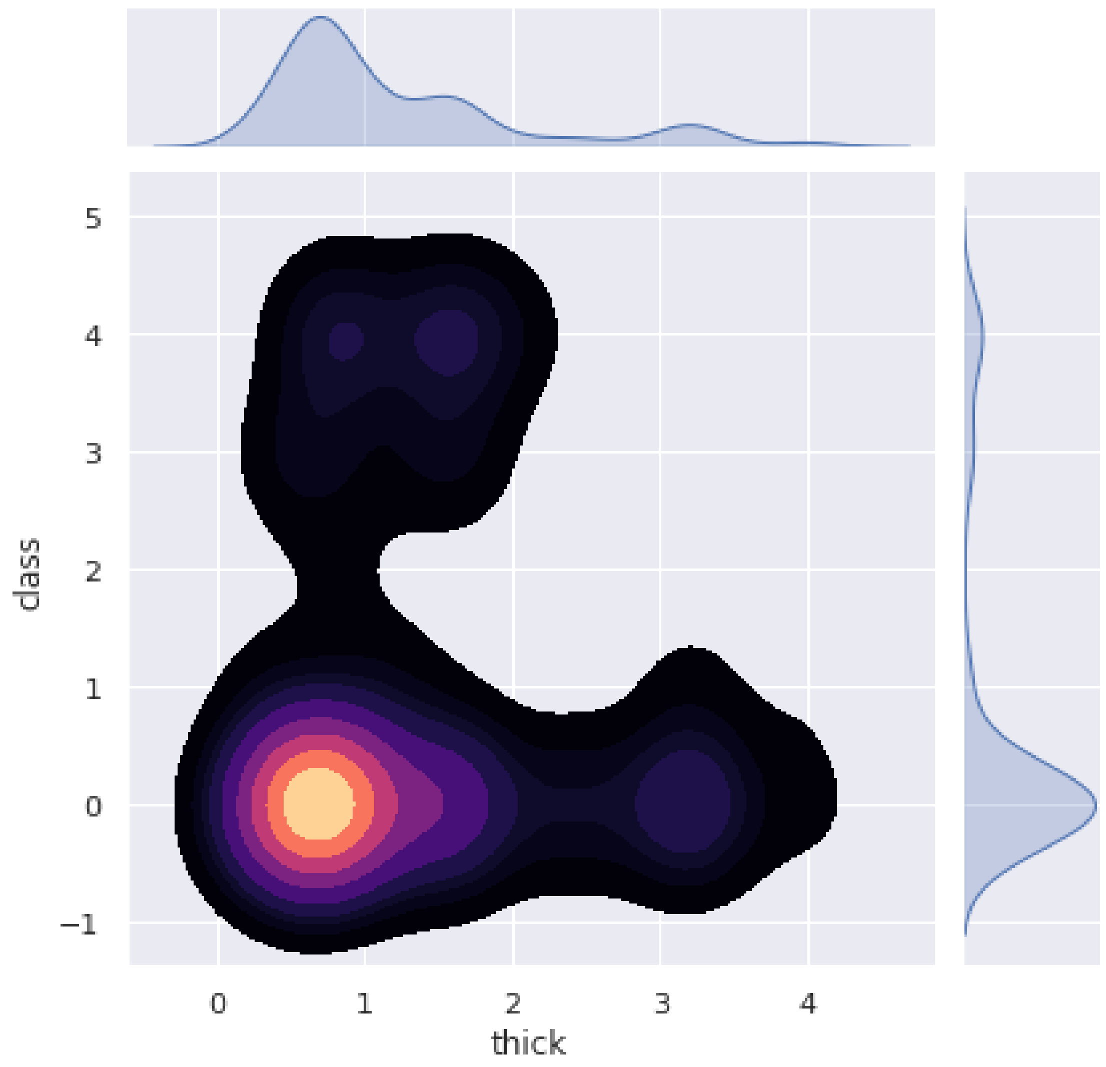
Pattern Recognition and Machine Learning
Lab 6

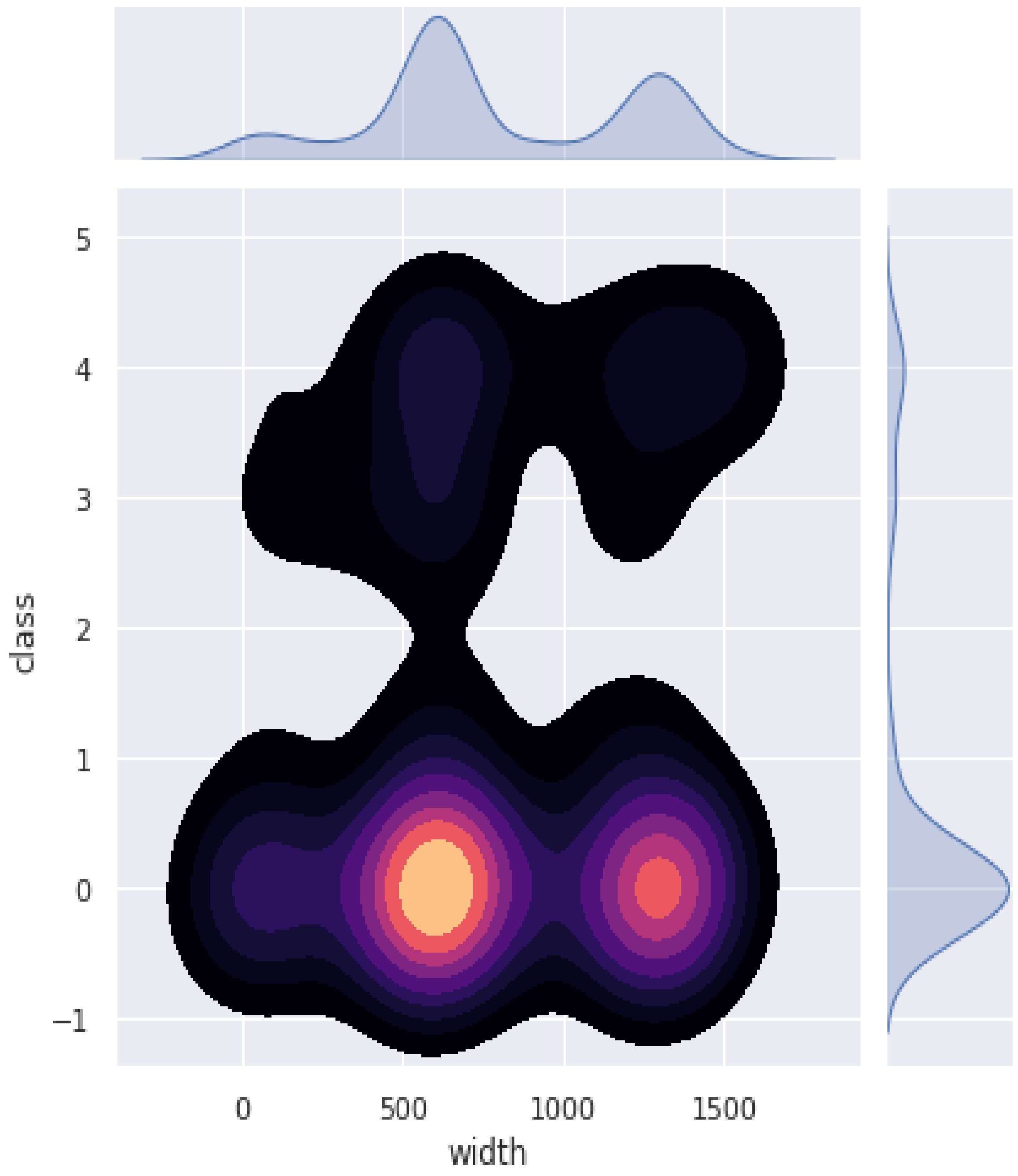
Question 1 - PCA

1. Pre-processing and meaningful data analysis- Preprocessing was done on the dataset to make it fit for the task to be done. 'Shape', 'bore', 'class', 'steel', and ‘class’ columns were encoded and columns like ‘temper_rolling’ and ‘non-aging’ were dropped because of a huge number of missing values.
   Joint plots of continuous features vs class were plotted to visualize the relation of continuous features and classes.
   The plot can be seen below:

Thick vs Class

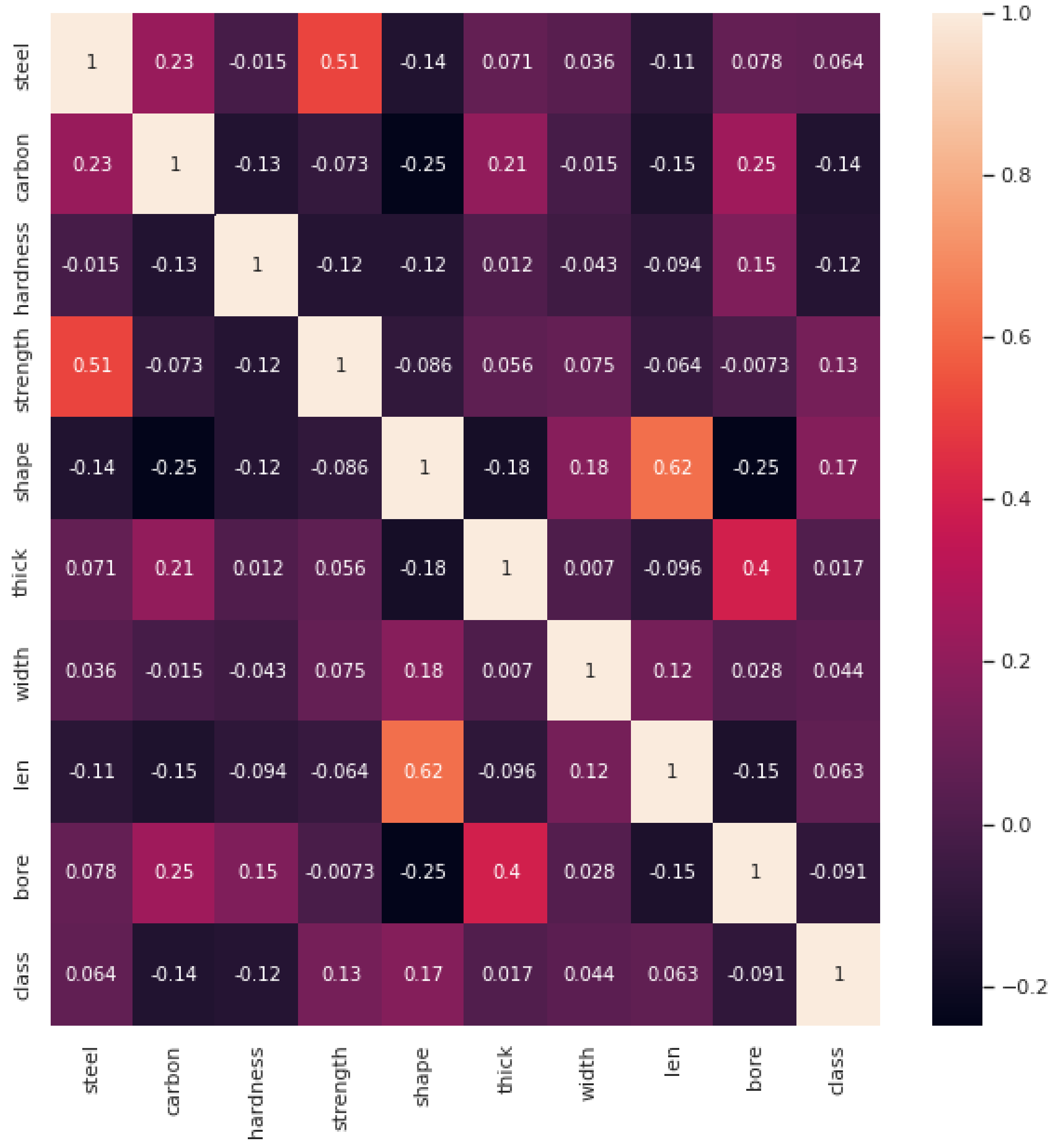


Width vs Class



A heatmap was also plotted to visualize the relationship between different features. ‘Strength’ and ‘Steel’ columns were highly correlated. ‘Class’ column was related to ‘shape’ and ‘strength’ columns. All the different relations can be visualized in the heatmap. df. corr() code was used to find the correlation between different features and classes.

The heatmap can be seen below:

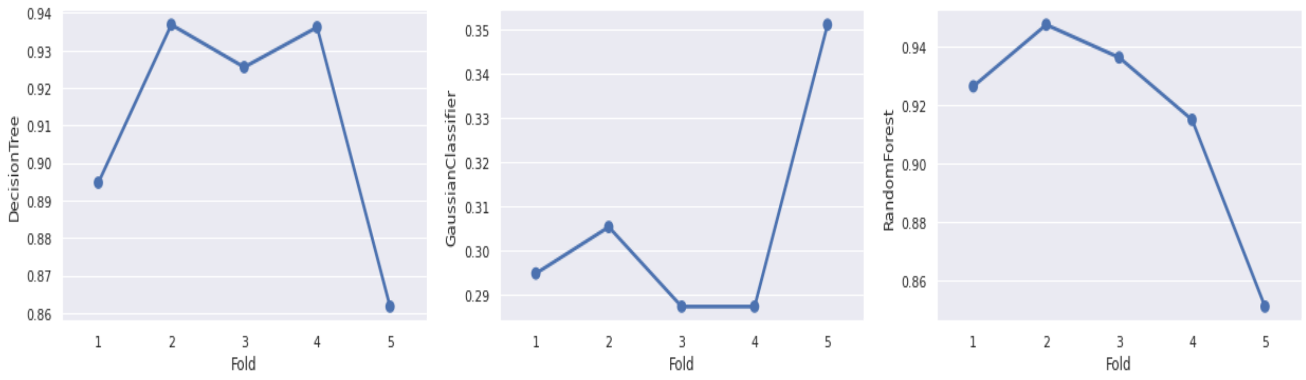


2. Preprocessing was done in the 1st subpart only. Codes can be found in the google colab notebook. All the columns having more than a specific amount of missing values were removed. After that, all the rows with missing values were removed, ensuring no missing values are left. The dataset was divided into a 65:35 ratio which can be seen in the notebook.

3. The three classification models used were the Random forest classifier, decision tree classifier, and GaussianNB. Random Forest and Decision Tree were chosen because of the fit of the data. The data consists of a lot

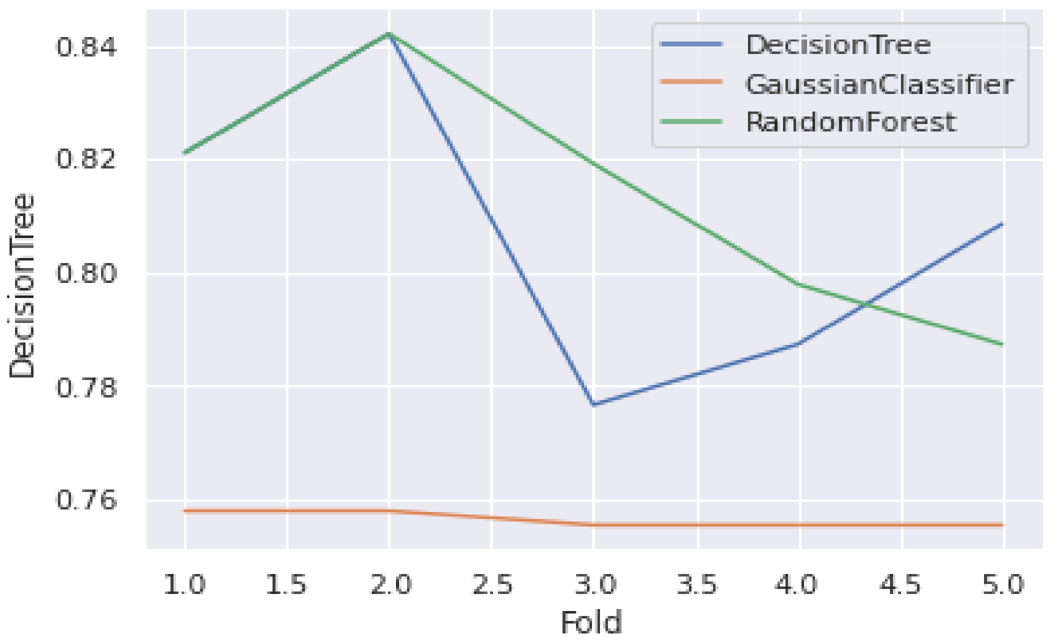
of categorical features which can be classified using Random Forest and Decision Tree. GaussianNB was chosen because of the two continuous features, ‘width’ and ‘thick’. 5-fold cross-validation was performed and the plots can be seen below, with Random Forest and Decision Tree models outperforming GaussianNB.

5-fold cross Validation



4. PCA was implemented from scratch and the required functions were included in the PCA class itself.
   The data was centralized by calculating the mean of the features and storing it as a vector. After that, the mean vector was subtracted from each row and was then divided by the standard deviation. The covariance matrix of which resulted in a matrix that had all its diagonal entries as 1. Eigenvectors and Eigen Values were calculated using the QR decomposition. QR decomposition was performed on the covariance matrix for 1000 iterations to get an approximation of the eigenvalues. Columns of the Q matrix represented the eigenvectors. These eigenvectors were sorted according to the eigenvalues in descending order and the first p eigenvectors were selected. ‘p’ was given as an input to the PCA class.
5. PCA was initialized with components equal to two. PCA was fitted with the dataset and the components were stored in the class itself. Chosen models were the same as in the previous part. 5-fold cross-validation results were stored for each model and the results were stored in a dictionary.

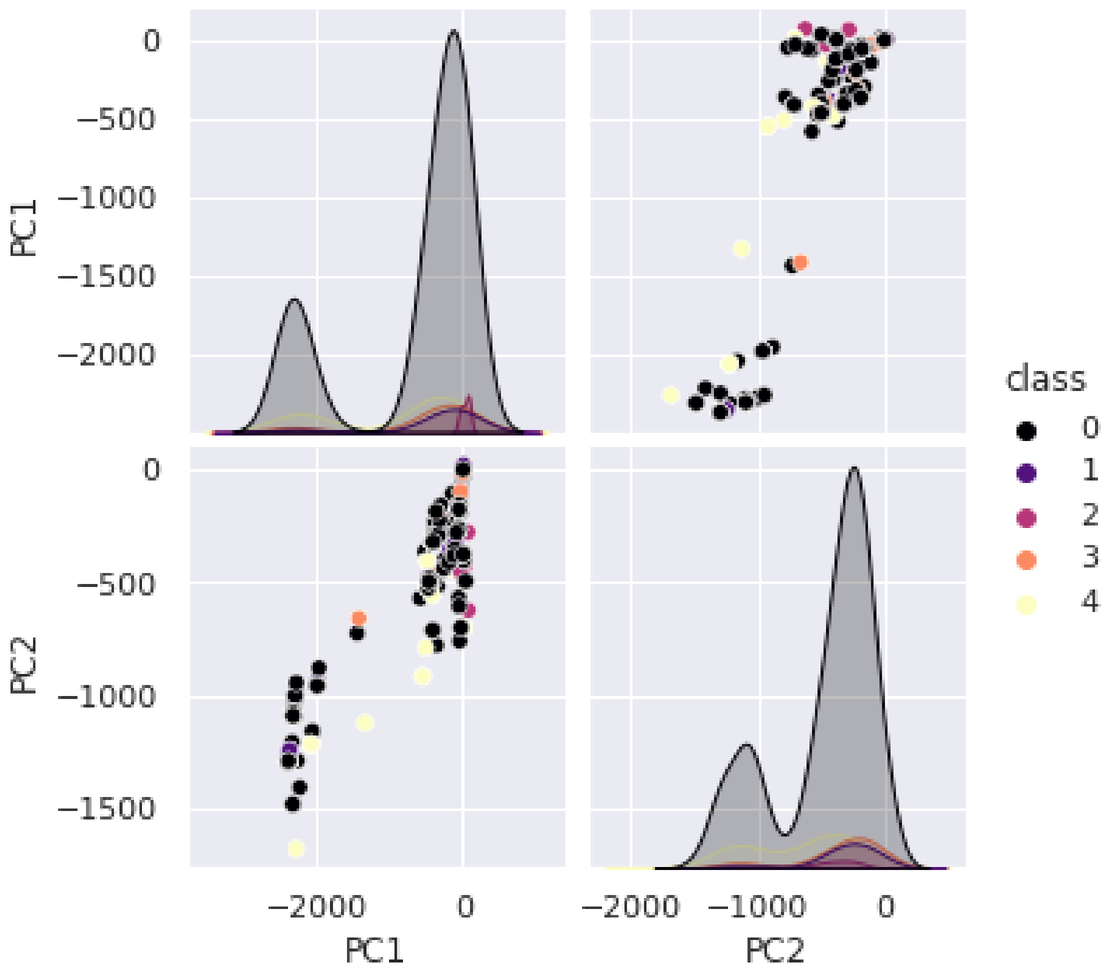
5-fold cross-validation can be seen in the plots below:



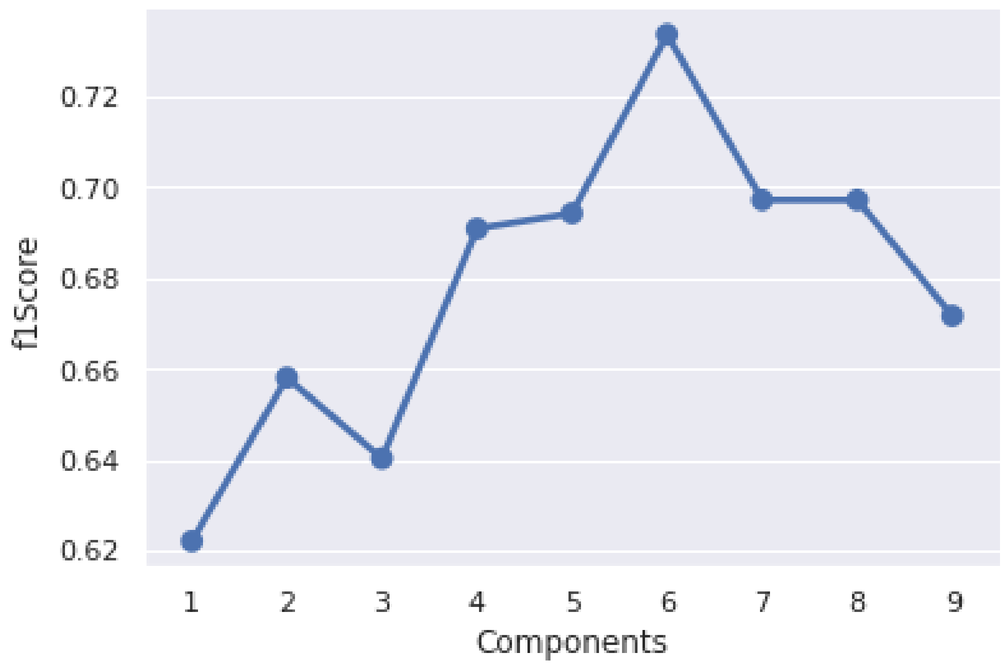
6. The same three models as trained in the previous subparts were chosen with accuracy and f1score as the parameter. f1score was chosen as the parameter because of the imbalance in the dataset (Class 0 dominates). F1score includes both ‘precision’ and ‘recall’ in the picture. The results can be seen below:
   DecisionTree :
   With PCA Accuracy: 0.7607843137254902
   Without PCA Accuracy: 0.8117647058823529
   With PCA f1Score : 0.6610407876230662
   Without PCA f1Score: 0.7439237206386936
   GaussianClassifier :
   With PCA Accuracy : 0.788235294117647
   Without PCA Accuracy : 0.5294117647058824
   With PCA f1Score : 0.17631578947368423
   Without PCA f1Score : 0.39242029075934365
   RandomForest :
   With PCA Accuracy: 0.7607843137254902
   Without PCA Accuracy: 0.8588235294117647
   With PCA f1Score : 0.6053316222364813
   Without PCA f1Score: 0.7341199528440908

RandomForest and DecisionTree performed equally well with RandomForest performing better in accuracy whereas Decision Tree performing better in terms of f1Score. After PCA, accuracy, and f1score slightly decreased for each model as there was information loss. The total variance was not conserved. Gaussian performed poorer than the other two models though its accuracy increased after applying PCA transformation. The features become less dependent, making it easy for gaussian's naive approach in classifying the data.

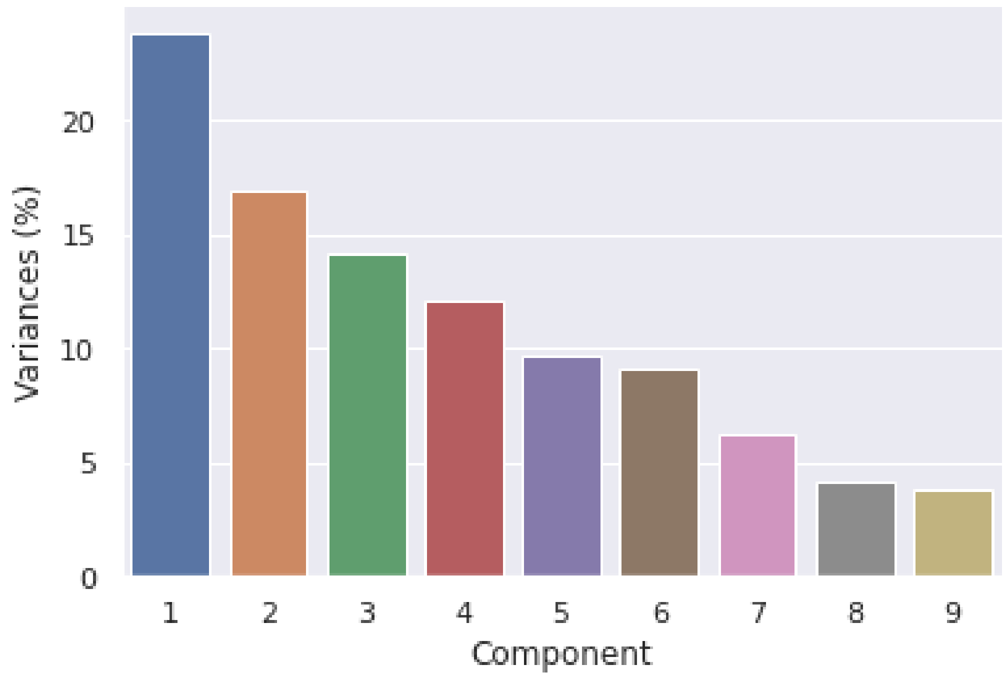
7. There was a remarkable improvement in the visualization of the dataset. The two principal components instead of a number of features made it easy to visualize the data. Nine features in the original dataset made it difficult to analyze the data. The distribution in the transformed data clearly draws a distinction between different classes. The plot below clearly shows the benefits of PCA:



A suitable graph could be of two types.
First could be checking f1scores for a different number of components and then selecting the desired ones.
A plot of which can be seen below:



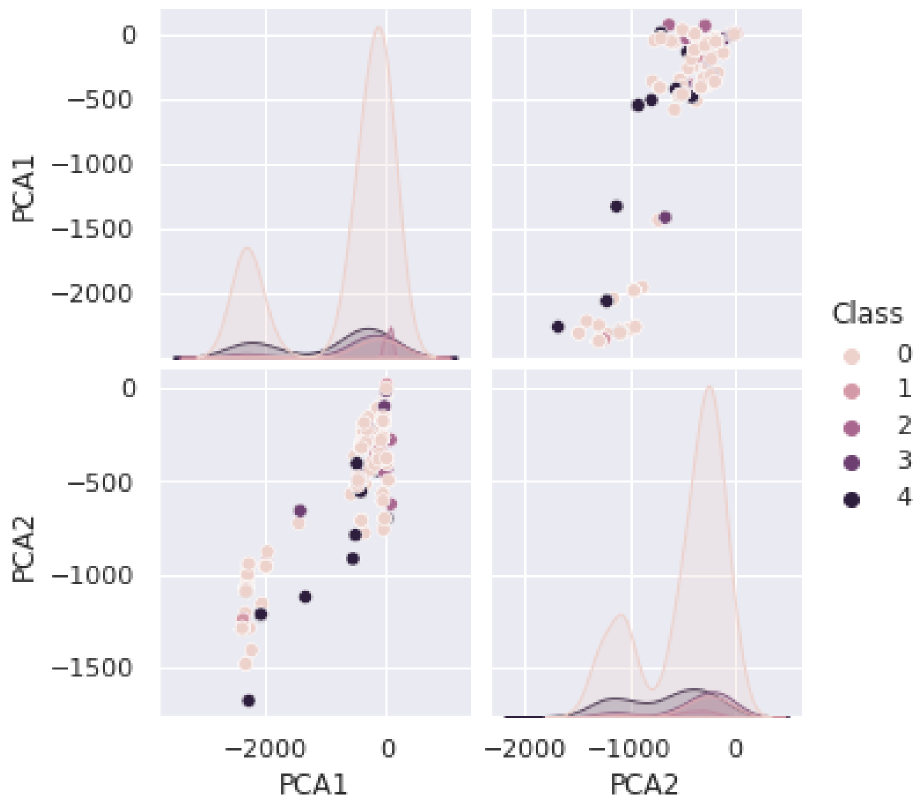
The second kind of plot can be the plot explaining the variances conserved by each component and then deciding the total components.



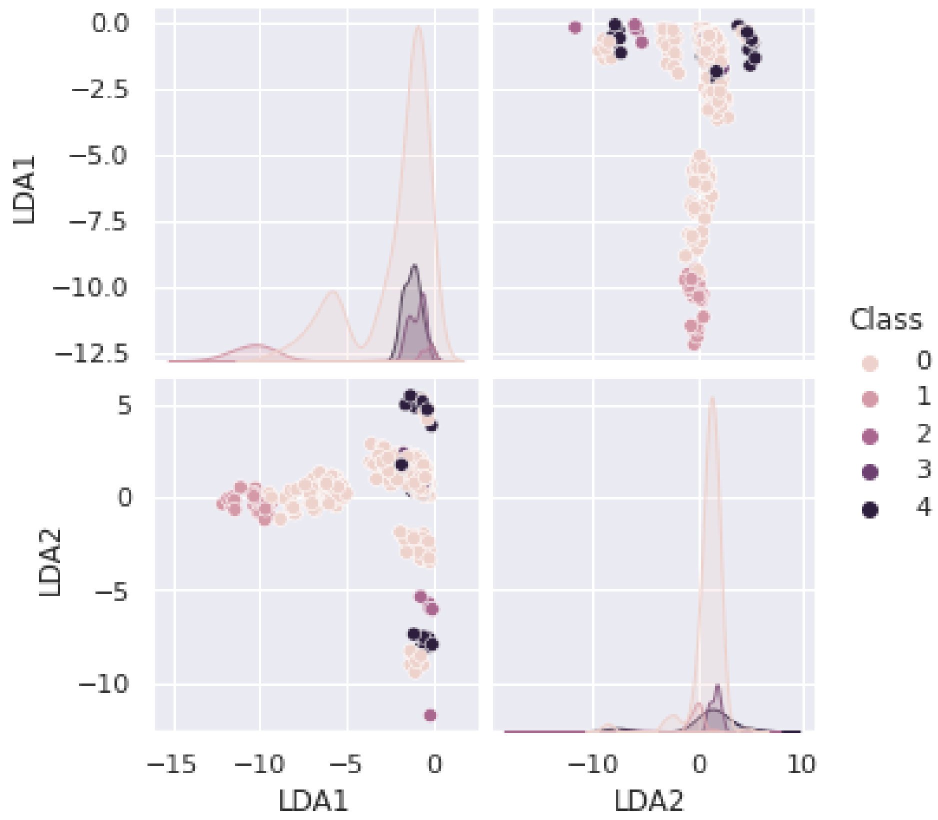
Question 2 - LDA

1. LDA was implemented from scratch and the functions for computing class within and class between matrices were included in the LDA class itself. A function that will automatically select the number of linear discriminants based on the variance to be conserved was also included in the LDA class. Variance to be stored was provided as an input to the class and function stored as many components that were necessary. A self-explanatory code of which can be found in the colab notebook.
2. PCA and LDA both were performed and the results were compared. RandomForest was trained on the transformed data and the results can be seen below:
   PCA (RandomForest)(Accuracy) --> 0.8433734939759037
   LDA (RandomForest)(Accuracy) -- > 0.9156626506024096
   PCA (RandomForest)(F1score) --> 0.47188811188811197
   LDA (RandomForest)(F1score) -- > 0.8381124913733609
   As we can see LDA transformation boosted the f1 score in the case of the RandomForest classifier.
   Plots explaining the transformation can be seen below:

PCA transformation



LDA transformation



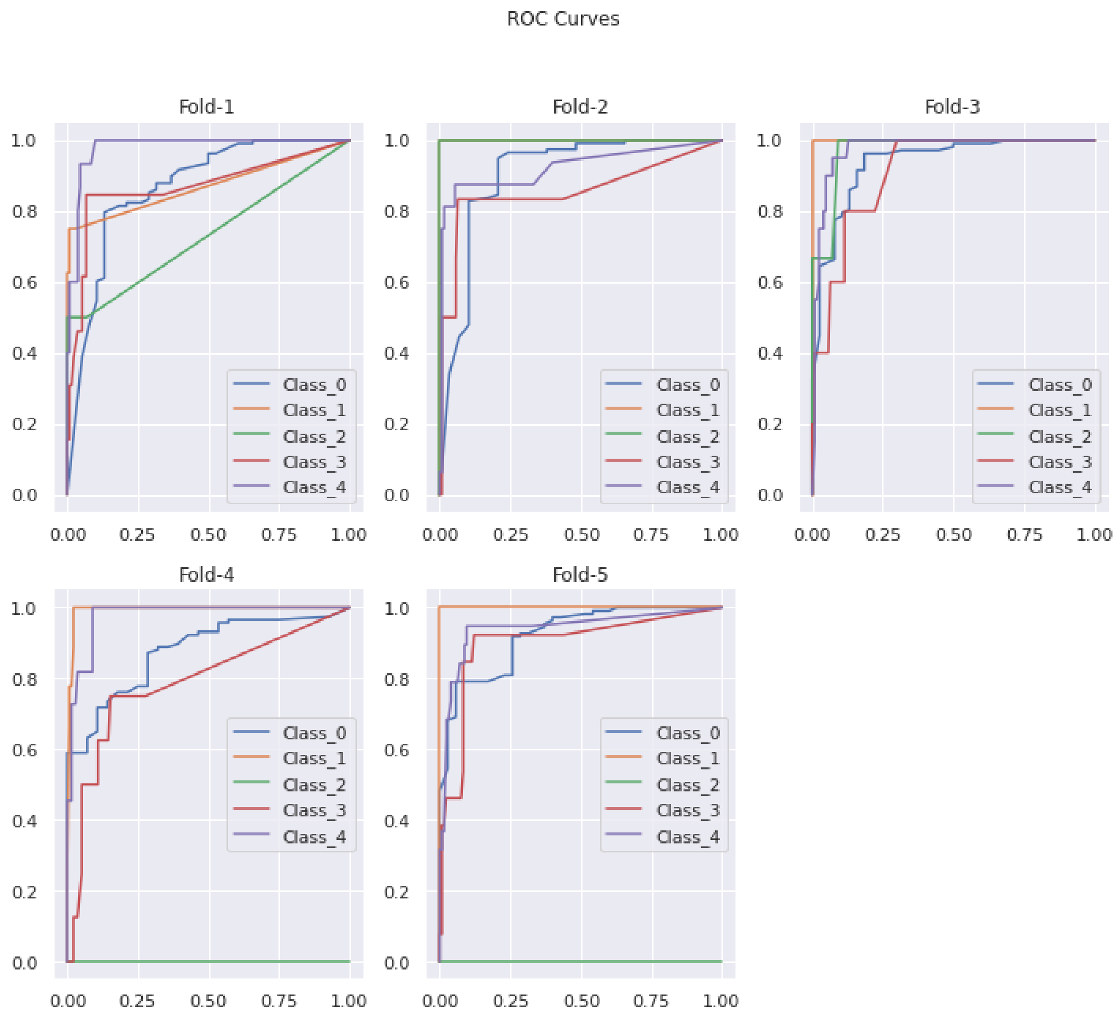
Distinctions between the classes are more clearly visible in the case of LDA as compared to PCA. In the LDA transformation, means of different classes were separated from each other as it is clearly visible in the plots. In the case of PCA, the data was transformed on the principal components and the transformation can be seen in the above plots.

3. Features having a higher impact on the Principal components and LDA were calculated using the most significant principal component and LDA component respectively. Greater the absolute value of the component at feature’s position more was its contribution. The function for it was included in the respective classes. The self-explanatory codes can be found in the colab notebook. Visualization of the PCA and LDA with two components was done in the previous part.
4. GaussianNaiveBayes and RandomForest Classifiers were used with LDA and PCA Feature Extraction. The results can be seen below:

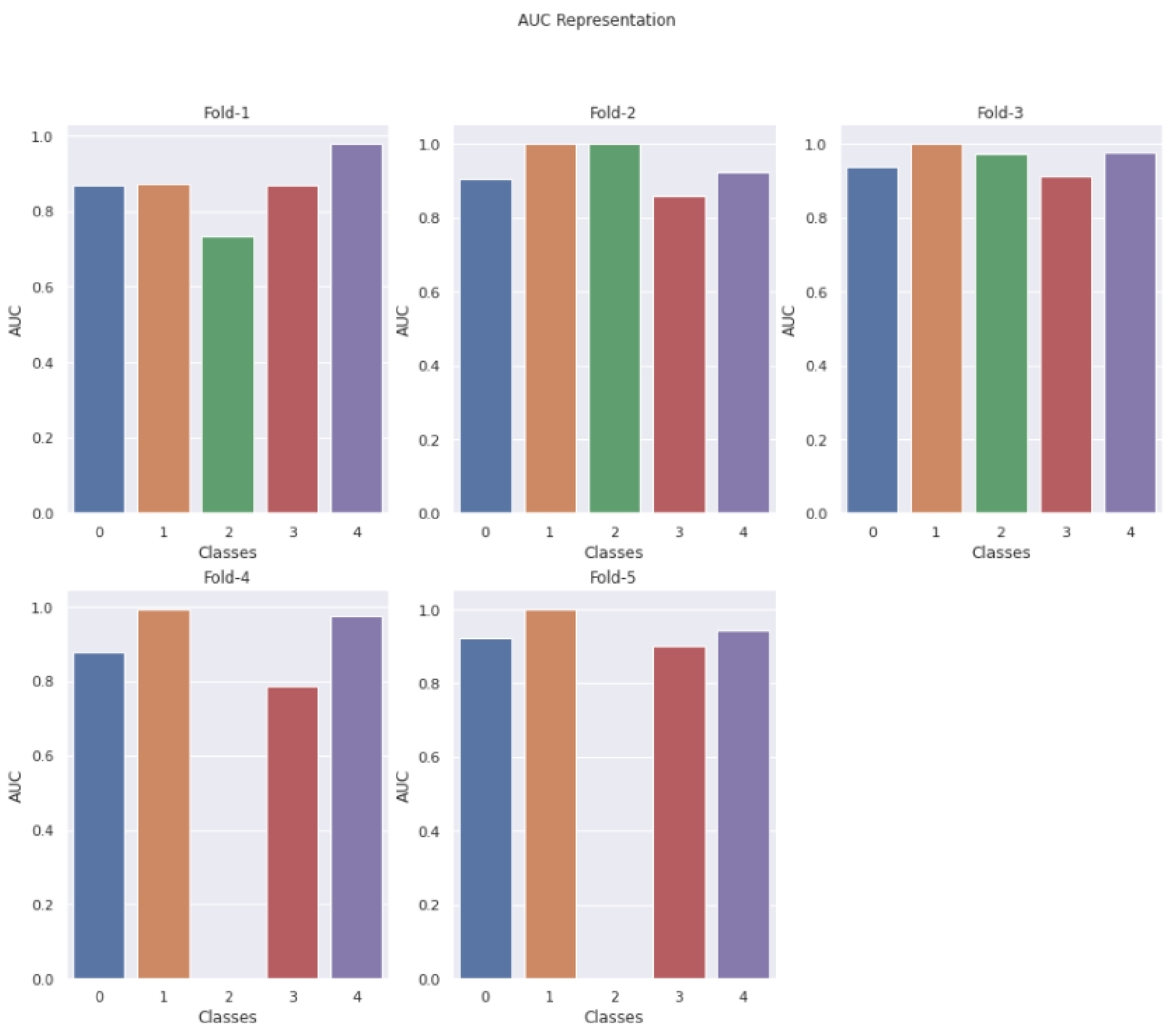
	RandomForest	GaussianNB
LDA	0.927710843373494	0.5301204819277109
PCA	0.8433734939759037	0.6746987951807228

Clearly, RandomForest performed better than GaussianNB and LDA gave better results than PCA feature extraction. Though accuracy for GaussianNB increased in the case of PCA feature extraction as the components are perpendicular to each other and the data is transformed on them. This allows the Gaussian algorithm to perform better. Overall, the RandomForest Classifier along with LDA performs than other permutations.

5. LDA was used as a classifier and the ROC plots were plotted for different folds



AUC representation -



Values for each AUC can be found in the colab notebook itself.

- **Piyush Arora (B20CS086)**