Project Report

Group-2
Ganiv Kaur MS21187
Abin Mathew MS21087
Berleen Kaur MS21089

Contributions:

Everyone contributed equally and wrote the code for different models for dimensionality reduction and classifier model and the model which gave the best accuracy was chosen.

Data location:

1. You can download the data from the google drive in your system and then upload the file from your computer which can be saved anywhere without providing the whole path of the file
2. You can also run the second code which uses the file saved in the google drive directly once you have the access to the drive folder /IDC409_PROJECT_Group2/ and it is saved in your /content/drive/My Drive. ('/content/drive/My Drive/IDC409_PROJECT_Group2/data_hep.csv')

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a powerful dimensionality reduction technique that plays a pivotal role in the world of machine learning and data analysis. It is utilised to transform high-dimensional data into a lower-dimensional space while preserving the most critical information regarding class separability. This intricate process involves several key steps that collectively contribute to the efficacy of LDA. In this comprehensive exploration, we will delve deeper into the inner workings of LDA, elucidating each step with precision and clarity.

LDA commences its journey by aiming to optimise class separability in a dataset with multiple classes. The central objective is to uncover a lower-dimensional representation of the data that maximises the distance between class centroids while minimising the variance within each class. This technique is invaluable in numerous applications, including pattern recognition, image analysis, and bioinformatics, where dimensionality reduction can significantly enhance the performance of machine learning models.

1. Calculating the between-class variance:

1. The initial phase of LDA involves calculating the between-class variance, a pivotal metric that quantifies the degree of separation between the different classes in the dataset.
2. To accomplish this, LDA computes the centroids (or means) of each class in the original feature space. These centroids represent the centre of each class, and they are instrumental in assessing class separability.
3. In addition to the class centroids, LDA determines the overall mean of all data points, which acts as the global centroid, serving as a reference point.
4. The between-class variance is calculated by measuring the sum of weighted squared distances between the class centroids and the global centroid. This distance evaluation effectively captures the extent to which the class centroids are spread apart in comparison to the global centroid. A larger between-class variance indicates better separation of classes in the original feature space.

$$s_1 = \sum_{x_i \in c_1} (x_i - \mu_1)(x_i - \mu_1)^T$$

www.geeksforgeeks.org

2. Calculating the within-class variance:

1. Following the determination of the between-class variance, LDA shifts its focus towards the within-class variance. This metric evaluates the compactness of data points within each class, an essential aspect of class separability.
2. LDA calculates a scatter matrix for each class, which essentially represents the variance within that particular class. This scatter matrix is computed by summing the weighted squared distances between each data point in the class and its respective class centroid.
3. The within-class variance is obtained by aggregating these scatter matrices from all classes. It offers a comprehensive view of the overall variance within the classes, emphasising the importance of minimising variance within each class to improve class separability.

$$s_w = s_1 + s_2$$

$$s_b = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$ Sw = within class matrix Sb = between class matrix

www.geeksforgeeks.org

3. Constructing the lower-dimensional space:

1. The ultimate objective of LDA is to identify a transformation matrix (P) that can effectively project the data into a lower-dimensional space. This transformation should be chosen in a manner that optimises class separability while minimising the within-class variance.
2. To achieve this, LDA adheres to Fisher's criterion, a mathematical principle that seeks to maximise the ratio of between-class variance to within-class variance. In simpler terms, LDA aims to find a transformation matrix (P) that maximises the separation between classes while minimising the variance within each class.
3. This mathematical optimization process is realised by identifying the eigenvectors of a specific matrix, namely, the matrix resulting from the multiplication of the inverse of the within-class scatter matrix (S_w) with the between-class scatter matrix (S_b). These eigenvectors constitute the columns of the transformation matrix P.
4. Once the transformation matrix P is successfully determined, data can be projected into the lower-dimensional space, thereby preserving class separability more effectively. The process ensures that the transformed data retains the essential discriminative information while being represented in a more compact and informative manner.

$$P_{lda} = \arg\max_{P} \frac{\left| P^T S_b P \right|}{\left| P^T S_w P \right|}$$

www.knowledgehut.com

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique that plays a pivotal role in improving the performance of machine learning models in classification problems. Its intricate methodology involves three fundamental steps: calculating the between-class variance, determining the within-class variance, and constructing the lower-dimensional space. LDA aims to find the optimal transformation that enhances class separability while minimising the variance within each class. This technique is widely applied in various domains where dimensionality reduction and enhanced discrimination between classes are of paramount importance.

.

Prediction of LDA using Bayes' Theorem

Linear Discriminant Analysis (LDA) in machine learning makes predictions using Bayes' Theorem. Let's break down the key concepts and the mathematical expression involved in this process:

1. Probability Estimation:

LDA leverages Bayes' Theorem to estimate the probability that a new input dataset belongs to each class. The goal is to assign a probability to each class and select the class with the highest probability as the output class.

2. Bayes' Theorem:

Bayes' Theorem is used to calculate the conditional probability that a new data point belongs to a specific class given the input data. It considers both the prior probability of each class (Plk) and the likelihood of the data point belonging to each class (fk(x)).

3. Components of the Equation:

1. The probability P(Y=x|X=x) represents the conditional probability that the output class (Y) is equal to a specific class (k) given the input data (X=x).
2. Plk is the prior probability, representing the base probability of each class observed in the training data. It is essentially the relative frequency of each class in the training dataset.
3. fk(x) is the estimated probability that the input data point x belongs to class k. It characterizes how well the input data aligns with the characteristics of each class.

4. Gaussian Distribution Function:

1. LDA often models the likelihood fk(x) as a Gaussian distribution. This is a common assumption in LDA, where it is assumed that the data within each class follows a multivariate Gaussian distribution.
2. The Gaussian distribution is defined by its mean (average) and variance (Σ^2) for each class. These parameters are estimated from the training data.

5. Discriminant Function:

1. The discriminant function Dk(x) is a key component in the prediction process. It is used to calculate the discriminant value for each class given the input data point x.
2. The discriminant function is expressed as $Dk(x) = x * (mean / \Sigma^2) - (mean^2 / (2 * \Sigma^2)) + \ln(Plk)$.
3. This function combines the input data, class-specific mean and variance, and the prior probability to calculate the discriminant score for class k.

6. Prediction:

1. To make a prediction, LDA calculates the discriminant function Dk(x) for each class k. The class with the highest discriminant value is considered the output class.
2. In other words, the class that maximises the discriminant function is chosen as the predicted class for the input data point.

LDA in machine learning uses Bayes' Theorem and Gaussian distribution modelling to estimate the probability that an input data point belongs to each class. The discriminant function is then used to calculate discriminant values for each class, and the class with the highest discriminant

value is predicted as the output class. This approach is effective for classification tasks and is based on sound statistical principles.

Random Forest Classifier (RFC)

Random Forest is indeed a popular machine learning algorithm that falls under the category of supervised learning techniques. In supervised learning, the algorithm is trained on labelled data, which means it learns to make predictions or decisions based on input features while having access to the correct target values or labels. Random Forest is widely used for both classification and regression tasks, making it a versatile and powerful tool in the field of machine learning

1. Supervised Learning Algorithm:

Random Forest is a supervised learning technique, which means it requires labelled data, where the target variable is known, to train a predictive model.

2. Versatility:

Random Forest can be used for both classification and regression problems. In classification, it predicts the class label of an input data point, while in regression, it predicts a continuous numerical value.

3. Ensemble Learning:

Random Forest is based on the concept of ensemble learning, which involves combining the predictions of multiple machine learning models to improve overall predictive accuracy and reduce the risk of overfitting.

4. Multiple Decision Trees:

1. In a Random Forest, multiple decision trees are constructed. Each tree is trained on a different subset of the dataset. This process is known as bootstrapping or bagging, and it involves randomly selecting subsets of the data with replacement.
2. These decision trees are known as "weak learners" because they may not be highly accurate individually, but they contribute to the overall accuracy of the model when combined.

5. Averaging Predictions:

1. When making predictions, the Random Forest algorithm takes the predictions from each of the decision trees. In the case of classification, it counts the votes of each tree for the class label, and the class with the majority of votes is chosen as the final prediction.
2. This majority voting mechanism helps improve the robustness and accuracy of the model. In the case of regression, it typically takes the average of the predictions from individual trees.

6. Preventing Overfitting:

1. One of the advantages of Random Forest is that it is less prone to overfitting compared to individual decision trees. By averaging predictions from multiple trees, it reduces the risk of a single tree learning the noise in the data.
2. Additionally, by using random subsets of the data for training each tree, Random Forest introduces diversity in the ensemble, further reducing the risk of overfitting.

7. Hyperparameter:

The number of decision trees in the Random Forest, known as the "n_estimators" hyperparameter, can be adjusted to control the trade-off between model complexity and accuracy. A larger number of trees generally leads to higher accuracy but also increases computational complexity.

Random Forest is a powerful ensemble learning algorithm that combines multiple decision trees to make robust predictions for classification and regression tasks. It is known for its versatility, high accuracy, and its ability to prevent overfitting, making it a popular choice in various machine learning applications.

Assumptions of Random Forest Classifier

1. Diverse Trees for Improved Accuracy:

1. One of the key strengths of Random Forest is its ability to leverage the wisdom of crowds by combining multiple decision trees. Each decision tree in the ensemble is trained on a different subset of the data.
2. The idea is that by having different trees with diverse perspectives, the collective ensemble can improve prediction accuracy. While some trees may make incorrect predictions, others may make the correct ones, and when their votes are combined, the overall prediction tends to be more accurate.

2. Actual Values for Accurate Predictions:

1. Random Forest, like most machine learning algorithms, benefits from having actual and informative values in the feature variables. The model relies on patterns and relationships in the data to make predictions.
2. It's crucial that the feature variables contain meaningful information rather than arbitrary or guessed values. High-quality data with real-world relevance generally leads to more accurate and reliable predictions.

3. Low Correlation of Predictions:

1. You mentioned that the predictions from each tree should have low correlations, which is an important aspect. This is often achieved through the randomness introduced during the construction of each decision tree.
2. Random Forest uses bootstrapping, where each tree is trained on a random subset of the data, and at each split in the tree, a random subset of features is considered. This introduces diversity in the predictions made by each tree, as different trees see different data and features.
3. By having low correlations between the predictions of individual trees, the ensemble can effectively reduce overfitting and produce more robust and accurate predictions.

Random Forest excels in harnessing the power of ensemble learning, where multiple decision trees work together to enhance prediction accuracy. The diversity of the trees, real values in feature variables, and low correlations between tree predictions are all important considerations to achieve the full potential of this algorithm in various machine learning tasks.

Random Forest Classifier Algorithm

There are two main phases of how a Random Forest works:

Phase 1: Random Forest Construction

1. Step 1: Data Sampling

Random Forest begins by selecting random subsets of the training data. These subsets are often referred to as "bootstrapped" samples, as they are created by randomly choosing data points with replacement. Each subset typically contains K data points, where K is less than or equal to the total number of data points.

2. Step 2: Decision Tree Building

A decision tree is constructed for each of these bootstrapped samples. Each tree is built independently and can have its own unique structure. The trees are typically deep and capture different patterns in the data, which helps in reducing overfitting and improving the overall predictive power of the ensemble.

3. Step 3: Number of Trees (N)

The user specifies the number of decision trees to create, denoted as N. The choice of N can impact the trade-off between model complexity and accuracy. A larger N generally leads to a more robust model but also increases computational resources required.

4. Step 4: Repetition

Steps 1 and 2 are repeated N times, resulting in N decision trees, each trained on a different bootstrapped subset of the data.

Phase 2: Making Predictions

5. Step 5: Prediction Aggregation

When making predictions for new data points, the Random Forest algorithm collects predictions from each of the N decision trees. In the case of classification tasks, it counts the votes for each class. The class with the most votes is considered the final prediction for the new data point. In regression tasks, it takes the average of the individual tree predictions as the final prediction.

Here is a simplified diagram to illustrate the process:

```
Phase 1: Random Forest Construction
------------------------------------------------
1. Data Sampling
  - Subset 1 (Bootstrapped)
  - Subset 2 (Bootstrapped)
  - ...
  - Subset N (Bootstrapped)

2. Decision Tree Building
  - Tree 1
  - Tree 2
  - ...
  - Tree N

Phase 2: Making Predictions
------------------------------------------------
3. For New Data Point:
  - Collect Predictions from Each Tree
  - Aggregate Predictions (Majority Voting for Classification or Average for Regression)
  - Final Prediction
```

Random Forest combines the strength of multiple decision trees to create a more accurate and robust predictive model. Each tree is constructed independently, and the final prediction is determined by aggregating the predictions from all trees. This ensemble approach is known for its effectiveness in various machine learning tasks.

Results

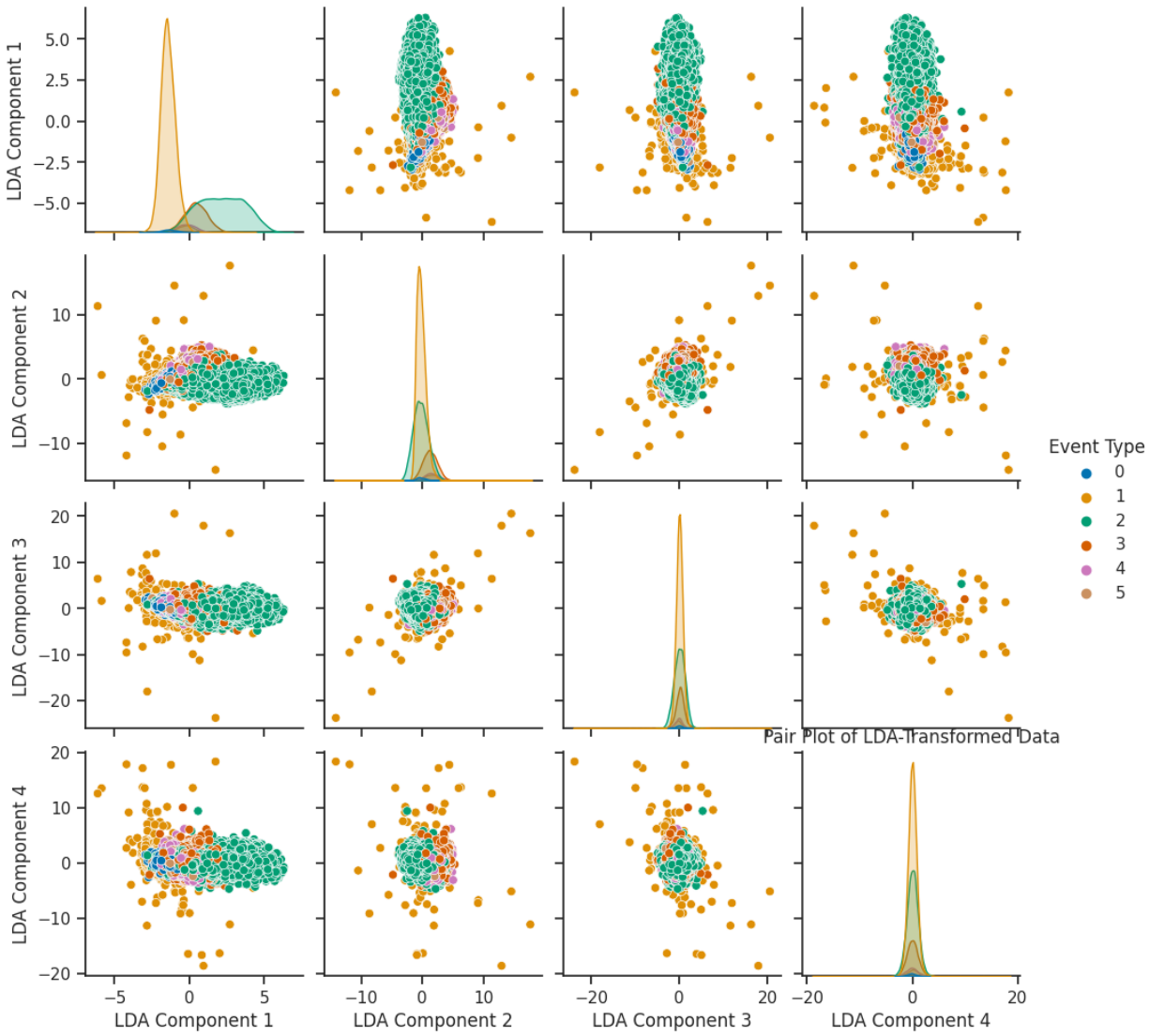
This is the pairplot of the four components of the four LDA components with respect to each other after classification by random forest classifier. The diagonal histograms provide insights into the distribution of individual variables, allowing us to understand their frequency and pattern. On the other hand, the scatterplots in both the upper and lower triangles reveal how two variables relate to each other, helping us observe any existing relationships or the absence of such connections.

Interpreting Scatterplots:

Positive/Negative Correlation:
If two variables have a positive correlation, points on the scatterplot tend to form an upward-sloping pattern. In contrast, a negative correlation will show a downward-sloping pattern.

No Correlation:
If there's no correlation, points are scattered randomly, and the scatter plot appears as a cloud.

Strength of Correlation:
The more closely the points cluster around a line, the stronger the correlation. A wider spread indicates a weaker correlation.



Interpretation of pairplot:

1. Components 2,3,4 are not correlated as stated above; they are randomly distributed and appear as a cloud.
2. Components 1 and 4 are positively correlated as they form an upward-sloping pattern and are quite strongly correlated as are clustered very closely.
3. Event type 1 is most abundant (orange in colour) in each of the LDA components as can be seen in scatter plots as well as the diagonal plots.
4. The second most abundant event type is Event 2.
5. The least abundant Events are 0, 4 and 5
6.

Accuracy and the Classification Report

Event Type: e+ e- -> Upsilon(4S) -> B+ B- (Label 0):
Accuracy: 0.99

	precision	recall	f1-score	support
0	0.99	1.00	1.00	14022
1	0.00	0.00	0.00	100
accuracy			0.99	14122
macro avg	0.50	0.50	0.50	14122
weighted avg	0.99	0.99	0.99	14122

Event Type: e+ e- -> Upsilon(4S) -> B0 B0-bar (Label 1):
Accuracy: 0.94

	precision	recall	f1-score	support
0	0.96	0.92	0.94	7021
1	0.93	0.97	0.95	7101
accuracy			0.94	14122
macro avg	0.95	0.94	0.94	14122
weighted avg	0.95	0.94	0.94	14122

Event Type: e+ e- -> c c-bar (Label 2):
Accuracy: 0.92

	precision	recall	f1-score	support
0	0.93	0.96	0.94	9507
1	0.90	0.85	0.87	4615
accuracy			0.92	14122
macro avg	0.92	0.90	0.91	14122
weighted avg	0.92	0.92	0.92	14122

Event Type: e+ e- -> u u-bar (Label 3):
Accuracy: 0.91

	precision	recall	f1-score	support
0	0.94	0.96	0.95	12469
1	0.66	0.52	0.58	1653
accuracy			0.91	14122
macro avg	0.80	0.74	0.77	14122
weighted avg	0.91	0.91	0.91	14122

Event Type: e+ e- -> d d-bar (Label 4):
Accuracy: 0.97

	precision	recall	f1-score	support
0	0.97	1.00	0.99	13733
1	0.41	0.03	0.06	389
accuracy			0.97	14122
macro avg	0.69	0.51	0.52	14122
weighted avg	0.96	0.97	0.96	14122

Event Type: e+ e- -> s s-bar (Label 5):
Accuracy: 0.98

	precision	recall	f1-score	support
0	0.98	1.00	0.99	13858
1	0.11	0.01	0.01	264
accuracy			0.98	14122
macro avg	0.55	0.50	0.50	14122
weighted avg	0.97	0.98	0.97	14122

This report tells us that

1. Our model is good at predicting if the event type is not ‘0’ but is very poor in predicting the presence of event type ‘0’.
2. Our model is very good at predicting if the event type is not ‘1’ as well as pretty good at predicting presence of event type ‘1’.
3. Our model is very good at predicting if the event type is not ‘2’ as well as good at predicting presence of event type ‘2’.
4. Our model is very good at predicting if the event type is not ‘3’ and decent in predicting presence of event type ‘3’.
5. Our model is very good at predicting if the event type is not ‘4’ and decent in predicting presence of event type ‘4’.
6. Our model is very good at predicting if the event type is not ‘5’ and relatively poor in predicting presence of event type ‘5’.

HeatMap

Heatmaps are valuable tools for visualising and interpreting data patterns, relationships, and trends.

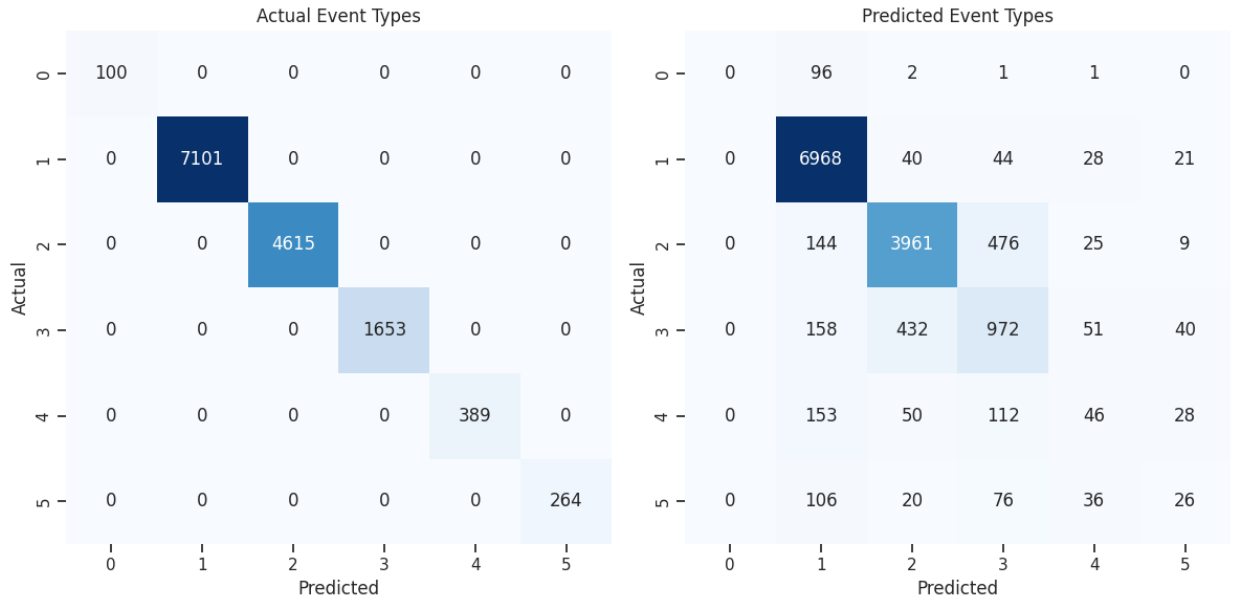
1. Color Coding and Scale:
Heatmaps use a color palette to represent data values. Typically, a gradient from hot (e.g., red or dark colors) to cold (e.g., blue or light colors) is used to indicate high-to-low values. Understanding the color scale and the value it represents is fundamental.

2. Identifying Patterns and Clusters:
The primary goal of analysing a heatmap is to identify patterns and clusters in the data. Look for regions where the color intensity is consistent, indicating a cluster of data points with similar values.

3. Correlation and Relationships:
Heatmaps are often used to visualise correlations between variables. Positive correlations (when one variable increases, the other also increases) are represented by similar colors, while negative correlations (one increases, the other decreases) are indicated by contrasting colors.

4. Interpretation and Inference:
Once you have identified patterns, clusters, correlations, and outliers, draw inferences and make interpretations based on your analysis.

Analysing a heatmap is a multi-step process that involves understanding the data, recognizing patterns, identifying clusters, and drawing meaningful inferences. It's a valuable tool for visual data exploration that can lead to deeper insights and help in making data-driven decisions.



Interpretation of HeatMap:

1. The 1st plot in the heatmap shows the actual event types where diagonals represent where the predicted events match with the actual event and the annotated numerical value indicates the number of matches between actual and predicted event types.
2. The second plot shows the heatmap of predicted values by random forest and the actual event types in the data given.
3. The second plot shows that the model was not able to predict the ‘0’ event type due very low abundance of the that event i.e. 100 out of 70606 data points
4. It also shows that the model was very good in predicting ‘1’ event type due its large abundance.
5. It can also be observed that the model did well at predicting event type ‘2’ but was not very efficient in predicting event type ‘3’,’4’ and ‘5’.
6. Wrong predictions are quite spread across all event types but are far less abundant as compared the whole dataset i.e. 70606 hence this much error is acceptable.

Confusion matrix

A confusion matrix is a tabular tool commonly employed in machine learning and statistics to evaluate the performance of a classification model. It provides a comprehensive summary of classification outcomes by quantifying the occurrences of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) in the model's predictions.

The confusion matrix serves as a means to compare the model's predicted labels with the actual labels of the data points, enabling the assessment of the model's effectiveness in classifying input instances. Typically, classification models are designed to assign categorical labels to each input, and the confusion matrix is an invaluable tool in this context.

The accuracy of a model, a widely-used performance metric, can be calculated using the following formula: accuracy = (TP + TN) / (TP + TN + FP + FN). This equation quantifies the proportion of correct predictions (both true positives and true negatives) in relation to all predictions, offering a measure of the model's overall correctness in classifying data points.

```
[[13855     3]
 [  263     1]]
```

The above shown is the confusion matrix for the test data set compared with the predicted values by random forest classifier.

TP: 13855
TN: 1
FP: 3
FN: 263

Accuracy: 0.98

Hence the model fits well with the dataset as the number of TP and TN are cumulatively far more than the FP and FN. and the accuracy also indicates that the model fits perfectly and gives quite accurate results and predictions.

References:

https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/
https://www.knowledgehut.com/blog/data-science/linear-discriminant-analysis-for-machine-learning
https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/
https://www.javatpoint.com/machine-learning-random-forest-algorithm
https://towardsdatascience.com/visualizing-data-with-pair-plots-in-python-f228cf529166
https://attentioninsight.com/heatmaps-101/