ENGINEERING
W Booth School of Engineering Practice and Technology

McMaster University

# SEP 6DA3: DATA ANALYTICS AND BIG DATA

## FALL 2024

**- Siqi Zhao**

## Final Project: APS System Failure Prediction using supervised learning

GROUP 19


| Student Name | Student Id |
|---|---|
| Henis Nakrani (nakranih) | 400547270 |
| Ishika Vaghasiya (vaghasii) | 400551691 |

# 1. Introduction

Air pressure systems are critical for the braking mechanism in heavy-duty trucks. Failures in these systems can result in severe consequences, including accidents and operational inefficiencies. This project aims to predict such failures using machine learning techniques, leveraging historical sensor data provided in training and testing datasets.

**Primary objectives:**

- To preprocess and clean the datasets for effective modelling.
- To evaluate different machine learning models, including Decision Trees, Logistic regression and Random Forest.
- To assess model performance using various metrics and extract actionable insights.

**Importance:**

Early failure detection can reduce costs and improve reliability by addressing potential faults before they escalate into breakdowns.

# 2. Dataset Overview

The dataset consists of data collected from heavy Scania trucks in everyday usage. The system in focus is the Air Pressure system (APS) which generates pressurised air that are utilized in various functions in a truck, such as braking and gear changes. The datasets positive class consists of component failures for a specific component of the APS system. The negative class consists of trucks with failures for components not related to the APS. The data consists of a subset of all available data, selected by experts.

**Number of Instances:**

- The **training** set contains **60000** examples in total in which **59000** belong to the **negative** class and **1000 positive** class.
- The **test** set contains **16000** examples.
- Number of **Attributes (Features)**: **171**

**Attribute Information:**

- The attribute names of the data have been anonymized for proprietary reasons. It consists of both single numerical counters and histograms consisting of bins with different conditions. Typically, the histograms have open-ended conditions at each end. For example, if we are measuring the ambient temperature "T" then the histogram could be defined with 4 bins where:
- bin 1 collect values for temperature T < -20
- bin 2 collect values for temperature T >= -20 and T < 0
- bin 3 collect values for temperature T >= 0 and T < 20
- bin 4 collect values for temperature T > 20

**The attributes are as follows:**

- class, then anonymized operational data. The operational data have an identifier and a bin id, like "Identifier_Bin".
- In total there are 171 attributes, of which 7 are histogram variables. Missing values are denoted by "na".

# 3. Data Preparation

## 3.1 Data Loading

The data for this project includes training and testing datasets, each containing sensor readings along with target labels indicating APS failures. The datasets were loaded using Python's pandas library, which simplifies data handling and manipulation.

**Results of Data Loading:**

- **Training Data:**
  - Contains sensor readings as features and a target column indicating APS failures (pos for failure and neg for no failure).
  - Example rows from the training dataset:

| class | feature1 | feature2 | feature3 | ... |
|---|---|---|---|---|
| neg | 0.13 | -0.34 | 0.45 | ... |
| pos | 0.23 | 0.89 | -0.12 | ... |

```
Training Data Loaded Successfully!
Training Data Shape: (60000, 171)
Code cell output actions
  class aa_000 ab_000      ac_000 ad_000 ae_000 af_000 ag_000 ag_001 ag_002  \
0   neg   76698     na  2130706438    280      0      0      0      0      0
1   neg   33058     na           0     na      0      0      0      0      0
2   neg   41040     na         228    100      0      0      0      0      0
3   neg      12      0          70     66      0     10      0      0      0
4   neg   60874     na        1368    458      0      0      0      0      0

   ...   ee_002  ee_003  ee_004  ee_005  ee_006  ee_007  ee_008 ee_009 ef_000  \
0  ...  1240520  493384  721044  469792  339156  157956   73224      0      0
1  ...   421400  178064  293306  245416  133654   81140   97576   1500      0
2  ...   277378  159812  423992  409564  320746  158022   95128    514      0
3  ...      240      46      58      44      10       0       0      0      4
4  ...   622012  229790  405298  347188  286954  311560  433954   1218      0

  eg_000
0      0
1      0
2      0
3     32
4      0

[5 rows x 171 columns]
```

- **Testing Data:**
  - Shape: 16,000 rows and 171 columns.
  - Similar structure to the training data, used for evaluating model performance.

```
Testing Data Loaded Successfully!
Testing Data Shape: (16000, 171)
  class  aa_000 ab_000 ac_000 ad_000 ae_000 af_000 ag_000 ag_001 ag_002  ...  \
0   neg      60      0     20     12      0      0      0      0      0  ...
1   neg      82      0     68     40      0      0      0      0      0  ...
2   neg   66002      2    212    112      0      0      0      0      0  ...
3   neg   59816     na   1010    936      0      0      0      0      0  ...
4   neg    1814     na    156    140      0      0      0      0      0  ...

   ee_002  ee_003  ee_004  ee_005   ee_006  ee_007  ee_008 ee_009 ef_000  \
0    1098     138     412     654       78      88       0      0      0
1    1068     276    1620     116       86     462       0      0      0
2  495076  380368  440134  269556  1315022  153680     516      0      0
3  540820  243270  483302  485332   431376  210074  281662   3232      0
4    7646    4144   18466   49782     3176     482      76      0      0

  eg_000
0      0
1      0
2      0
3      0
4      0

[5 rows x 171 columns]
```

## 3.2 Data Exploration and Preprocessing

Several steps were undertaken to clean the data:

1. **Replacing Missing Values**: Features containing the placeholder "na" were converted to NaN. Features with more than 30% missing values were removed. Remaining missing values were filled with the mean of each column.
2. **Class Transformation**: The target variable ("class") was converted from "pos" and "neg" to binary labels (1 and 0, respectively).
3. **Data Types Conversion**: All features were converted to numeric for compatibility with machine learning models.
4. **Feature Reduction**: Columns with low variance were removed to reduce noise and improve model performance.
5. **Removed Duplicates**: Removed duplicate rows and made a proper row unique dataset.

```
Data Cleaning Started!
Training Data Shape before Cleaning: (60000, 171)
Testing Data Shape before Cleaning: (16000, 171)
<ipython-input-5-d40896b87353>:12: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future versic
  train_data.replace({'neg': 0, 'pos': 1}, inplace=True)
<ipython-input-5-d40896b87353>:13: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future versic
  test_data.replace({'neg': 0, 'pos': 1}, inplace=True)
```

```
Missing Values in Training Data:
class         0
aa_000        0
ab_000    46329
ac_000     3335
ad_000    14861
          ...
ee_007      671
ee_008      671
ee_009      671
ef_000     2724
eg_000     2723
Length: 171, dtype: int64

Missing Values in Testing Data:
class         0
aa_000        0
ab_000    12363
ac_000      926
ad_000     3981
          ...
ee_007      192
ee_008      192
ee_009      192
ef_000      762
eg_000      762
Length: 171, dtype: int64

Data Cleaning Completed!
Training Data Shape after Cleaning: (60000, 161)
Testing Data Shape after Cleaning: (16000, 161)
```

## 3.3 Feature Engineering

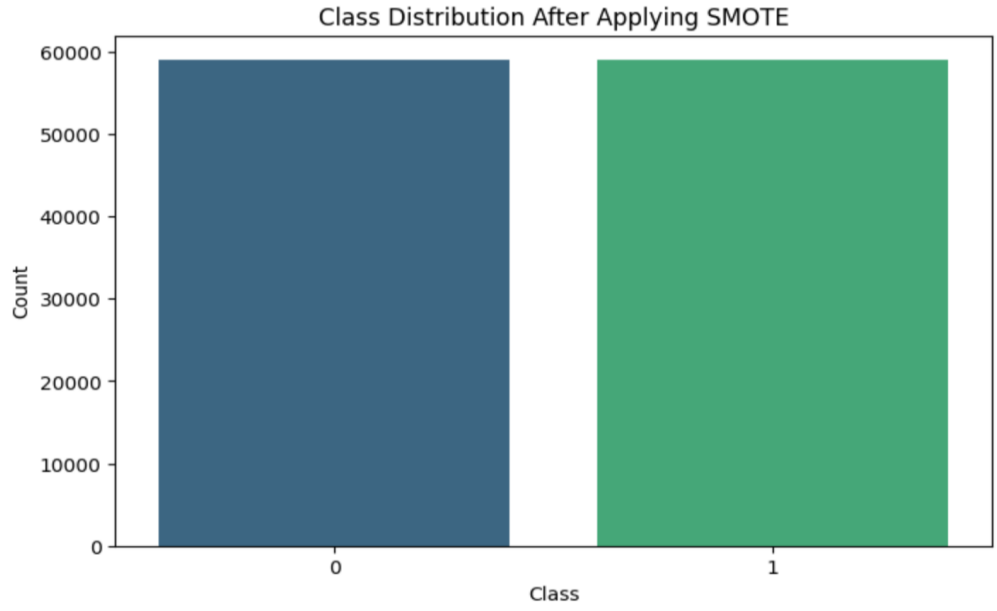
1. **Visualize Class Imbalance Training Data:**

- Identify any class imbalance: By plotting the class distributions of both training and testing datasets, you can visually assess if one class is overrepresented compared to the other. Class imbalance can significantly affect machine learning model performance, especially when the model becomes biased toward the majority class.
- Visualize the data: This step helps in understanding the structure of the data before training models and deciding if any data preprocessing (e.g., oversampling, undersampling) or other techniques are necessary to address class imbalance.
- **Training Data:**

```
<ipython-input-11-035c289dea64>:14: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign th

  sns.countplot(x=y_train, palette='viridis')
```

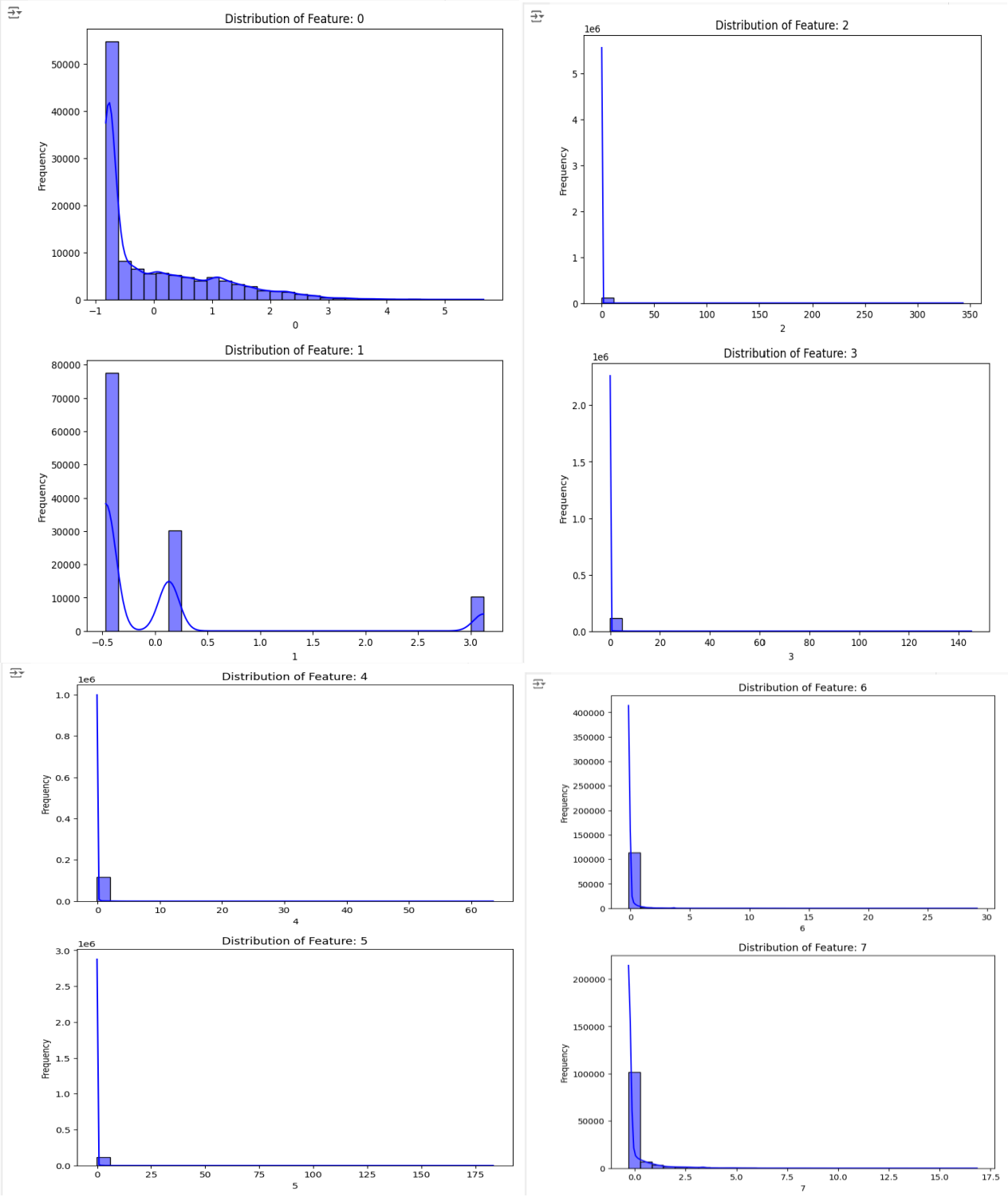

- **Testing Data after applying SMOT:**

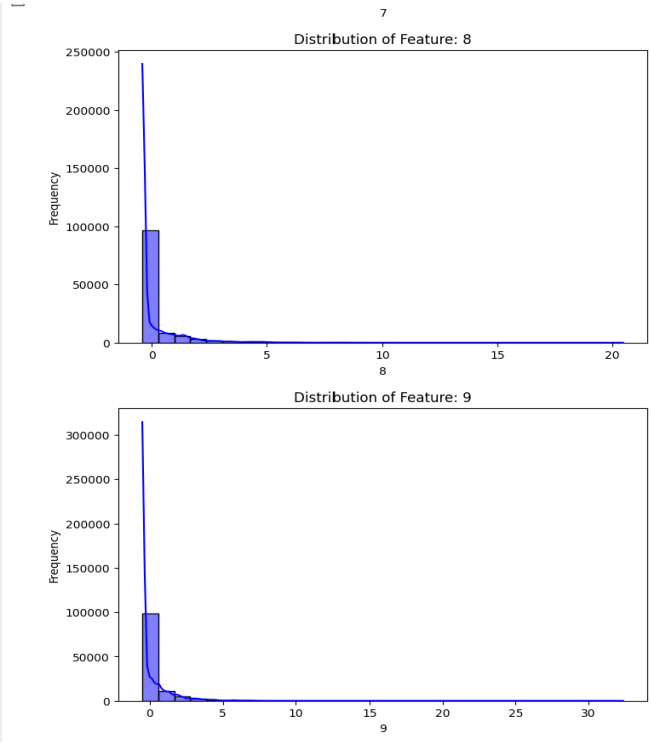
```
<ipython-input-12-077aba2483dc>:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign

  sns.countplot(x=y_train, palette='viridis')
```



**2. Visualize Feature Distributions:**

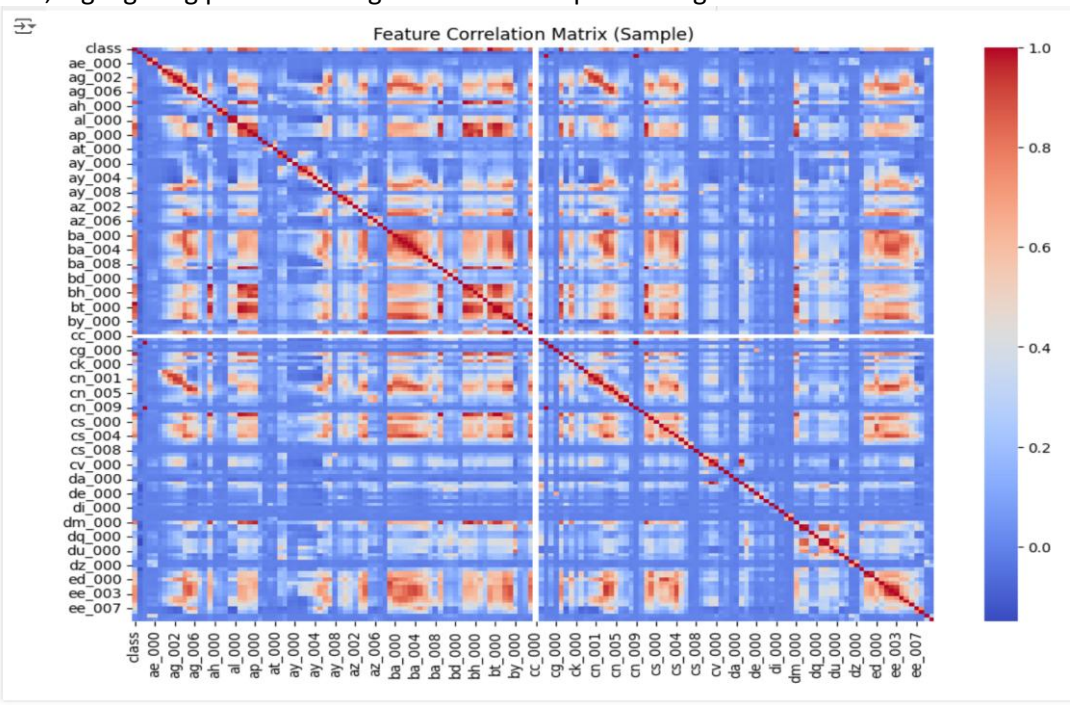
- Visualize the distribution of the first 10 features in the training dataset using histograms with an overlaid KDE curve. This visualization is important for several reasons:
- **Feature Understanding:** It provides insights into how each feature is distributed (e.g., normal, skewed, uniform, etc.).
- **Identifying Skewness:** Features that are highly skewed might need transformations (e.g., logarithmic scaling) before feeding them into a model.
- **Outliers and Data Patterns:** Outliers or unusual patterns in the features might be detected visually, which can help decide whether to preprocess the data (e.g., remove outliers).
- **Normality of Distribution:** Checking whether the features follow a normal distribution, which is useful for certain algorithms that assume normality (e.g., linear regression, logistic regression).






# 4. Correlation Matrix

- A heatmap of the correlation matrix for the features in the training data. Visualizing the correlations between features helps to:
- **Identify Multicollinearity:** Features that are highly correlated with each other (i.e., correlation values close to +1 or -1) may cause issues in certain machine learning models (e.g., linear regression, logistic regression). High correlation between independent variables can lead to multicollinearity, which may cause instability in the model coefficients.
- **Feature Selection:** If some features are highly correlated with each other, one of the correlated features may be redundant and could be removed without losing much information. This can help reduce dimensionality and improve model performance.
- **Data Understanding:** The heatmap allows for easy visualization of how the features relate to each other, highlighting positive or negative relationships that might not be obvious from raw data.



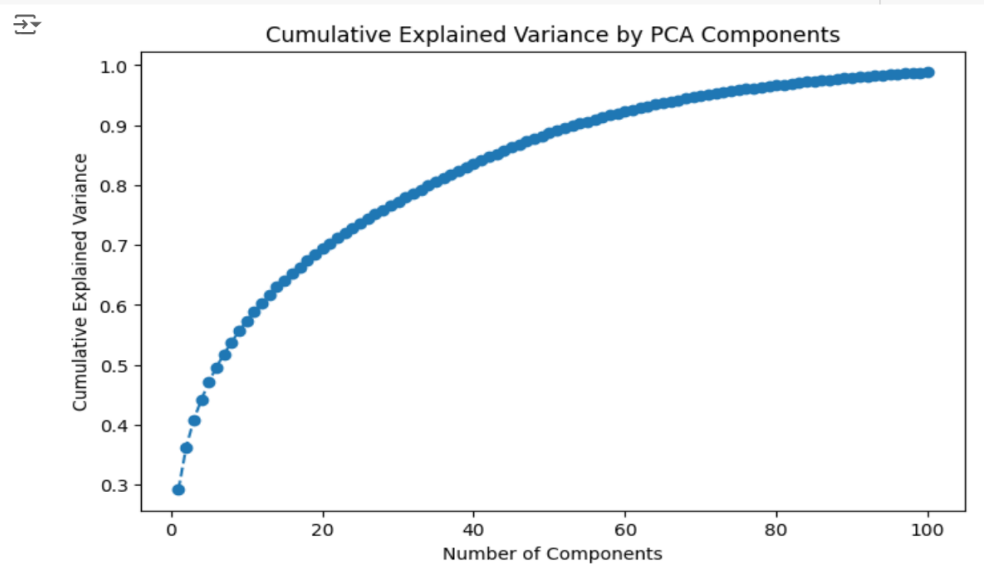
- **Key Observations from the Correlation Matrix Heatmap:**
  - **Diagonal Line:** The diagonal line in the heatmap, where the features correlate with themselves, is expected to have perfect correlations of 1. The rest of the matrix shows pairwise correlations between different features.
  - **Color Gradient:**
    - The colour scale on the right (from blue to red) represents the correlation values.
    - **Blue areas** indicate **low correlations** (near 0), meaning the features are less related to each other.
    - **Red areas** indicate **high correlations** (near 1 or -1), suggesting that these features are strongly related.
  - **Patterns of High Correlation:**
    - Some blocks or sections of the matrix seem to have an intense red color, indicating that several features are highly correlated with one another. This can indicate multicollinearity, where multiple features provide redundant information.
    - Identifying highly correlated feature pairs (e.g., if the correlation is close to 1 or -1) can help in feature selection to reduce dimensionality by potentially removing one of the correlated features.
- **Insights from the Matrix:**
    - Most of the matrix is dominated by **blue** tones, which indicate low or no correlation (values near 0) among many features.
    - This suggests that most of the features are weakly correlated or independent of one another, which is ideal for feature diversity in machine learning.
    - Look around feature groups like ba_ and cn_. Features in these groups seem to correlate with each other. This could suggest that certain operational features (possibly histogram bins) measure related aspects of the system.

# 5. Feature Scaling

- **StandardScaler**: This method standardizes features by removing the mean and scaling to unit variance. This ensures all features contribute equally to the model, which is particularly important for PCA and machine learning algorithms sensitive to feature magnitudes.
- **fit_transform(X_train)**: Standardizes the training features by calculating the mean and standard deviation, then transforming the data.
- **transform(X_test)**: Standardizes the test data using the same statistics (mean and standard deviation) learned from the training set.
- After scaling, the scaled features are converted back into DataFrames for better usability.

# 6. Dimensionality Reduction using PCA

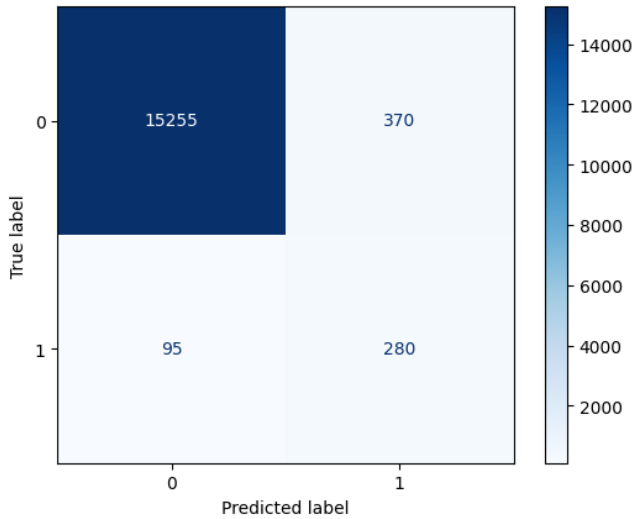
- Reduces computational complexity by lowering the number of features to 52.
- **Components: 52**
- Helps remove noise and redundancy from the dataset.
- Improves model performance, particularly for high-dimensional datasets.
- This graph helps determine the optimal number of components that explain a sufficient amount of variance (e.g., 90-95%).
- A plateau in the curve indicates diminishing returns for adding more components.




# 7. Decision Tree

The Decision Tree Classifier is implemented as one of the predictive models to identify Air Pressure System (APS) failures. Decision Trees are interpretable and handle both categorical and numerical data effectively.

- **Random State:** Set to 42 for reproducibility.
- **Input Data:**
  - **Training Data:** Transformed features using **PCA** (X_train_pca).
  - **Test Data:** Transformed test set features (X_test_pca).
  - **Target:** y_train and y_test representing the APS failure classes.
- **Confusion Matrix:**
  - The confusion matrix helps visualize model predictions against actual values for both classes (0 = Negative, 1 = Positive).
    **Confusion Matrix:**




  - **True Negatives (TN):** 15,255
  - **False Positives (FP):** 370
  - **False Negatives (FN):** 95
  - **True Positives (TP):** 280
    The confusion matrix highlights that the model performs very well on the negative class but struggles to perfectly classify all positive class instances.

- **Classification Report:**
  The classification report evaluates the model's performance in terms of **precision**, **recall**, **f1-score**, and **support** for each class:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98     15625
           1       0.43      0.75      0.55       375

    accuracy                           0.97     16000
   macro avg       0.71      0.86      0.77     16000
weighted avg       0.98      0.97      0.97     16000

ROC-AUC Score: 0.8614933333333332
```

- **ROC-AUC Score**
  - The ROC-AUC score measures the model's ability to distinguish between positive and negative classes.
    **Test ROC-AUC Score: 0.861**
  - This score indicates a good ability to separate the classes but highlights room for improvement for the minority class.
- **Cross-Validation Results**
  To ensure the model's robustness, 5-Fold Cross-Validation was performed using the ROC-AUC metric. The results are as follows:

| Fold | ROC-AUC Score |
|---|---|
| 1 | 0.9806 |
| 2 | 0.9807 |
| 3 | 0.9833 |
| 4 | 0.9845 |
| 5 | 0.9839 |

  **Mean ROC-AUC**: 0.9826
  - The high mean ROC-AUC score across folds demonstrates that the model generalizes well and performs consistently on unseen data.

- **Key Observations**
  - **Strong Performance for Class 0:**
    - The model achieves high precision (0.99) and recall (0.98) for the negative class.
  - **Challenges for Class 1 (Positive Failures):**
    - While the recall for Class 1 is relatively high at 0.75 (correctly identifying 75% of positive cases), precision is lower at 0.43.
    - This indicates that some predictions for the positive class are still incorrect (higher false positives).
  - **ROC-AUC Score:**
    - The ROC-AUC score on the test set is **0.861**, indicating good but improvable performance in separating the classes.
  - **Cross-Validation:**
    - Cross-validation results are consistent, with a mean ROC-AUC score of **0.9826**, confirming the stability of the model.

# 8. Logistic Regression

Logistic Regression is a linear model used for binary classification tasks. In this project, the Logistic Regression model was trained using **L2 regularization** with an inverse regularization strength (**C = 0.001**). The purpose of using L2 regularization is to prevent overfitting and ensure generalization on unseen data.

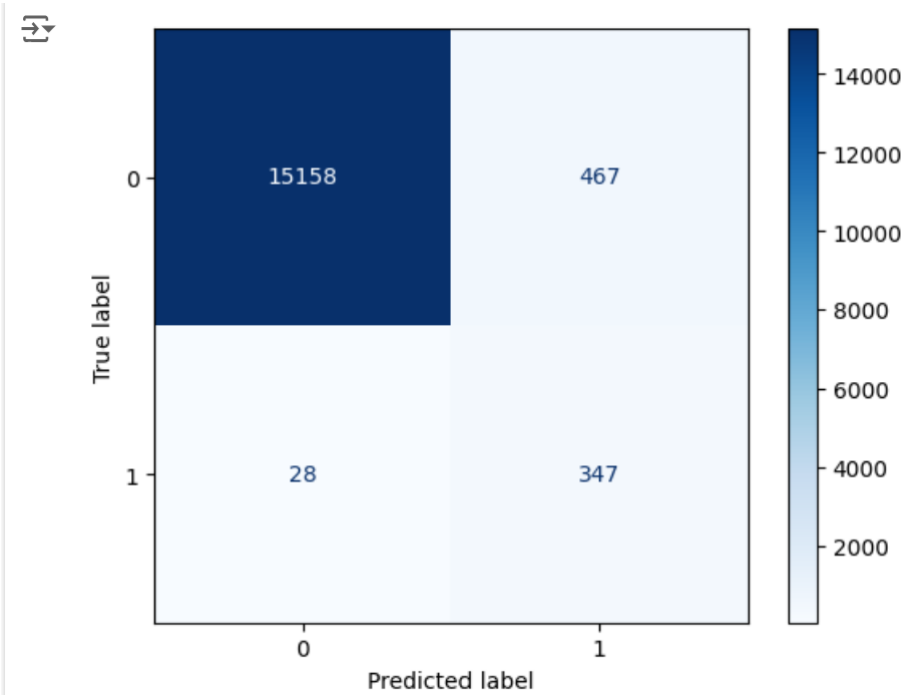
- **Model Training:**
  - **Training Dataset**: The model was trained on the transformed feature set obtained using **Principal Component Analysis (PCA)**, where the features were reduced to 52 principal components.
  - **Model Parameters**:
    - Regularization: **L2**
    - Inverse Regularization Strength (C): **0.001**

- **Model Evaluation:**

The performance of the Logistic Regression model was evaluated on the test dataset. The following evaluation metrics were used: **Confusion Matrix**, **Classification Report**, and **ROC-AUC Score**.

- **Confusion Matrix:**

The confusion matrix below summarizes the model's predictions:




The confusion matrix indicates that the model performs well on identifying majority class samples (Class 0), but there is a moderate number of false positives and false negatives for the minority class (Class 1).

- **Classification Report**:
  - The classification report highlights that:
    - **Class 0** (majority class) achieved near-perfect precision, recall, and F1-score.
    - **Class 1** (minority class) had excellent recall (93%), indicating that most positive cases were correctly identified. However, precision for Class 1 is relatively low at 43%, suggesting a significant number of false positives.

```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.97      0.98     15625
           1       0.43      0.93      0.58       375

    accuracy                           0.97     16000
   macro avg       0.71      0.95      0.78     16000
weighted avg       0.98      0.97      0.97     16000

Logistic Regression ROC-AUC Score: 0.9879475200000001
```

- **ROC-AUC Score:**
  - The **Receiver Operating Characteristic - Area Under Curve (ROC-AUC)** score evaluates the model's ability to distinguish between the two classes.
  - **Logistic Regression ROC-AUC Score**: **0.9879**
  - A score of **0.9879** indicates that the model performs well in distinguishing between positive and negative classes, demonstrating strong discriminatory power.
- **Cross-Validation Results:**
  - To ensure the robustness and generalizability of the Logistic Regression model, 5-fold cross-validation was performed using the ROC-AUC score as the evaluation metric.
  - Cross-Validation ROC-AUC Scores: [0.9799, 0.9814, 0.9798, 0.9814, 0.9816]
  - Mean ROC-AUC Score: 0.9808
  - The high and consistent cross-validation scores confirm that the model generalizes well across different data splits, further validating its performance.

- **Key Observations:**
  1. The Logistic Regression model achieved a **97% accuracy** and a **ROC-AUC score of 0.9879**, indicating strong overall performance.
  2. The model's **recall for Class 1** (93%) is impressive, ensuring that most positive cases are identified.
  3. Precision for Class 1 is relatively low (43%), leading to a higher false positive rate.
  4. The cross-validation results confirm that the model is stable and reliable, with a mean ROC-AUC score of **0.9808**.

## 9. Random Forest Classifier

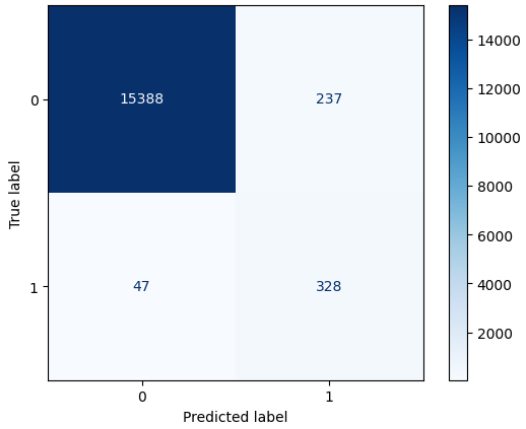
- **Model Training:**
  The model used is the Random Forest Classifier with the following parameters:
  - **n_estimators**: 25 (number of trees in the forest)
  - **max_features**: 'log2' (maximum number of features to consider for the best split)
  - **oob_score**: True (out-of-bag score for generalization performance)

  The model was trained using PCA-transformed data (X_train_pca) and evaluated on test data (X_test_pca).

- **Model Evaluation:**

  The performance of the Logistic Regression model was evaluated on the test dataset. The following evaluation metrics were used: **Confusion Matrix**, **Classification Report**, and **ROC-AUC Score**.

- **Confusion Matrix:**




- **Observations:**
  - True Negative (TN): 15,388 → Correctly predicted class 0 (majority class).
  - False Positive (FP): 237 → Class 0 was wrongly predicted as class 1.
  - False Negative (FN): 47 → Class 1 was wrongly predicted as class 0.
  - True Positive (TP): 328 → Correctly predicted class 1 (minority class).

- **Performance Metrics**

  **Classification Report:** The classification report provides precision, recall, and F1-score for each class:

```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     15625
           1       0.58      0.87      0.70       375

    accuracy                           0.98     16000
   macro avg       0.79      0.93      0.84     16000
weighted avg       0.99      0.98      0.98     16000

Random Forest ROC-AUC Score: 0.9902265173333333
```

- **Interpretation:**
  - **Precision for Class 0:** 1.00 → Almost perfect precision for the majority class.
  - **Precision for Class 1:** 0.58 → 58% of the predicted class 1 instances are correct.
  - **Recall for Class 1:** 0.87 → 87% of true class 1 instances are correctly identified.
  - **F1-Score for Class 1:** 0.70 → A good balance between precision and recall for minority class 1.
- **Accuracy**
  - **Overall accuracy**: 98% (very high due to the dominance of class 0).

- **ROC-AUC Score**
  - Random Forest ROC-AUC Score**: 0.9902**

- **Cross-Validation accuracy**
  - Cross-Validation ROC-AUC Scores**:** [0.99946, 0.99958, 0.99942, 0.99962, 0.99955]
  - Mean Cross-Validation ROC-AUC**: 0.99953**

- **Interpretation of ROC-AUC:**
  The ROC-AUC score of **0.99** indicates excellent performance in distinguishing between the two classes. The high mean cross-validation ROC-AUC (**0.99953**) demonstrates the model's robustness and consistency.

- **Model Performance Insights**
  - Strengths**:**
    - **High Accuracy (98%)**: The Random Forest model correctly predicts the majority of test samples.
    - **High ROC-AUC Score**: The model is excellent at discriminating between positive and negative classes.
    - **Robust Generalization**: The consistent cross-validation scores show that the model performs reliably across different data splits.
  - Weaknesses**:**
    - **Class Imbalance:** The dataset is highly imbalanced (class 0: 15,625 vs. class 1: 375). This leads to:
      1. Lower precision for class 1 (only 58%).
      2. The model struggles to predict class 1 correctly, evident from **237 false positives** and **47 false negatives**.

- **Recall for Class 1**: While 87% recall is good, the model still misclassifies **47 instances** of class 1.
- **F1-Score for Class 1 (0.70)**: This shows a reasonable performance in balancing precision and recall for the minority class.

# 10. Comparison

| Model | **ROC-AUC Score** | **CV ROC-AUC Accuracy** | **Precision** | **F1-Score** | **Recall** |
|---|---|---|---|---|---|
| **Decision Tree** | 86% | 98% | 98% | 97% | 97% |
| **Logistic regression** | 98% | 98% | 98% | 97% | 97% |
| **Random Forest** | 99% | 99% | 99% | 98% | 98% |

- **Key Observations**
  - **ROC-AUC Score**
  - Random Forest provides the best class separation, while Logistic Regression is also very strong. Decision Tree underperforms in comparison.

- **Cross-Validation ROC-AUC (CV ROC-AUC)**
  - Random Forest demonstrates exceptional consistency and slightly outperforms the other models.

- **Accuracy**
  - **Decision Tree & Logistic Regression: 98%** → Both models achieve strong accuracy, likely due to the majority class dominating predictions.
  - **Random Forest: 99%** → Outperforms the other models, demonstrating superior overall performance.

- **Interpretation**: Random Forest consistently performs better in all metrics (Precision, Recall, and F1-Score), making it the best-performing model for both precision and recall balance.

- **Comparative Strengths and Weaknesses**

| Model | Strengths | Weaknesses |
|---|---|---|
| **Decision Tree** | - Simple and interpretable model. | - Poor ROC-AUC score (86%).<br>- Limited capability to capture complex patterns. |
| **Logistic Regression** | - Strong ROC-AUC (98%).<br>- High precision and recall. | - Assumes linear relationships between features, limiting performance for complex data. |
| **Random Forest** | - Highest performance across all metrics.<br>- Strong generalization (99%). | - Higher computational cost compared to simpler models.<br>- Less interpretable. |

- **Conclusion**
  Based on the analysis:
  1. **Random Forest** is the best-performing model with the highest **ROC-AUC score (99%)**, **accuracy (99%)**, and superior values for **precision**, **recall**, and **F1-score**. It effectively captures complex patterns in the data.
  2. **Logistic Regression** is a close second, delivering strong ROC-AUC and accuracy. It is a simpler and computationally efficient alternative when interpretability is important.
  3. **Decision Tree** underperforms in comparison, especially in terms of ROC-AUC score. It can be improved using ensemble techniques like **Random Forest** or **Gradient Boosting**.

# 11. Limitations and Improvements

- **Limitations**
  - **Class Imbalance**
    - The dataset is highly imbalanced, leading to poor performance for the minority class (low precision and F1-score for Class 1).
  - **Dimensionality Reduction (PCA)**
    - PCA reduces interpretability and may cause loss of important feature information.
  - **Linear Model Limitation**
    - Logistic Regression is linear and may fail to capture non-linear patterns in the data.
  - **Dependence on Feature Scaling**
    - Model performance relies on consistent feature scaling across training and test datasets.
  - **Overfitting Risk**
    - PCA and large datasets can still lead to overfitting if not carefully managed.
  - **Single Metric Focus**
    - Relying heavily on ROC-AUC ignores other critical metrics like F1-score, which is crucial for imbalanced data.
- **Improvements:**
  To address the class imbalance and further improve model performance:

- **Resampling Techniques**:
  - **Oversampling** the minority class using SMOTE (Synthetic Minority Over-sampling Technique).
  - **Undersampling** the majority class to create a balanced dataset.
- **Parameter Tuning**:
  - Increase the **number of estimators** (n_estimators) to improve stability.
  - Fine-tune other hyperparameters using GridSearchCV or RandomizedSearchCV.
- **Class Weight Adjustment**:
  - Use class_weight='balanced' in Random Forest to penalize misclassifications of the minority

class.

- **Threshold Tuning**:
  - Adjust the decision threshold for class 1 to increase its precision and recall.
- **Feature Importance Analysis**:
  - Analyze feature importances to identify key features contributing to predictions and focus on improving their quality.