ISYE 6740 – Fall 2024
Final Report

Team Member Names: MinYoung Kim

Project Title: Predicting Hard Drive Failure Using S.M.A.R.T Statistics

Problem Statement

Backblaze, a company who provides high performance cloud storage solutions, released a dataset containing daily snapshots of hard drive S.M.A.R.T (Self-Monitoring, Analysis, and Reporting Technology) statistics on Kaggle [1]. S.M.A.R.T statistics is a monitoring system within hard drives that collect various data such as temperature or unexpected power loss count.

Although spinning disk are outdated compared to newer storage technology such as SSDs, the importance of having operational, performant storage remains unchanged. For example, storage failure at a hospital system may mean electronic health information (EHR) data is inaccessible for nurses, or that data is corrupt/lost and requires a downtime to restore from backup.

The goal of our project is to identify characteristics in our data that predict a hard drive's failure using various models including Gaussian Naïve Bayes (NB), logistic regression, and K-Nearest Neighbors (KNN).

Data Processing

This was the biggest challenge in the project. The dataset contains around 3.2M rows of data, with each row corresponding to a single hard drive's daily snapshot. A daily snapshot captures a hard drive's S.M.A.R.T statistics, including a "failure" column that is 0 if the hard drive is operational, and 1 if it was the hard drive's last operational day before failing. After failure, the hard drive's data is no longer tracked.

A single snapshot row has the following columns (95 columns total):

Column Name	Column Description
Date	Date of disk snapshot
Serial_Number	Number that uniquely identifies an single physical drive
Model	Model of a drive
Capacity_Bytes	Storage capacity of a drive in bytes
SMART_n_normalized	SMART statistic values normalized from 0 - 255, where n = 1 … 45
SMART_n_raw	Raw SMART statistic values from 0 - 255, where n = 1 … 45
Failure	0 if drive is operational, 1 if it was the last day before failing

The first step was to reduce the dimensionality of our data as much as possible, as using all 94 columns in our models poses the risk of overfitting. We performed the following:

- Drop all 45 SMART_n_raw columns, as we already have a normalized SMART_n_normalized column in our dataset
- Drop 'Date' column as a hard drive's failure shouldn't depend on a calendar date
- Drop 'Serial_Number' as it identifies a unique physical drive and it provides no predictive value
- Drop Capacity_Bytes as all drives seem to have the same value

This reduces our columns to just 'model', 'faillure', and 45 SMART_n_normalized columns.

Not all SMART_n_normalized columns are useful because the "drive manufacturer decides which SMART attributes to populate" and "different vendors use a different attribute for basically the same thing" [3]. Therefore, unpopulated columns with all NaN values were dropped. This fact also prohibits us from creating a single prediction model that can work with different drive models, as a S.M.A.R.T statistic for one drive may mean something different in another drive. Therefore, we created three smaller datasets filtered on the following drive

models to work on independently - ST4000DX000, WDC WD20EFRX. and WDC WD1600AAJS. We chose these specific models due to limitations detailed below.

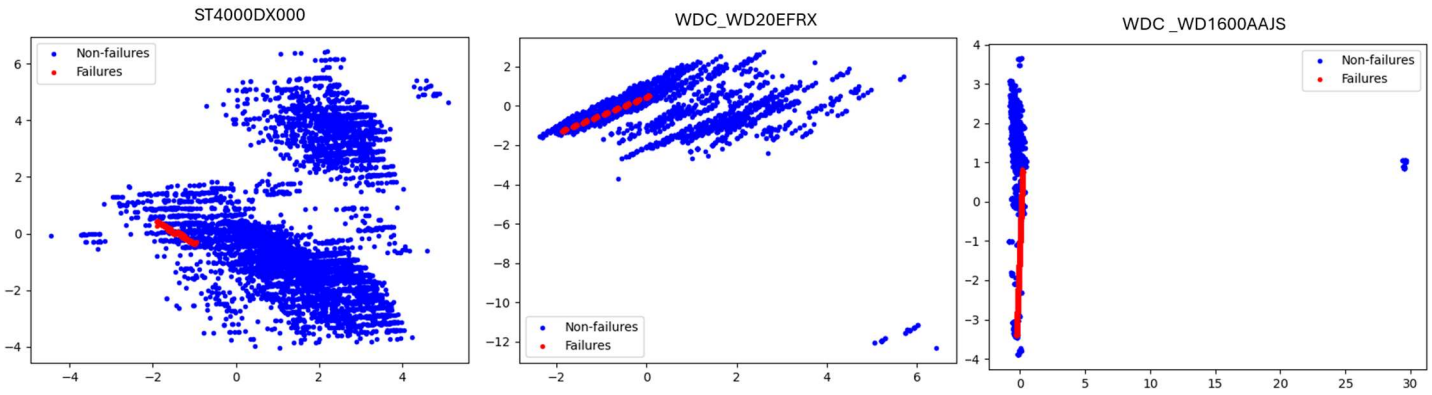
We also reduced the number of rows down from 3.2M to around 10,000 as we ran into memory issues when training our Gaussian NB model. This is because Gaussian NB has memory requirements of $O(n + n^2)$, where n is the number of data points [5]. Therefore, we chose drive models with a smaller number of rows - ST4000DX000 (10847 rows), WDC WD20EFRX (6871 rows), and WDC WD1600AAJS (5003 rows) – to create smaller datasets. Another reason these specific models were chosen is because the data is extremely imbalanced, and many models had just one observed class (no drive "failures").

The next problem is to address is the class imbalance, shown below.

	Number of Rows	
Hard Drive Model	'Failure' = 0	'Failure' = 1
ST4000DX000	10845	2
WDC WD20EFRX	6869	2
WDC WD1600AAJS	5001	2

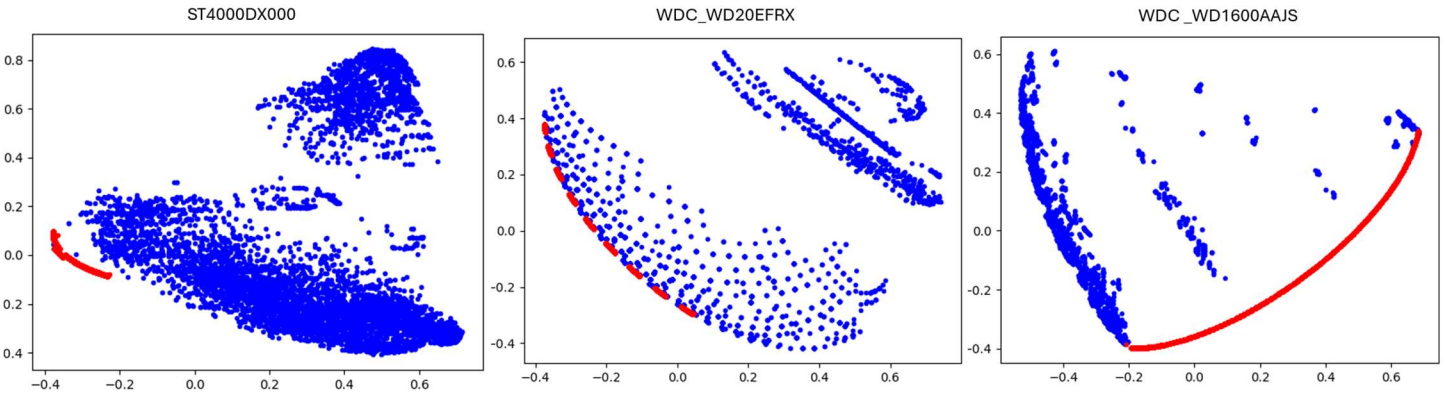
Many drive models had no failures recorded at all. Imbalanced data is a problem as our models will train mostly on the majority class, and won't learn enough about the minority class to accurately predict it. In our case, we have 2 observations of failures, so the only way to train and test our models is if we put 1 failure class in the train set and 1 failure class in the test set. Synthetic Minority Oversampling Technique (SMOTE) was used to oversample the minority class and undersample the majority class with a target of 50%/50% class distribution [1].

Lastly, we further reduced the dimensionality of our data via Principal Component Analysis (PCA). Linear PCA proved inadequate as plotting our transformed data along the first and second largest principal components showed our two classes were linearly inseparable.



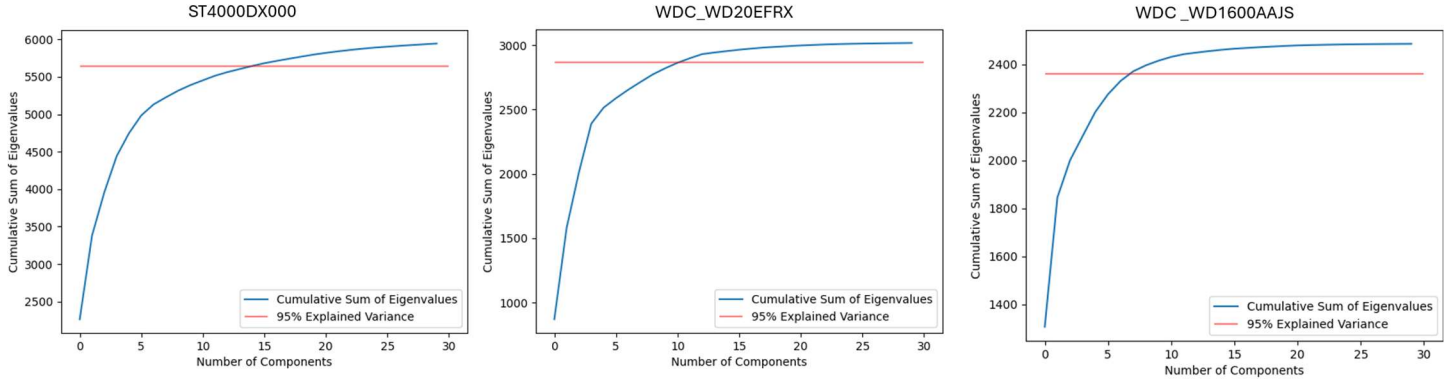
Above: Data plotted along first and second largest components of linear PCA

Linear PCA attempts to linearly transform our data to find principal components that maximize the variance in our data [2]. From the figures above, the two classes appear linearly inseparable. However, performing a non-linear transformation of our data using Gaussian Kernel PCA produces the below results, which look more separable.



Above: Data plotted along first and second largest components of Gaussian Kernel PCA. Failed drives in red, non-failed drives in blue

Two principal components were chosen in the above figures for visual purposes only. For training our model, we picked principal components with the n largest eigenvalues such that the sum of their eigenvalues was greater than 95% of the sum of all components' eigenvalues [4].



Above: Cumulative eigenvalues of principal components sorted in descending order

Using this method, we chose the following number of principal components for each model:

Hard Drive Model	Number of Principal Components
ST4000DX000	16
WDC WD20EFRX	12
WDC WD1600AAJS	8

Methodology

After standardizing, randomly shuffling, and splitting our dataset with a 75%/25% split into training and testing set, we trained three different models – Gaussian NB, Logistic Regression, and KNN. For each model, we chose the Receiving Operator Curve Area Under Curve (ROC AUC) as the metric to assess and compare the model's performance on the test set.

<u>Gaussian NB</u>

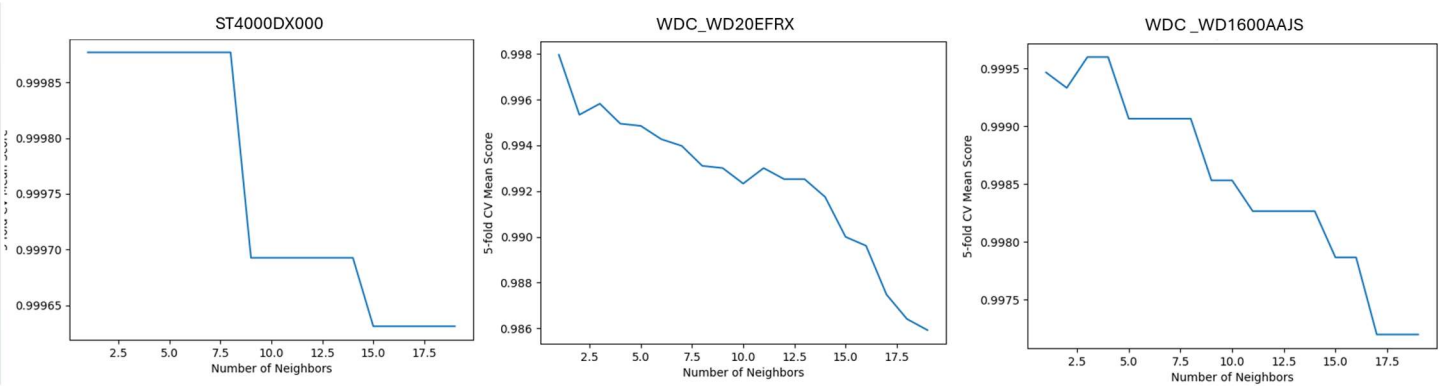
Default parameters from scikit-learn was used as our model. This means the prior probabilities were auto-calculated based on the data.

Logistic Regression

Logistic regression was used to test linear separation of our data. L2-norm was used in regularization in order to prevent risk of overfitting.

K-Nearest Neighbor (KNN)

KNN was tested to account for complex, dynamic decision boundaries, as it is a non-parametric model unlike Gaussian NB and logistic regression. The parameter to be tuned is the number of neighbors to use when predicting the class of a data point. The following number of neighbors n were chosen by performing 5-fold cross validation from $n = 1 \ldots 20$ and picking the number with highest accuracy. The below plot shows the cross validation accuracy score for different n. In cases where multiple number of neighbors had the same validation score, the largest n was chosen as a larger n means more data points are used to predict classes and thus less likely to overfit.



Above: 5-fold CV score for different KNN number of neighbors

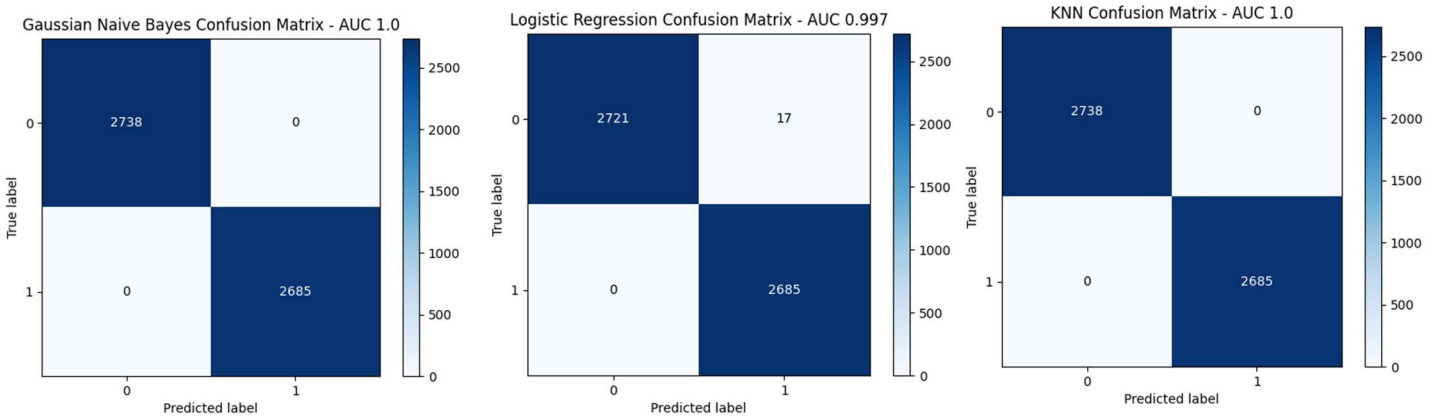
Hard Drive Model	Number of Neighbors
ST4000DX000	8
WDC WD20EFRX	1
WDC WD1600AAJS	4

Evaluation and Final Results

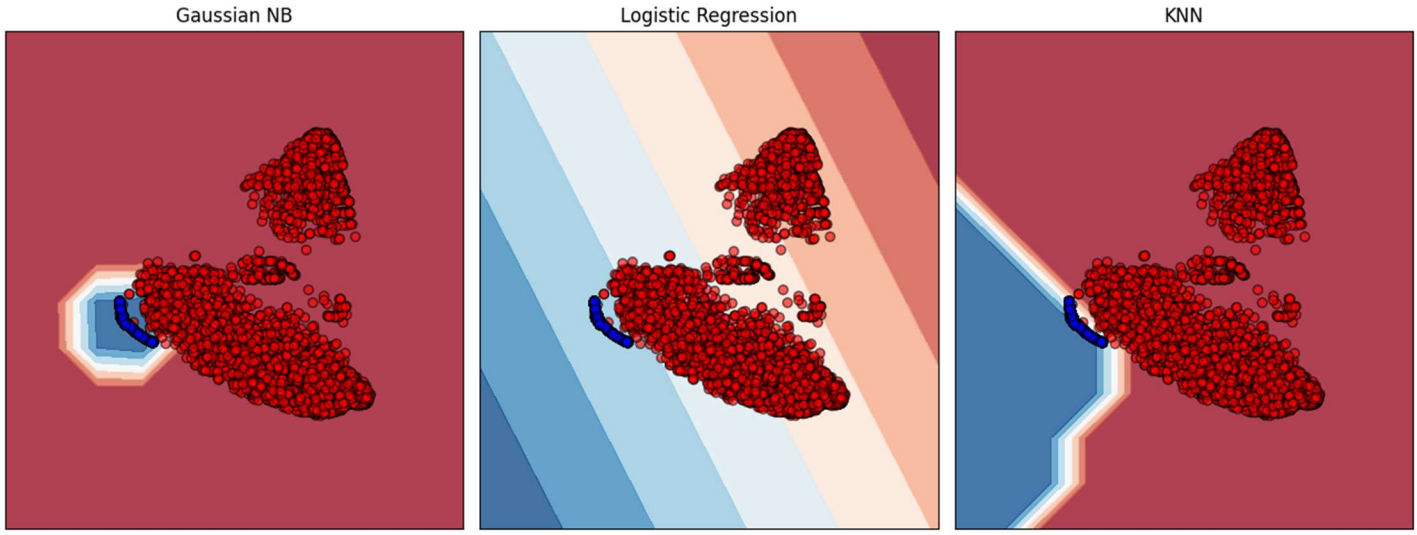
Confusion matrices and ROC AUC scores against the test dataset are shown below. For each model, the decision boundaries were also plotted against the training dataset. The two largest principal components were used for visualization purposes.

ST4000DX000

Both the Gaussian NB and KNN had very good ROC AUC scores. Logistic regression model had a slightly worse score. This is likely due to the data not being perfectly linearly separable (curved patterns in below plots).



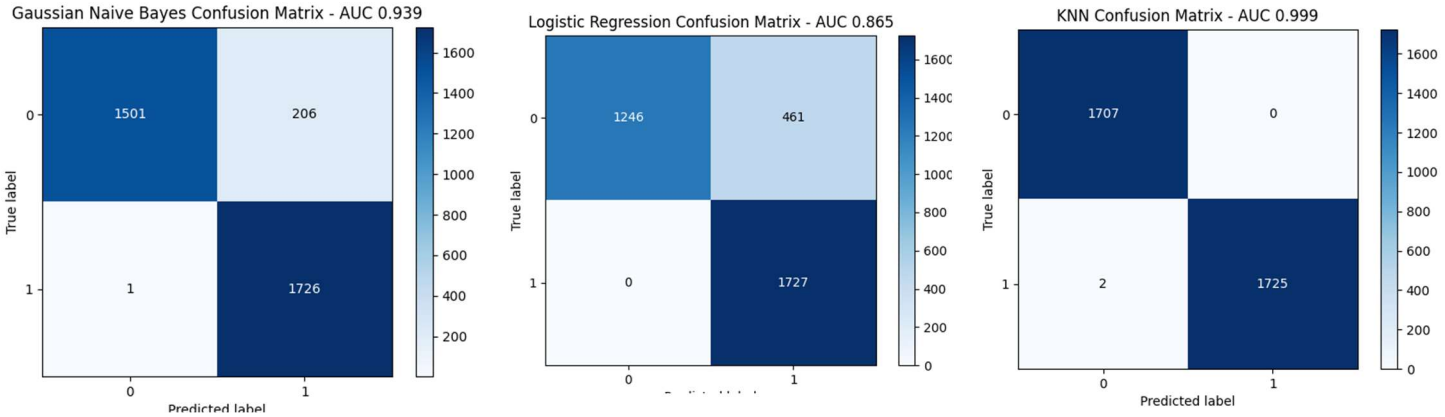
Above: Confusion matrix for each model with ROC AUC score in title



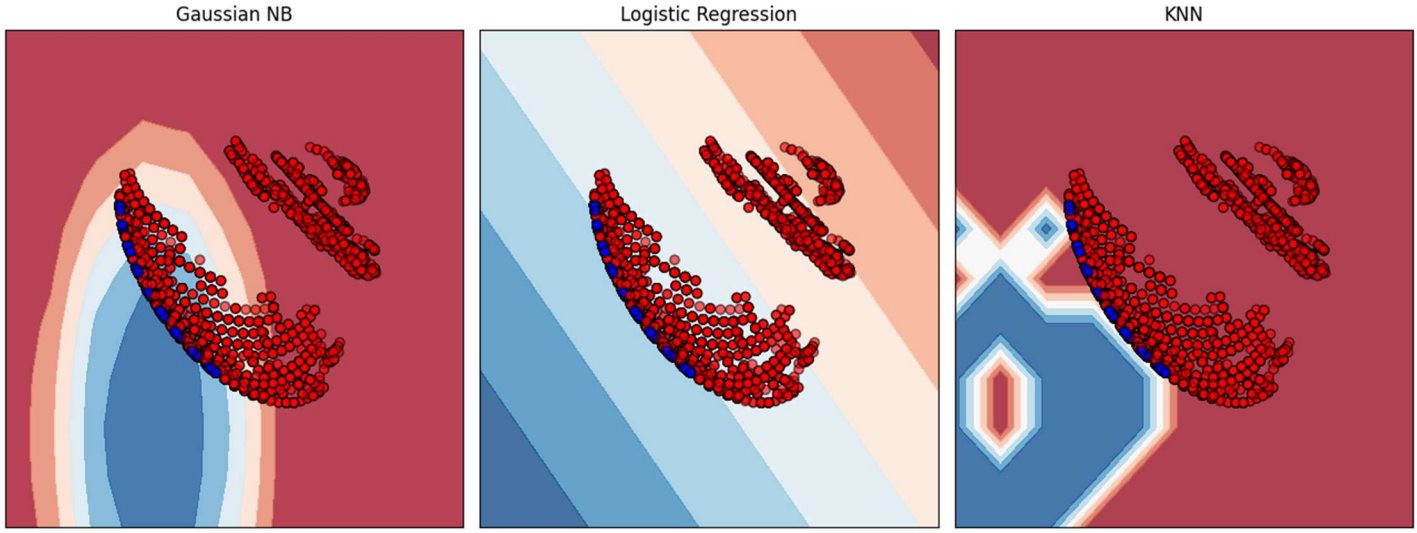
Above: Decision boundaries plotted against train dataset, with drive failure in blue, red otherwise.

<u>WDC_WD20EFRX</u>

Gaussian NB had a higher ROC AUC score than the logistic regression model. However, the ROC AUC scores for the two models are less than ST4000DX000 likely because in the below 2D-plot, the failure classes are more closely mixed with the non-failure classes in our train dataset. The failure classes are harder to separate out. KNN performed much better than the other two models, likely because number of neighbors used was 1. Even though different classes were closely mixed, using one neighbor to predict classes effectively drew a “finer” distinction within that mixture.



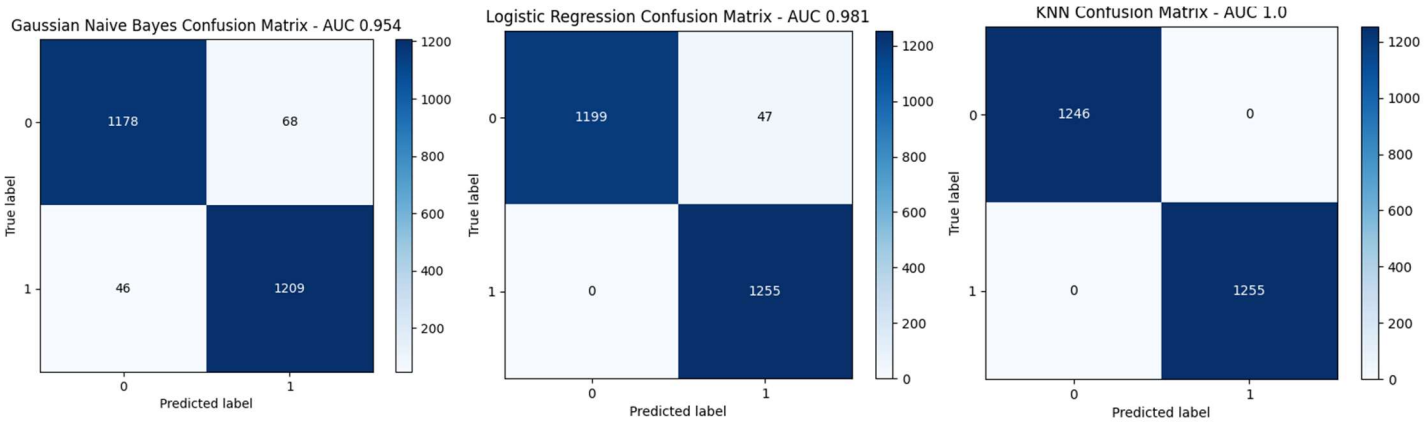
Above: Confusion matrix for each model with ROC AUC score in title



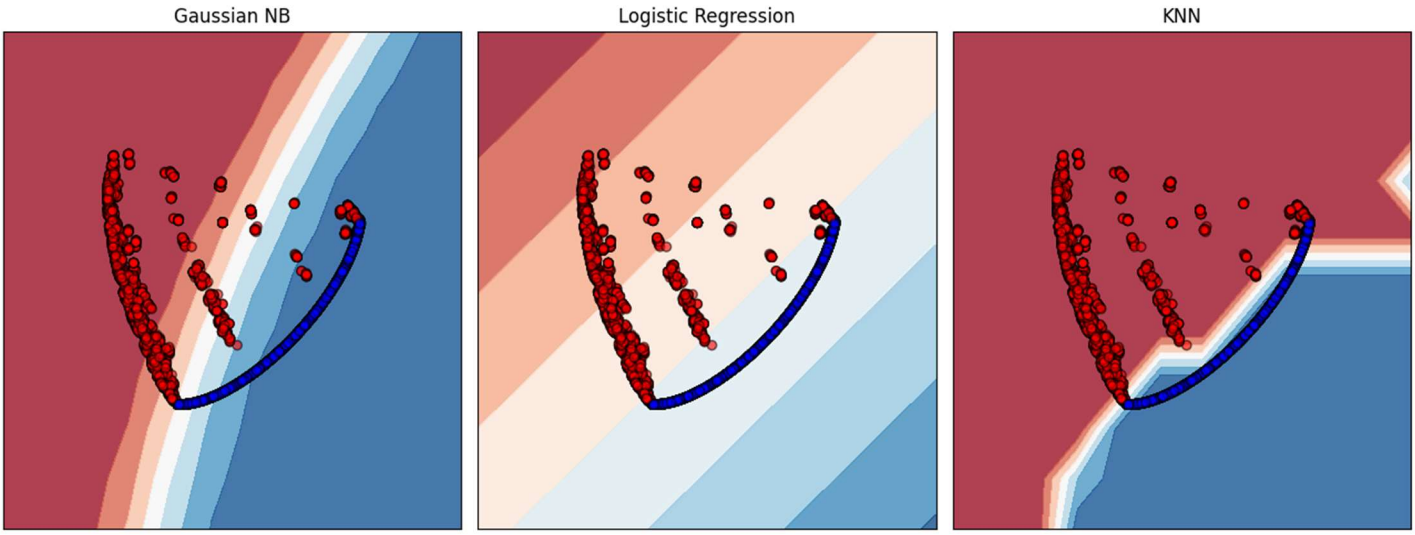
Above: Decision boundaries plotted against train dataset, with drive failure in blue, red otherwise.

<u>WDC WD1600AAJS</u>

Logistic regression performed better than Gaussian NB unlike other drive models. The failed drives appear relatively linearly separable, potentially explaining this result. KNN performed very well – the blue points appear much more densely packed with each other than red points, and since KNN uses a distance metric for prediction, it is able to predict with high accuracy.



Above: Confusion matrix for each model with ROC AUC score in title



Above: Decision boundaries plotted against train dataset, with drive failure in blue, red otherwise.

Future Considerations

Here are some future considerations to improve the goal of this project based on limitations we faced.

- Train models on hardware with more memory/CPU
    - Memory limitations stopped us from working with datasets larger than around 10,000 rows
    - Initially planned to perform non-linear dimension reduction via ISOMAP but faced memory limitations
- Contextual information on S.M.A.R.T statistics
    - Because different models can have different meanings and values populated for the same S.M.A.R.T statistic, it would be good to know the specifics for a given dataset. Subject matter experts can then weigh in on which metrics may be more significant.
- Imbalanced dataset
    - More observations of failure class to improve both our train and test dataset

References

[1]Backblaze. “Hard Drive Test Data.” *Kaggle*, 5 Nov. 2016, www.kaggle.com/datasets/backblaze/hard-drive-test-data/code.
[2] Chawla, N. V., et al. “Smote: Synthetic Minority over-Sampling Technique.” *arXiv.Org*, 9 June 2011, arxiv.org/abs/1106.1813.
[3] *CS 429: Computer Vision Lecture 2 PCA_handout*, www.cs.princeton.edu/courses/archive/fall08/cos429/CourseMaterials/lecture2/lecture1.pdf. Accessed 8 Dec. 2024.
[4] Klein, Andy. “Making Sense of SSD Smart Stats.” *Backblaze Blog | Cloud Storage & Cloud Backup*, 12 Mar. 2024, www.backblaze.com/blog/making-sense-of-ssd-smart-stats.
[5] *Lesson 11: Principal Components Analysis (PCA)*, online.stat.psu.edu/stat505/book/export/html/670. Accessed 7 Dec. 2024.
[6] Xie, Yao. *ISYE 6740 Module 6 Density Estimation*, p. 33.