop splitting the node in the decision tree - 'early stopping')
3) 'n_estimators' -> the number of trees to merge

Each of these hyperparameters have been optimised separately for when the data is reduced using PCA (or) LDA (or) rawdata.

Optimising Decision Tree when the data is reduced using PCA :

- ***Optimising the number of PCA components :***

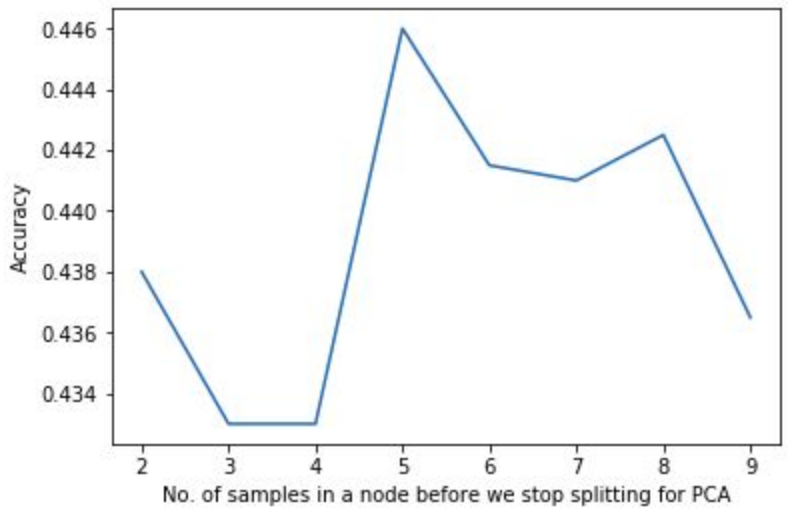
Keeping the other hyperparameters (min samples split, numtrees) constant (assigned 5 and 300 respectively), I have computed the accuracy of the Decision tree classifier each time, varying only the number of PCA components from 50 to 250.
The above graph (No. of PCA components vs Accuracy) is plotted.



From the graph, we can see that the optimal value of number of PCA components is **100.**

- ***Optimising the min samples split :***

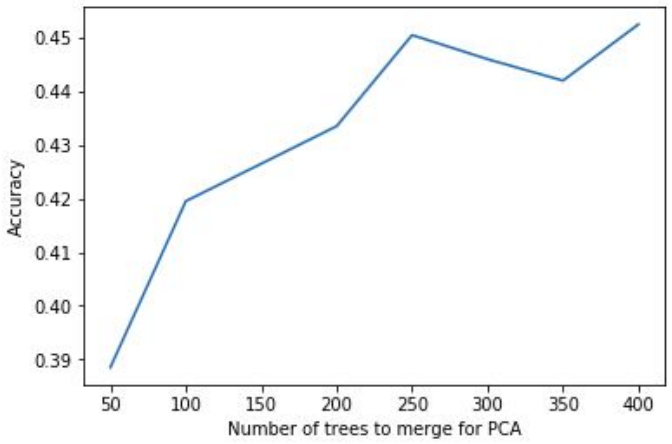
Reducing the data with PCA (number of components is kept 100 - optimal). Keeping the 'numtree' parameter constant (300), the 'min samples split' parameter is varied between 1 to 10 and the graph is plotted between this parameter and accuracy.



From the above graph, we can see that the optimal value of the parameter 'min samples split' is **5.**

- ***Optimising the 'n_estimators' parameter :***

Reducing the data with PCA (number of components is kept 100 - optimal). Keeping the 'min samples split' parameter (5 - found optimal above) to be constant, the 'numtree' parameter is varied between 50 and 400 and the graph is plotted between this parameter and accuracy.



From the above graph, we can see that the optimal value of **n_estimators** parameter is **250.**

Now all the optimal values of the hyperparameters is obtained (when the data is reduced with PCA). The optimal values are as follows :

- Number of PCA components = **100**
- Min samples split parameter (minimum number of nodes in a node such that we stop splitting the node in the decision tree) = **5**
- 'n_estimators' parameter (the number of trees to merge) = **250**

Decision classifier is run with the above mentioned parameters.

Accuracy obtained : 0.468

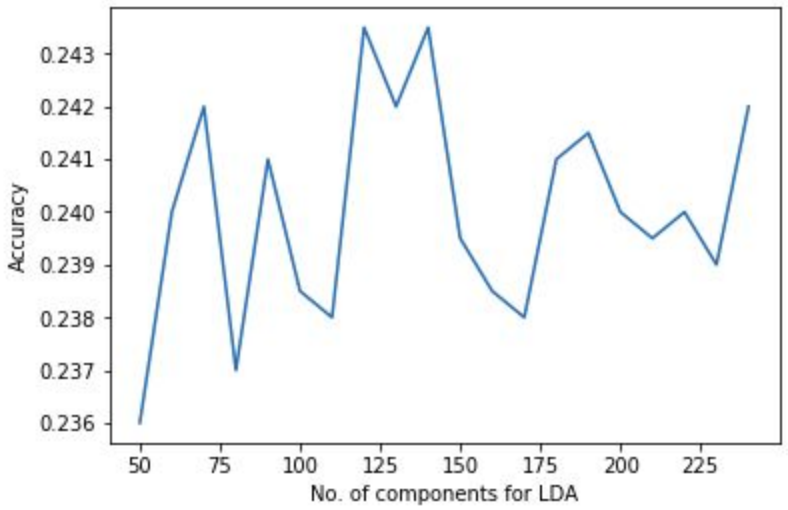
F-score obtained : 0.468

- ***Optimising Decision Tree classifier when the data is reduced using LDA :***

The same approach is followed as above (One hyperparameter is varied while keeping the other hyperparameters constant) Then the optimal value of that hyperparameter is obtained through the graph plotted.

Results obtained when data is reduced using LDA :

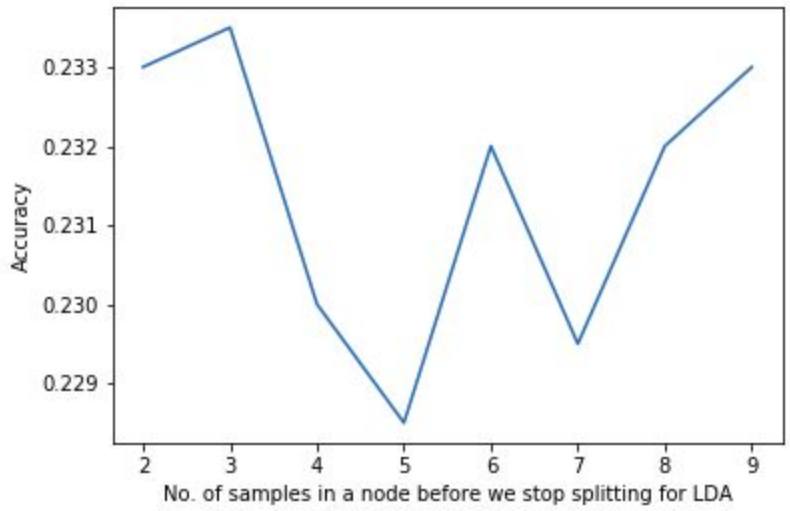
- ***Optimising the number of LDA components :***



From the graph, we can see that the optimal value of number of LDA components is **120.**

- ***Optimising the min samples split :***

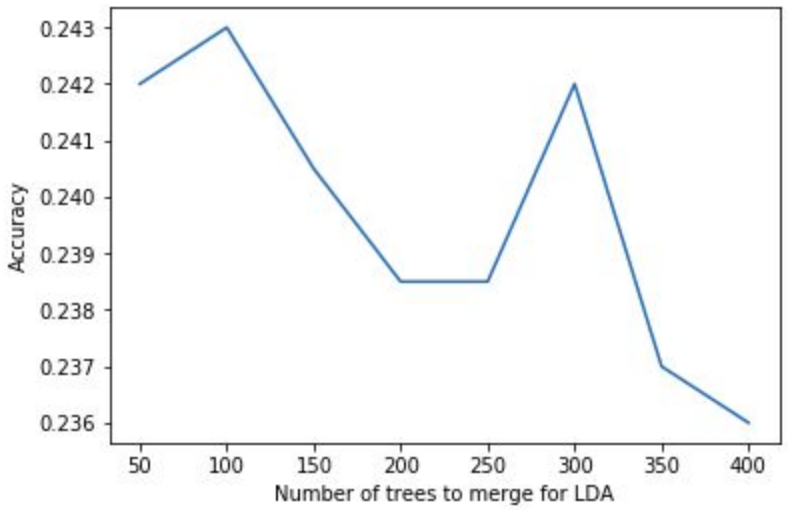
Reducing the data with LDA (number of components is kept 120 - optimal). Keeping the 'numtree' parameter constant (300), the 'min samples split' parameter is varied between 1 to 10 and the graph is plotted between this parameter and accuracy.



From the above graph, we can see that the optimal value of the parameter 'min samples split' is **3.**

- ***Optimising the* 'n_estimators' *parameter :***

Reducing the data with LDA (number of components is kept 120 - optimal). Keeping the 'min samples split' parameter (3 - found optimal above) to be constant, the 'numtree' parameter is varied between 50 and 400 and the graph is plotted between this parameter and accuracy.



From the above graph, we can see that the optimal value of **n_estimators** parameter is **300.**

Now all the optimal values of the hyperparameters is obtained (when the data is reduced with LDA). The optimal values are as follows :

- Number of LDA components = 120
- Min samples split parameter = 3
- 'n_estimators' parameter = 300

Decision Tree classifier is run with the above mentioned parameters.

Accuracy obtained : 0.2425
F-score obtained : 0.2425

Analysis : The accuracy obtained using Decision Tree classifier almost becomes half when the data is reduced using LDA instead of PCA.

Similarly when we run the Decision Tree classifier with the above found optimal hyperparameters the results obtained are as follows :

Accuracy obtained : 0.4401 (44.01%)
F-score obtained : 0.4401 (44.01)

Classifier 3 : Logistic Regression (LR)

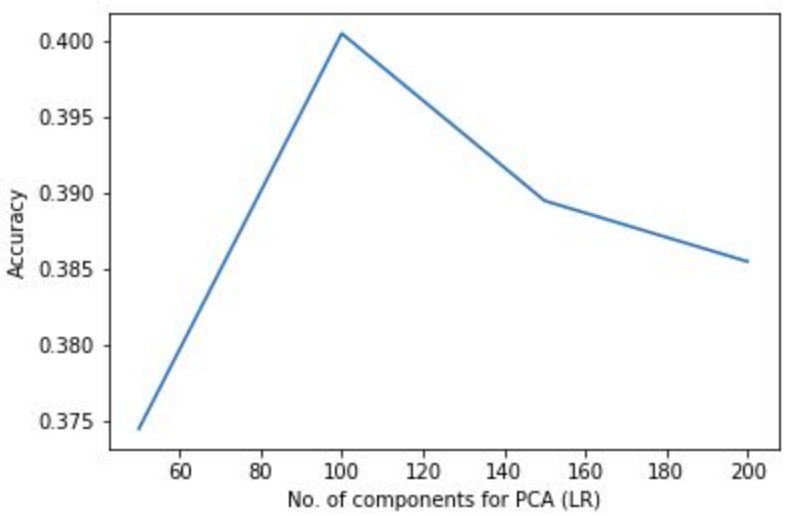
Hyperparameters optimised :

1) **Number of principal components** (Both for PCA and LDA)
2) **Max number of iterations** (This refers to the maximum number of iterations taken for the solvers to converge)
3) **'C' value** (C is the inverse of regularization strength; must be a positive float. Like in SVMs, smaller values specify stronger regularization.)

Each of these hyperparameters have been optimised separately for when the data is reduced using PCA (or) LDA (or) rawdata.

Optimising Logistic Regression when the data is reduced using PCA :

- ***Optimising the number of PCA components :***

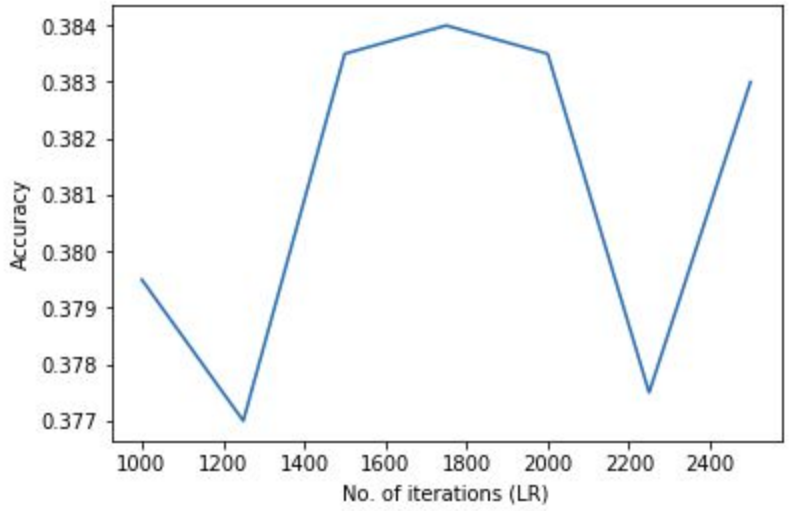


Keeping the other hyperparameters (max number of iterations, and c value) constant (2000 and 5 respectively) , I have computed the accuracy of the Logistic Regression classifier each time, varying only the number of PCA components from 50 to 250.
The above graph (No. of PCA components vs Accuracy) is plotted.

From the graph, we can see that the optimal value of number of PCA components is **100.**

- ***Optimising the maximum number of iterations :***

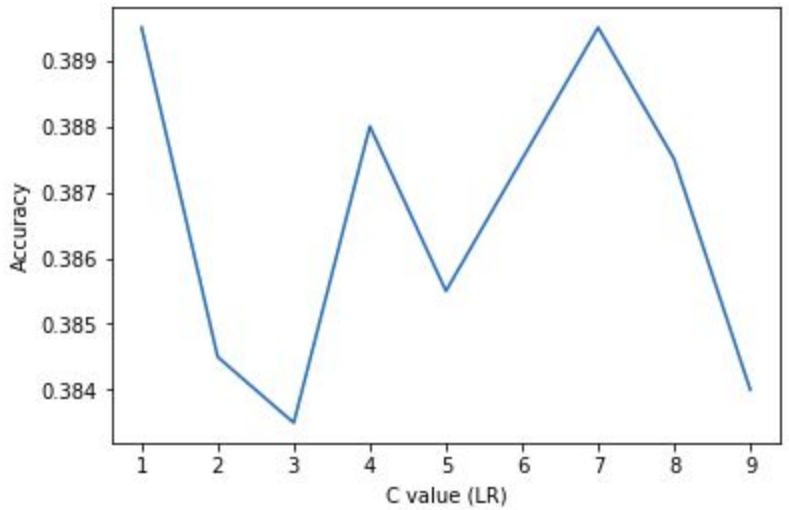
Reducing the data with PCA (number of components is kept 100 - optimal). Keeping the c value (5) constant, the maximum number of iterations is varied between 1000 to 2500 and the graph is plotted between the number of maximum number of iterations and accuracy.



From the graph, we can see that the optimal value of maximum number of iterations is **1700.**

- ***Optimising the c value :***

Reducing the data with PCA (number of components is kept 100 - optimal). Keeping the maximum number of iterations 1700 (optimal value found above) the c value is varied between 1 and 10 and the graph is plotted between the c value and accuracy.



From the graph, we can see that the optimal value of c is **7.**

Now all the optimal values of the hyperparameters is obtained (when the data is reduced with PCA). The optimal values are as follows :

- Number of PCA components = 100
- maximum number of iterations = 1700
- C value = 7

Logistic Regression classifier is run with the above mentioned parameters.

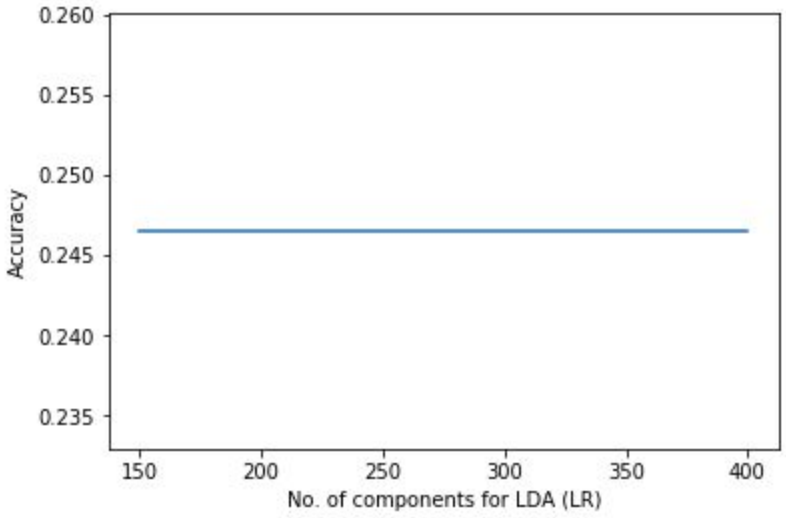
Accuracy obtained : 0.3895
F-score obtained : 0.3895

- ***Optimising Logistic Regression classifier when the data is reduced using LDA :***

The same approach is followed as above (One hyperparameter is varied while keeping the other hyperparameters constant) Then the optimal value of that hyperparameter is obtained through the graph plotted.

Results obtained when data is reduced using LDA :

- ***Optimising the number of LDA components :***

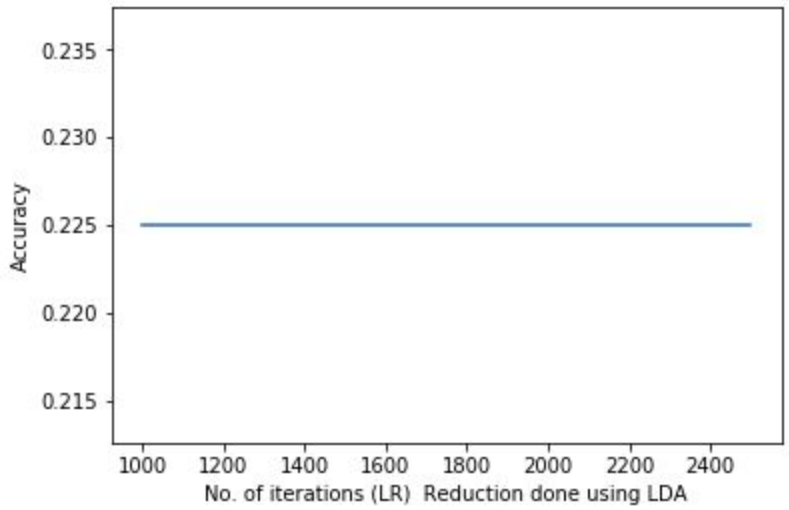


From the graph, we can see that the accuracy of the classifier remains constant irrespective of the number of components.

- ***Optimising the max number of iterations :***

Reducing the data with LDA (number of components is kept 100 - any value is optimal). Keeping the c value (5) constant, the maximum number of iterations is varied between 1000 to 2500 and the graph is

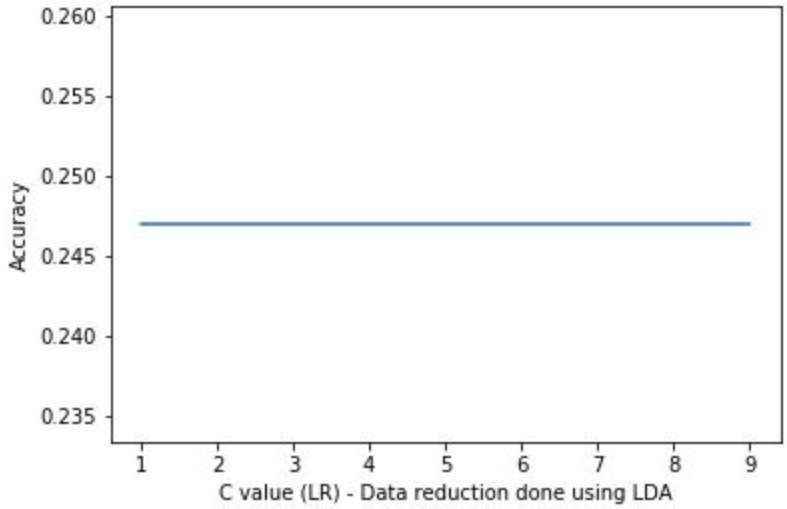
plotted between the number of maximum number of iterations and accuracy.



From the graph, we can see that the accuracy of the classifier remains constant irrespective of the maximum number of iterations.

- ***Optimising the c value :***

Reducing the data with LDA (number of components 100 - as any number of components is optimal). Keeping the maximum number of iterations 2500 (any value is optimal as found above) the c value is varied between 1 and 10 and the graph is plotted between the c value and accuracy.



From the graph, we can see that the accuracy of the classifier remains constant irrespective of the c value also.

Now all the optimal values of the hyperparameters is obtained (when the data is reduced with LDA). The optimal values are as follows :

- Number of LDA components => any value is optimal (doesn't affect)
- Max number of iterations => any value is optimal (doesn't affect)
- C value => any value is optimal (doesn't affect)

Logistic Regression classifier is run with the above mentioned parameters.

Accuracy obtained : 0.2395
F-score obtained : 0.2395

Analysis :

1) The accuracy obtained using Logistic Regression classifier decreases significantly when the data is reduced using LDA instead of PCA.
2) Once the data is reduced using LDA, even changing any of the other parameters does not make any change to the accuracy of the classifier.

Classifier 4 : Support-Vector Machine (SVM)

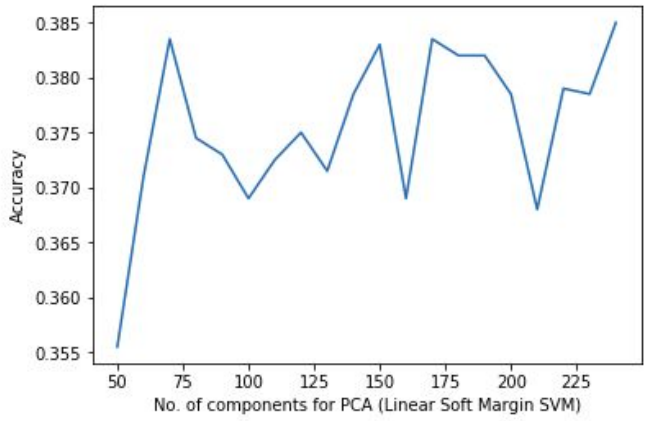
Hyperparameters optimised :

1) **Number of principal components** (Both for PCA and LDA)
2) **'kernel'** (This parameter specifies the type of kernel ('poly','sigmoid', 'linear','rbf') to be used in the SVM classifier. If no particular value is given to this parameter then the default value 'rbf' will be used.)
3) **Maximum number of iterations** (It has the value equal to hard limit on iterations within solver, or -1 for no limit.)

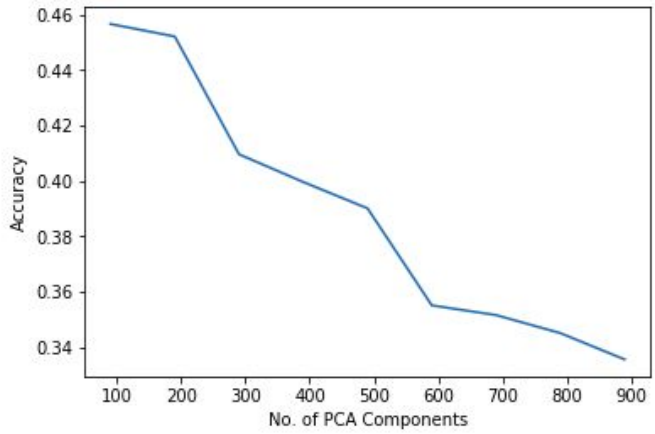
Optimising SVM when the data is reduced using PCA :

- ***Optimising the number of PCA components :***

1) Without including the 'whiten' parameter for PCA the maximum accuracy obtained was just around 0.385.



2) Including the 'whiten' parameter for PCA the maximum accuracy obtained increased significantly to 0.45.



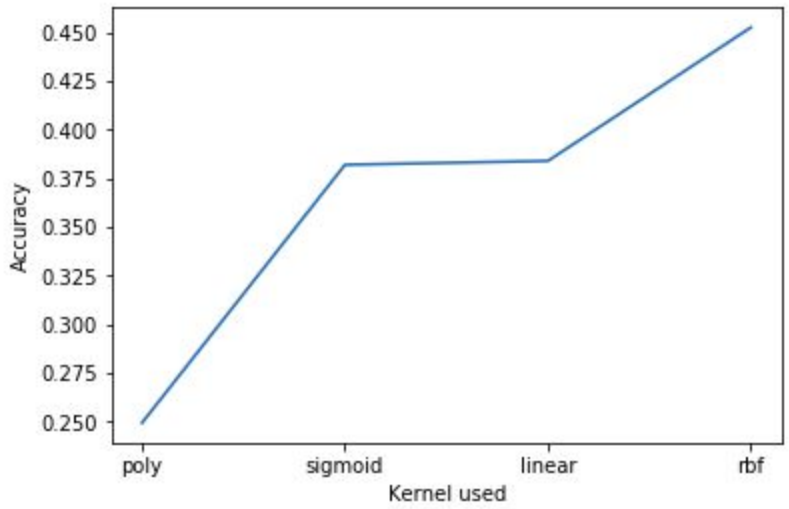
Keeping the other hyperparameters (kernel type : rbf kernel) constant, I have computed the accuracy of the SVM classifier each time, varying only the number of PCA components from 80 to 900. The above graph (No. of PCA components vs Accuracy) is plotted.

From the graph, we can see that the optimal value of number of PCA components is **90**. (whiten parameter mentioned in PCA function)

- ***Optimising the type of kernel :***

Reducing the data with PCA (number of components is kept 90 along with the whiten parameter - optimal as found above). The type of

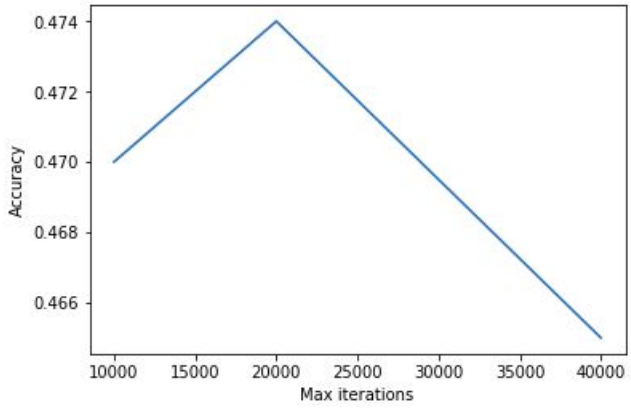
kernel is varied ('poly','sigmoid', 'linear','rbf') and the graph is plotted between the type of kernel and accuracy.



From the graph, it can be inferred that 'rbf' kernel gives the most optimal accuracy of 0.45 and 'poly' kernel gives very low accuracy compared to the above (0.25). Sigmoid and linear kernels gives approximately similar accuracies.

- ***Optimising the maximum number of iterations :***

Reducing the data with PCA (number of components is kept 90 - optimal). Keeping the type of kernel as 'rbf' (optimal as found above) the maximum of iterations is varied between 10000 and 40000 and the graph is plotted between the maximum number of iterations and accuracy.



From the above graph, we can see that the optimal value of maximum number of iterations is **20000.**

Now all the optimal values of the hyperparameters is obtained (when the data is reduced with PCA). The optimal values are as follows :

- Number of PCA components = 90
- Type of kernel = 'rbf'
- Maximum number of iterations = 20000

SVM classifier is run with the above mentioned parameters.

Accuracy obtained : 0.4761
F-score obtained : 0.4761

- ***Optimising Logistic Regression classifier when the data is reduced using LDA :***

Running the SVM classifier with number of LDA components = 90, type of kernel = 'rbf' and maximum number of iterations = 20000 (optimal values for PCA) :

Results obtained when data is reduced using LDA :

Accuracy obtained : 0.2565
F-score obtained : 0.2565

Similarly when we run the SVM classifier with the above found optimal hyperparameters the results obtained are as follows :

Accuracy obtained : 0.4614 (46.14%)
F-score obtained : 0.4614 (46.14)

Preprocessing of train and test data :

Preprocessing of train and test data is done using preprocessing.StandardScaler() in sklearn. It finds the standard deviation and the mean on a training set so as to be able to later apply the same transformation (using these metrics) on the testing set.

Practical issues with building a classifier from the data:

1) **Computational Time :**

   In CIFAR-10 dataset, the images are of size 32x32 (colour) and the dataset consists of 60000 images which we split into 80% training data and 20% validation data. Each image when flattened, gives rise to 32*32*3 [3072] sized feature vectors which have to be further processed and fed to the classifiers for optimal classification results. This induces very high computational time, if no preprocessing is performed.

   Feature vector size can be reduced by reducing the size of the image and then performing classification on the reduced data. We are performing “PCA” and “LDA” for dimensionality reduction which indirectly improve the computation time.

2) **Overfitting :**

   For showcasing the overfitting in the model I simultaneously plotted accuracy graphs for the training data and testing data, where the training data is drawn in “red” while testing data is plotted in “blue”.

   The obtained graph is shown below :

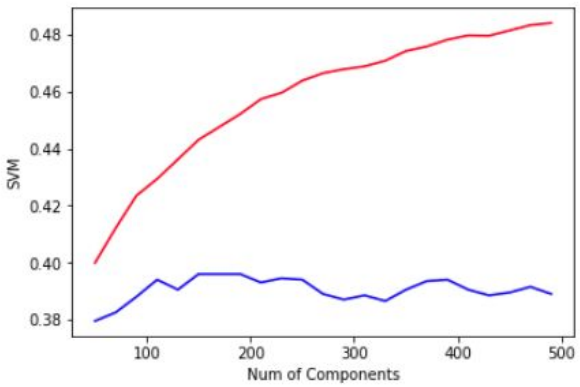


Fig.1

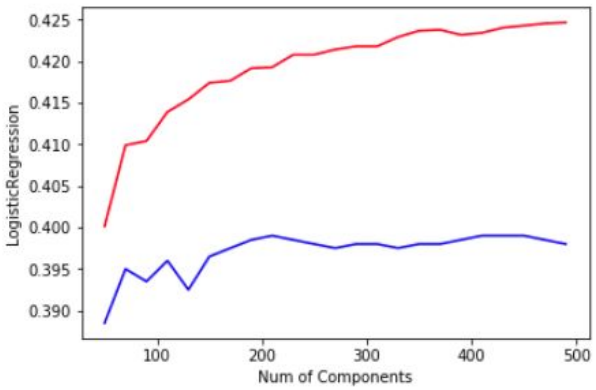


Fig.2

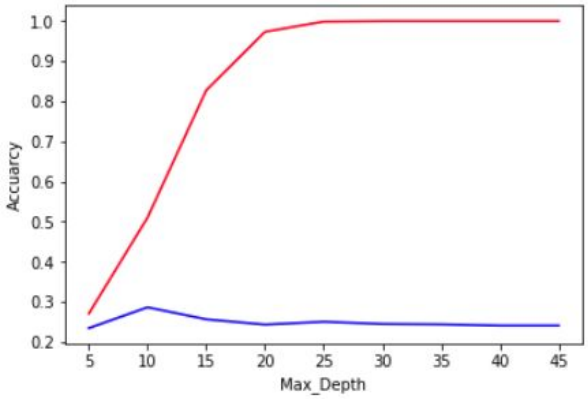


Fig.3

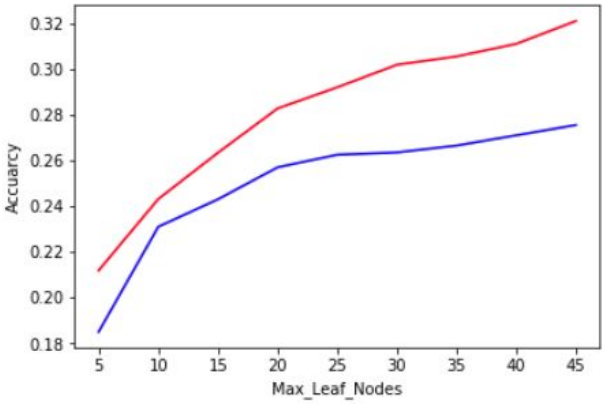


Fig.4

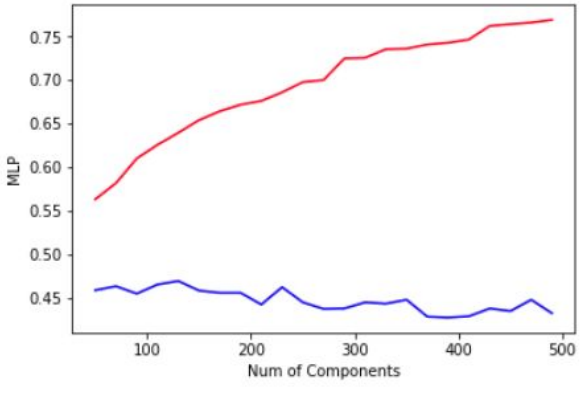


Fig.5

a) In fig.1, fig.2, fig.5 , we can see that on varying the number of principal components, the test data accuracy reaches a peak and then decreases on increasing the number

of principal components. Theoretically, as we take more number of principal components, the accuracies need to increase as we are considering more information (features) with less data compression. So, the variation here is due to the overfitting in the model. As we tune the parameters more and more to fit the training data, we are reducing the accuracy when performed on various other test datasets (which are different from the given training data set).

b) In fig.3 and fig.4, we can see that on increasing or tuning the parameters, "max_depth" and "max_leaf_nodes" corresponding to Decision Tree Classifier, accuracies reach a peak and then decrease gradually, this is again due to overfitting in the model. For example, as we increase the number of leaf nodes required, we are focussing more on the training data features and distribution so, the classification tree becomes more comfortable with training data, hence the overfitting.

c) Similarly in SVM classifier, if the maximum number of iterations is increased beyond 20000 the accuracy decreases due to overfitting.

d) Also, so far we found the best value for one hyperparameter keeping others constant. To get the best accuracy we considered the classifier by clubbing the hyperparameter values giving best accuracies individually. This may not give best overall accuracy always, as the hyperparameters are not linearly additive. For example, in MLP classifier accuracy is highest if the number of principal components= 80 (keeping others constant), similarly accuracy is highest if number of hidden layers = 3 (keeping others constant), each hidden layer having 3000 nodes (keeping others constant) and maximum number of iterations being 3000 (keeping others constant). But the accuracy wasn't always (across multiple runs) the highest if the hyperparameters are set to these values(though they are individually the best.)

3) **Minor changes in the accuracy result obtained :**

Every time dataset is read newly the train_test_split in sklearn.model_selection selects the train and validation datasets randomly in 80:20 ratio, thereby giving new splitting each time and hence the accuracy may change by 1%-3% by reading dataset newly.