EE 559 - MATHEMATICAL PATTERN RECOGNITION

Spring 2020 Project

MAY 8, 2020

UNIVERSITY OF SOUTHERN CALIFORNIA
Viterbi School of Engineering

EE 559 – Mathematical Pattern Recognition

(Spring 2020)

Motion Capture based Hand Posture Recognition System with Graphical User Interface

Author: Shashank Nelamangala Sridhara

(nelamang@usc.edu)

Aagam Manish Shah

(aagamman@usc.edu)

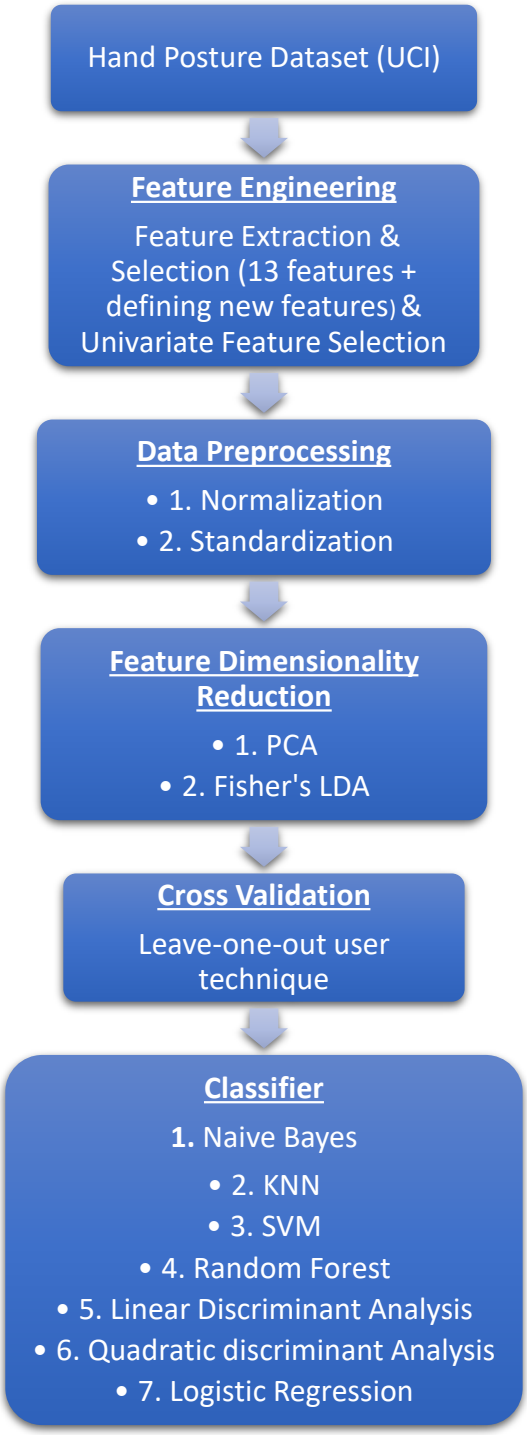
GitHub Link - https://github.com/Shashank-95/EE559-Hand-Postures

May 8, 2020

1. ABSTRACT:

Motion Capture based Hand Posture The gist of the entire project can be summarized here and the flowchart for the same is shown as below:

Flowchart of the entire process is as follows:



The raw dataset of the Motion Capture based Hand Posture Recognition system was taken from the UCI Machine Learning Repository. The detailed description of the dataset is given below in the Introduction section. The dataset basically consists of the 12 users out of which 9 are used for the training of the model and the rest 3 are used for testing. Each user performs the experiment several times for each class contributing to the generation of each data point. Thus, this data is stored in the .csv file which is then converted into a matrix or an array for further processing. In the Feature Engineering block, we perform the feature extraction from the raw data (described in later section) and then we select the features employing the Univariate Feature selection method. The features obtained from this block need to be preprocessed to bring down all the features in the same range. So, the preprocessing techniques used here are either Normalization or Standardization and the effect of each preprocessing technique is analyzed and compared in the later section. In order to reduce the computation complexity as well as computation time for further processing we still need to reduce the dimension of the features obtained which can be done by either Principle Component Analysis or the Fisher's Linear Discriminant Analysis, depending on the effect of each technique on the particular form of classifier. Followed by the dimensionality reduction, now in order to avoid the overfitting issue we implement the leave-one-out user cross-validation technique which is described in detail in the later section. Finally, the processed data is fed to the classifiers (Naïve Bayes, SVM, KNN, Random Forest, LDA, QDA and Logistic Regression) to train the model and then apply the test set to the trained model to obtain the test set accuracy through which we can justify that under what parameter settings which classifier performs better on the test dataset to give the optimum results. The best result obtained is reported briefly in the Result section.

2. INTRODUCTION:

2.1 Problem Statement and Goals:

Mathematical Pattern Recognition is a branch of machine learning which mainly focuses on the recognition of the regularities as well as the pattern in the data. In most of the cases the pattern recognition systems are trained using the labelled training data which is known as **Supervised Learning** whereas when the labelled training data is not available, various other algorithms are employed to decipher the unknown patterns generally known as **Unsupervised Learning.**

So, here we are basically set to develop a pattern recognition system for the hand postures based on the motion capture. The dataset used here was generated using a Vicon Capture camera system in an indoor environment. So, the problem statement here is basically given the location of markers placed in the left-hand glove, the designed system should be able to predict the hand posture. So, in this experiment the dataset was generated using the left hand glove on which 15 markers were attached, out of which 4 markers which are situated at the back of the hand forms the rigid pattern which provides the identification of the hand's position as well as orientation in order to generate the local coordinate system for the rest 11 markers, which are unlabeled markers and do not belong to either the rigid pattern or the skeleton, the reason being all the angles and distances between all these markers are flexible. Also, additionally few markers are obstructed by other fingers or other portion of hand which results in variation of recorded markers from one data point to other, which means it is a user independent pattern recognition problem.

As there are 5 different hand postures generated (fist, stop, point with one finger, point with two finger and grab) which indicates that it is basically a multi-class problem. To begin with the entire hand posture dataset is pre-divided into two parts i.e., the training data (having 9 users out of 12) and the test data (having 3 users out of 12). To make a note, the entire training of the model was done using only the training dataset and the testing dataset was untouched. Although the data is partially preprocessed but before passing the training data directly to the classifier for training the model it requires a significant amount of preprocessing such handling of the missing data points, balancing of the data, normalization or rescaling, standardization, feature extraction and selection, feature space dimensionality adjustment and cross validation. Now the processed training data is fed into the classifiers such as Naïve Bayes, SVM, KNN, RF, LR, LDA and QDA in order to train the classifier and use the same trained classifier on the test dataset for the classification. Also, the manner in which the data was captured, it is likely that for a particular given record and the user there might exist a near duplication of the record coming from the same user. So, therefore for evaluation of classification algorithms we implement the leave-one-user out technique in which each user is iteratively left out from the training and used as a test dataset.

3. APPROACH AND IMPLEMENTATION:

3.1 Preprocessing:

As the data given in the dataset is evenly balanced among all the classes so we do not face the issue of unbalanced data and there is no need for performing the data balancing among the classes.

As in the given dataset there are several missing values for each datapoint (for every user for each class), so while performing the preprocessing steps as well as the feature extraction procedure we deal with this **NaN** values for each datapoint by using the *nanmean & nanstd* while calculating the mean and the standard deviation of the of each feature. For example while calculating the mean or standard deviation of the X marker value of a particular record having missing entries are computed using *nanmean* or *nanstd* and similarly for Y as well as Z markers for the rest of the datapoints.

The best optimum results for each classifier with and without preprocessing are reported in the Results section 4.

3.1.1 Normalization: [6]

The data normalization is basically the preprocessing technique used for generation of the data for the further machine learning process. This preprocessing technique is basically done only when the features of the dataset are in the different ranges otherwise the data normalization is not required. Normalization is basically the scaling technique where the values are rescaled as well as shifted to make them into the range of 0 to 1.The main role of the data normalization is to transform the data values of a column in the given dataset to a particular common scale without disturbing the differences in the ranges of the numeric

values of the dataset. It is also known as the Min-Max Scaling. The formula for the data normalization is given as follows:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

Normalization is performed on the data which does not follow the Gaussian distribution and can be helpful for the algorithms which doesn't follow any kind of assumption that the distribution of the data is like KNN or neural networks.

3.1.2 Standardization: [6]

Standardization is also another kind of preprocessing technique which is used for scaling the input data. In this preprocessing method i.e., standardization which is nothing but another kind of the scaling technique where the data values are basically centered around the mean with the standard deviation equal to 1 which implies that the mean of the features change to zero and the standard deviation of the distribution is 1. The formula for the standardization is given as follows:

$$X' = \frac{X - \mu}{\sigma}$$

Where, μ is the mean of the feature datapoints and the σ is the standard deviation of the feature datapoints.
Standardization on the other hand is performed on the dataset which follows the Gaussian distribution; however, this is not the case always. As compared to the normalization, the standardization does not have any kind of the bounding range, which means that the standardization process remains unaffected even if there are outliers.

3.2 Feature Engineering:

3.2.1 Feature Extraction: [9]

Feature extraction in machine learning, is basically the process of dimensionality reduction through which the initial chunk of raw data is lowered to more manageable clusters for further processing.[10] As to the initial dataset is very large which consists of large number of variables and hence in turn it will require a huge amount of computations for processing, so in turn to reduce it we basically perform the feature extraction procedure which is a method which basically chooses and fuses various variables into appropriate features thereby reducing the amount of data required for processing effectively but at the same time not compromising on the data as that of the original dataset or losing the important information.
In this project basically we perform the feature extraction in the following way as described. As the given data is in its raw form which demands for the extraction of the

features as the number of features varies from one data point to another as well as the markers being unlabeled, we are set to develop the data to get it converted into the usable form. So, for this we choose out first 13 set of features for each data point (for each particular user and the class to which it belongs to) to be such as:

1. The number of the recorded markers.
2. Mean of the *x* marker locations
3. Mean of the *y* marker locations
4. Mean of the *z* marker locations
5. Standard deviation of the *x* marker locations
6. Standard deviation of the *y* marker locations
7. Standard deviation of the *z* marker locations
8. Maximum of the *x* marker locations
9. Maximum of the *y* marker locations
10. Maximum of the *z* marker locations
11. Minimum of the *x* marker locations
12. Minimum of the *y* marker locations
13. Minimum of the *z* marker locations

3.2.2 Feature Selection:

Feature selection is basically the process in which the features are chosen from the data which contribute most towards the target, which means *we choose the best predictors for the target value.*[3]

The classes in the ***sklearn.feature_selection*** *module are used for feature selection/dimensionality reduction to improve estimators' accuracy scores or to boost the performance on the high-dimensional dataset.*[3]

The advantages of the Feature Selection technique are as follows:

(i) Feature Selection basically reduces the overfitting of the data, which is nothing, but the less redundant data is eliminated which means the data having the least probability of forming decisions.

(ii) Feature selection basically selects only the prominent features which in turn reduces the size of the data and hence less the data implies the lesser training time.

(iii) This technique also eliminates the misleading data which in turn helps in improving the model accuracy.

3.2.2.1 F_classif feature selection:

The degree of the linear dependency among the various features is estimated using the F-test estimate. On the application of this particular method, the input is given as the scoring function which in turn will return the univariate scores and then the p-values which exhibits the probability of the given model such that the statistical summary of the model can either be same or else greater in magnitude than the observed actual results.

3.2.2.2 Mutual_info_classif:

This type of feature selection is basically used for the univariate feature selection process. So, this type of feature selection technique is basically used to estimate the mutual information of the particular discrete target variable. So basically, the dependency between the two variables is measured using the Mutual Information between the 2 random variables which is a non-negative value. If the 2 random variables are independent of each other then it is equal to 0 if the values are higher than it implies that it has higher dependency. *The mutual_info_classif function basically depends on the nonparametric methods which are based on the entropy estimation from the k-nearest neighbors' distances.*[3]

3.3 Feature Dimensionality Adjustment:

3.3.1 PCA (Principle Component Analysis):

Principle Component Analysis (PCA) is basically the technique which is used to find the underlying variables which are most commonly known as the principle components which are the most prominent ones among the other which helps to differentiate the best data points from the others. Principle components can be expressed in either single or multiple variables. These components are basically the dimensions along which the data points are mostly spread out. PCA is most commonly performed either by using the singular value decomposition technique or the eigen value decomposition of the covariance matrix.

Basically, we use PCA in order to project the high dimensional data to the lower dimensional space. But generally, the original data is normalized before applying PCA to it. The data normalization procedure basically implies the mean centering of each feature in which the mean of that particular data is calculated, and it is then subtracted from each data value such that its empirical mean is zero. Additionally, to the mean centering or so-called data normalization sometimes standardization is also performed in which the variance of that particular variable is computed to make it equal to 1.

However, the PCA does not seem to work well as desired for all the kinds of models which are used in this project. The reason might be that, generally, *applying PCA before the model is built doesn't help in improving accuracy due to the fact that the PCA algorithm doesn't consider the response prediction target into account. So, basically the PCA will consider the features which have the large variation but practically the features with large variation has nothing to do with the prediction target, which basically means that a lot of useless features can be generated and eliminate the useful features after PCA.* [2]

If we compare the PCA method with the Logistic regression then we can make an argument that PCA considers only the variance of the independent variable and not the response variable, while the logistic regression considers that in what sense every independent variable will impact on the response variable.

PCA is basically the statistical procedure which is based on the orthogonal transformation in order to convert the set of observations of correlated variables to the set of linearly uncorrelated variables which are known as Principle Components.[1]

PCA and the Logistic regression methods are completely different. *PCA can be used to remove the dimensions which has the string correlations prior to the PCA transformation.*[1] The drawback of the PCA is that doesn't focus into the categories and instead it cares only about the dimensions, due to which we choose the linear discriminant analysis (LDA). LDA is basically a PCA variant but keeps the categories different which makes logistic regression faster and simpler.

3.3.2 Fisher's Linear Discriminant Analysis [5]

As already we know that the Principle Component Analysis is one of the famous dimensionality reduction technique. PCA basically finds for the direction in the data which has the greatest variance and the projects the data onto it. So, by this approach we basically obtain the projection of data from the high dimensional subspace to the lower dimensional subspace which aids in the removal of the noisy directions. This basically implies that PCA is an unsupervised method as it doesn't incorporate the information of the labels of the data. FLDA comprises of the continuous independent variables as well as the class dependent variable.

As here we have multiclass problem, we can employ the Fisher Discriminant Analysis which can be stretched in order to find the subspace which consists of all the class variability. The generalization of the FLDA for multiclass problem can be given as, let us assume that there are C classes which has a mean of μi and similar covariance ∑, then the sample covariance of the class means can be used to define the scatter between the class variability as

$$\Sigma_b = \frac{1}{C}\sum_{i=1}^{C}(\mu_i - \mu)(\mu_i - \mu)^T$$

Where μ is the mean of the class means. The direction ω which is for the class separation can be given as

$$S = \frac{\vec{w}^T \Sigma_b \vec{w}}{\vec{w}^T \Sigma \vec{w}}$$

This implies that when ω is an eigenvector of $\Sigma^{-1}\Sigma_b$ *then the separation is equal to the corresponding eigenvalue.* [4]

If $\Sigma^{-1}\Sigma_b$ *is diagonalizable then the subspace which is spanned by the eigenvectors corresponding to C-1 greatest eigen values will consist of the variability between the features.* [4] Similar to PCA, we use these eigen vectors for the feature reduction. It becomes mandatory to use regularization due to the reason that the eigenvectors which are corresponding to the least eigen values would be very sensitive to a particular selection of the training data.

3.4 Dataset Usage:

The procedure followed using the given dataset is already described in detail in each of the particular section but briefly it can be described as follows:

i. The raw data for the Motion Capture based Hand Posture Recognition system is taken from the UCI Machine Learning Repository and it is stored in the .csv file which is then converted into a matrix or an array for further processing.

ii. As the data is randomly arranged and have the unlabeled markers so we extract the features in the manner described briefly in the **feature extraction** section. The **feature selection** is performed using the Univariate Feature Selection method and the reason for choosing this selection method is also justified in the Feature selection section.

iii. In the **Feature preprocessing step**, the data given in the dataset is evenly balanced among all the classes, so we do not face the issue of unbalanced data and there is no need for performing the data balancing among the classes.

iv. As in the given dataset there are several missing values for each datapoint (for every user for each class), so while performing the preprocessing steps as well as the feature extraction procedure we deal with this **NaN** values for each datapoint by using the *nanmean & nanstd* while calculating the mean and the standard deviation of the of each feature.

v. After the preprocessing of the obtained data we perform the dimensionality reduction using the PCA or Fisher's LDA technique in order to reduce the computation complexity as well as the computation time.

vi. To avoid the overfitting issue of the data we employ the cross-validation technique as described below:

Cross Validation: [7]

Cross validation is basically the method used for the evaluation of the model and to prevent the overfitting issue. In cross validation procedure we don't use the entire dataset for training the model, instead we remove some amount of data before the training procedure and the data which was removed before the training procedure can be then used in order to test the performance of the trained model on the new data. There are basically various types of the cross validation procedures available but in this project we are set to develop the leave-one-out user cross validation technique which is nothing but the *K-fold cross validation technique which is basically taken to its logical extreme, with K equal to N,*[8] where N is nothing but the number of the data points in the set. This basically means that the function approximator is trained N different times on all the data except the one datapoint for which the prediction is to be made for. Prior to this and in order to evaluate the model the average error is calculated. Although the results obtained by the leave-one-out user validation are good but still in the first pass the computations are very costly. But on the other hand the leave-out-one predictions can be made as easy as the regular predictions using the locally weighted learners, which implies that the calculation of the leave-one-out cross validation takes more amount of computation time as compared to that of the residual error.

We have used 9 folds & 900 runs (10^{-3} to 10^{-9}) for cross-validation.

The cross validation was used multiple times so basically, they were arranged in the nested loops due to multiple usage. In this the decisions were made based on the validation-set results.

vii. Last but not the least, the processed data is fed to the classifiers (Naïve Bayes, SVM, KNN, Random Forest, LDA, QDA and Logistic Regression) to train the model and then apply the test set to the trained model to obtain the test set accuracy through which we can justify that under what parameter settings which classifier performs better on the test dataset to give the optimum results. In the classification step we use

the test dataset and we used the test dataset approximately 95-100 times for each different combination of the parameter setting.

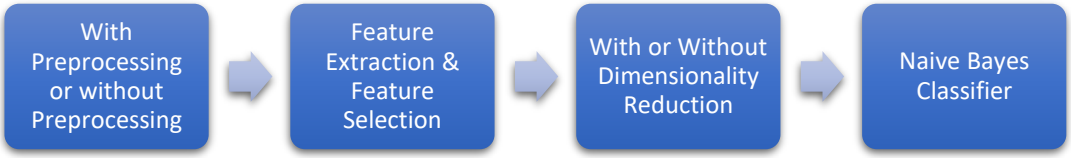
viii. To make a point, in order to state that we used 12000 datapoints for the training, 1500 datapoints for validation and 21099 datapoints for the testing.

3.5 Training and Classification:

3.5.1 Classifiers:

Here in this project we basically implement the five different types of classifiers namely: Naïve Bayes, SVM, Random Forest, KNN and Logistic Regression. Each of them is described in detail below. Basically, the cross validation is implemented for various classifiers in order to get the optimum parameter choice.

The in general block diagram of the training and the classification can be given as follows:



---

1. Naïve Bayes Classifier

Naïve Bayes Classifier is basically one of the supervised learning algorithms. The assumption made here is that each pair of features are independent, which implies that more the independence of the features of each pair the better is performance of the classifier. It should be noted that the preprocessing step such as the standardization of the training dataset may regulate the dependence between each feature. So, as a result for Naïve Bayes classifier it is logical to rescale the training dataset in order to maintain the dependency unaffected between each feature. Naïve Bayes classifier is highly scalable and the only advantage of using Naïve Bayes classifier is that for the classification, it basically requires a small number of training data in order to estimate the parameters.

2. SVM Classifier

Support Vector Machines are basically the supervised learning models which includes the learning algorithms which will analyze the data which is in turn used for the regression analysis as well as for the classification. Support vector machine is based on the principle that the SVM model is nothing but the representation of sample of the data points in the space which are mapped in such a way that the samples of various different categories are divided by the gap which is as wide as possible to make a clear distinction between the categories. Then later on the new sample data points are then mapped in the same space and then they are predicted to which category it will belong to based on which side of the

gap they will fall. Also, an important advantage of using SVM classifier is that performs very well in the non-linear classification case employing the kernel trick, which implicitly maps the input data to the high dimensional feature space. In this approach we will basically try with two different kernels of the SVM classifier namely, the Linear and the RBF kernel. After employing both the kernels for the SVM classifier we found out that the RBF kernel gives a better performance with the flexibility of changing the gamma parameter which is mainly used for the cross validation and the penalty factor C.

3. KNN Classifier

KNN (k-nearest neighbors) is basically the non-parametric technique which is used for the classification as well as the regression. In k-NN classification the input data basically consists of the k nearest training samples in the feature space and the output consists of the members of the class. The classification of an object is basically done by the majority of vote of the neighbors. The object will then be assigned to the class which will have the most representatives among the k-nearest neighbors of the point. KNN is also known as one of the instance-based learning which means that the approximation of the function is done locally and rest all the computations are suspended until the evaluation of the function. *For the KNN classification the assigning of weights to the contribution of the neighbors will be a helpful method, the reason being that the nearer neighbors will contribute more towards the average as compared to that of the distant ones.*[11] The most commonly used weighting method for each neighbor can be 1/d, where d is nothing but the distance to the neighbor. The unique property of the KNN classifier is that the sensitivity of the KNN algorithm is more towards the local form of data.

4. Logistic Regression

Logistic regression method is one of the statistical models which uses the basic form of the logistic function to model the multinomial variables when the outputs are more than two. Although logistic regression model basically models the probability of the output in terms of the input, but it is not the statistical classifier. The logistic regression can be made into classifier but by setting various cutoff values and then classifying the inputs based on their probability values such that the inputs having the probability value greater than a particular cutoff are classified as one class and the rest having lower probability value are classified as another class. So, the 3 basic steps which we perform for the logistic regression are:

- Firstly, in order to predict the assessment result of the input data we find an appropriate hypothesis function.
- Secondly, we build the loss function or also known as the cost function which basically indicates the difference between the training dataset and the predicted output.
- Lastly, in order to find the optimal solution of the loss function or also known as the cost function we apply the gradient descent.

5. Random Forest

Random forest is basically an ensemble learning technique for the classification which is based on the construction of the decision trees at the time of training and then class which is the mode of all the classes are generated as the output of the individual trees. Random Forest Classification method has the advantage of correction of the overfitting of the training data. There are 3 basic concepts of the Random Forest Classification namely the entropy, information and the information gain. Discussing each one of them briefly is as follow:

i. The 3 concepts such as the entropy, information and the information gain are the prominent parameters of the decision tree which serve as the basis for the determination of the feature selection order at the time when the decision tree makes use of these features to classify them. By Shannon, *information is used to eliminate random uncertainties.*[12] In machine learning, for the case of decision tree if the classification of the set of things is categorized into multiple categories, then the information of the class is given by the formula as given below:

$$I(X = x_i) = -log_2 p(x)$$

Where, I(X) represents the random variable information and the *p(xi)* is nothing but the probability of the occurrence of *xi.*

The entropy is basically used to measure the uncertainty, which means that the greater is the entropy the larger is the uncertainty of X = Xi & the other way around as well. The same concept applies to the machine learning problems as well.

Last but not the least, in the decision tree algorithm the features are selected using the information gain. The greater is the information gain then the selectivity of the feature is better.

ii. The **decision tree** is the basic structure of the Random Forest Classifier. The structure of the decision tree is basically in the form of tree where each internal node of the tree signifies the test of a feature and each branch as the output of the test data and each leaf node as a category. The most commonly used decision tree algorithms are CART, C4.5 and ID3.

iii. The single prediction issues can be solved by employing the integrated learning which is achieved by the combination of several models. The working principle states that the independent prediction of each other are made by generating various classifiers or models and learning them. These obtained predictions are then finally aggregated into single predictions which becomes superior to any other prediction made by the single classification.

6. Linear Discriminant Analysis

Linear Discriminant analysis basically can operate in both the modes i.e., as the dimensionality reduction as well as the classifier. The most important aspect of the Linear Discriminant Analysis is that separates out the samples of the classes by linearly shifting them to another feature space, which implies that if the data is linearly separable then on applying LDA classifier will fetch the optimum results. On the other hand, if the data isn't linearly separable then it will basically attempt to reorganize the data in the different space

where the maximum linear separability can be achieved. Using LDA as a classifier instead of the dimensionality reduction can be achieved by partitioning the classes and then using the LDA to classify each partition, which can be done by using the 'one vs rest' in which all the datapoints are placed into one single group and the rest others in another class and then we apply the LDA method. This will basically give us the results from the C different classifiers which are then combined to produce the unified output. Another approach can be the pairwise classification where for each pair of various classes are created using a new classifier and then finally all the separate classifiers are combined to generate a final unified classification output.

7. Quadratic Discriminant Analysis

Quadratic Discriminant Analysis is very much similar to the Linear Discriminant Analysis with the only difference that the covariance matrix can be different for each class hence the estimation of the covariance matrix is computed separately for each class *k.* The Quadratic discriminant function is given as:

$$\delta_k(x) = -\tfrac{1}{2}\log|\Sigma_k| - \tfrac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log\pi_k$$

As it is quadratic discriminant function so it will contain the second order terms in it and also the decision boundaries will be quadratic in nature. The QDA is more preferable over the LDA due the reason that the quadratic decision boundaries allow more flexibility for the covariance matrix which in turn aids in the fitting of the data more better as compared to that of the of LDA, but at the same time this results in the estimation of more number of parameters which are significantly high for QDA as compared to that of the LDA due to the fact of having separate covariance matrix for each class. But, QDA provides better and infact more accurate non-linear classification decision boundaries.

4. ANALYSIS: COMPARISON OF RESULTS, INTERPRETATION

4.1 Effect of preprocessing:

Classification accuracy is recorded with different preprocessing techniques like Normalization and Standardization. Tables below show the result of different classifiers using standardization and normalization.

With Standardization and Without Dimensionality Reduction

Classifier	Train Accuracy	Test Accuracy	F1-Score (Weighted)
Naive Bayes	0.89088	0.80202	0.78998
SVM (RBF)	0.99844	0.95781	0.95784
Random Forest	1	0.68491	0.70052

K-Nearest Neighbor (K = 5)	0.99896	0.79899	0.79384
LDA	0.9388	0.9272	0.9283
QDA	0.99	0.8490	0.8356
Logistic Regression	0.96088	0.9121	0.91125

With Normalization and Without Dimensionality Reduction

Classifier	Train Accuracy	Test Accuracy	F1-Score
Naive Bayes	0.78881	0.66813	0.62527
SVM (RBF)	0.79807	0.56125	0.48786
Random Forest	1	0.64458	0.64004
K-Nearest Neighbor (K = 5)	0.99785	0.64723	0.63453
LDA	0.8968	0.8227	0.8223
QDA	0.9866	0.7166	0.6738
Logistic	0.8377	0.6371	0.6159

As we can see that the classification accuracy falls drastically if we use normalization instead of standardization. This is because normalization rescales the data in between 0 and 1, whereas standardization rescales the data so each feature should have 0 mean and unit variance. This choice seems trivial, but it matters a lot for classifiers like SVM, which assumes the data to be standardized and each feature should be distributed and transformed in the same way. **So, standardization is more suitable for the given dataset.**

4.2 Effect of Feature Reduction:

After choosing standardization to be the suitable preprocessing technique, we have decided which feature dimensionality reduction technique is more suitable for the data set. So, we experimented with PCA, FLD and univariate feature selection techniques.

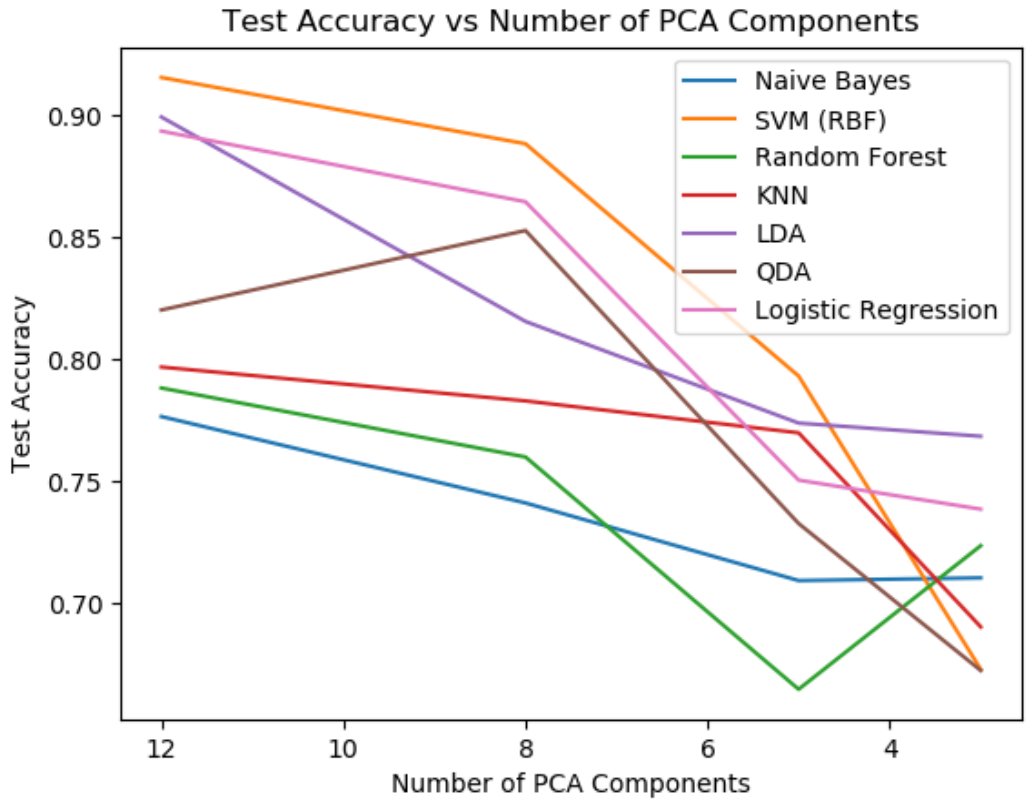
Tables below show the results of each classifier with different Feature reduction techniques.

With Standardization, With PCA

Reduced Dimension = min (n_features, n_classes - 1)

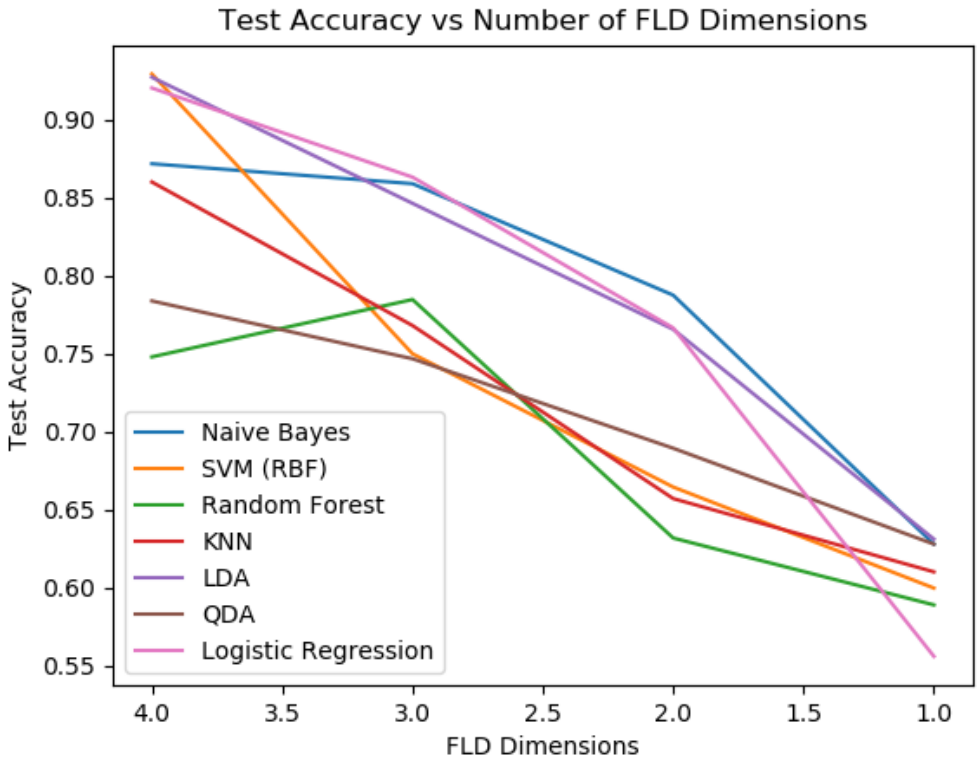
Classifier	4 - components	3- components	2- components	1- components
Naive Bayes	TrainAcc: 0.9521 Test Acc: 0.8717	TrainAcc: 0.9362 Test Acc: 0.8589	TrainAcc: 0.9523 Test Acc: 0.7874	TrainAcc: 0.7842 Test Acc: 0.6277

	Fscore : 0.8744	Fscore : 0.8541	Fscore : 0.7888	Fscore : 0.6084
SVM (RBF)	TrainAcc: 0.9976 Test Acc: 0.9293 Fscore : 0.9295	TrainAcc: 0.9868 Test Acc: 0.7498 Fscore : 0.7258	TrainAcc: 0.9807 Test Acc: 0.6643 Fscore : 0.6350	TrainAcc: 0.8302 Test Acc: 0.5995 Fscore : 0.6049
Random Forest	TrainAcc: 1.0 Test Acc: 0.7478 Fscore : 0.7149	TrainAcc: 1.0 Test Acc: 0.7846 Fscore : 0.7743	TrainAcc: 1.0 Test Acc: 0.6316 Fscore : 0.6104	TrainAcc: 0.9994 Test Acc: 0.5887 Fscore : 0.5937
K-Nearest Neighbor (K = 5)	TrainAcc: 0.9979 Test Acc: 0.8600 Fscore : 0.8588	TrainAcc: 0.9962 Test Acc: 0.7679 Fscore : 0.7482	TrainAcc: 0.9883 Test Acc: 0.6569 Fscore : 0.6402	TrainAcc: 0.8771 Test Acc: 0.6099 Fscore : 0.6154
LDA	TrainAcc: 0.9388 Test Acc: 0.9272 Fscore : 0.9283	TrainAcc: 0.9359 Test Acc: 0.8463 Fscore : 0.8486	TrainAcc: 0.9541 Test Acc: 0.7655 Fscore : 0.7637	TrainAcc: 0.7940 Test Acc: 0.6309 Fscore : 0.6149
QDA	TrainAcc: 0.9571 Test Acc: 0.7837 Fscore : 0.7553	TrainAcc: 0.9492 Test Acc: 0.7466 Fscore : 0.7283	TrainAcc: 0.9574 Test Acc: 0.6892 Fscore : 0.6781	TrainAcc: 0.7842 Test Acc: 0.6277 Fscore : 0.6084
Logistic Regression	TrainAcc: 0.9430 Test Acc: 0.9203 Fscore : 0.9196	TrainAcc: 0.9362 Test Acc: 0.8631 Fscore : 0.8660	TrainAcc: 0.9472 Test Acc: 0.7664 Fscore : 0.7628	TrainAcc: 0.6552 Test Acc: 0.5557 Fscore : 0.5060



Fischer's Linear Discriminant with Standardization.
Reduced Dimension = min(n_features, n_classes - 1)

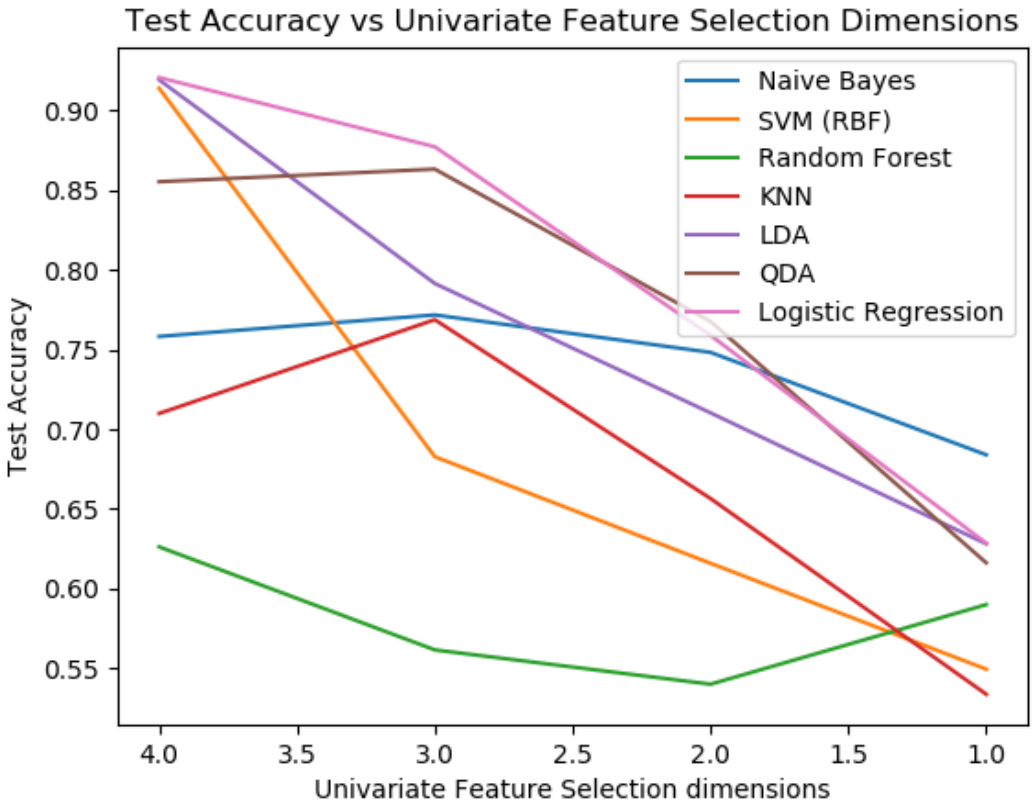
Classifier	4 - components	3- components	2- components	1- components
Naive Bayes	TrainAcc: 0.9521 Test Acc: 0.8717 Fscore : 0.8744	TrainAcc: 0.9362 Test Acc: 0.8589 Fscore : 0.8541	TrainAcc: 0.9523 Test Acc: 0.7874 Fscore : 0.7888	TrainAcc: 0.7842 Test Acc: 0.6277 Fscore : 0.6084
SVM (RBF)	TrainAcc: 0.9976 Test Acc: 0.9293 Fscore : 0.9295	TrainAcc: 0.9868 Test Acc: 0.7498 Fscore : 0.7258	TrainAcc: 0.9807 Test Acc: 0.6643 Fscore : 0.6350	TrainAcc: 0.8302 Test Acc: 0.5995 Fscore : 0.6049
Random Forest	TrainAcc: 1.0 Test Acc: 0.7478 Fscore : 0.7149	TrainAcc: 1.0 Test Acc: 0.7846 Fscore : 0.7743	TrainAcc: 1.0 Test Acc: 0.6316 Fscore : 0.6104	TrainAcc: 0.9994 Test Acc: 0.5887 Fscore : 0.5937
K-Nearest Neighbor (K = 5)	TrainAcc: 0.9979 Test Acc: 0.8600 Fscore : 0.8588	TrainAcc: 0.9962 Test Acc: 0.7679 Fscore : 0.7482	TrainAcc: 0.9883 Test Acc: 0.6569 Fscore : 0.6402	TrainAcc: 0.8771 Test Acc: 0.6099 Fscore : 0.6154
LDA	TrainAcc: 0.9388 Test Acc: 0.9272 Fscore : 0.9283	TrainAcc: 0.9359 Test Acc: 0.8463 Fscore : 0.8486	TrainAcc: 0.9541 Test Acc: 0.7655 Fscore : 0.7637	TrainAcc: 0.7940 Test Acc: 0.6309 Fscore : 0.6149
QDA	TrainAcc: 0.9571 Test Acc: 0.7837 Fscore : 0.7553	TrainAcc: 0.9492 Test Acc: 0.7466 Fscore : 0.7283	TrainAcc: 0.9574 Test Acc: 0.6892 Fscore : 0.6781	TrainAcc: 0.7842 Test Acc: 0.6277 Fscore : 0.6084
Logistic Regression	TrainAcc: 0.9430 Test Acc: 0.9203 Fscore : 0.9196	TrainAcc: 0.9362 Test Acc: 0.8631 Fscore : 0.8660	TrainAcc: 0.9472 Test Acc: 0.7664 Fscore : 0.7628	TrainAcc: 0.6552 Test Acc: 0.5557 Fscore : 0.5060



With Standardization, With Feature Selection (f_classif)

Classifier	12 - components	8- components	5- components	3- components
Naive Bayes	TrainAcc: 0.8861 Test Acc: 0.7582 Fscore : 0.7599	TrainAcc:0.9055 Test Acc: 0.7717 Fscore : 0.7707	TrainAcc: 0.8837 Test Acc: 07482 Fscore : 0.7620	TrainAcc: 0.8231 Test Acc: 0.6840 Fscore : 0.6842
SVM (RBF)	TrainAcc: 0.9985 Test Acc: 0.9138 Fscore : 0.9160	TrainAcc: 0.9977 Test Acc: 0.6828 Fscore : 0.6576	TrainAcc: 0.9942 Test Acc: 0.6160 Fscore : 0.5944	TrainAcc: 0.9456 Test Acc: 0.5496 Fscore : 0.4660
Random Forest	TrainAcc: 1.0 Test Acc: 0.6263 Fscore : 0.6402	TrainAcc: 1.0 Test Acc: 0.5617 Fscore : 0.5413	TrainAcc: 1.0 Test Acc: 0.5402 Fscore : 0.5018	TrainAcc: 0.9999 Test Acc: 0.5901 Fscore : 0.5560
K-Nearest Neighbor (K = 5)	TrainAcc: 0.9990 Test Acc: 0.7099 Fscore : 0.7049	TrainAcc: 0.9991 Test Acc: 0.7687 Fscore : 0.7637	TrainAcc: 0.9981 Test Acc: 0.6566 Fscore : 0.6510	TrainAcc: 0.9872 Test Acc: 0.5339 Fscore : 0.4711
LDA	TrainAcc: 0.9376 Test Acc: 0.9191 Fscore : 0.9201	TrainAcc: 0.9162 Test Acc: 0.7914 Fscore : 0.7934	TrainAcc: 0.9021 Test Acc: 0.7103 Fscore : 0.7016	TrainAcc: 0.84 Test Acc: 0.6282 Fscore : 0.6072

QDA	TrainAcc: 0.9908 Test Acc: 0.8552 Fscore : 0.8406	TrainAcc: 0.9836 Test Acc: 0.8631 Fscore : 0.8624	TrainAcc: 0.9528 Test Acc: 0.7676 Fscore : 0.7844	TrainAcc: 0.8558 Test Acc: 0.6163 Fscore : 0.5911
Logistic Regression	TrainAcc: 0.9590 Test Acc: 0.9205 Fscore : 0.9200	TrainAcc: 0.9599 Test Acc: 0.8771 Fscore : 0.8758	TrainAcc: 0.9065 Test Acc: 0.7589 Fscore : 0.7413	TrainAcc: 0.8448 Test Acc: 0.6286 Fscore : 0.5915



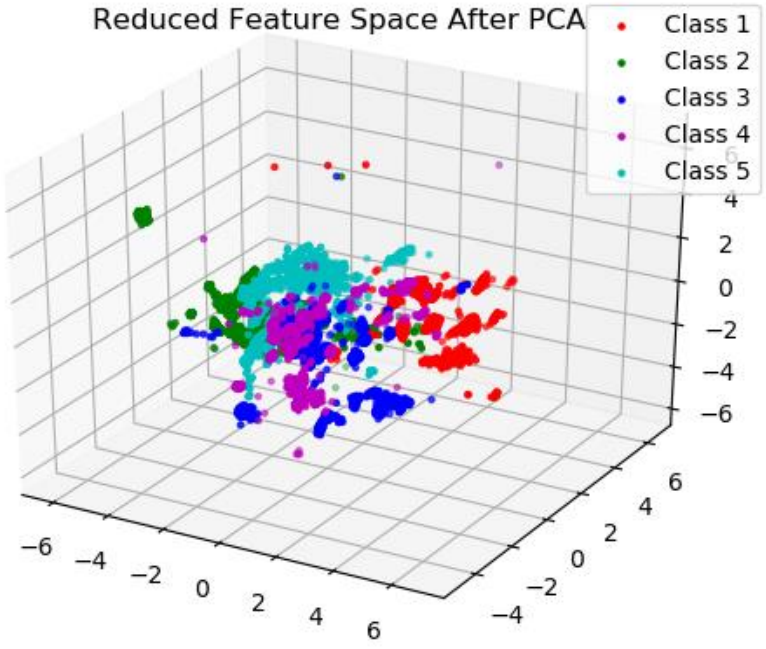
As we can see from the above plots, the accuracy decreases with each Feature reduction technique as the number of features decrease. The reason can be analyzed using the **Singular Values of the covariance matrix of the feature matrix.** Below table shows the singular values of the covariance matrix.

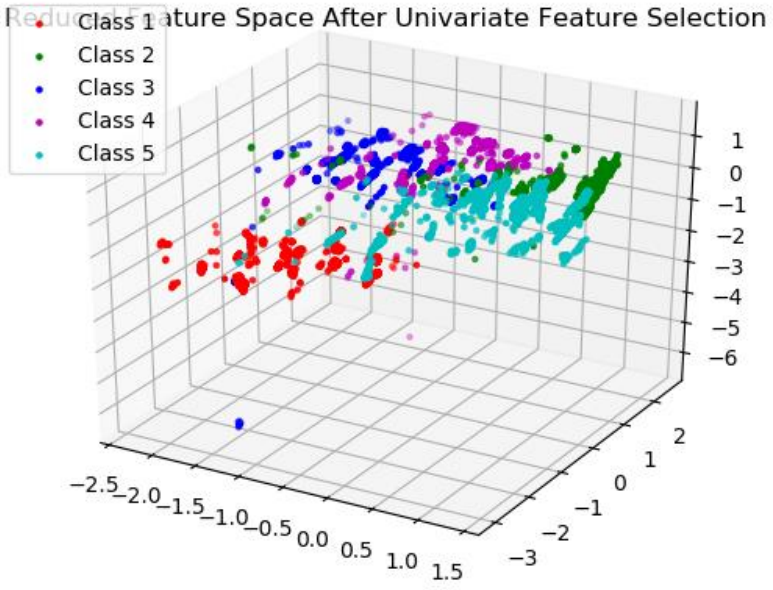
Singular Values of the covariance matrix

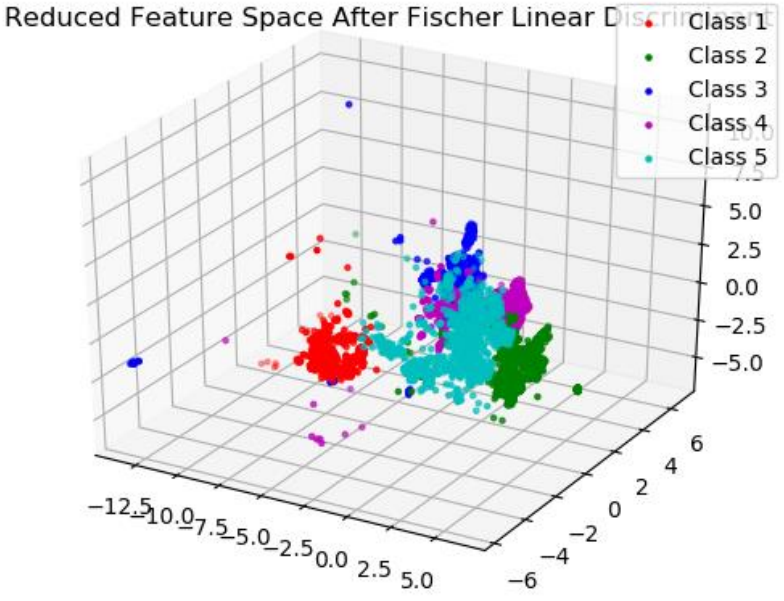
4.77	3.63	1.38	1.24	0.86	0.55	0.31	0.08	0.04	0.03	0.02	0.01	0.01

Since, first 7 singular values are prominent, reducing the number of features to anything below 7 will badly affect the accuracy. This result can be seen in the above plots where the accuracy decreases drastically for features 5 and 3 for PCA and Univariate Selection. But this isn't true for Fisher's Linear Discriminant method, because unlike PCA it provides optimal subspace by considering the class labels of the data points.

Below figures show the 3D plot of each of the Dimensionality reduction techniques.




Class 1
Class 2
Class 3
Class 4
Class 5


Reduced Feature Space After Fischer Linear D
Class 1
Class 2
Class 3
Class 4
Class 5

From the above 3D plots, we can notice only FLD gives strong discriminant power among the features after feature reduction. So, we conclude FLD technique is better for this dataset.

4.3 Results of each Classifier:

4.3.1 Naive Bayes Classifier (Baseline):

With Standardization and FLD with 4 dimensions

Density Function	Train Accuracy	Test Accuracy	F-Score
Multinomial Naive Bayes (Input Feature Matrix have negative values)	NA	NA	NA
Gaussian Naive Bayes	**0.9521**	**0.8717**	**0.8744**
Bernoulli Naive Bayes (Input Feature Matrix have negative values)	NA	NA	NA

Multinomial and Bernoulli density functions will not work if the feature matrix has negative values in it. Although experiments were performed by trying **min-max normalization**, the accuracy was very low. So, experiments were only performed on Gaussian density function.

The confusion matrix of Naïve Bayes classifier with Gaussian Density function as class conditional density is shown below.

```
[[4171   48  238    0    9]
 [   0 4252   66   52   32]
 [   0    0 3802  977    0]
 [   0    0 1089 2825    0]
 [   0   10   47  139 3342]]
```

From the above confusion matrix, we can observe that the highest misclassification is recorded between Class 4 and 3 data points, whereas Class 2 data points are classified with highest accuracy. So, classes 3 and 4 are difficult to classify and often misclassified.

4.3.2 Support Vector Machine

With Standardization and FLD with 4 dimensions

Kernels	Train Accuracy	Test Accuracy	F-Score
RBF	0.9976	0.9293	0.9292
Linear	0.9739	0.9097	0.9088

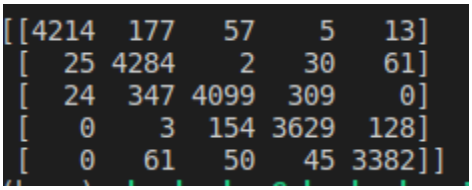
Sigmoid	0.5862	0.6878	0.6772
Polynomial	0.9931	0.7768	0.7547

Weight Vectors for Linear Kerner:

```
[[-0.82973034 -2.32996207  0.27468514  0.01251163]
 [ 0.4206486  -2.648934   -0.34283891 -2.01103828]
 [-1.24668608 -1.50393572  0.68277815 -0.36172032]
 [-1.80749041  0.0551773  -0.82181934  0.67659366]
 [ 0.60112449 -0.66004586 -0.18695095 -0.19788429]
 [ 0.64882043 -0.49267395 -0.10670932 -0.00480305]
 [ 1.85523947  2.35078438 -0.51514337  0.89235518]
 [-1.34604352  0.18232291  0.53365913  0.60193876]
 [-0.71661606  1.93786988 -0.55912123  0.85057193]
 [-0.47161673  1.07428774 -0.76262707 -0.27295965]]
```

Model Parameter Selection: The model parameters for SVM are chosen by implementing **leave-one-user-out cross validation.** The features are segregated based on User_ID and respective validation and training sets are created. Model parameters which resulted in highest accuracy are chosen as the final parameter.

The confusion matrix of SVM-Classifier using RBF kernel is shown below.



```
[[4214  177   57    5   13]
 [  25 4284    2   30   61]
 [  24  347 4099  309    0]
 [   0    3  154 3629  128]
 [   0   61   50   45 3382]]
```

Final Model Parameters for SVM:

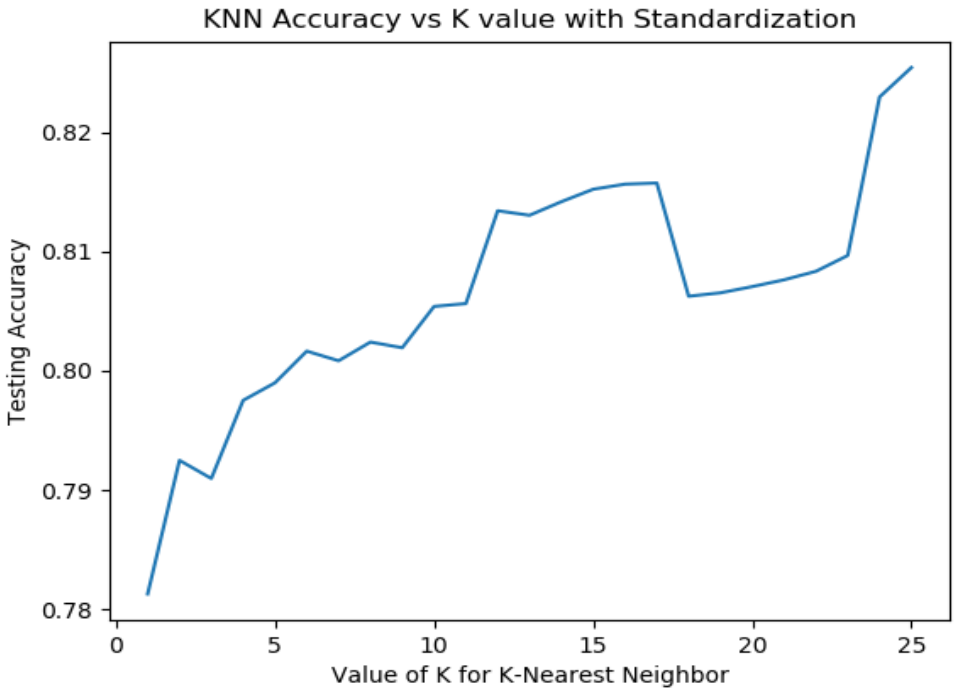
Test Accuracy: 0.9293 and F-Score:0.9292

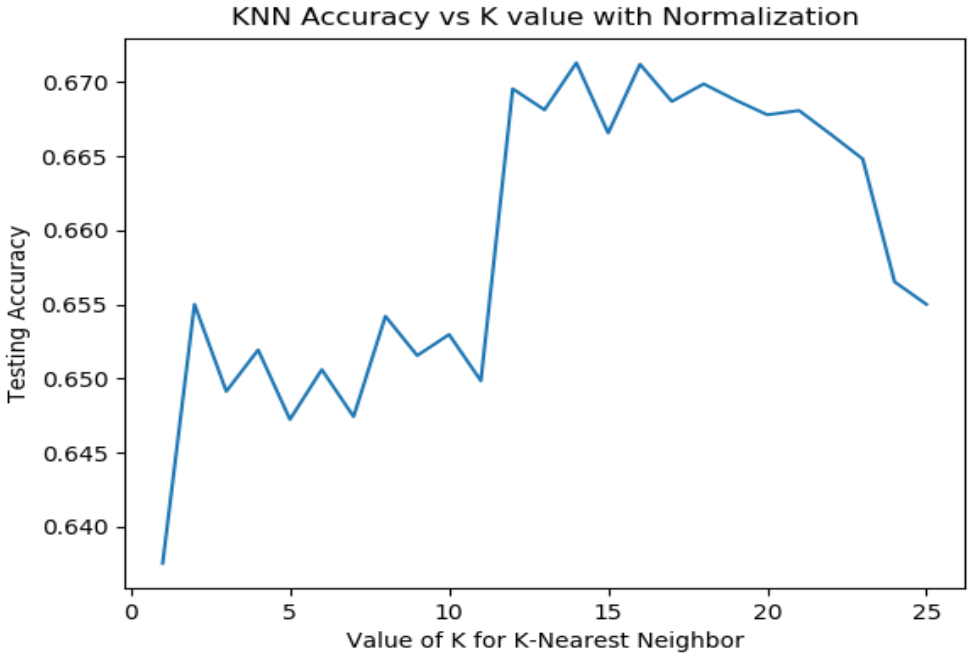
Kernel	Gamma	C
RBF	0.036	1000

From the above confusion matrix, we can observe that the highest misclassification is recorded between Class 3 and 2 data points, whereas Class 1 data points are classified with highest accuracy. So, classes 3 and 2 are difficult to classify and often misclassified.

4.3.3 K-Nearest Neighbors:

The K value in K-NN (number of points to be included in the region) is decided by plotting different values of K vs **accuracy of validation data**. Based on different experiments, we decided the optimal value of K to be 5. Below figures show the plot of validation accuracy vs K values.





Choosing K Value: As we can see from the above plots, with standardization, the accuracy on the validation set is increasing as the K value increases. But with normalization, the validation accuracy decreases after a point. As we have discussed in the previous section, standardization

resulted in better accuracies on all classifiers compared to Preprocessing. So, K value will be decided based on results obtained with standardization. An optimal value of K=5 is chosen. Although higher values of K yield better results, the chances of overfitting is more. So, an optimal value is chosen based on the above plots.

Best result for KNN is obtained using Standardization and Fishers' Linear Discriminant.

With Standardization and FLD with 4 dimensions

KNN	Train Accuracy	Test Accuracy	F-Score
K = 5 (with Standardization and FLD)	0.9979	0.8600	0.8588

The confusion matrix for KNN classifier with K=5 is shown below.

```
[[4311   74   81    0    0]
 [  29 3525    2   65  781]
 [ 213   18 4248  299    1]
 [   0    0 1096 2680  138]
 [   0   15   40  100 3383]]
```

From the above confusion matrix, we can observe that the highest misclassification is recorded between Class 4 and 3 data points, whereas Class 1 data points are classified with highest accuracy. So, classes 3 and 4 are difficult to classify and often misclassified.

4.3.4 Linear Discriminant Analysis:

Linear Discriminant analysis assumes the class conditional density to be Gaussian with equal covariance matrix. It derives linear boundaries using Maximum Value Method.

The results of Linear Discriminant Analysis with standardization and Fishers' Linear Discriminant with 4 dimensions is shown below.

With Standardization and FLD with 4 dimensions

	Train Accuracy	Test Accuracy	F-Score
LDA	0.9388	0.9272	0.9283

The confusion matrix of LDA is shown below.

```
[[4360   48   58    0    0]
 [  30 4253   14  105    0]
 [  13    0 3683 1083    0]
 [   0   48    2 3864    0]
 [   0    7    2  124 3405]]
```

As we can observe, LDA doesn't overfit the data. In all the experiments we can observe the same thing. This is because LDA doesn't give complicated boundaries unlike SVM, Quadratic Discriminant Analysis.

Also, from the above confusion matrix we can observe that the highest misclassification is recorded between Class 3 and 4 data points, whereas Class 1 data points are classified with highest accuracy. So classes 3 and 4 are difficult to classify and often misclassified.

4.3.5 **Quadratic Discriminant Analysis:** Like LDA, Quadratic Discriminant Analysis also fits the class conditional densities with Gaussian function. But it doesn't assume the covariance matrix of each class to be the same. Instead it calculates the Covariance matrix for each class separately. So, this provides complex decision boundaries.

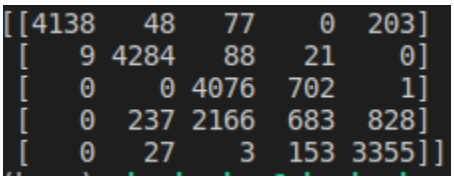
The results of Quadratic Discriminant Analysis with standardization and Fishers' Linear Discriminant with 4 dimensions is shown below.

With Standardization and FLD with 4 dimensions

	Train Accuracy	Test Accuracy	F-Score
QDA	0.9571	0.7837	0.7553

As we can see, the training accuracy for QDA is good enough but the accuracy dropped significantly on test data. This is because QDA draws complex decision boundaries which tends to overfit the data. This problem is not seen in LDA.

The confusion matrix for QDA is shown below.



From the above confusion matrix, we can observe that the highest misclassification is recorded between Class 3 and 4 data points, whereas Class 2 data points are classified with highest accuracy. So, classes 4 and 3 are difficult to classify and often misclassified

4.3.6 **Logistic Regression:** Logistic regression provides probability for each data point with respect to each class. It uses a soft-max function to get the final decision. The class assignment is done by taking the result of highest probability.

The best result for logistic regression is obtained by using lbfgs solver along with One-vs-Rest for multi class classification. The results are shown in the below table.

With Standardization and FLD with 4 dimensions

Logistic Regression	Train Accuracy	Test Accuracy	F-Score
Iteration= 100 Solver= lbfgs Multi-class = OvR	0.9430	0.9203	0.9196

The confusion matrix is shown below.

```
[[4188   48  176    0   54]
 [  51 4199   39  113    0]
 [  11   48 4700   20    0]
 [   0  912   75 2927    0]
 [   0    9   58   67 3404]]
```

From the above confusion matrix, we can observe that the highest misclassification is recorded between Class 4 and 2 data points, whereas Class 3 data points are classified with highest accuracy. So, classes 4 and 2 are difficult to classify and often misclassified.

4.3.7 <u>Random Forest</u>: Random Forest is a decision tree approach for classification. The parameters are chosen heuristically. The results of Random Forest are not good compared to other classifiers. The result of Random Forest is shown in the below table.

With Standardization and FLD with 4 dimensions

Random Forest	Train Accuracy	Test Accuracy	F-Score
num_trees= 80 max_depth=100	1.0	0.7478	0.7149

The confusion matrix for Random Forest is shown below

```
[[4357   49   18   42    0]
 [  25 3464    0   32  881]
 [   6  295 3963  504   11]
 [   0 1995 1331  588    0]
 [   0   12   49   71 3406]]
```

From the above confusion matrix, we can observe that the highest misclassification is recorded between Class 4 and 2 data points, whereas Class 1 data points are classified with highest accuracy. So, classes 4 and 2 are difficult to classify and often misclassified.

4.4 Comparison with the Baseline Classifier:

All the classifiers are compared against Naive Bayes classifier (Baseline classifier) under similar settings i.e, with standardization with Fishers' Linear Discriminant feature reduction technique. Table below summarizes the comparison.

With Standardization and FLD with 4 dimensions

Classifier	Train Accuracy	Test Accuracy	F1-Score
Naive Bayes (Baseline)	0.9521	0.8717	0.8744
SVM (RBF)	0.9976	0.9293	0.9292
Random Forest	1.0	0.7478	0.7149
K-Nearest Neighbor (K = 5)	0.9979	0.8600	0.8588
LDA	0.9388	0.9272	0.9283
QDA	0.9571	0.7837	0.7553
Logistic	0.9430	0.9203	0.9196

As we can see from the above table, the accuracy on Naive Bayes (baseline) classifier is 87.17%. Quadratic Discriminant analysis, K-NN and Random Forest performed poorly when compared to the baseline classifier. All other classifiers performed well with the highest being SVM classifier with RBF kernel. Logistic Regression and Linear Discriminant Analysis performed equally well when compared SVM. So, the order of performance with respect to test accuracy is as follows.

SVM > LDA>Logistic Regression> Naive Bayes>KNN>QDA>RF

5. CONTRIBUTION OF EACH TEAM MEMBER:

Date	Task	Done By	Progress
21st April - 22nd April	Preprocessing and Feature Extraction	Shashank	Done
23rd April	Feature Space Dimensionality adjustment	Aagam	Done
24-26th April	Cross Validation	Shashank	Done
27-30th April	Training Different Classification Algorithm	Aagam (3) Shashank(4)	Done
1-2nd April	Evaluation on Test set and fine tune	Aagam	Done

2nd-4th May	Improvise Feature space by defining new features	**YTD**	Pending
5th-6th May	Report Drafting	Aagam Shashank	Done
7th-8th May	Graphical User Interface.	Shashank	Done

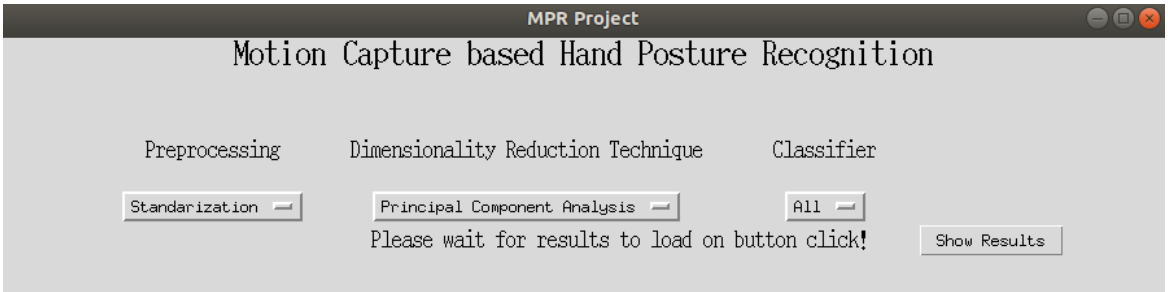
6. SUMMARY AND CONCLUSIONS

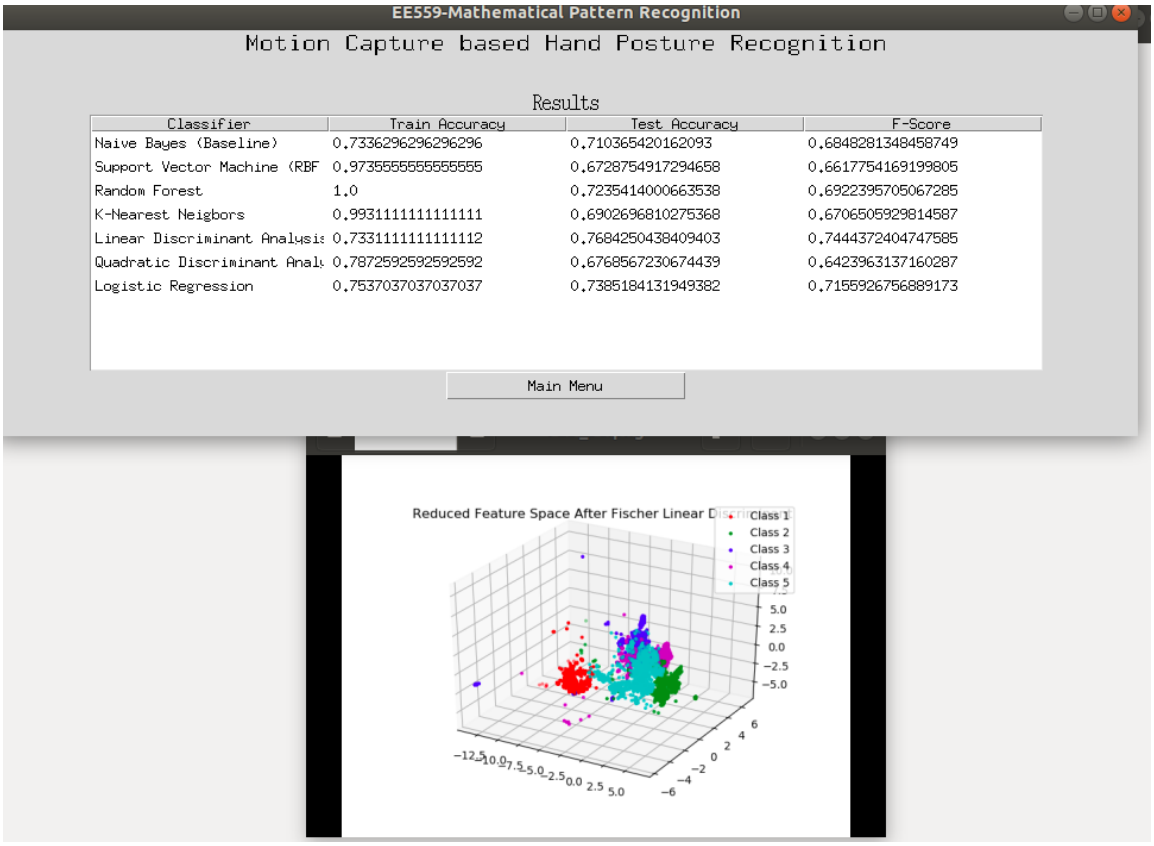
In this project, the given data set is used to classify different hand postures and the results of each classifier is analyzed. We experimented with the classification process using different preprocessing methods and Feature dimensionality reduction techniques. We observed that using standardization as a preprocessing method, highest test accuracy is achieved compared to normalization for most of the classifiers. We also tried different feature reduction techniques like Principal Component Analysis (PCA), Fishers' Linear Discriminant (FLD) and Univariate Feature Selection. We observed that using Fishers' linear discriminant method highest accuracy is achieved on the test dataset. In addition to that FLD gave better results with minimum number of dimensions compared to PCA and Univariate Feature Selection.

We evaluated each classifier using standardization and FLD with 4 dimensions. We obtained highest accuracy for the SVM classifier with the RBF kernel. Logistic Regression and Linear Discriminant analysis performed equally well compared to SVM. Random Forest gave lowest accuracy among all the classifiers because the model parameters are chosen heuristically. By analyzing the confusion matrix of each classifier, we can conclude that data points of class 3 and 4 have the highest misclassifications.

Future work: We would like to define more features like distance of markers to the origin, median of X, Y and Z coordinates. Adding features with more discriminant power provides better accuracy. Also, we would like to train the model on larger dataset and compare the result of training on larger dataset. In addition to this, we also want to improvise the GUI of the project making it more interactive with capabilities to display necessary plots.

7. PREVIEW OF THE IMPLEMENTATION





REFERENCES:

1. Online:https://stats.stackexchange.com/questions/244677/how-to-decide-between-pca-and-logistic-regression/244680#244680
2. Online:https://stats.stackexchange.com/questions/303602/why-does-pca-feature-reduction-make-accuracy-dramatically-worse/303607
3. Online:https://scikitlearn.org/stable/modules/feature_selection.html#univariate-feature-selection
4. Online: Linear Discriminant Analysis – Wikipedia.com
5. Fisher Linear Discriminant Analysis, Max Welling Department of Computer Science University of Toronto 10 King's College Road Toronto, M5S 3G5 Canada welling@cs.toronto.edu
6. Online:https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/
7. Online:https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.LeaveOneOut.html
8. Online: https://www.cs.cmu.edu/~schneide/tut5/node42.html
9. Online: https://deepai.org/machine-learning-glossary-and-terms/feature-extraction
10. Online: Feature Extraction – Wikipedia.com
11. KNN – Wikipedia.com
12. Random Forest – Wikipedia.com