Industrial Machinery Predictive Maintenance

CONTENTS

I. PROBLEM SETTING:

As Industry 4.0 develops quickly and manufacturing systems become more automated and complex, it becomes more important than ever to guarantee the dependability, safety, and maintainability of production equipment over its whole lifecycle. Achieving these objectives requires effective preventive maintenance, which can provide savings, improved safety, and decreased downtime. Nonetheless, maximizing resource utilization and reducing the chance of equipment failure present difficulties in contemporary production settings. Unexpected equipment failures in industrial settings result in large losses in output, income, and safety hazards. Conventional maintenance approaches, including reactive or preventative maintenance, are frequently expensive and ineffective. By predicting equipment breakdowns before they happen, predictive maintenance with data analytics provides a proactive solution to these problems. Nevertheless, there are unique difficulties associated with putting predictive maintenance systems into practice, such as problems with data quality, intricate feature engineering, and interpretability of models. The primary goal is to develop a model to predict the failure scenarios and apply them in real time environment to assess the machine failures and start the maintenance activity before it starts failing.

II. PROBLEM DEFINITION:

The creation of a predictive maintenance model for the vital machinery in a manufacturing facility is the specific issue this project attempts to solve. Predicting equipment failures ahead of time allows for timely intervention and reduces operational downtime. Multiclass classification models are being used to predict machine failure to overcome these difficulties. These models can then be tested against tangible results to determine their dependability and performance. Data-driven approaches have become more popular because theoretical models cannot fully capture the intricate processes of production equipment degradation brought on by uncertainties and shifting operating conditions. Predicting equipment failures ahead of time will allow for prompt intervention and minimize downtime. Important queries which are all addressed in this project are as follows:

1. Can we use past data to predict machinery failures with any degree of accuracy?

2. Which factors are the most important in predicting equipment failures?
3. How can the predictive maintenance model's interpretability and dependability be guaranteed?

III. DATA SOURCE:

This project's main source of data is a predictive maintenance dataset hosted on the UCI Machine Learning Repository that mimics actual situations that arise with industrial machinery. UDI, product ID, air temperature, process temperature, rotational speed, torque, tool wear, and machine failure indicators are among the 14 features that make up the 10,000 data points in the dataset.

DATASET URL:

https://archive.ics.uci.edu/dataset/601/ai4i+2020+predictive+maintenance+dataset

As there is a challenge of acquiring genuine predictive maintenance datasets, particularly due to limitations on availability and publication of real-world machinery maintenance data, so this is a synthetic dataset generates to emulate real-world scenarios encountered in industrial predictive maintenance to the best extent possible.

To explore more into the generation of synthetic datasets, we have delved into the research article by Barbiero et al. titled "A Methodology for Controlling Bias and Fairness in Synthetic Data Generation." In this study, a pioneering method is introduced to mitigate bias and ensure fairness in machine learning algorithms through synthetic data generation. The research article explores a methodology for generating synthetic datasets with controlled bias and fairness, addressing the critical need for unbiased and fair machine learning algorithms. It begins by discussing the challenges of collecting real-world data and the prevalence of bias in existing datasets. The proposed methodology utilizes structural equation modelling (SEM) to model bias through a probabilistic network, allowing users to set and adjust bias parameters.

The process involves several steps:

- Defining a probabilistic network representing feature dependencies using Structural Equation Modelling (SEM).

- Altering bias by tuning the direct influence among attribute pairs and overall bias of specific attributes.
- Deriving a multivariate probabilistic distribution summarizing the network.
- Sampling from the distribution to generate a latent dataset.
- Converting the latent dataset into categorical datasets.
- The methodology significantly reduces parameter count compared to traditional Bayesian Networks, while maintaining dataset balance. Evaluation metrics such as mutual information (MI) and demographic parity (DP) are used to assess bias and fairness.

Experiments demonstrate the methodology's capability to generate datasets with controlled bias and fairness, offering flexibility to simulate various scenarios for machine learning model development and evaluation. Examples include studying the impact of edge weights and conditional variances on mutual dependencies between nodes, as well as generating datasets to illustrate fair and unfair scenarios such as loan approval discrimination.

In conclusion, the proposed methodology provides a streamlined approach to generating synthetic datasets with controlled bias and fairness, essential for developing and evaluating bias mitigation approaches in machine learning models and classifiers. Future developments may focus on creating integrated tools to further streamline the generation process and enhance discrimination mitigation techniques.

IV. DATA DESCRIPTION:

The dataset includes unique identifiers for tracking machinery and product variants, as well as a variety of features related to predictive maintenance, including torque, temperature readings, rotational speed, and tool wear. The "failure type" designates the failure mode, while the "machine failure" label indicates whether a failure has happened. The dataset, which has 10,000 rows and 14 columns, offers a thorough foundation for developing and assessing predictive maintenance models.

The following table summarizes the data attributes from the dataset.

Variable Name	Role	Type	Description
UDI	ID	Integer	unique identifier ranging from 1 to 10000
Product ID	ID	Categorical	consisting of a letter L, M, or H for low (50% of all products), medium (30%), and high
			(20%) as product quality variants and a variantspecific serial number
Type	Feature	Categorical	consisting of a letter L, M, or H for low (50% of all products), medium (30%), and high (20%) as product quality variants
Air temperature	Feature	Integer	Random generated data later normalized to a standard deviation of 2 K around 300 K. Unit-K (Kelvin)
Process temperature	Feature	Integer	Random generated data normalized to a standard deviation of 1 K, added to the air temperature plus 10 K. Unit-K (Kelvin)
Rotational speed	Feature	Integer	calculated from power of 2860 W, overlaid with a normally distributed noise Unit-rpm (rotation per minute)
Torque	Feature	Integer	torque values are normally distributed around 40 Nm Unit-Nm (newton-meter)
Tool wear	Feature	Integer	Time taken for wear of the used tool Unit-min (minutes)
Machine failure	Target	Integer	Machine failure represents machine failed (1) or not (0)
TWF	Target	Integer	Failure type - tool wear failure (TWF)
HDF	Target	Integer	Failure type - heat dissipation failure (HDF)
PWF	Target	Integer	Failure type - power failure (PWF)
OSF	Target	Integer	Failure type - overstrain failure (OSF)

RNF	Target	Integer	Failure type - random failures (RNF)

V. DATA PREPROCESSING:

The first step in the preprocessing involves the checking of any missing values in the dataset, with that it is found no missing values are present in the dataset.

```
# Checking for missing values
missing_values = df.isnull().sum()
print("Missing Values:\n", missing_values[missing_values > 0])
```

```
Missing Values:
 Series([], dtype: int64)
```

Then all the numeric columns were converted into float datatype to make the subsequent processing steps simpler. Then the 'UDI' and product id columns were removed as they do not add value to the classification models.

```
df.drop(['UDI', 'Product ID'], axis=1, inplace=True)
```

Then the machine failure columns such as 'TWF', 'HDF', 'PWF', 'OSF' & 'RNF' were converted into a single machine failure column based on assigning failure codes for each type of failures to perform multiclass classification, then eliminating the original columns. And then performing the one-hot encoding on categorical column 'type'.

```
df['Machine failure'] = 0

# Assigning failure codes based on different factors
df.loc[df['TWF'] == 1, 'Machine failure'] = 1
df.loc[df['HDF'] == 1, 'Machine failure'] = 2
df.loc[df['PWF'] == 1, 'Machine failure'] = 3
df.loc[df['OSF'] == 1, 'Machine failure'] = 4
df.loc[df['RNF'] == 1, 'Machine failure'] = 5

# Dropping derived features
df.drop(['TWF', 'HDF', 'PWF', 'OSF', 'RNF'], axis=1, inplace=True)
```

VI. FEATURE ENGINEERING:

As part of feature engineering, new features based on existing columns in the Data Frame were created, which can potentially provide additional insights for analysis or modeling purposes.

Features generated are as below:

Power = Rotational speed [rpm] * Torque [Nm]: Calculates the power by multiplying the rotational speed (in revolutions per minute) with the torque (in Newton meters).

Power wear = Power * Tool wear [min]: Calculates the power wear by multiplying the power with the tool wear (in minutes).

Temperature difference = Process temperature [K] - Air temperature [K]: Computes the temperature difference by subtracting the air temperature (in Kelvin) from the process temperature (in Kelvin).

Temperature power = Temperature difference / Power: Calculates the temperature power by dividing the temperature difference by the power.

```
# Calculating new features based on the existing columns
df['Power'] = df['Rotational speed [rpm]'] * df['Torque [Nm]']
df['Power wear'] = df['Power'] * df['Tool wear [min]']
df['Temperature difference'] = df['Process temperature [K]'] - df['Air temperature [K]']
df['Temperature power'] = df['Temperature difference'] / df['Power']
```

```
df.describe(include='all').T
```

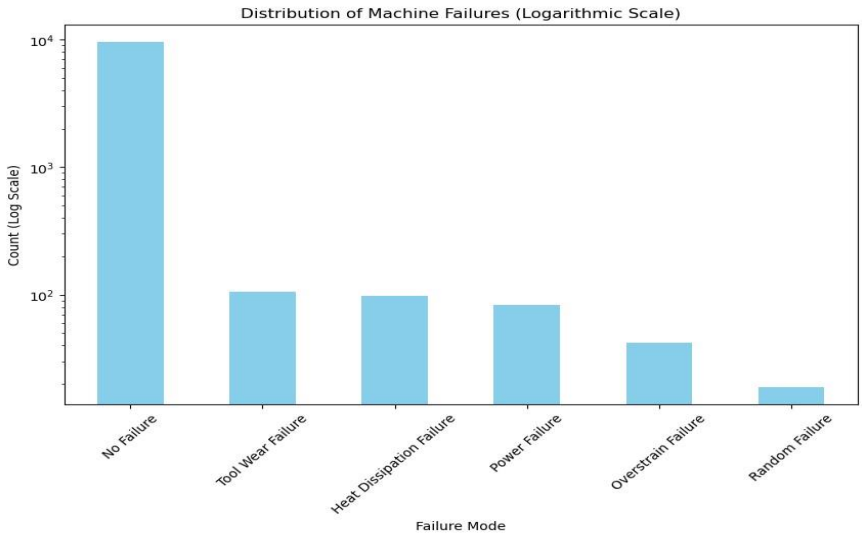
	count	mean	std	min	25%	50%	75%	max
Air temperature [K]	10000.0	2.995522e+02	2.014071e+00	295.000000	2.980000e+02	3.000000e+02	3.010000e+02	3.040000e+02
Process temperature [K]	10000.0	3.095520e+02	1.511597e+00	305.000000	3.080000e+02	3.100000e+02	3.110000e+02	3.130000e+02
Rotational speed [rpm]	10000.0	1.538776e+03	1.792841e+02	1168.000000	1.423000e+03	1.503000e+03	1.612000e+03	2.886000e+03
Torque [Nm]	10000.0	3.953340e+01	9.975662e+00	3.000000	3.300000e+01	4.000000e+01	4.600000e+01	7.600000e+01
Tool wear [min]	10000.0	1.079510e+02	6.365415e+01	0.000000	5.300000e+01	1.080000e+02	1.620000e+02	2.530000e+02
Machine failure	10000.0	9.900000e-02	5.619881e-01	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	5.000000e+00
Type_L	10000.0	6.000000e-01	4.899224e-01	0.000000	0.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
Type_M	10000.0	2.997000e-01	4.581494e-01	0.000000	0.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00
Power	10000.0	5.926834e+04	1.027410e+04	8658.000000	5.240400e+04	5.920000e+04	6.619250e+04	9.945000e+04
Power wear	10000.0	6.396723e+06	3.978889e+06	0.000000	3.056418e+06	6.197430e+06	9.399364e+06	2.067156e+07
Temperature difference	10000.0	9.999800e+00	1.078386e+00	7.000000	9.000000e+00	1.000000e+01	1.100000e+01	1.300000e+01
Temperature power	10000.0	1.744762e-04	4.148819e-05	0.000081	1.472624e-04	1.688582e-04	1.947894e-04	1.155001e-03

VII. DATA EXPLORATION:

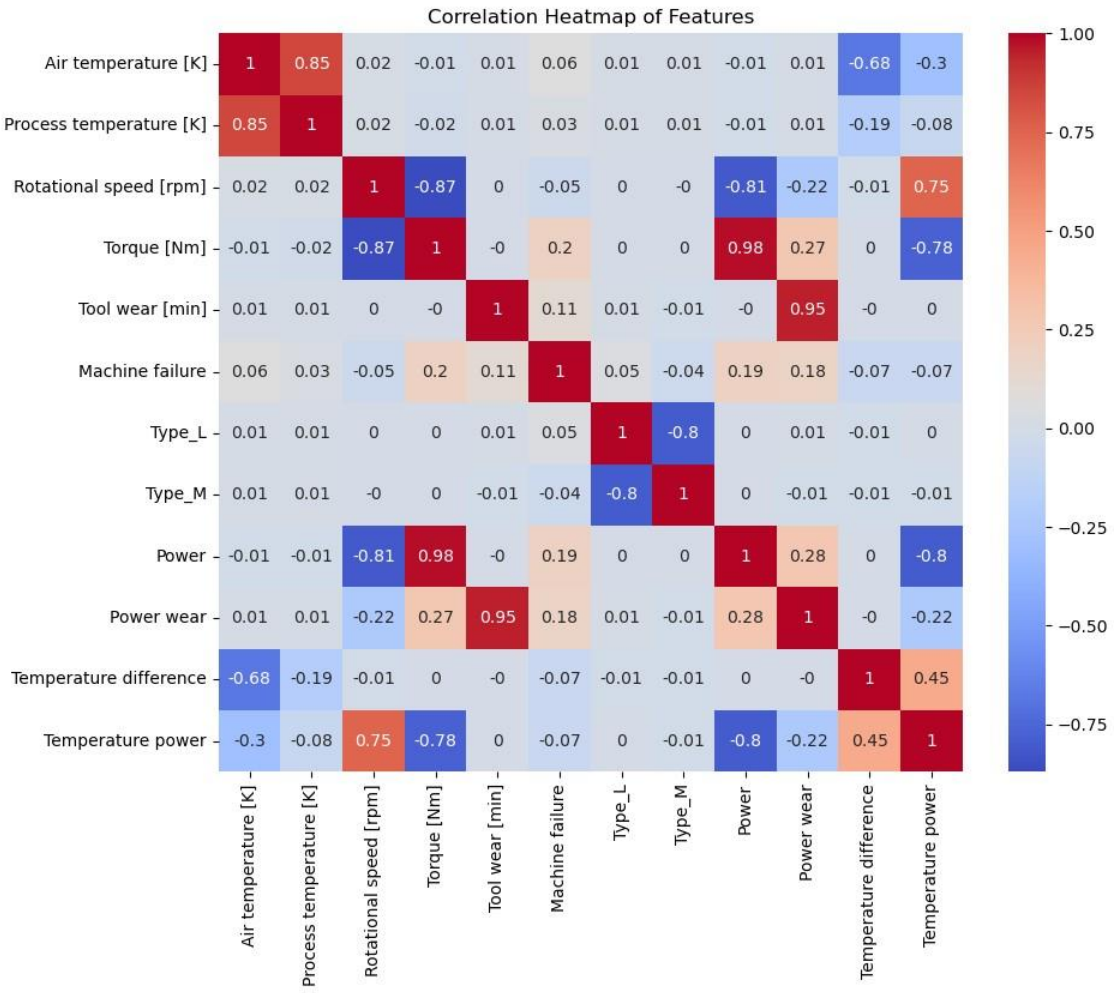
The logarithmic scale of distribution of machine failures below indicates that there is huge imbalance between non-failures cases and failures condition in the dataset.

The bars denoting the real modes of failure—tool wear failure, heat dissipation failure, power failure, overstrain failure, and random failure—are notably smaller than the count of "No

failures." The two that seem to occur most frequently among them are "tool wear failure" and "heat dissipation failure," with "power failure" and "overstrain failure" occurring less frequently. Out of all the failure modes, "random failure" has the lowest count, meaning it happens less frequently.



Correlation Heatmap of the features:

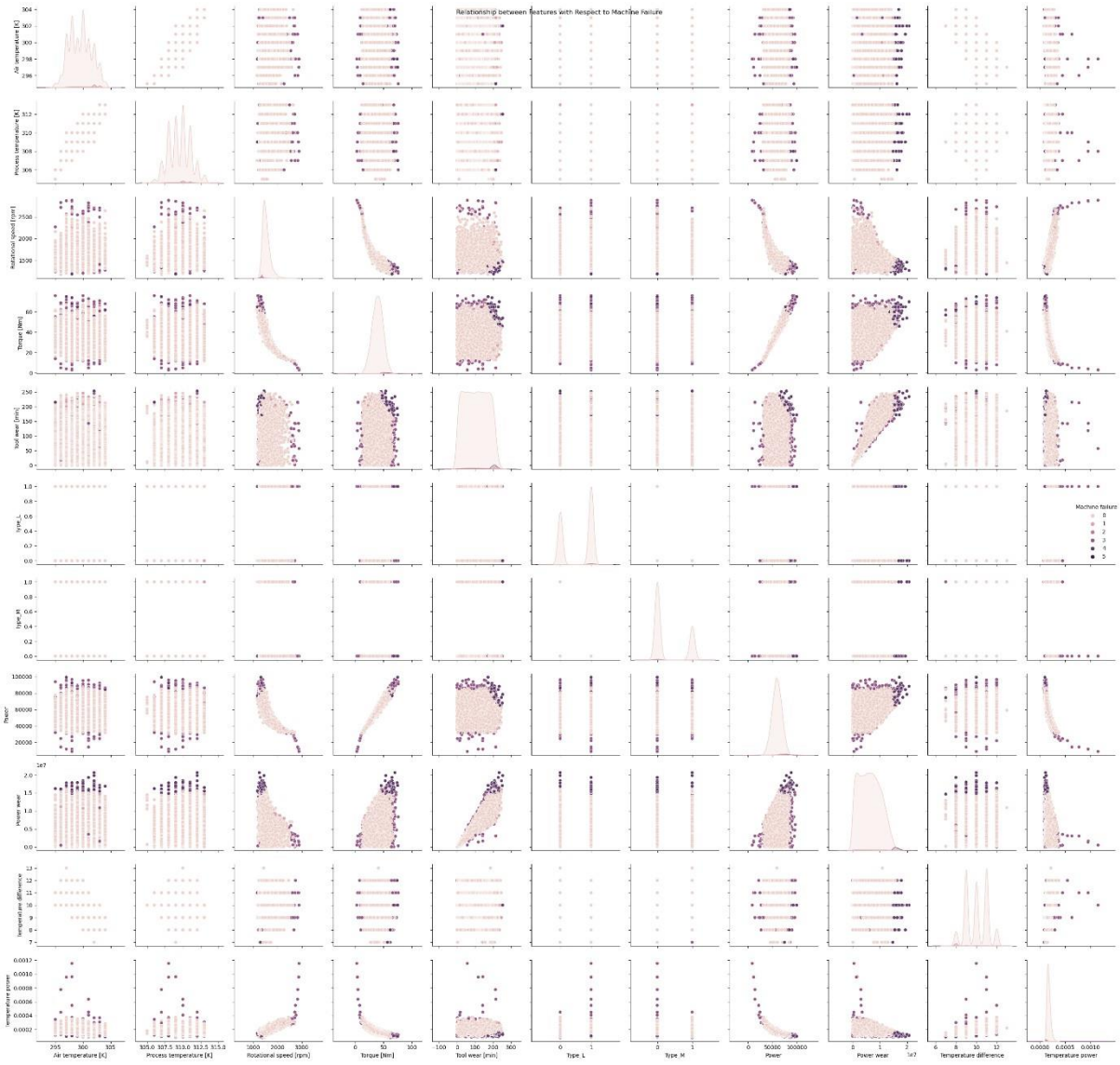


The following is an analysis of some significant relationships found on the heatmap:

- The strong positive correlation between **Air temperature [K]** and **Process temperature [K]** (0.85) indicates that the process temperature tends to rise along with the air temperature.
- **Torque [Nm]** and **Rotational speed [rpm]** have a strong negative correlation (-0.87), meaning that lower torque and greater rotational speeds are related.
- The negative correlation between **Power** and **Rotational speed [rpm]** is -0.81, suggesting that greater speeds may require less power because of possible machine design features.
- The extraordinarily strong positive association between **Torque [Nm]** and **Power** (0.98) indicates that power rises as torque does.

- **Tool wear [min]** and **Power wear** have a strong positive association (r = 0.95), showing that when tools deteriorate, power wear rises as well, signaling inefficiencies or higher energy use.
- The substantial negative correlation of -0.8 between the variables **Type_L** and **Type_M** may suggest that they are mutually exclusive conditions or types.
- **Temperature difference** has a moderately positive association and a high negative correlation with **Air temperature [K]** (-0.68) and **Power** (-0.78).

Pair Plot of Features with Respect to Machine Failure:

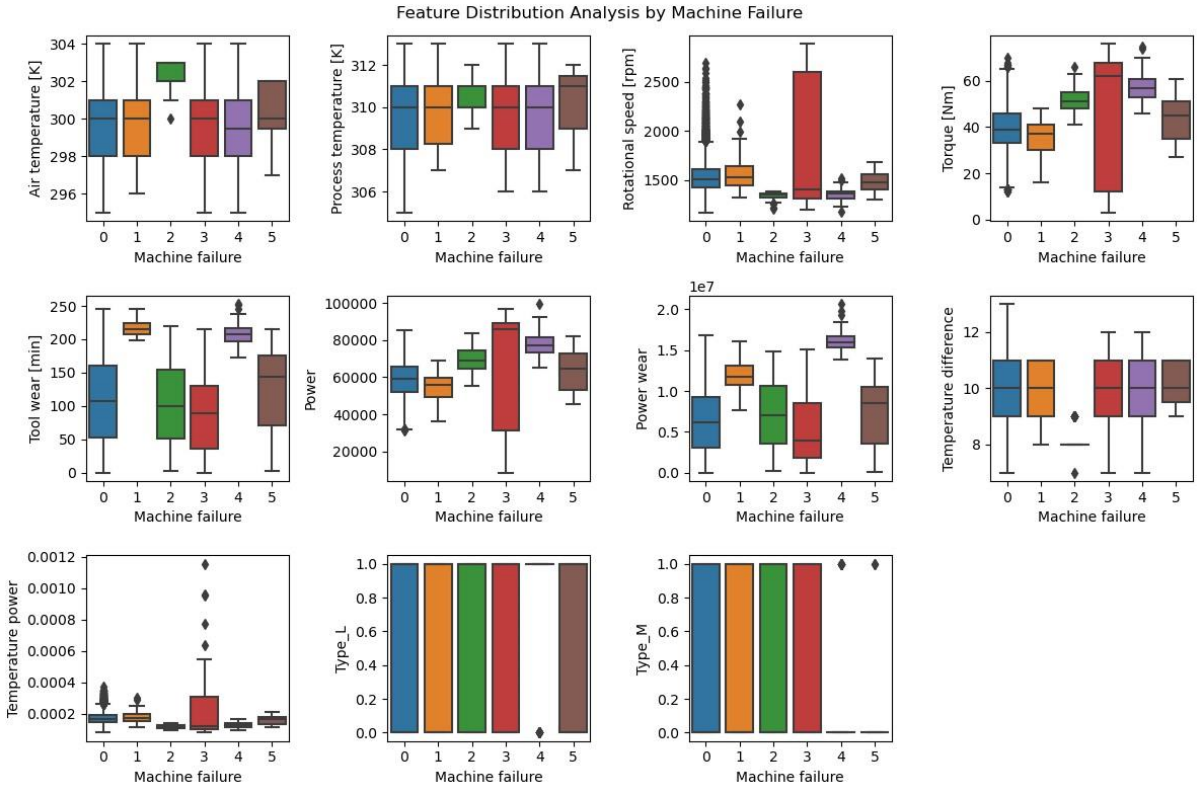


The pair plot shows the relationships between different variables in the dataset, along with their distributions with respect to machine failures:

- **Air temperature [K] and Process temperature [K]:** These two variables seem to have a strong positive correlation, indicated by the diagonal line in their joint distribution plot. This suggests that as one temperature increases, the other tends to increase as well.
- **Rotational speed [rpm] and Torque [Nm]:** There appears to be some relationship between rotational speed and torque, as indicated by the scatter plot. However, the relationship may not be linear, and further analysis would be needed to understand the nature of this relationship.

- **Power and Power wear:** There seems to be a positive correlation between power and power wear, suggesting that higher power usage might lead to more wear on the machine. Again, further analysis would be needed to confirm this relationship.
- **Temperature difference and Temperature power:** These variables might represent derived features or calculated values. The pair plot could show their relationships with other variables in the dataset, providing insights into their importance or relevance in the context of the problem domain.

Feature Distribution Comparison Based on Machine Failure:



The above boxplots explain about the distribution among all the features based on machine failures showing the median and outliers. The following are some of the key observations noted in the box plot analysis:

- **Air temperature [K]:** With minor fluctuations in the median and spread, it is constant across all machine failure types.

- **Process temperature [K]:** Variability is somewhat more than air temperature, especially for categories 2 and 3, whose medians are higher.
- **Rotational Speed [rpm]:** This feature varies significantly, particularly for category 3, which has a significantly larger median and outliers. This suggests that when this kind of failure happens, rotational speed is highly variable.
- **Torque [Nm]:** Like rotational speed, torque varies among several machine failure categories (Nm). Once more, Category 3 is notable due to its greater median torque.
- **Tool wear [min]:** There is a large variation in the tool wear times across all categories, but category 3 has a significantly higher median value. This could indicate that the tools in this category have been in use for longer lengths of time before they fail.
- **Power:** A wide range is observed here, especially in categories 3 and 4, which may suggest that these kinds of failures entail considerable variations in electrical current.
- **Power wear:** There is variation in the median values and ranges; category 3 once more exhibits a higher median and more variability.
- **Temperature difference:** There are differences in the spread and median; category 1 displays the lowest median temperature difference, while category 3 displays the largest.
- **Temperature power:** This plot is distinct from others since it deviates from the conventional boxplot structure, resembling a feature with discrete, category values.
- **Type_L:** There is a similarity in the Type_L feature across all categories with an exception for category 4 which has a has a significantly lower median value.
- **Type_M:** Like Type_L, the Type_M feature shows a similarity across all categories, except for category 4 and 5 which have a significantly lower a showing some similarities.

VIII. STANDARDIZATION:

We are employing PCA, LDA, and feature selection techniques to preprocess our data before training models aiming to enhance model performance and interpretability, those processes were used as explained below.

Standardization is the method in which features are ensured that they have a mean of 0 and a standard deviation of 1, making them suitable for algorithms that assume normally distributed data or require standardized features for optimal performance. This is crucial because features often have vastly different ranges of values, which can complicate the model's ability to identify relationships among them.

By centering the distribution around 0, standardization reduces the impact of outliers, which can disproportionately distort the data distribution. This guarantees that the model's performance is duly impacted by extreme values and aids in preserving the data's integrity. The coefficients of features become directly comparable when they are standardized to the same scale. This makes it possible to more easily interpret the model's performance based on the size of the coefficients, which is especially helpful for determining the relative importance of various model features. The code snippet below shows the step we used that involves the separation of predictor and target to X and Y variables then applying standardization to predictors which is X_scaled.

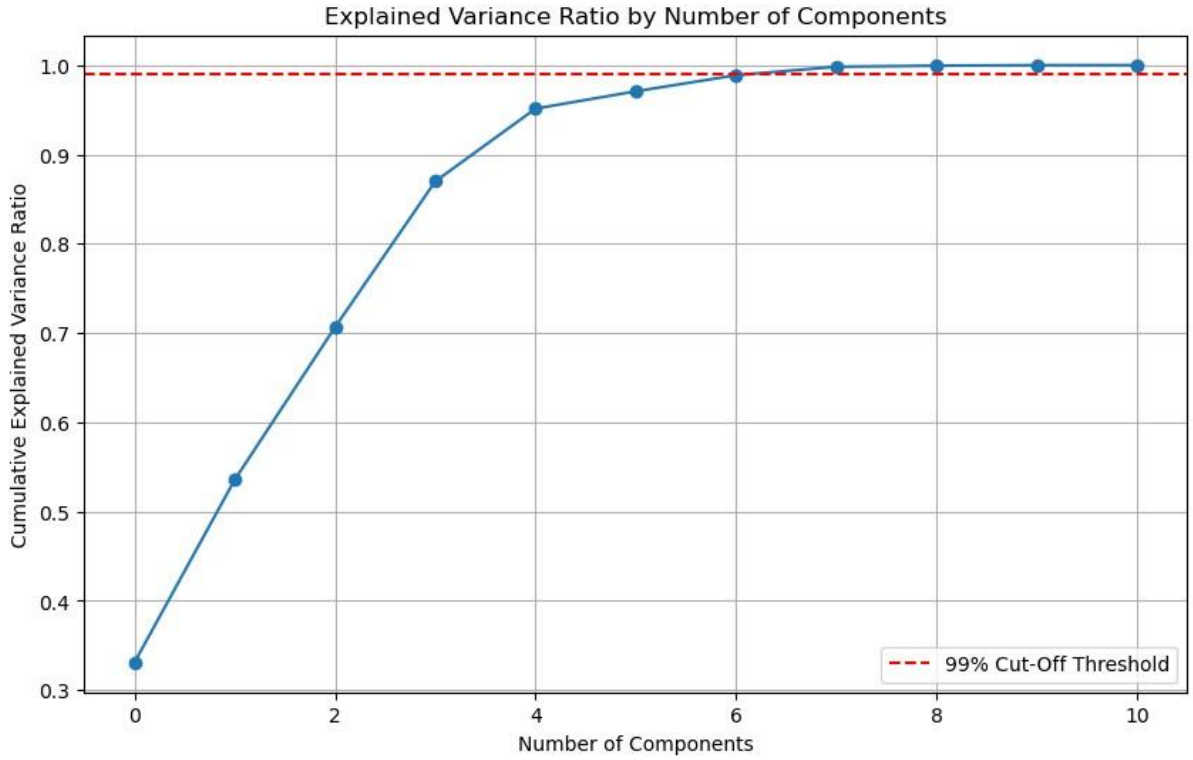
```
# Separating features and target variable
X = df.drop('Machine failure', axis=1)
y = df['Machine failure']

# Standardizing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

IX. DIMENSIONALITY REDUCTION:

Principal Component Analysis

We are using PCA as part of the dimensionality reduction process, as Principal Component Analysis reduces the number of dimensions in datasets to principal components that retains maximum original information, it is done by transforming correlated variables into smaller set of variables known as principal components. We have applied PCA on predictor dataset X_scaled. And we got 11 principal components selected by PCA. Then we have plotted the variance ratio plot from the analysis which is as below.



Based on our analysis of the cumulative explained variance ratio plot, we have determined that 99% of the variance in the data can be explained by retaining only the first 6 principal components. Consequently, as part of the dimensionality reduction process, we have opted to retain solely these 6 principal components. This decision lets us capture most of the variance within the data while significantly reducing its dimensionality, which can offer computational benefits, enhance model performance, and facilitate clearer data interpretation.

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a statistical technique used for dimensionality reduction and classification. It works by finding a linear combination of features that best separates different classes of data. LDA projects data onto a lower-dimensional space to maximize class separability, focusing on the variance between class means. We have used LDA also as one of the dimensionality reduction methods before training our models.

Feature Selection

We are implementing machine learning models using exclusively the features that have been engineered through our feature engineering process as part of feature selection. These features, namely `Power`, `Power wear`, `Temperature difference`, and `Temperature power`, have been

derived to capture relevant information from the dataset. Each of these features plays a critical role in understanding and predicting the various failure modes observed in the manufacturing process.

Specifically, the engineered features encapsulate distinct failure modes such as tool wear failure (`TWF`), heat dissipation failure (`HDF`), power failure (`PWF`), overstrain failure (`OSF`), and random failures (`RNF`). For instance, `TWF` occurs when the tool reaches a randomly selected wear time between 200 and 240 minutes, resulting in either replacement or failure. `HDF` occurs when the temperature difference between air and process falls below 8.6 K, coupled with a rotational speed below 1380 rpm. `PWF` manifests when the power required for the process, derived from the product of torque and rotational speed, falls outside the range of 3500 W to 9000 W. `OSF`, on the other hand, is triggered when the product of tool wear and torque exceeds specific thresholds for different product variants. Additionally, a small percentage of random failures (`RNF`) occur independent of process parameters.

X. DATA SPLITTING:

Data Splitting

The dataset will be split into train and test sets by `80:20` ratio

```
# Splitting the dataset into train and test sets (80% train, 20% test) for PCA
X_train_pca, X_test_pca, y_train_pca, y_test_pca = train_test_split(X_pca, y, test_size=0.2, random_state=42)
```

```
# Splitting the dataset into train and test sets (80% train, 20% test) for LDA
X_train_lda, X_test_lda, y_train_lda, y_test_lda = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Splitting the dataset into train and test sets (80% train, 20% test) for Feature Selection
X_train_fs, X_test_fs, y_train_fs, y_test_fs = train_test_split(X_fs, y, test_size=0.2, random_state=42)
```

The code snippet outlines the process of dataset splitting, employing an 80:20 train-test split ratio. This ratio ensures that 80% of the data is allocated for training purposes, while the remaining 20% is reserved for testing of three different techniques: Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Feature Selection.

In the first section, the dataset is divided into training and testing sets for PCA technique. The training and testing subsets are denoted as (X_train_pca, y_train_pca, X_test_pca, y_test_pca), with a test size of 20% and a fixed random state for reproducibility.

The subsequent section focuses on splitting the dataset for LDA technique. Like PCA, the dataset is split into training and testing subsets (X_train_lda, y_train_lda, X_test_lda, y_test_lda) with consistent parameters for test size and random state.

Finally, the third block addresses dataset splitting for feature selection, a process aimed at identifying a subset of relevant features for model building. The training and testing sets (X_train_fs, y_train_fs, X_test_fs, y_test_fs) are defined with identical splitting parameters to maintain consistency across techniques.

XI. DATA IMBALANCE HANDLING:

Data Imbalance Handling

Performing oversampling only on the training data to avoid data leakage

In [30]:
```
ros = RandomOverSampler(random_state=42)
```

Data Imbalance Handling for Principal Component Analysis (PCA)

In [31]:
```
# Applying RandomOverSampler to address data imbalance only on the training data of PCA
X_train_resampled_pca, y_train_resampled_pca = ros.fit_resample(X_train_pca, y_train_pca)

# Checking the shape of the resampled training data of PCA
print("Shape of X_train_resampled_pca:", X_train_resampled_pca.shape)
print("Shape of y_train_resampled_pca:", y_train_resampled_pca.shape)
```

```
Shape of X_train_resampled_pca: (46302, 6)
Shape of y_train_resampled_pca: (46302,)
```

Data Imbalance Handling for Linear Discriminant Analysis (LDA)

In [32]:
```
# Applying RandomOverSampler to address data imbalance only on the training data of LDA
X_train_resampled_lda, y_train_resampled_lda = ros.fit_resample(X_train_lda, y_train_lda)

# Checking the shape of the resampled training data of LDA
print("Shape of X_train_resampled_lda:", X_train_resampled_lda.shape)
print("Shape of y_train_resampled_lda:", y_train_resampled_lda.shape)
```

```
Shape of X_train_resampled_lda: (46302, 11)
Shape of y_train_resampled_lda: (46302,)
```

Data Imbalance Handling for Feature Selection

In [33]:
```
# Applying RandomOverSampler to address data imbalance only on the training data of LDA
ros = RandomOverSampler(random_state=42)
X_train_resampled_fs, y_train_resampled_fs = ros.fit_resample(X_train_fs, y_train_fs)

# Checking the shape of the resampled training data of LDA
print("Shape of X_train_resampled_fs:", X_train_resampled_fs.shape)
print("Shape of y_train_resampled_fs:", y_train_resampled_fs.shape)
```

```
Shape of X_train_resampled_fs: (46302, 4)
Shape of y_train_resampled_fs: (46302,)
```

In addressing the issue of class imbalance within our training datasets, particularly the overrepresentation of the '**No failure**' class (**class 0**), we employed a RandomOverSampler approach. This technique involved replicating existing samples from the underrepresented classes to create new instances, thereby balancing the distribution of classes. By adopting this strategy, we aimed to mitigate potential biases in our predictive models and enhance their generalizability when confronted with new data.

Through the application of RandomOverSampler, we augmented the minority class instances within our training data, resulting in a balanced dataset comprising 46,302 samples across six characteristics specifically for the Principal Component Analysis (PCA) dataset. Furthermore, we extended this oversampling technique to our training datasets for both linear discriminant analysis (LDA) and feature selection algorithms ensuring consistency across techniques.

It is imperative to note that all oversampling procedures were confined solely to the training data to prevent any data leaks and uphold the integrity of our test data. By implementing these measures, our objective was to improve the accuracy of our models and enhance their ability to predict instances of minority classes, a crucial skill in datasets characterized by imbalances.

XII. MODEL IMPLEMENTATION:

In our study, we have implemented a total of four distinct machine learning models: logistic regression, random forest, decision trees, and xgboost. Each model serves as a unique approach to analysing and predicting outcomes within our dataset. To ensure a comprehensive analysis, we have integrated various dimensionality reduction and feature selection techniques, including Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and feature selection of engineered features.

To further refine the hyperparameter tuning process and optimize model performance, we utilized GridSearchCV with the F1 weighted score as the metric for improvement. The F1 weighted score is particularly well-suited for imbalanced classification tasks, as it provides a balanced assessment of precision and recall across different class labels by considering both the number of true positives and the respective class frequencies. Through GridSearchCV, we aim to fine-tune each model's hyperparameters to achieve the best possible performance, thereby enhancing predictive accuracy and minimizing overfitting

By incorporating the F1 weighted score as the optimization metric within GridSearchCV, we aimed to identify hyperparameter configurations that maximize the overall predictive performance of each model while accounting for class imbalances. This approach ensures that the selected hyperparameters yield models capable of achieving high precision and recall

across all classes, thereby enhancing the robustness and generalizability of our predictive models.

Logistic Regression:

Logistic regression is a statistical method used in machine learning for binary classification problems. It estimates the probability of an input point belonging to a certain class using a link function (logit function). Logistic regression can be extended to handle multi-class classification problems using multiple binary classifiers or a multinomial logistic regression approach. Performance metrics like accuracy, precision, recall, and F1-score provide insights into the model's performance and areas for improvement. The model's performance can be evaluated using metrics such as accuracy, precision, recall, and F1-scores.

Logistic Regression with Principal Component Analysis (PCA):

Summary of Class-Wise Performance:

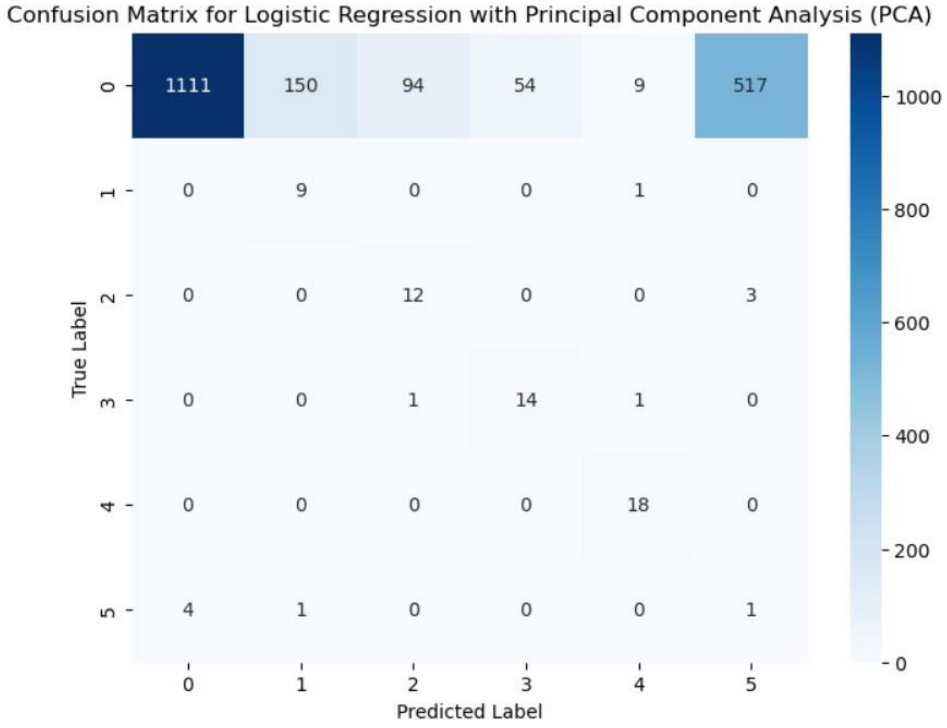
- **Class 0:** The model achieves perfect precision, indicating all predicted instances are correct. However, its recall is moderate, suggesting it fails to identify all actual instances of this class.
- **Class 1:** While exhibiting high recall, this class suffers from weak precision, indicating numerous false positives.
- **Class 2:** Deficient performance is observed in both precision and recall, resulting in an exceptionally low F1-score.
- **Class 3:** Moderately low precision is accompanied by high recall for this class.
- **Class 4:** This class demonstrates the best performance with high precision and perfect recall, leading to a high F1-score. However, it is worth noting that this evaluation is based on a small sample size of only 18 instances.
- **Class 5:** The model shows zero precision and extremely low recall for this class, indicating significant underperformance.

Overall Model Performance Metrics:

- **Accuracy:** The model achieves an accuracy of 0.58, correctly predicting 58% of all classes.

- **Macro Average:** Macro average precision is low at 0.33, indicating a high rate of false positives across classes. However, macro average recall is relatively high at 0.72, suggesting the model's ability to identify positive instances. The macro average F1score stands at 0.36, reflecting the balance between precision and recall across all classes, though skewed towards recall.
- **Weighted Average:** The weighted average precision is notably high at 0.97, primarily due to accounting for class imbalance. However, the weighted average recall and F1score align closely with accuracy, as they are aggregate metrics for all classes.

In summary, while the model demonstrates strengths in identifying positive instances, particularly for Class 4, it also exhibits significant weaknesses, such as high false positives and poor performance in certain classes. The evaluation underscores the importance of achieving a better balance between precision and recall for improved model performance.



Logistic Regression with Linear Discriminant Analysis (LDA):

Summary of Class-Wise Performance:

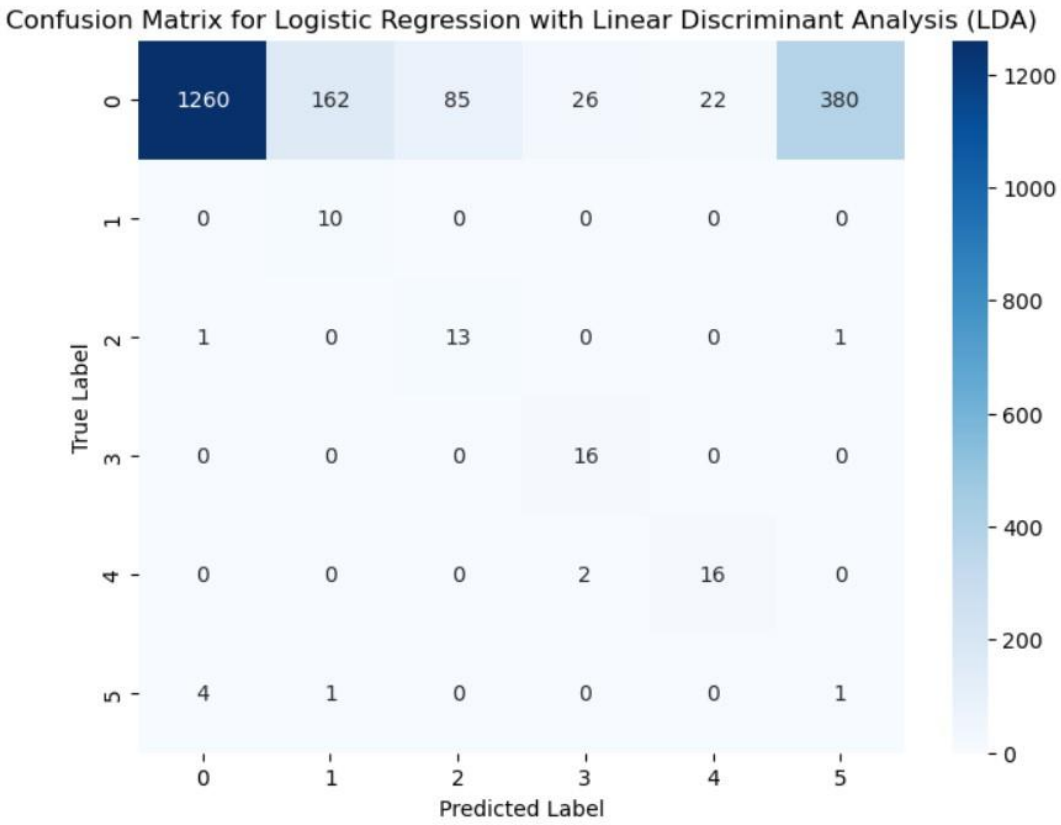
- **Class 0:** Achieves perfect precision but misses some true positives, maintaining a good balance between precision and recall.
- **Class 1:** Shows low precision with perfect recall, indicating numerous false positives and resulting in a low F1-score.

- **Class 2:** Exhibits low precision and high recall, with an imbalance between precision and recall reflected in a low F1-score.
- **Class 3:** Demonstrates moderate precision alongside perfect recall, leading to a higher F1-score compared to other classes.
- **Class 4:** Displays better precision and high recall, resulting in a moderate F1-score.
- **Class 5:** Indicates incredibly low precision and recall, leading to a dismal F1-score, signifying poor performance.

Overall Model Performance Metrics:

- **Accuracy:** The model achieves an overall accuracy of 0.66, correctly predicting 66% of instances in the test dataset.
- **Macro Average:** Macro average precision (0.33) and recall (0.76) highlight the model's struggle with precision across all classes but effectiveness in identifying true positives. The F1-score (0.37) reflects the balance between precision and recall, albeit skewed towards recall.
- **Weighted Average:** The weighted average precision is high at 0.97, considering class support. The weighted average recall and F1-score align with accuracy, indicating the aggregation of metrics across all instances.

In summary, the model demonstrates varied performance across different classes, with notable strengths in certain classes but significant weaknesses in others. The evaluation underscores the importance of achieving a better balance between precision and recall for improved overall model performance.



Logistic Regression with Feature Selection:

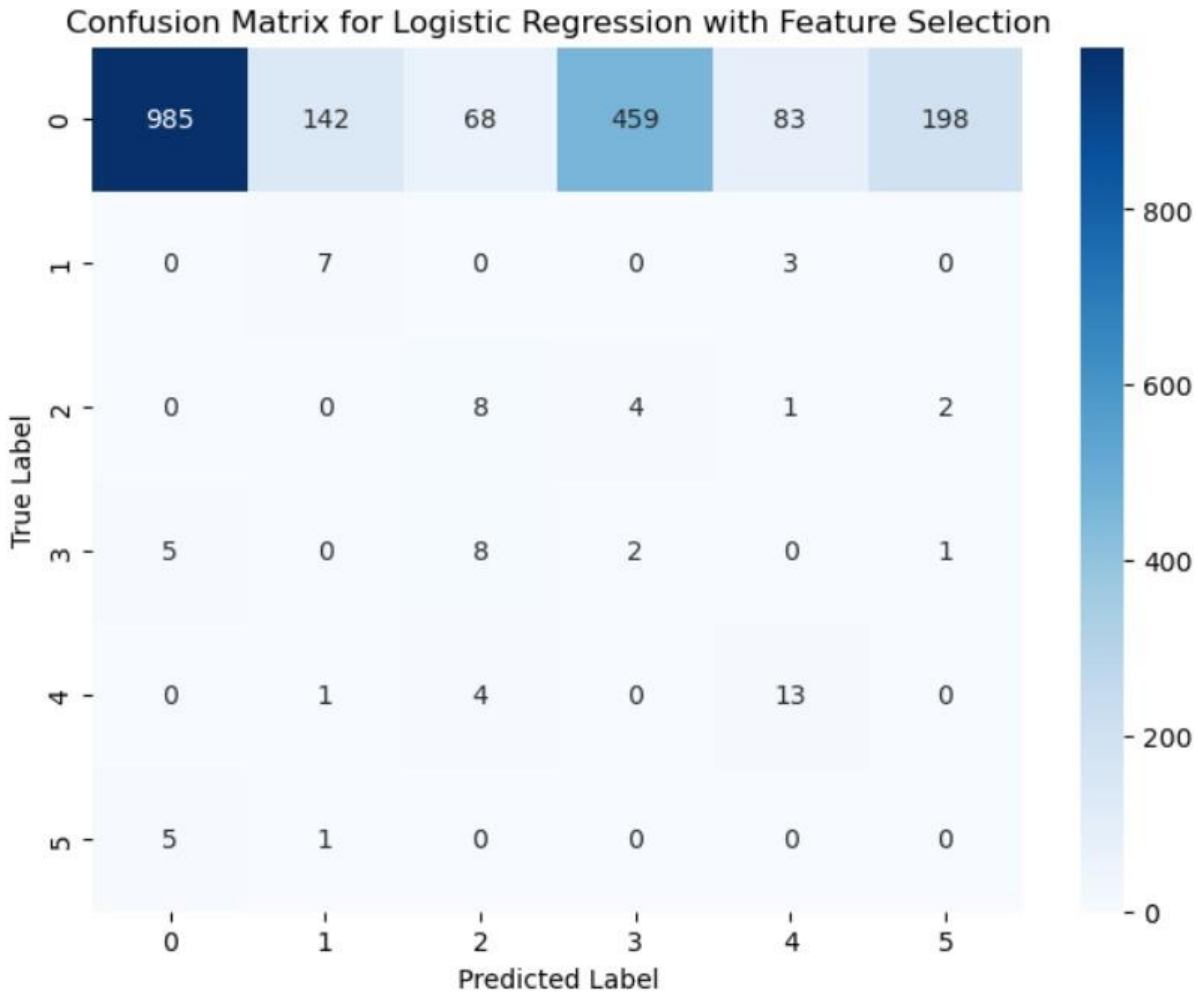
Summary of Class-Wise Performance:

- **Class 0:** Achieves nearly perfect precision, indicating high accuracy in predictions for class 0, but struggles with recall, missing about half of the actual instances. The balance between precision and recall is moderate.
- **Class 1:** Shows moderate precision and higher recall, indicating the model's capability to identify class 1 instances, though it also misclassifies many other instances. The balance between precision and recall is low.
- **Class 2:** Similar to class 1, with low precision and moderate recall, reflecting an imbalance between precision and recall.
- **Class 3:** Exhibits extremely low precision, failing to predict any instances correctly for class 3, and low recall, indicating a failure to capture most actual instances.
- **Class 4:** Demonstrates low precision but higher recall, suggesting the model's ability to identify class 4 instances, albeit with many false positives.
- **Class 5:** Fails to correctly identify any instances of class 5, with both precision and recall being 0.00.

Overall Model Performance Metrics:

- **Accuracy:** The model achieves an overall accuracy of 0.51, correctly predicting 51% of all cases.
- **Macro Average:** With a low macro average precision (0.21), the model shows poor performance across all classes, irrespective of class imbalance. The macro average recall (0.43) and F1-score (0.19) also indicate an ineffective balance between precision and recall across the classes.
- **Weighted Average:** The high weighted average precision (0.96) is influenced by the model's strong performance on the majority class (0), which has the most instances. However, this skews the metric. The weighted average recall and F1-score align with the overall accuracy (0.51 and 0.65, respectively), reflecting the model's performance across all instances.

In summary, the model exhibits imbalanced precision and recall across classes, achieving moderate accuracy overall with notable discrepancies in macro average precision and recall, indicating the need for improved balance in classification performance.



Random Forest:

A popular machine learning approach for classification problems is Random Forest. To get a result, it builds several decision trees during training and then aggregates the predictions made by each tree. In contrast to individual decision trees, Random Forest promotes model robustness and generalisation by training each tree on a random selection of characteristics and data, which reduces overfitting.

Random Forest with Principal Component Analysis (PCA):

Summary of Class-Wise Performance:

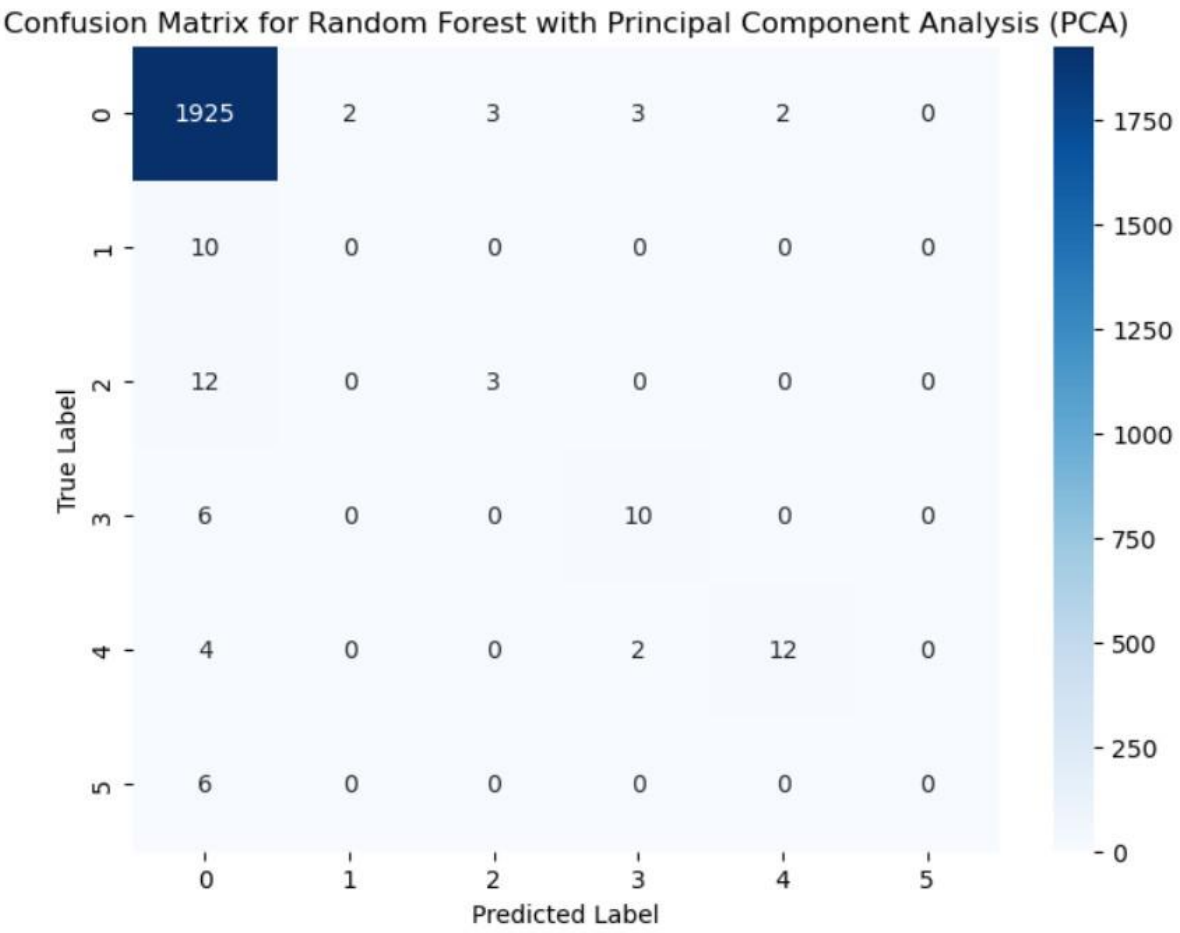
- **Class 0:** Demonstrates excellent precision and recall, indicating high accuracy in predicting class 0 instances.
- **Class 1:** Shows a complete failure in correctly identifying any instances of class 1, raising concerns about the representation or distinguishability of class 1 features in the dataset or after PCA transformation.

- **Class 2:** Exhibits moderate precision but low recall, resulting in a relatively low F1score, suggesting limited capability in identifying class 2 instances.
- **Class 3:** Displays higher precision and recall compared to class 2, resulting in a moderate F1-score, indicating better performance in identifying class 3 instances.
- **Class 4:** Achieves high precision and reasonable recall, leading to a strong F1-score, indicating effective identification of class 4 instances.
- **Class 5:** Similar to class 1, the model fails to correctly identify any instances of class 5.

Overall Model Performance Metrics:

- **Accuracy:** The model achieves a high overall accuracy of 0.97, correctly predicting 97% of all cases. However, this might be misleading due to class imbalance, as evidenced by the low recall for several classes.
- **Macro Average:** The average precision, recall, and F1-score across all classes are relatively low, indicating inconsistent performance across classes.
- **Weighted Average:** The weighted average precision and recall are both high at 0.97, influenced by the robust performance on the majority class (class 0), which skews these metrics. However, the weighted average F1-score reflects the model's performance across all instances.

In summary, the model demonstrates varying performance across classes, achieving high accuracy overall but facing challenges with class imbalance, particularly in identifying instances of classes 1 and 5.



Random Forest with Linear Discriminant Analysis (LDA):

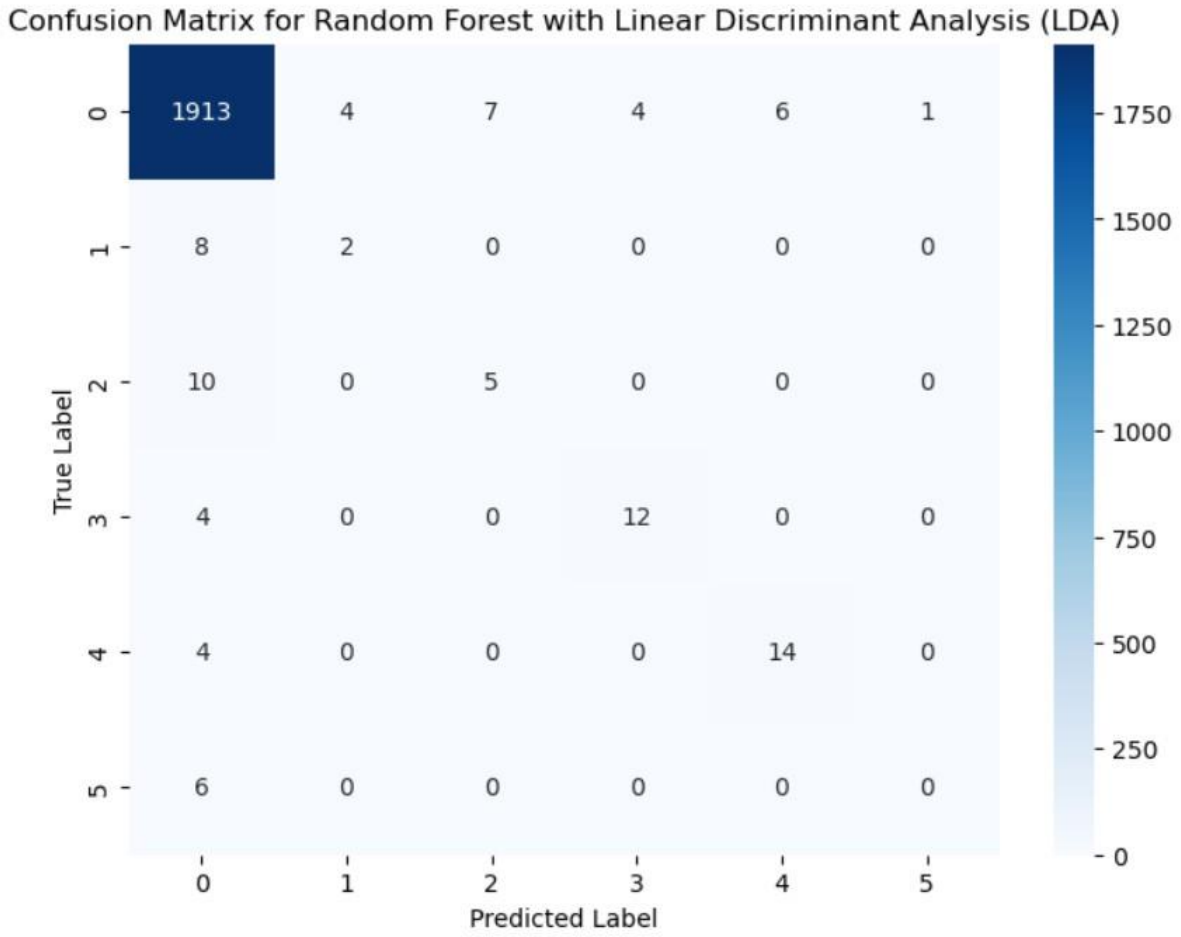
Summary of Class-Wise Performance:

- **Class 0:** Demonstrates excellent precision, recall, and F1-score, indicating exceptional performance for this class.
- **Class 1:** Shows lower precision, recall, and F1-score, suggesting the model struggles to correctly identify and predict this class.
- **Class 2:** Exhibits moderate precision but low recall and F1-score, indicating difficulty in handling this class.
- **Class 3:** Displays high precision, recall, and F1-score, suggesting effective performance for this class.
- **Class 4:** Demonstrates good precision, recall, and F1-score, indicating the model's effectiveness in predicting this class.
- **Class 5:** Completely fails to identify this class, with both precision and recall being 0.00.

Overall Model Performance Metrics:

- **Accuracy:** High overall accuracy of 0.97, influenced by the high support for class 0, which masks deficiencies in other classes.
- **Macro Average:** Reflects unbalanced performance across classes, with precision, recall, and F1-score indicating variability in model performance.
- **Weighted Average:** Reflects the influence of class 0 in the overall metrics due to its large support, with precision, recall, and F1-score being 0.97 when weighted by class support.

In summary, the model demonstrates robust performance for most classes, achieving high accuracy overall, but struggles with identifying certain classes, notably class 1 and class 5.



Random Forest with Feature Selection:

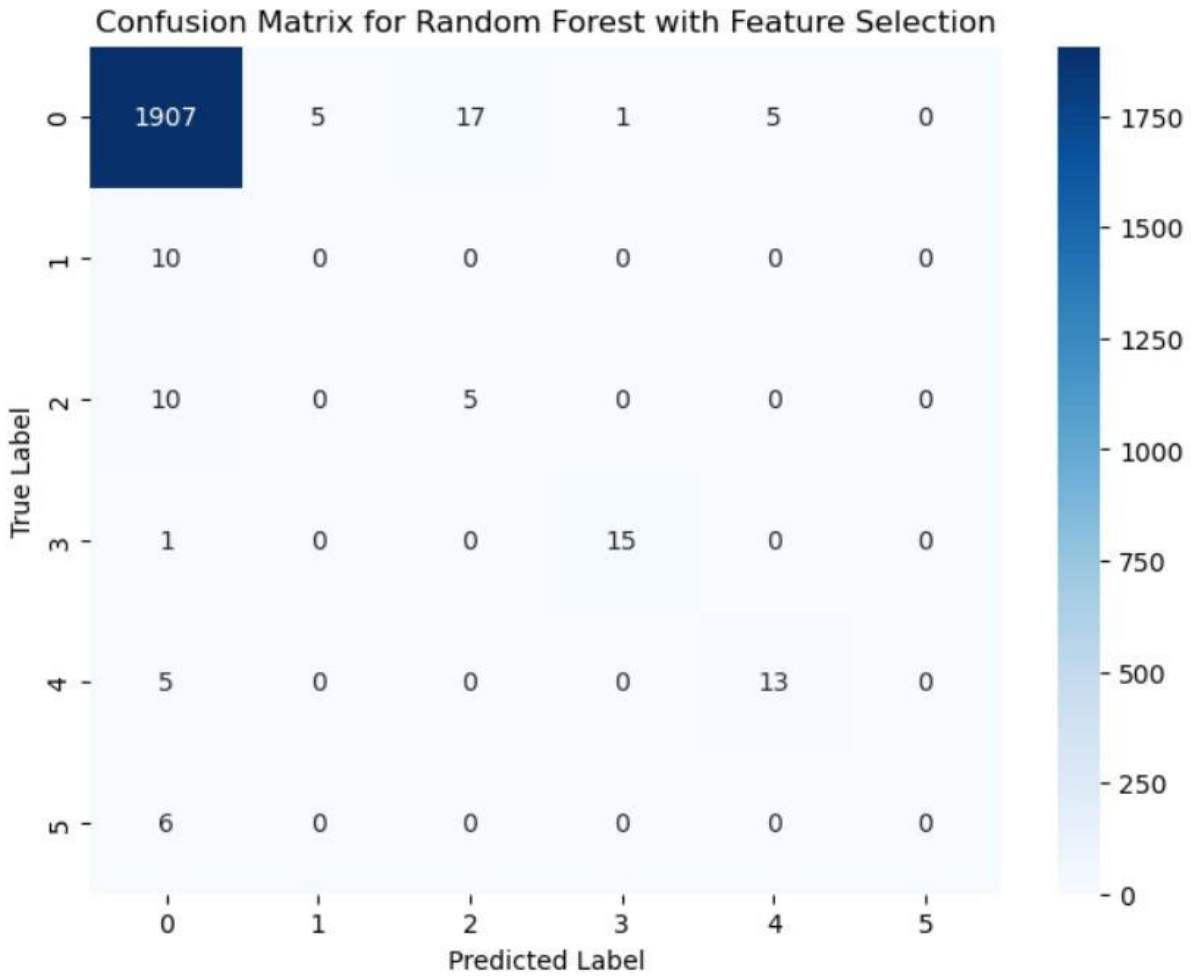
Summary of Class-Wise Performance:

- **Class 0:** Exhibits perfect precision and recall, indicating excellent performance.
- **Class 1:** Fails to predict any instance correctly, with zero scores for precision, recall, and F1-score.
- **Class 2:** Shows low precision and recall, resulting in a low F1-score.
- **Class 3:** Demonstrates high precision and recall, leading to a high F1-score.
- **Class 4:** Displays moderate precision and recall, resulting in a moderate F1-score.
- **Class 5:** Like class 1, the model fails to identify any instance of class 5.

Overall Model Performance Metrics:

- **Accuracy:** The model achieves a high overall accuracy of 97%, but its effectiveness might be misleading due to potential class imbalance.
- **Macro Average:** Precision, recall, and F1-score are all below 0.50, indicating inconsistent performance across classes.
- **Weighted Average:** All metrics are 0.97, reflecting the influence of the majority class (class 0) on the overall performance.

In summary, the model demonstrates strong performance in predicting certain classes, such as class 0 and class 3, while struggling with classes 1 and 5. Despite the high overall accuracy, macro average metrics suggest that the model's performance is not consistent across all classes, potentially due to class imbalance.



Decision Tree:

Classification problems are performed with decision trees, which are flexible and easily interpreted machine learning models. They produce a tree-like structure of decision nodes and leaf nodes by recursively dividing the feature space into smaller subsets according to the values of the features. To maximise the purity or homogeneity of the generated subsets, the algorithm chooses the feature at each decision node that splits the data the best. Decision trees provide insights into the decision-making process since they are simple to read. They can, however, overfit, particularly when working with intricate datasets. Pruning and restricting the depth of the tree are two strategies that reduce overfitting and enhance generalisation.

Decision Tree with Principal Component Analysis (PCA):

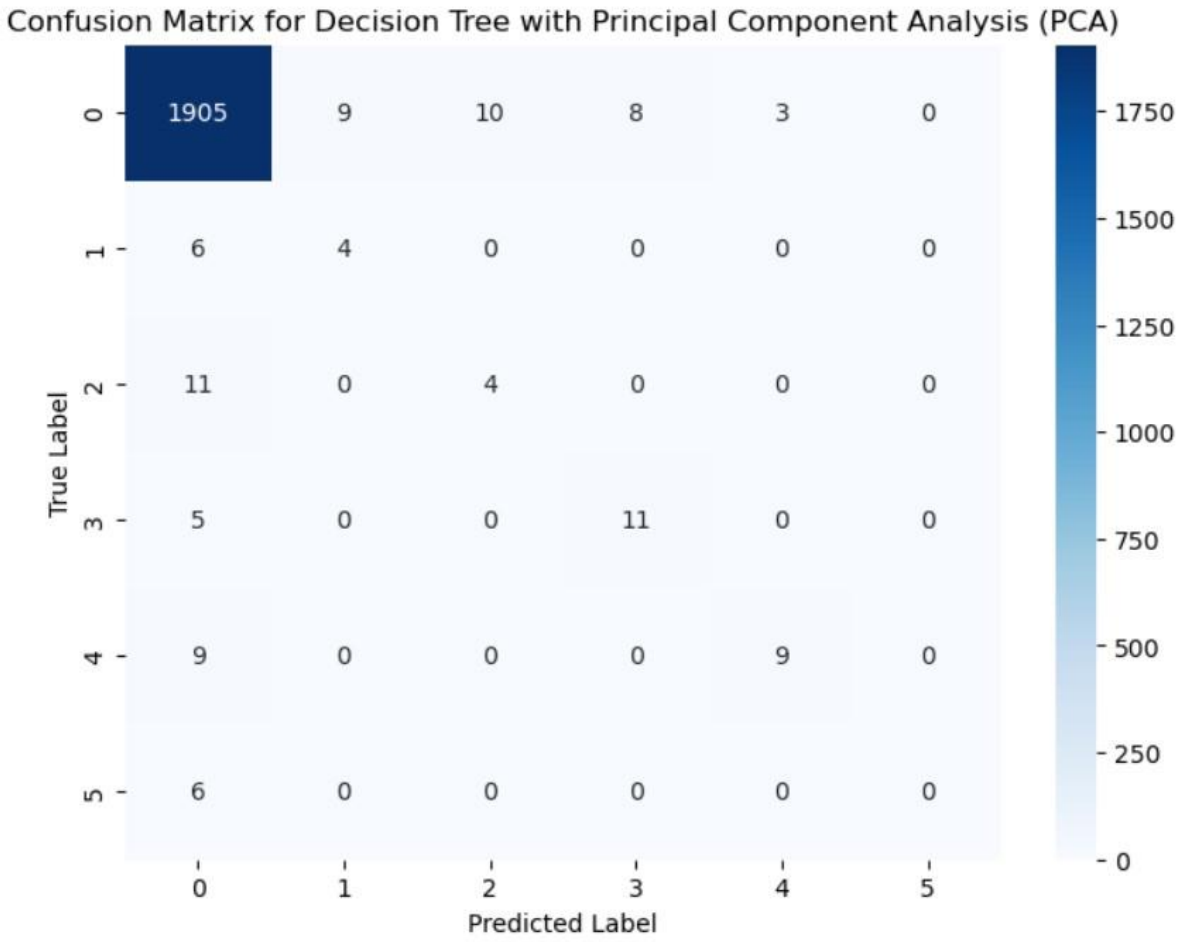
Summary of Class-Wise Performance:

- **Class 0:** The model shows remarkable precision and recall, leading to an excellent F1score, indicating perfect identification of instances for this class.
- **Class 1:** Precision, recall, and F1-score are low, suggesting the model struggles to correctly identify this class, with high false positives and false negatives.
- **Class 2:** Similar to class 1, the model demonstrates inferior performance with low precision, recall, and F1-score.
- **Class 3:** The model performs moderately with better precision and recall compared to classes 1 and 2, resulting in a fair F1-score.
- **Class 4:** Exhibits the highest precision and recall among the minority classes, with a moderate F1-score indicating a moderate level of performance.
- **Class 5:** The model fails to correctly predict any instances of class 5, as shown by zero precision and recall, resulting in an F1-score of zero.

Overall Model Performance Metrics:

- **Accuracy:** High accuracy of 0.97 suggests a considerable proportion of correct predictions by the model. However, it may be misleading in the presence of class imbalance.
- **Macro Average:** Reflects average performance across all classes without considering support. With precision, recall, and F1-score below 0.50, it indicates underperformance for several classes.
- **Weighted Average:** Accounts for class imbalance by weighting each class's performance by its support. Skewed by high performance on class 0, the weighted metrics are 0.96 for precision and 0.97 for both recall and F1-score.

In summary, the model performs exceptionally well for class 0 but struggles with other classes, notably class 1 and class 5. Despite high overall accuracy, macro average metrics suggest underperformance for several classes, highlighting potential issues with class imbalance.



Decision Tree with Linear Discriminant Analysis (LDA):

Summary of Class-Wise Performance:

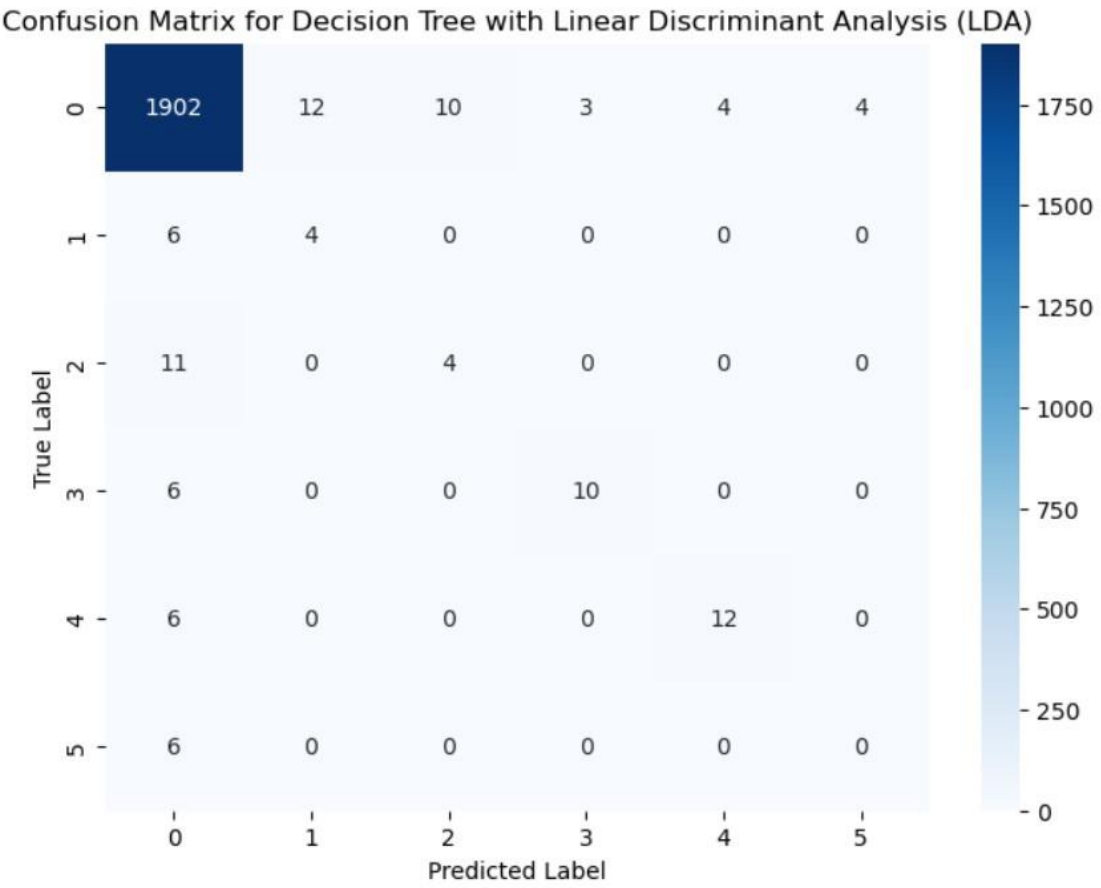
- **Class 0:** Exhibits almost perfect precision and recall, leading to an exceedingly high F1-score, indicating excellent identification of instances for this class.
- **Class 1:** Shows low precision and moderate recall, resulting in a low F1-score, suggesting the presence of false positives in addition to correctly identified instances of class 1.
- **Class 2:** Demonstrates similarly low precision and recall, with a low F1-score, indicating mediocre performance in identifying class 2 instances.
- **Class 3:** Performs better with good precision and fair recall, resulting in a higher F1score, indicating a good balance between avoiding false positives and correctly identifying true positives for class 3.

- **Class 4:** Displays high precision and good recall, reflected in a solid F1-score, suggesting effective prediction of class 4.
- **Class 5:** Both precision and recall are 0, indicating failure to identify any true instances of class 5, resulting in a poor F1-score.

Overall Model Performance Metrics:

- **Accuracy:** The overall accuracy is extremely high at 0.97, suggesting strong performance. However, lower performance metrics for several classes indicate uneven distribution of accuracy across different classes.
- **Macro Average:** Provides an unweighted average of precision, recall, and F1-score across all classes, resulting in scores around 0.50, indicating moderate average performance when each class is treated equally.
- **Weighted Average:** Takes the support of each class into account. The high weighted average precision, recall, and F1-score (0.97) are influenced by the dominant performance of class 0, which skews the average.

In summary, the model performs well for certain classes but struggles with others, notably class 5. While overall accuracy is high, macro average metrics indicate moderate average performance, highlighting potential issues with class imbalance.



Decision Tree with Feature Selection:

Summary of Class-Wise Performance:

- **Class 0:** The precision, recall, and F1-score for class 0 are high, indicating accurate predictions for this class.
- **Class 1:** Unfortunately, the model fails to correctly identify any instances of class 1, resulting in a zero precision, recall, and F1-score.
- **Class 2:** The precision for class 2 is low, and while recall is slightly better, the F1-score remains relatively low, indicating subpar performance for this class.
- **Class 3:** Class 3 shows decent precision and recall, resulting in a relatively high F1score, suggesting effective classification for this class.
- **Class 4:** The precision and recall for class 4 are moderate, leading to a reasonable F1score, indicating acceptable performance for this class.
- **Class 5:** Similar to class 1, the model fails to identify any instances of class 5 correctly, resulting in zero precision, recall, and F1-score.

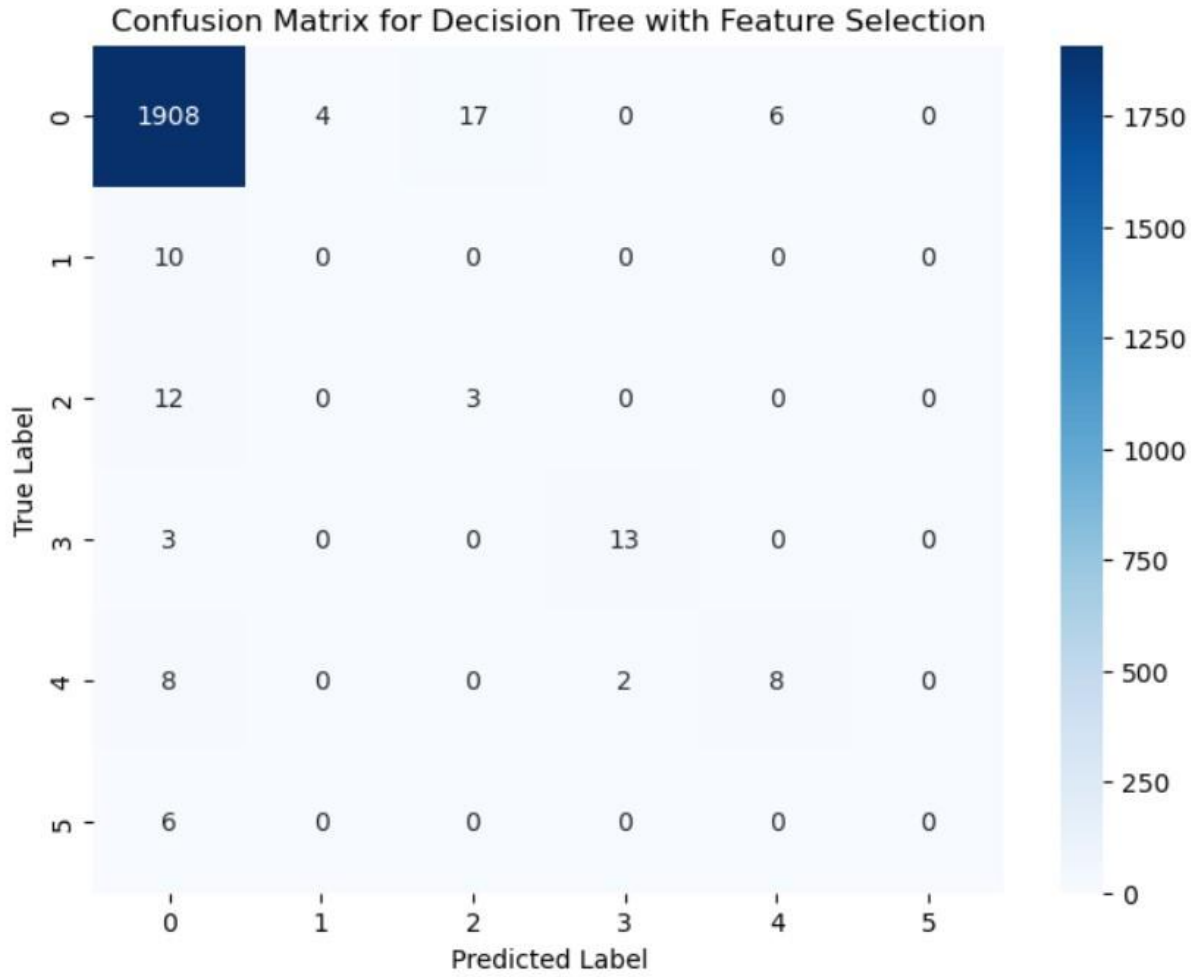
Overall Model Performance Metrics:

Accuracy: The overall accuracy of the model is high at 97%, suggesting effective classification overall.

Macro Average: The macro-average precision, recall, and F1-score are relatively low, indicating inconsistent performance across all classes.

Weighted Average: The weighted average precision, recall, and F1-score are high, influenced by the dominant class (class 0), which has the most instances.

In summary, while the model demonstrates excellent performance for class 0 and reasonable performance for classes 3 and 4, it faces significant challenges in correctly identifying instances from classes 1, 2, and 5.



XGBoost:

Extreme Gradient Boosting, or XGBoost for short, is a potent and effective machine learning algorithm that excels in classification. It is built on the gradient boosting framework and falls under the category of ensemble learning techniques. By building a series of decision trees iteratively and concentrating on the mistakes made by its forebears, XGBoost can increase forecast accuracy over time. It has various improvements over classical gradient boosting, including support for multiple objective functions, parallel processing for faster training, and regularisation strategies to avoid overfitting.

XGBoost with Principal Component Analysis (PCA):

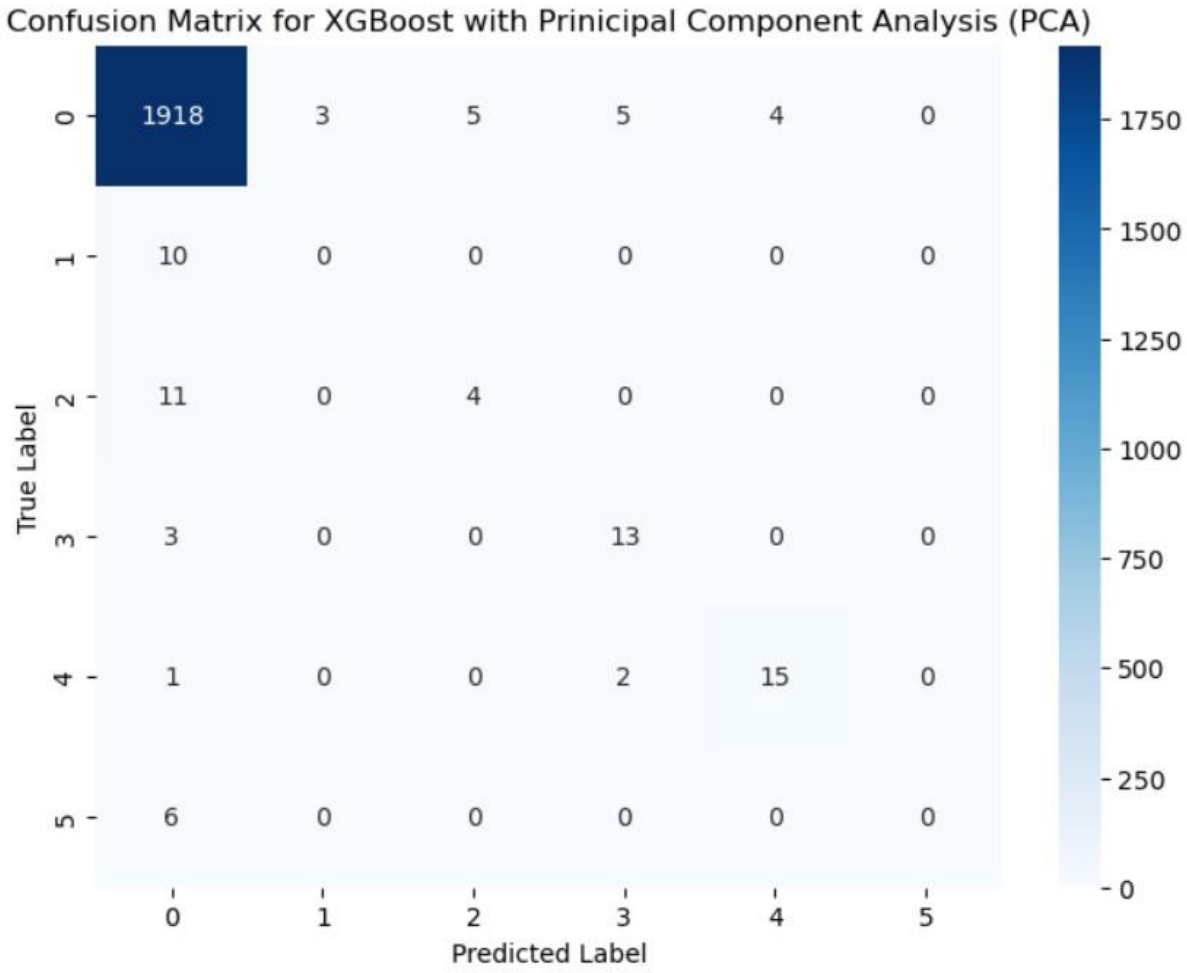
Summary of Class-Wise Performance:

- **Class 0:** The precision, recall, and F1-score for class 0 are high, indicating accurate predictions for this class.
- **Class 1:** Once again, the model fails to correctly identify any instances of class 1, resulting in zero precision, recall, and F1-score.
- **Class 2:** While the precision for class 2 is moderate, the recall is relatively low, resulting in a moderate F1-score, suggesting room for improvement in classifying this class.
- **Class 3:** Class 3 shows decent precision and recall, leading to a relatively high F1score, indicating effective classification for this class.
- **Class 4:** The precision and recall for class 4 are high, resulting in a high F1-score, indicating effective classification for this class.
- **Class 5:** Similar to class 1, the model fails to identify any instances of class 5 correctly, resulting in zero precision, recall, and F1-score.

Overall Model Performance Metrics:

- **Accuracy:** The overall accuracy of the model is high at 97%, suggesting effective classification overall.
- **Macro Average:** The macro-average precision, recall, and F1-score are moderate, indicating somewhat consistent performance across all classes.
- **Weighted Average:** The weighted average precision, recall, and F1-score are high, influenced by the dominant class (class 0), which has the most instances.

In summary, the model demonstrates excellent performance for class 0 and reasonable performance for classes 3 and 4. However, it faces significant challenges in correctly identifying instances from classes 1, 2, and 5, similar to other models evaluated.



XGBoost with Linear Discriminant Analysis (LDA):

Summary of Class-Wise Performance:

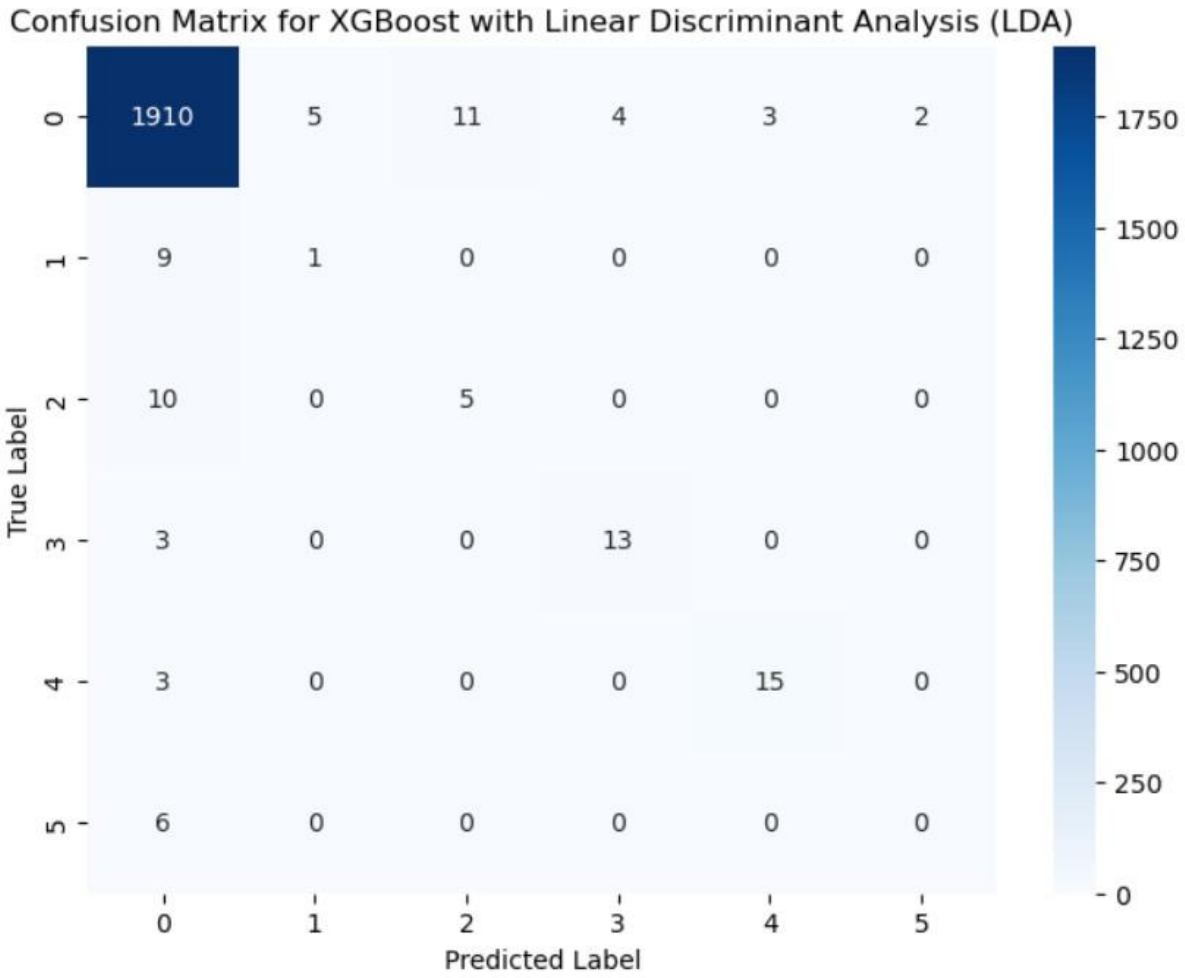
- **Class 0:** The precision, recall, and F1-score for class 0 are high, indicating accurate predictions for this class.
- **Class 1:** The precision for class 1 is relatively low, and while recall is slightly better, the F1-score remains low, suggesting subpar performance for this class.
- **Class 2:** Both precision and recall for class 2 are moderate, resulting in a moderate F1score, indicating reasonable performance for this class.
- **Class 3:** Class 3 shows good precision and recall, leading to a relatively high F1-score, indicating effective classification for this class.

- **Class 4:** The precision and recall for class 4 are high, resulting in a high F1-score, indicating effective classification for this class.
- **Class 5:** Similar to class 1, the model fails to identify any instances of class 5 correctly, resulting in zero precision, recall, and F1-score.

Overall Model Performance Metrics:

- **Accuracy:** The overall accuracy of the model is high at 97%, suggesting effective classification overall.
- **Macro Average:** The macro-average precision, recall, and F1-score are moderate, indicating relatively consistent performance across all classes.
- **Weighted Average:** The weighted average precision, recall, and F1-score are high, influenced by the dominant class (class 0), which has the most instances.

In summary, the model demonstrates excellent performance for class 0 and reasonable performance for classes 2, 3, and 4. However, it faces challenges in correctly identifying instances from classes 1 and 5, similar to other models evaluated.



XGBoost with Feature Selection:

Summary of Class-Wise Performance:

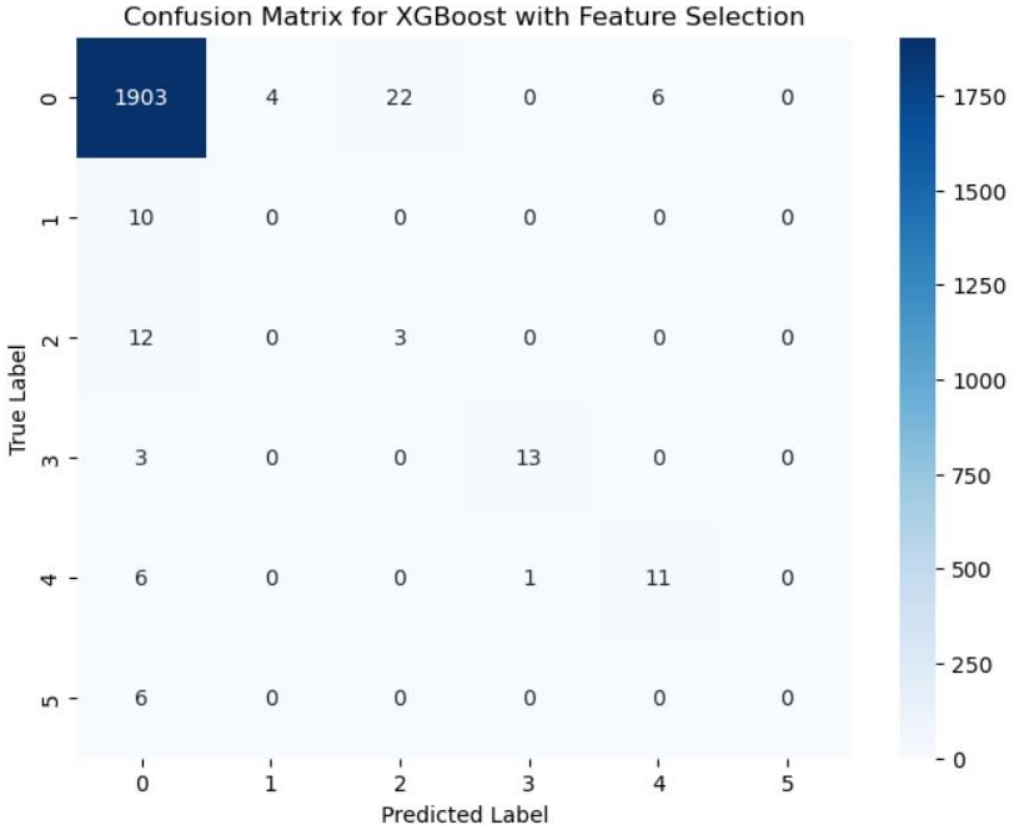
- **Class 0:** The precision and recall for class 0 are high, indicating accurate predictions for this class.
- **Class 1:** Once again, the model fails to correctly identify any instances of class 1, resulting in zero precision, recall, and F1-score.
- **Class 2:** While the precision for class 2 is low, the recall is slightly better, resulting in a low F1-score, indicating subpar performance for this class.
- **Class 3:** Class 3 shows high precision and reasonable recall, leading to a high F1-score, indicating effective classification for this class.

- **Class 4:** The precision and recall for class 4 are moderate, resulting in a moderate F1score, indicating acceptable performance for this class.
- **Class 5:** Similar to classes 1 and 5, the model fails to identify any instances of class 5 correctly, resulting in zero precision, recall, and F1-score.

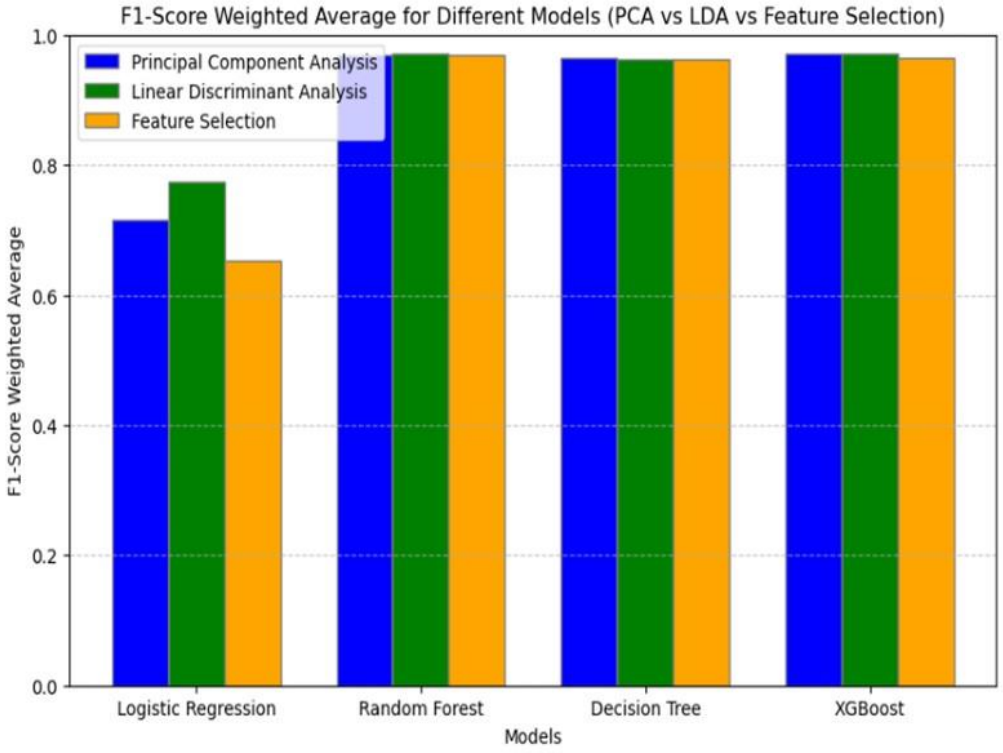
Overall Model Performance Metrics:

- **Accuracy:** The overall accuracy of the model is high at 96%, suggesting effective classification overall.
- **Macro Average:** The macro-average precision, recall, and F1-score are moderate, indicating somewhat consistent performance across all classes.
- **Weighted Average:** The weighted average precision, recall, and F1-score are high, influenced by the dominant class (class 0), which has the most instances.

In summary, the model demonstrates excellent performance for class 0 and reasonable performance for class 3 and class 4. However, it faces significant challenges in correctly identifying instances from classes 1, 2, and 5, like other models evaluated.



XIII. MODEL PERFORMANCE AND EVALUATION:



x

The graph shows the F1-score weighted average for different classification models, comparing the impact of three feature engineering techniques: Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Feature Selection.

The F1-score is a harmonic mean of precision and recall, and in this context, it is weighted to account for the true distribution of classes. The weighted average is particularly important in imbalanced datasets where some classes are underrepresented.

Here is a breakdown of the performance of each model with different feature engineering techniques:

- **Logistic Regression:**
- **PCA:** The F1-score weighted average is around 0.72
- **LDA:** The F1-score weighted average is slightly below 0.77.
- **Feature Selection:** The F1-score weighted average is just above 0.65.
- **Random Forest:**
- **PCA:** The F1-score weighted average is approximately 0.97.

- **LDA:** The F1-score weighted average is around 0.97, which is the highest among the Random Forest techniques.
- **Feature Selection:** The F1-score weighted average is 0.97.

- **Decision Tree:**
- **PCA:** The F1-score weighted average is about 0.97.
- **LDA:** The F1-score weighted average is just above 0.97.
- **Feature Selection:** The F1-score weighted average is just below 0.96.

- **XGBoost:**
- **PCA:** The F1-score weighted average is 0.97, indicating almost perfect performance.
- **LDA:** The F1-score weighted average is also 0.97.
- **Feature Selection:** The F1-score weighted average is at 0.96, indicating great performance.

Overall, the graph indicates that:
- XGBoost performs exceptionally well regardless of the feature engineering technique used, achieving near-perfect F1-scores.
- Random Forest performs same with LDA, Feature Selection and PCA.
- Logistic Regression has similar performance across all feature engineering techniques but better with LDA.
- Decision Tree benefits more from PCA and LDA than feature selection by a minute margin.

XIV. PROJECT RESULTS:

Using a variety of feature engineering strategies, the F1-score weighted averages for several classification models that were produced allowed for the following summary of the project's main findings:

1. Overall Effectiveness of Feature Engineering Techniques:

- When compared to PCA and feature selection, **Linear Discriminant Analysis (LDA)** produces the greatest F1-score weighted average, making it the most successful feature engineering technique for Logistic Regression.

- With very high F1-scores at or at 0.97,

Principal Component Analysis (PCA) and LDA are similarly successful for the Random Forest and Decision Tree models.

- Feature Selection for the Decision Tree model and Logistic Regression somewhat underperforms PCA and LDA, suggesting that the features chosen might not have as much predictive potential as the converted features from PCA and LDA

2. Model Performance

- Regardless of the feature engineering technique employed, Random Forest and XGBoost exhibit superior performance with near-perfect F1-score weighted averages, suggesting robustness to various feature sets and possibly an ability to handle class imbalance effectively.

- Decision Tree performs well when used with PCA and LDA in particular but performs a little worse when used with Feature Selection. This could be the result of the chosen features not capturing enough variance or overfitting.

- Logistic Regression performs the worst out of all the models, yet it still gains from LDA, suggesting that the model's susceptibility to feature extraction and selection may exist.

3. Implications for Imbalanced Datasets:

- The dataset exhibits class imbalance as evidenced using F1-score weighted averages.

- In practice, the models that appear to handle imbalance the best are Random Forest and XGBoost, which have the greatest weighted F1-scores.

- The improvement of Logistic Regression with LDA implies that the technique of feature engineering and extraction can have a substantial impact on the performance of certain models when working with imbalanced datasets.

4. Selection of Models and Methods:

- XGBoost and Random Forest are the top performers and thus good choices for implementation in comparable tasks, particularly in cases when the class distribution is skewed.

Given that PCA and LDA have demonstrated comparable performance for tree-based models, the choice between them can be made depending on the dataset's unique characteristics and computational efficiency.

5. Suggestions for Enhancing the Model:

- Additional fine-tuning of model hyperparameters may result in improvements, particularly for Decision Trees and Logistic Regression.

- Researching feature selection strategies may shed light on why this method did not perform as well as PCA and LDA and determine whether other selection criteria could be used.

It is important to do cross-validation to make sure that these conclusions hold up across various data subsets.

To sum up, this experiment shows that ensemble models perform well on datasets that may be unbalanced and emphasises the value of feature engineering strategies in maximising model performance, especially for non-ensemble models.

XV. IMPACT OF PROJECT OUTCOMES:

The following are some prominent effects of the predictive maintenance project's results and the data mining activities involved:

Cost savings: By anticipating equipment failures, businesses can lower the risk of unscheduled downtime by using proactive maintenance practices. By preventing production losses, lowering maintenance costs, and maximizing resource use, this results in significant cost savings.

Increased Safety: By reducing accidents and the risks connected to equipment breakdowns, prompt intervention based on predictive maintenance models improves workplace safety. Ensuring the dependability of equipment helps make the workplace safer for workers.

Enhanced Operational Efficiency: By scheduling maintenance tasks during scheduled downtimes, predictive maintenance helps businesses minimize production schedule delays. As a result, the manufacturing facility's overall productivity and operational efficiency rise.

Longer Equipment Lifespan: By detecting problems early and taking action before they worsen, proactive maintenance based on predictive models helps increase the lifespan of machinery and equipment. By doing this, assets' operational lives are extended, which lowers the need for regular replacements and capital expenses.

XVI. REFERENCES:

1. Matzka, S. (2020). Explainable Artificial Intelligence for Predictive Maintenance Applications. 2020 Third International Conference on Artificial Intelligence for Industries (AI4I), 69-74.
2. Barbierato E, Vedova MLD, Tessera D, Toti D, Vanoli N. A Methodology for Controlling Bias and Fairness in Synthetic Data Generation. Applied Sciences. 2022; 12(9):4619. https://doi.org/10.3390/app12094619