STAT 576 - FINAL PROJECT

A. Introduction and Motivation

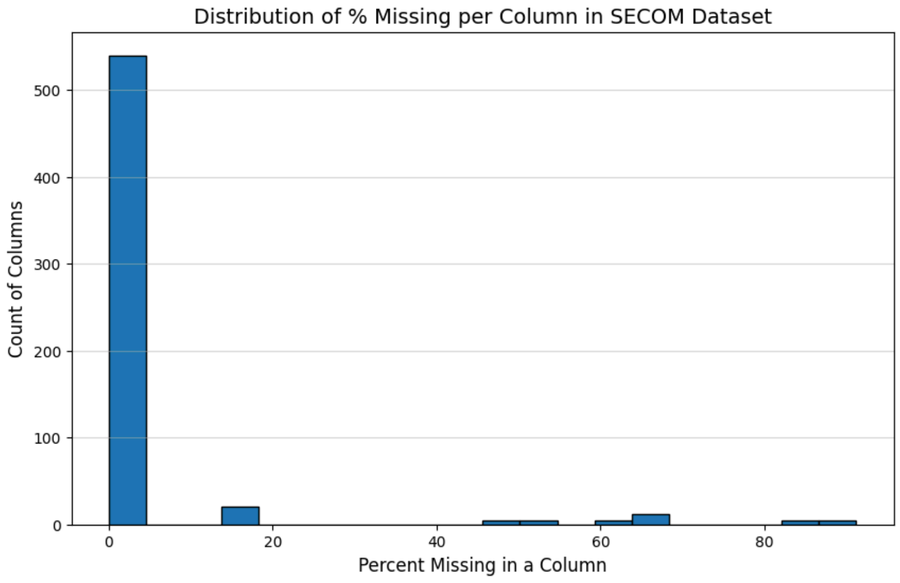
1. ***Background:***
   In modern semiconductor manufacturing, hundreds of sensors monitor the production process to ensure product quality. The SECOM dataset records these sensor readings along with a final pass/fail label for each product. In such high-dimensional settings, unsupervised learning can help uncover hidden process states, fault modes, and abnormal operating conditions.
2. ***Project goal:***
   The main objective of this project is to find the best clustering structure for the SECOM dataset and to interpret the resulting clusters in the context of semiconductor manufacturing quality. Specifically, I compare multiple clustering algorithms and dimensionality-reduction methods, and evaluate how well the clusters align with the original pass/fail labels.

B. Data Description and Preprocessing

1. ***Data Overview***

- The SECOM dataset consists of 1567 observations and 592 features, where each feature corresponds to a sensor measurement or test variable in the production process. In addition, a binary label indicates whether the final product passed (label = 1) or failed (label = -1) quality control.
- There are 590 predictors, one variable for target (Pass/Fail), and one Tine variable.
- Many of the sensor variables are continuous and operate on different scales, resulting in a high-dimensional, heterogeneous feature space. The dataset also contains a non-negligible amount of missing values.

2. ***Missing Data and Feature Cleaning***



- There are 538 variables containing missing values. To deal with this, different strategies have been applied:
  - Feature removal: 32 variables with more than 30% missing values are removed from the dataset.
  - Row removal: missing values of a variable with less than 5% of the total observations are dropped.
  - Imputation: Remaining missing entries are imputed using mean or median based on the variable distribution (normal or skewed distribution respectively). This step will be done after exploring the distribution of 20 variables that still have missing
- Since the features have different units and scales, we standardize each continuous variable to have zero mean and unit variance using z-score standardization. This step is particularly important for distance-based clustering algorithms such as k-means, where differences in scale can disproportionately affect the distance computation.
- After cleaning and preprocessing, our final analysis dataset contains 1393 observations and 560 variables. These preprocessed features serve as the input for both our baseline clustering and dimensionality reduction methods.

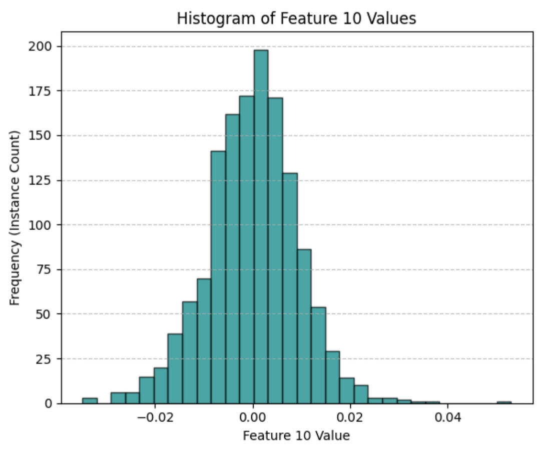
C. Descriptive Statistics & Exploratory Analysis

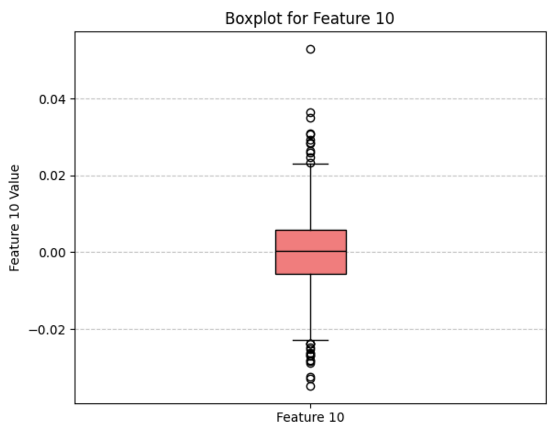
1. ***Summary statistics from selected variables:***

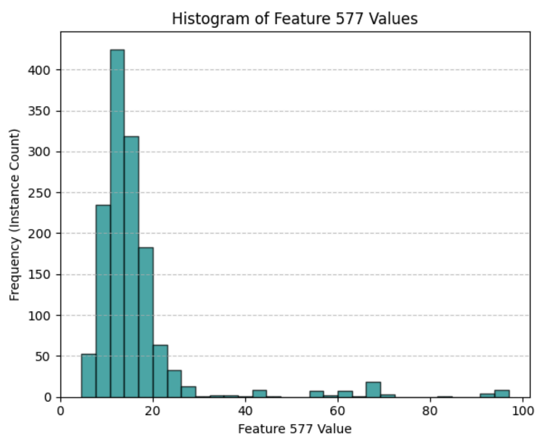
Variable	count	mean	std	min	25%	50%	75%	max
59	1393	3.108	9.69	-28.988	-1.782	0.974	4.441	168.146
164	1393	0.127	0.236	0	0.069	0.09	0.117	1.817
99	1393	0.001	0.064	-0.528	-0.031	0	0.03	0.885
64	1393	20.538	4.989	6.448	17.377	20.031	22.794	48.988
124	1393	15.798	0.113	15.43	15.73	15.78	15.88	16.1
205	1393	9.216	12.168	2.3	6.06	7.82	10.03	320.05

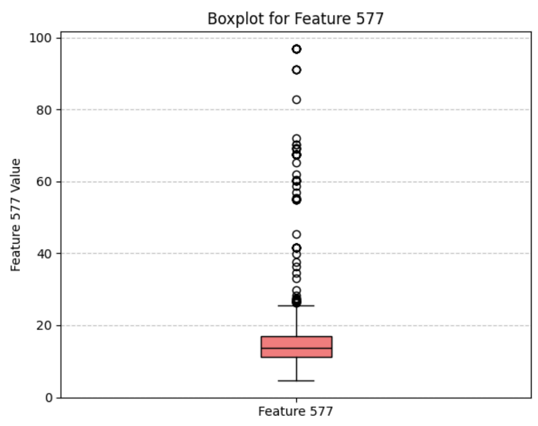
2. ***Distributions:***

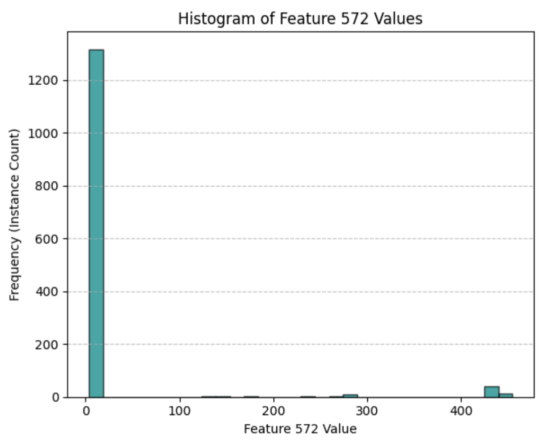
   The distribution of variables are very varied and diverse including normal distribution and right/left skewed distribution. Few of them show a constant value.
   The below plot are example from some variables in the dataset:

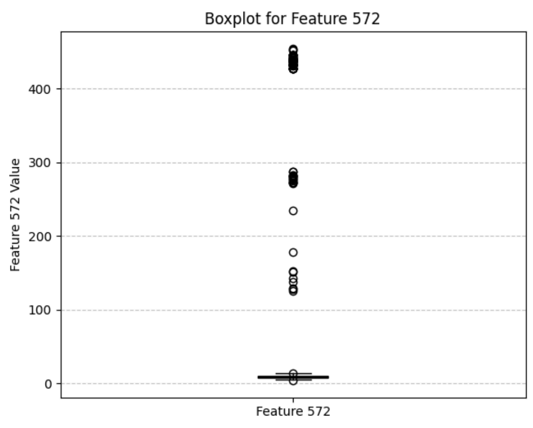








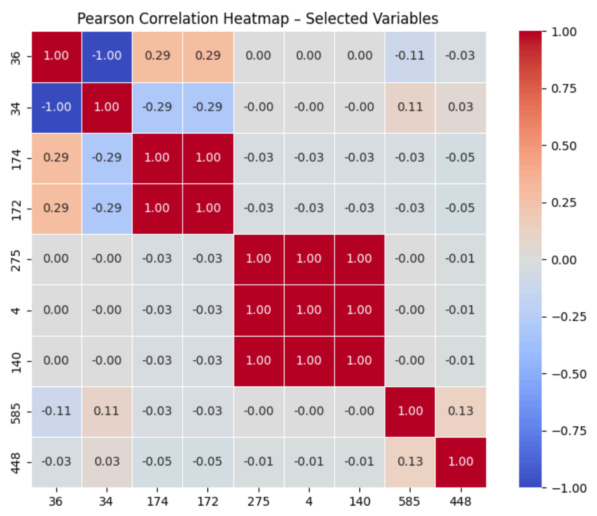


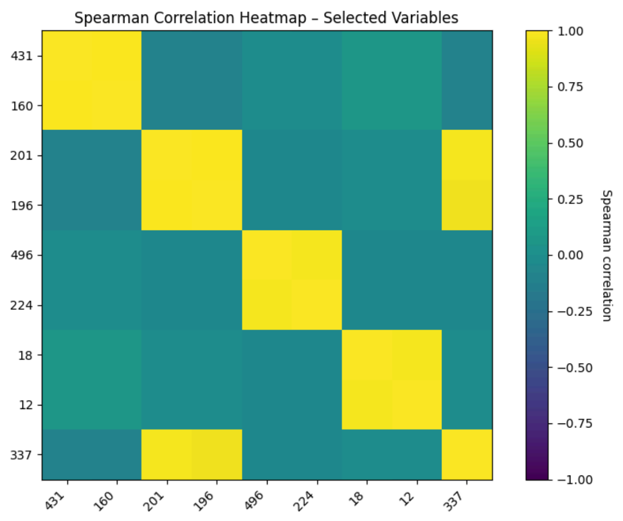


D. Dimensionality Reduction

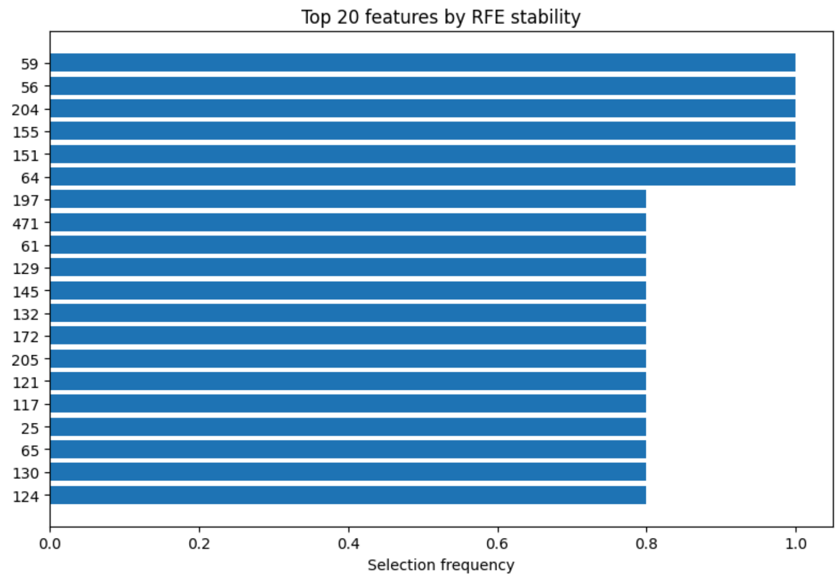
1. Feature Selection:

- Low Variance feature: 126 variables that have more than 90% the same values are dropped. The number of variables drops from 560 to 434.
- Multicollinearity: Apply correlation coefficient to remove variables that are highly correlated with other variables in the dataset. Variables:
    - Pearson correlation coefficient test is used for linear associations: Drop 164 vars.
    - Spearman's rank coefficient test is used for non-linear associations: Drop 5 vars.





- Recursive Feature Elimination:
  After handling missing values, removing near-constant variables, and pruning highly correlated sensors, we applied recursive feature elimination with cross-validation (RFECV) using a class-weighted logistic regression base estimator. All preprocessing (median imputation, z-scaling) and feature selection were encapsulated in a Pipeline and evaluated under a nested 5-fold stratified cross-validation scheme. In the inner loop, RFECV iteratively removed the least important features (10% per step), and selected the number of features that maximized average precision; the outer loop provided unbiased estimates of generalization performance. We additionally computed the selection frequency of each feature across outer folds to quantify stability and defined a 'stable' subset as features selected in at least 60% of folds.



2. ***Feature Extraction by Deep Autoencoder:***

a) Strategy and Rationale:

- I employed a Deep Autoencoder (DAE), a type of unsupervised Artificial Neural Network (ANN), for non-linear dimensionality reduction and robust feature extraction.
- The SECOM dataset contains 590 features, many of which are redundant, highly correlated, or noisy (as confirmed by the large number of missing values and the subsequent feature cleaning steps). Traditional linear methods (like PCA) often struggle with such complex, non-linear relationships. The DAE forces the model to learn the most essential, compressed representation of the data.
- The primary goal is to use the latent representation (Z)—the output of the encoder—as the new, compact, and highly informative set of features for subsequent modeling tasks like classification of 'Pass/Fail'.

b) *Model Architecture and Hyperparameters:*

- Encoder: Maps the high-dimensional input (590 → 512 → 128 → 64 → 10) down to the latent space.
- Decoder: Maps the latent space back up to the original input dimension (10 → 64 → 128 → 512 → 590).
- Activation: The ReLU function was used in the hidden layers to introduce non-linearity, enabling the model to capture complex relationships.

c) *Final Reconstruction MSE:*

Input Features	Reconstruction MSE	Conclusion on Information Density
558 features (Full Data) to 10 Z-latent variables	0.397584	High MSE. The model struggles to compress the noisy, redundant input.
20 features (Cleaned Data) to 10 Z-latent variables	0.059158	Very Low MSE. The model successfully captures the information in the cleaned input.

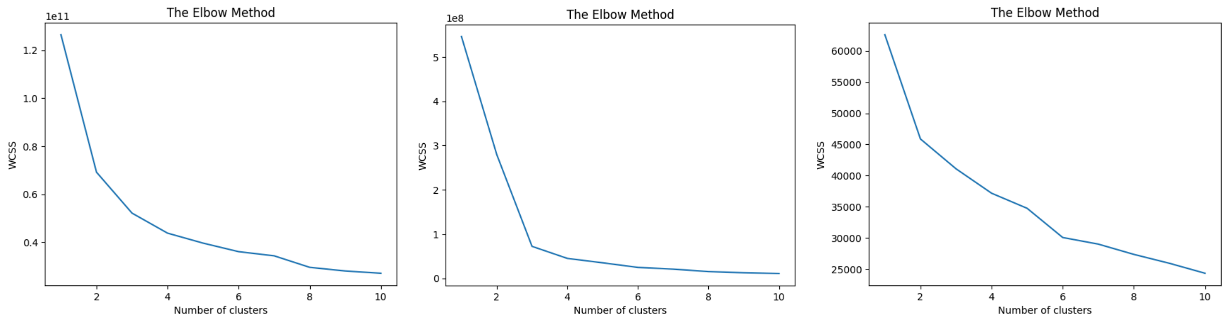
E. Clustering and Logistic Regression

1. ***Clustering:***

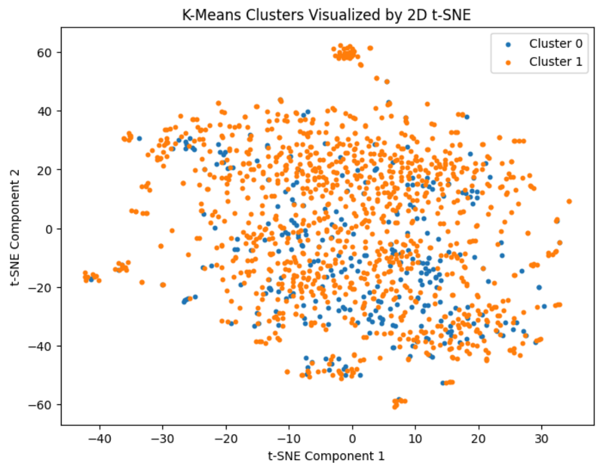
- As default clustering baselines I applied **K-means** and **DBSCAN** to three feature sets:
  (i) the full cleaned dataset (585 variables),
  (ii) the top 20 variables selected by RFE, and
  (iii) the 10-dimensional latent representation from the autoencoder.
- All features were standardized before clustering.
- Why Kmeans: Kmeans is a simple, widely used distance-based method that partitions all observations into k compact, roughly spherical clusters. It scales well to the SECOM dataset (≈1,400 observations, hundreds of continuous sensor variables) and provides a natural baseline for "globular" structure in the data. Because K-means assigns every point to a cluster, it is useful for this case when we want to observe two clusters for a true label.
- *Why DBSCAN*: DBSCAN is a density-based algorithm that can discover clusters of arbitrary shape and explicitly label low-density points as noise. This is attractive for

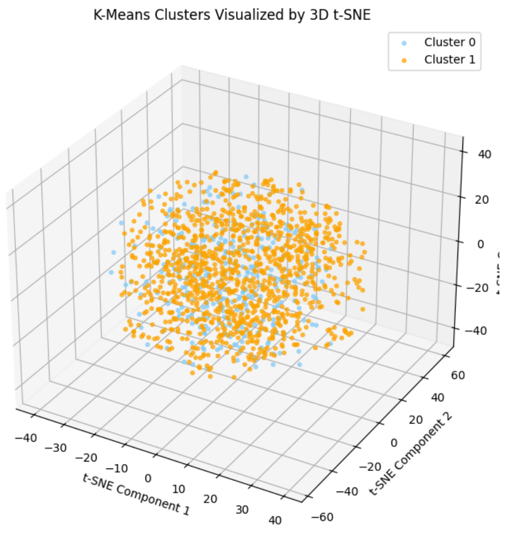
SECOM because the failure class is rare: only 99 of 1,393 units fail. In such imbalanced settings we expect failures to form small, dense pockets in feature space rather than large, balanced clusters. DBSCAN is therefore well suited to identify small abnormal regions and to separate them from the bulk of normal production.

- K-means and DBSCAN are trained in the original high-dimensional feature spaces, where direct visualization is impossible. To inspect the cluster structure, I project the data into 2D/3D using **t-SNE** and then color the points by their cluster labels.
- The Elbow method at three stages respectively:

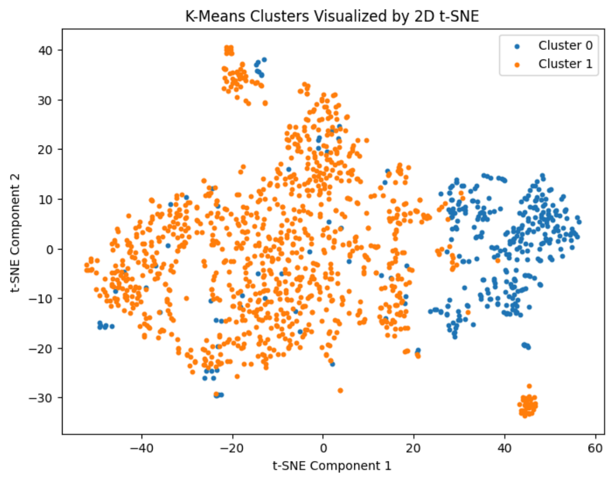


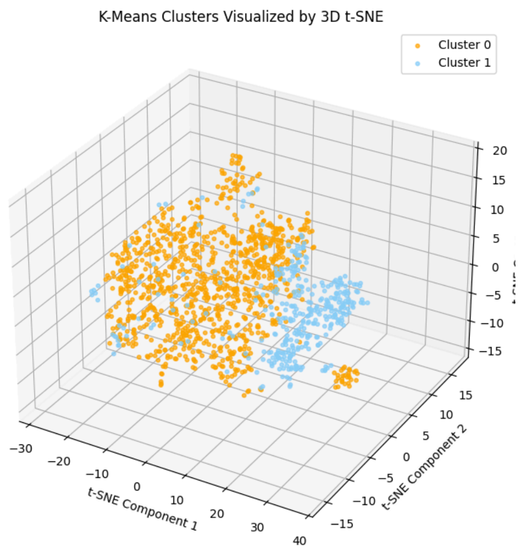
a) Combining Kmeans and t-SNE:
   Assign 2 clusters in Kmeans method:
   - Full cleaned dataset: 585



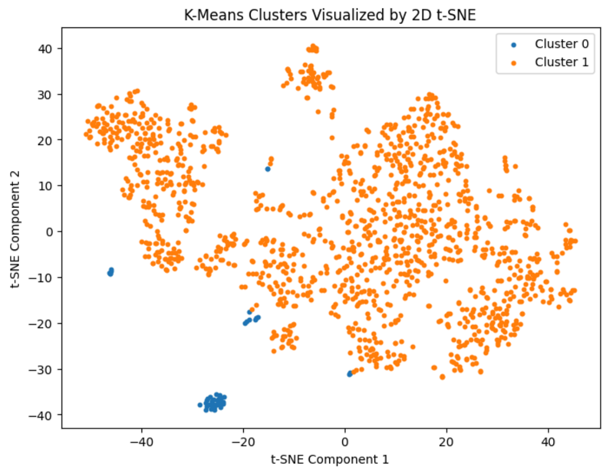


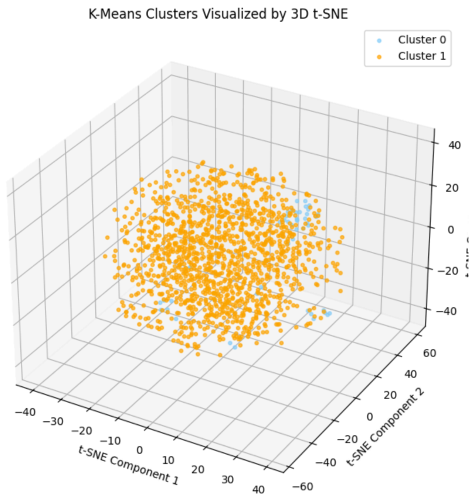
   - Top 20 features:





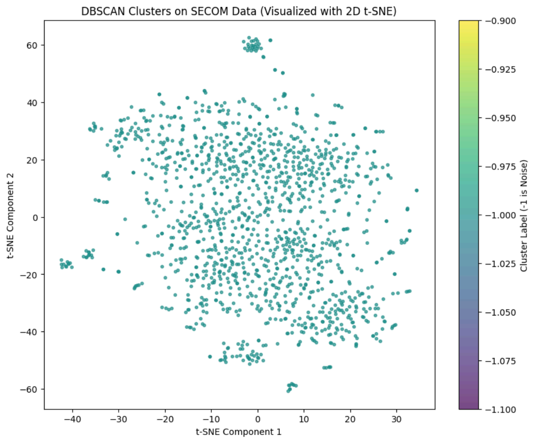
- 10 latent variables:



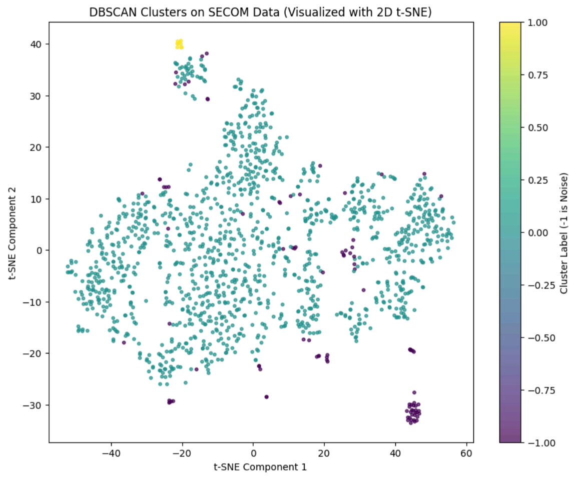


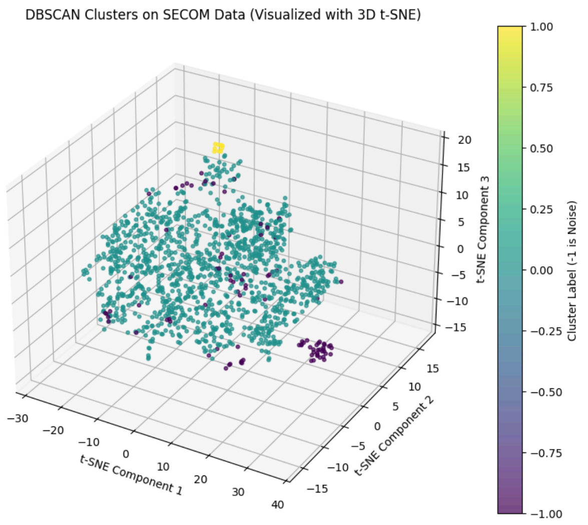
b) Combining DBSCAN and t-SNE:
DBSCAN again finds one large cluster and two small dense clusters, very similar to Stage 2, with the small clusters dominated by failure units. This stability across two different representations (20 raw features vs 10 latent features) strengthens the evidence that these minority clusters are not artifacts but real patterns in the data.

- Full cleaned dataset: 585

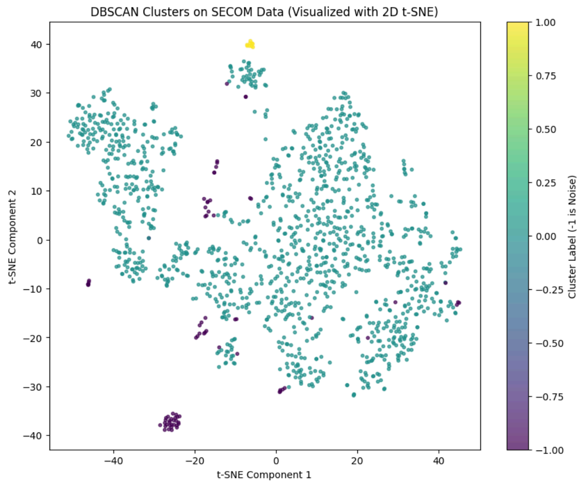


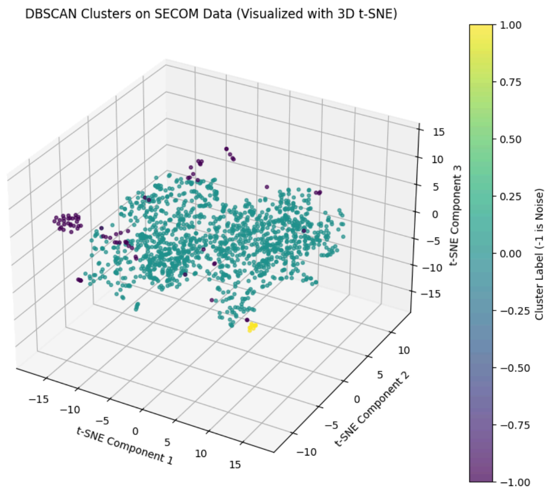
- Top 20 features:





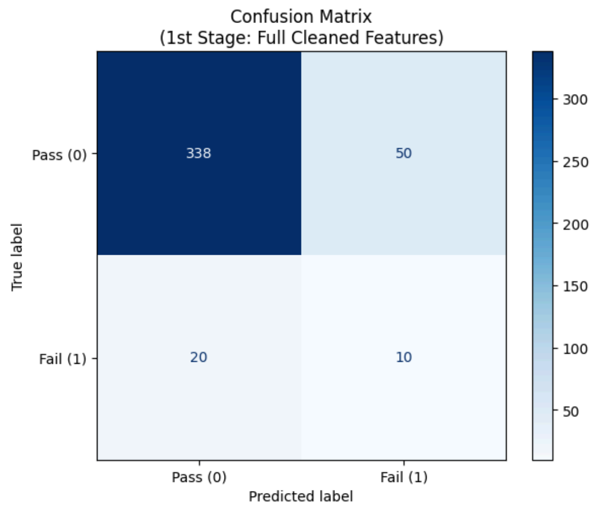
- 10 latent variables:

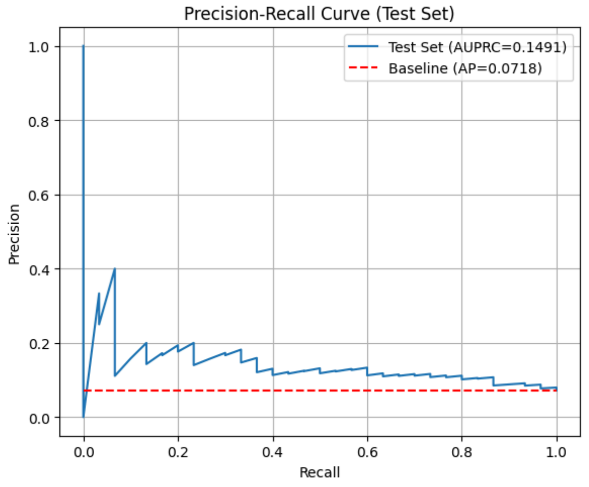


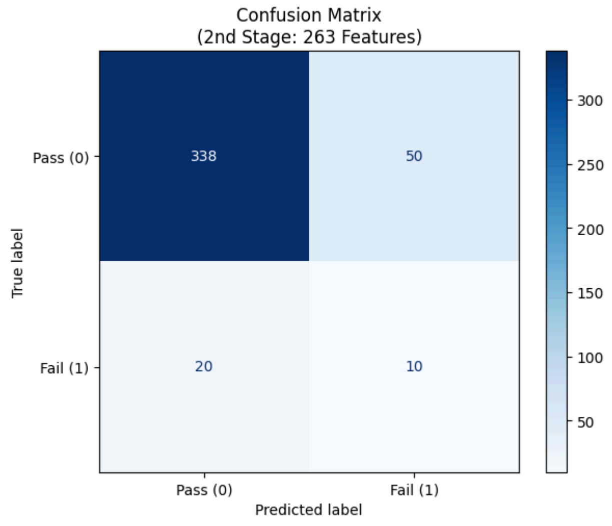


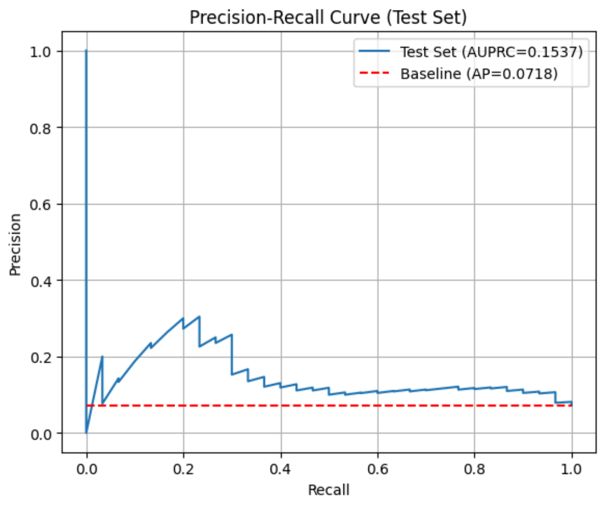
2. ***SMOTE Logistic Regression:***

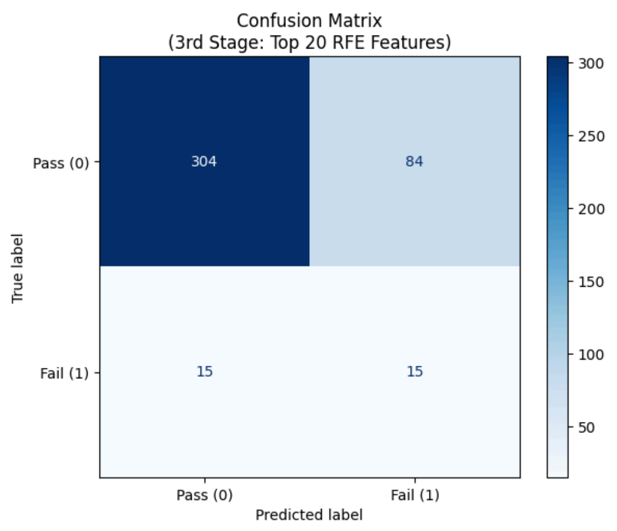
- To evaluate how informative the different feature representations are for predicting failures, I trained a SMOTE-logistic regression classifier in four stages:
    - Full cleaned dataset (585 features)
    - Further cleaned set (263 features)
    - Top 20 RFE features
    - 10 latent features from the autoencoder
- In each stage I used the same pipeline. First, the Pass/Fail label was recorded to a binary target y with Pass = 0 and Fail = 1. The data were then split into training (70%) and test (30%) sets using a stratified split, so that the strong class imbalance is preserved in the test set.

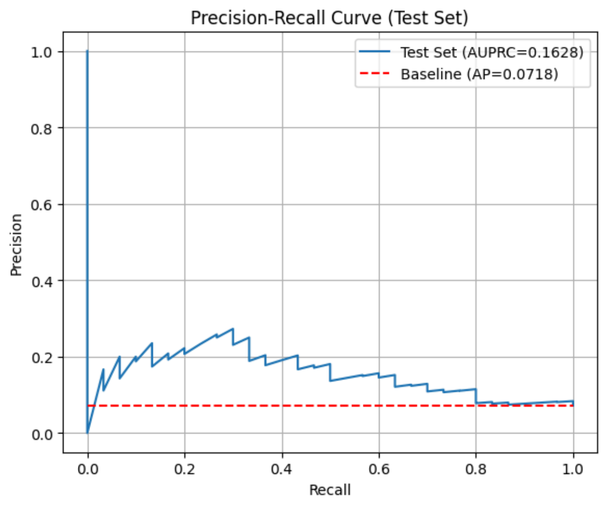


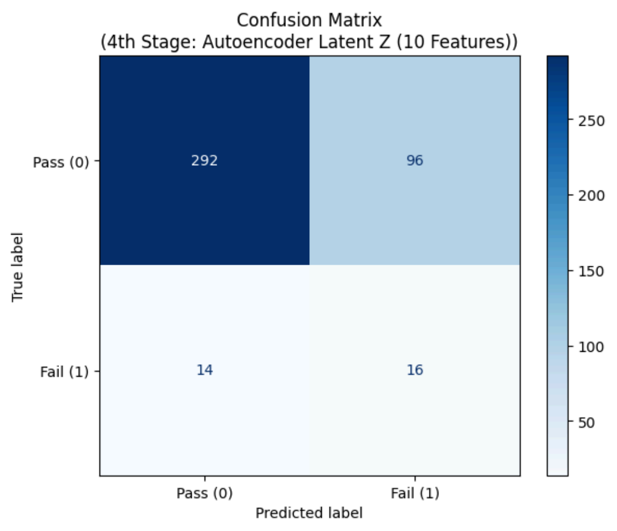


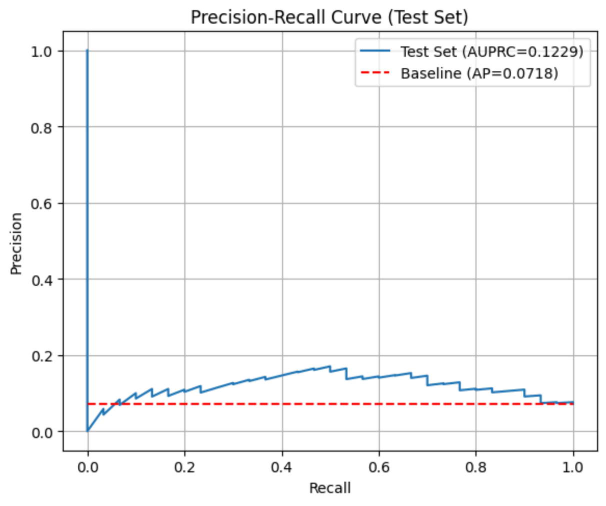












F. Comparison and Discussion

1. ***Clustering Distribution each stage of dataset:***

Stage	**True label**	**Full Dataset (585)**		**Top 20 rfe variables**		**10 latent variables**	
		KMeans	**DBSCAN**	**KMeans**	**DBSCAN**	**KMeans**	**DBSCAN**
Majority	**1294**	1101	1393	1035	1287	1347	1304
Minority	**99**	292	0	358	97+9=106	46	80+9=89

From the comparison table, we can see that the default clustering method **DBSCAN** in stages 2 and 3 produces cluster distributions that closely match the true Pass/Fail labels.

2. ***SMOTE Logistic Regression:*** Performance Metrics (Test Set)

Full Dataset (585)		**Clean Dataset (263)**		**Top 20 rfe variables**		**10 latent variables**	
F1-score	**AUPRC**	**F1-score**	**AUPRC**	**F1-score**	**AUPRC**	**F1-score**	**AUPRC**
0.2222	0.1491	0.2222	0.1537	0.2326	0.1628	0.2254	0.1229

- The F1-score is very similar for all models, ranging from about 0.22 to 0.23. The top 20 RFE variables give the highest F1-score (0.2326), followed closely by the 10 latent variables (0.2254), while the full and cleaned datasets have the same F1-score (0.2222).
- For AUPRC, performance improves slightly from the full dataset (0.1491) to the cleaned dataset (0.1537), and is highest for the top 20 RFE variables (0.1628). In contrast, the 10 latent variables stage shows the lowest AUPRC (0.1229)

G. Conclusion

- Across all stages of , the majority of observations form a single large cluster that corresponds to normal operating conditions and successful products (Pass). After careful feature selection and non-linear feature extraction, small, well-separated clusters appear that are strongly associated with the Fail label. This suggests that quality failures in this semiconductor process are driven by a few specific combinations of sensor readings rather than by gradual degradation across the whole operating range. In practice, the variables that contribute most to the 20-feature subset and the autoencoder latent space could be monitored to detect when the process enters one of these small "high-risk" regions, providing early warning for potential failures and helping engineers focus on a manageable subset of key sensors.
- SMOTE–logistic regression achieves only modest performance on this highly imbalanced problem: it can do better than random at identifying failures, but many failing units are still missed. The best results are obtained when training on the 20 RFE-selected variables, indicating that careful feature cleaning and selection are more beneficial for this linear classifier than the unsupervised autoencoder representation. This is consistent with the clustering results, which also show clearer separation between normal and abnormal operating modes after feature selection.

H. Future Improvement

- First, the clustering step could be refined by performing a more systematic search over the hyperparameters of K-means (number of clusters) and DBSCAN (ε, min_samples), and by comparing with additional algorithms such as Gaussian mixture models or HDBSCAN, which are designed for imbalanced, irregular densities.
- Addressing more techniques or strategies is necessary to improve both clustering quality and failure prediction.

I. Reference

- Lecture slides of STAT576 from Dr. Joon
- Dataset source: https://www.kaggle.com/datasets/paresh2047/uci-semcom/code
- https://www.kaggle.com/code/jengel/dealing-with-imbalance-autoencoder-sampling
- https://ieeexplore.ieee.org/document/9643228
- https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/