# CREDIT CARD FRAUD DETECTION

*Colin Blake, Kevin Wang, Eric Wenner*

## ABSTRACT

With the rise of online commerce and use of credit cards for day-to-day transactions, credit card fraud has become a significant issue for both consumers and card-issuing banks. As data is collected about these transactions, applying machine learning techniques to the problem of fraud detection has become an area of interest to researchers. Due to how these data are usually collected, credit card transaction datasets are often imbalanced, incomplete, and/or contain numerous features which may not contribute meaningfully to the classification. This paper aims to explore methods for addressing these issues with preprocessing techniques such as regression imputation, over-sampling, and feature reduction. With the application of the developed pipeline, a semi-successful model is created.

## 1. INTRODUCTION

Credit cards have become a part of everyday life, and potential fraudsters have adapted and found ways to take advantage of this. Based on the reports received by the Federal Trade Commission, the credit card is the most common form of payment method involved in fraudulent transactions, with 88,354 reports and $181M lost in 2021 [1].

Datasets available for analysis of this problem often involve some combination of class imbalance, missing values, and large feature space which create interesting challenges for the development of a machine learning model.

### 1.1 Incomplete Data

As a dataset becomes larger and larger, it becomes more and more likely that a few data points will fail to be collected (such as due to sensor error, respondents not filling out all survey fields, etc. depending on the data collection method). Handling these missing data is a necessary part of preprocessing, as many model types can't tolerate missing values in the input X.

There are generally three ways in which data can be missing from a dataset: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). When data is said to be missing completely at random, the missing values do not depend on any aspect of the dataset. When data is said to be missing at random, missing values depend only on the *observable* data. When data is said to be missing not at random, the missing values depend both on the observable values and the other missing values. [2]

The simplest method for handling missing data is deletion: simply remove any sample that has one or more features missing. This works when missing data is MCAR, but not when missing data is MAR or MNAR. This method also suffers for small datasets, where deleting data can significantly reduce the amount of data available for training. [2]

Another method which can be used when missing data is MCAR is replacing a missing value with the mean for that feature. Like deletion, this method is only unbiased when data is MCAR [3], but it doesn't require one to dispose of otherwise potentially useful data from the rest of that sample.

When missing data is MAR, it is possible to predict values for missing data based on the observable data. Data imputation algorithms typically assume that missing values are MAR. A common technique for this is regression imputation, in which a regression model is trained on all of the complete samples, and then missing values are calculated using this model (Emmanuel, Maupong, Mpoeleng, et al. 2021) [2]. SKLearn's implementation of this works in a round-robin fashion, where a new regression model is generated to predict missing values for each feature, trained with the data from all the other features.

### 1.2 Imbalanced Data

In some binary classification problems, a case can arise where there is a clear majority and minority class, and the difference in proportion of the dataset they make up is significant. If one were to train a model to classify this data, the model might optimize its classification accuracy or loss function simply by always predicting the majority class - with a large enough class imbalance this would be an appealing strategy simply to optimize classification accuracy. This isn't desirable, as in these imbalanced classification tasks we want to capture the patterns which lead to this minority class being present, and be able to predict it.

A common technique for addressing this issue is *over-sampling*, where synthetic samples from the minority class are generated based on the real data until a specified class ratio is reached (often 1:1). This is often done with SMOTE, or a variation of it.

### 1.3 Large Feature Space

In high-dimensional data, it can be useful to reduce the number of features to only the ones with a meaningful contribution to the classification output. This is helpful to reduce training time, simplify future data collection, and reduce the amount of data needed for training.

In addition to feature selection, feature *reduction* can be used for a similar purpose, but with the property of not preserving the original feature meanings. Principal Component Analysis (PCA) is commonly used for this task. The PCA algorithm creates a new basis for the input X, with the first feature in this new space being in the direction of maximum variance, the second feature being in the direction of second-most variance (while perpendicular to the first feature), and so on. The algorithm also provides the proportion of the total variance contributed by each feature, meaning that the user can select the features with the largest variance contributions, and they can select how many features they want to keep in order to preserve some proportion of the original variance in the data.

## 2. RELATION TO PREVIOUS WORK

There has been much research done in the field of dealing with the issues of class imbalance, missing data, and large feature space. In particular, the present work takes inspiration from a paper by Chen, Dewi, Huang, et al. [4] which describes a process for assessing the performance of several different feature reduction techniques using several different classifiers on several different datasets. This work was important for us to see how exactly we could compare these feature elimination techniques and assess the performance of machine learning models in this context.

The present work, however, has a much greater focus on a particular application (fraud detection) while the focus in Chen, Dewi, Huang et al.'s paper [4] is on assessing performance of models. As a consequence, the present work is more exploratory and less structured than the work presented in [4].

## 3. METHODS

The dataset chosen can be found on Kaggle [4]. This dataset contains 121 features and a single target class denoting whether the transaction is fraudulent. Values are missing from a large proportion of the features in this dataset.

In creating a model, a pipeline is employed (shown in Figure 1) which is meant to address potential performance issues which may result from the choice of an imbalanced and incomplete dataset. This includes deletion and imputation to address the incompleteness of the dataset, oversampling to address imbalance in the target class, feature selection to reduce the risk of overfitting and to improve training time, and model selection to achieve a good precision-recall curve and F1 score.

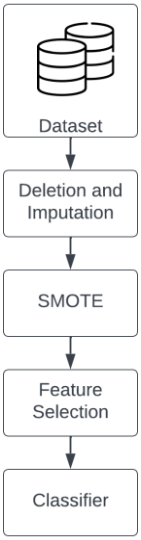


Figure 1: Pipeline used for Credit Card Fraud Classification

For the first stage in the pipeline, deletion and imputation, a combination of techniques is employed. First, if any feature contains more than 20% missing values, the feature is removed from consideration. For numeric features, missing values are assumed to be MAR, permitting the use of imputation. A form of regression imputation is employed, in which a regression model is fitted to each feature based on all other features, and missing values are predicted in a round-robin fashion. This is done using the SKLearn IterativeImputer. Missing values in the categorical features are imputed with the mean value for that feature.

For the next stage, SMOTENC (a variant of SMOTE which can support both numeric and categorical features) is employed in order to alleviate the effects of class imbalance.

Then, several methods for feature reduction are applied with the goal of reducing the risk of overfitting and permitting easier data collection going forward. With each method of feature reduction, several classifier types are employed with the goal of finding a combination which yields the best precision-recall curve and F1 score. Special emphasis was placed on the Random Forest Classifier, a literature review suggested that this model would work well with feature reduction [4].

A second dataset was used to test how the classifiers work on different datasets. This dataset includes 31 unidentified columns due to privacy reasons but there is still a substantial number of data points to work with.

## 4. EXPERIMENTS

After performing our imputation and oversampling techniques, examination of our 121 feature dataset showed

that not all features may be important in the target class. Feature reduction techniques can help improve a model's performance by eliminating irrelevant or redundant features, and focusing on the most informative features. In our case, we wanted to reduce the number of features in our dataset in order to reduce any noise or redundancy, prevent overfitting, and increase efficiency and scalability while still maintaining or improving the model's performance.

To accomplish this, we applied three feature reduction techniques to our dataset: selectkbest, recursive feature elimination, and principal component analysis (PCA).

### 4.1 Finding a Baseline

Before applying feature reduction techniques to our dataset, we first established a baseline performance by training the three classifiers (Random Forest, Decision Tree, and XGBoost) on the dataset after imputation, without any feature reduction or oversampling. This allowed us to establish a certain level of performance and compare the performance of the classifiers before and after applying feature reduction techniques.

After applying mode and regression imputation, the categorial features were given a label encoder and the models were trained accordingly. Figures 4.1-4.3 shown below shows the performance of our three classifiers on the imputed data.

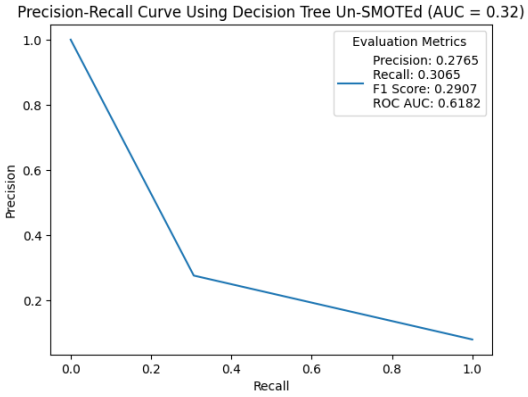


Figure 4.1: Decision Tree Performance on Imputed Data

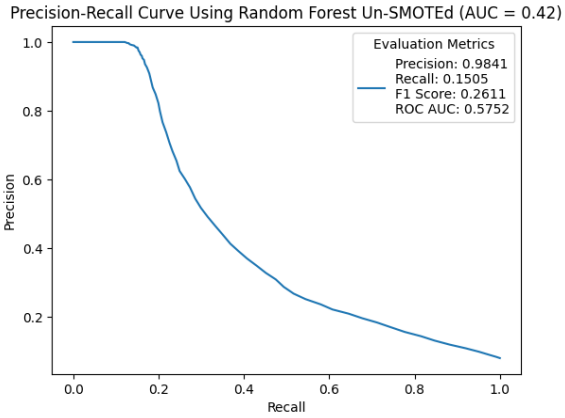


Figure 4.2: Random Forest Performance on Imputed Data

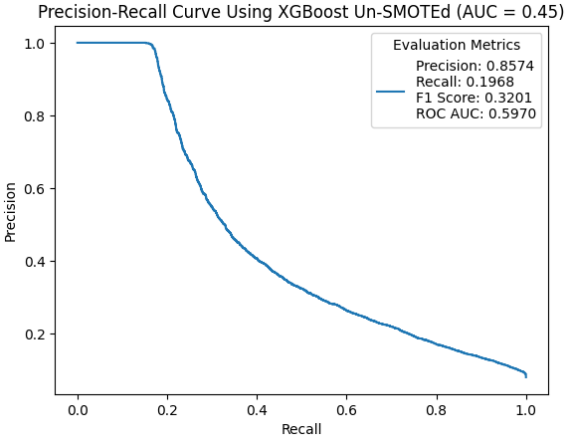


Figure 4.3: XGBoost on Imputed Data

### 4.2 SMOTE-NC Performance

Now we must perform an oversampling technique known as SMOTE-NC onto our training set in order to oversample the minority class, which should help our models receive more data and learn about the minority class better. We chose SMOTE-NC because this method can handle numerical and categorical features. After using SMOTE-NC and training our model using the oversampled training data, we still test using our un-smoted test set to ensure our models can handle the original unbalanced dataset. Our results are seen in figures 4.4-4.6.

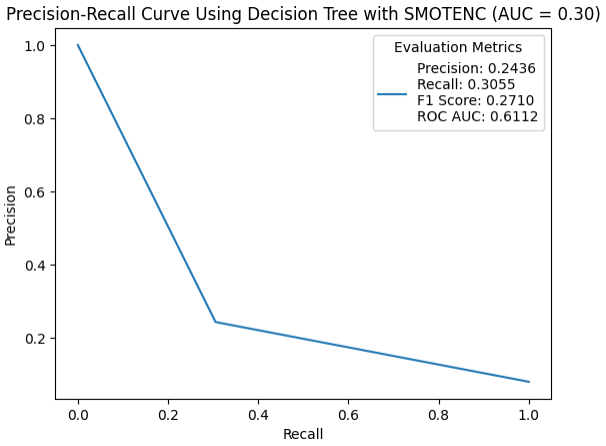


Figure 4.4: Decision Tree using SMOTE

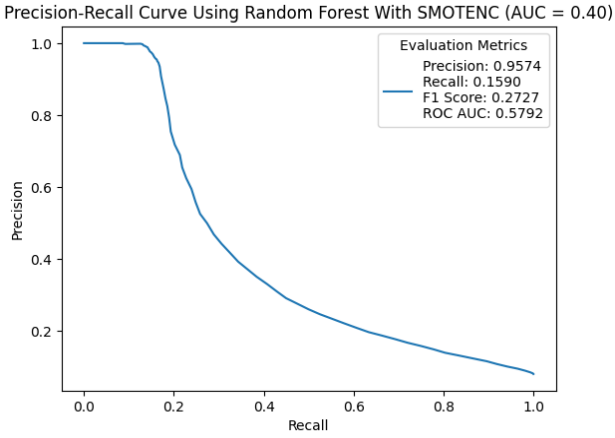


Figure 4.5: Random Forest using SMOTE

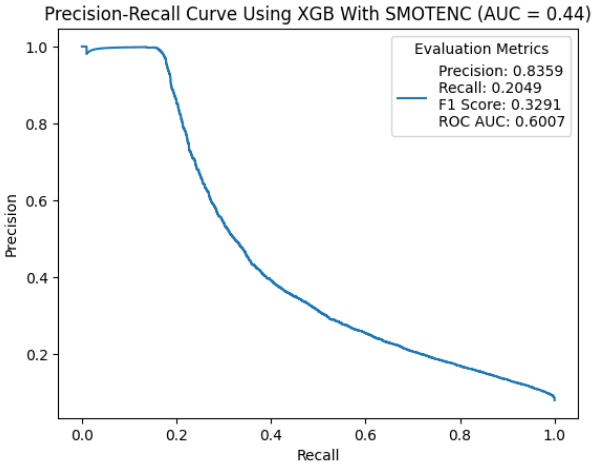


Figure 4.6: XGBoost using SMOTE

After analyzing my results, we can see that SMOTE-NC has led to a slight improvement in the F1 score for the Random Forest and XGBoost classifiers, indicating that the algorithm was able to somewhat address the class imbalance issue in my dataset. However, it is surprising to both find that the Decision Tree classifier did not show any improvement in performance, and that the improvement was so small after applying SMOTE-NC.

Our results suggest that the effectiveness of SMOTE-NC in improving classification performance may depend on the specific characteristics of the dataset and the underlying assumptions of the classifiers. It is possible that Decision Trees are less sensitive to class imbalance compared to other classifiers like Random Forest and XGBoost, and thus may not benefit from the oversampling of the minority class that SMOTE-NC provides. Our results show that other factors such as the quality and relevance of the features, or the complexity of the classification problem, may also play a role in determining the effectiveness of SMOTE-NC.

### 4.3 Feature Reduction

Next, we perform feature reduction on our large dataset. Our goal is to improve, or at least maintain, the predictive accuracy of our models while reducing the computational complexity and preventing overfitting. We chose to reduce the features down to 10.

Our first method of feature reduction is SelectKBest, which is a simple method that selects the K most important features based on their statistical significance. SelectKBest works by evaluating each feature individually, and selecting the K features with the highest scores.

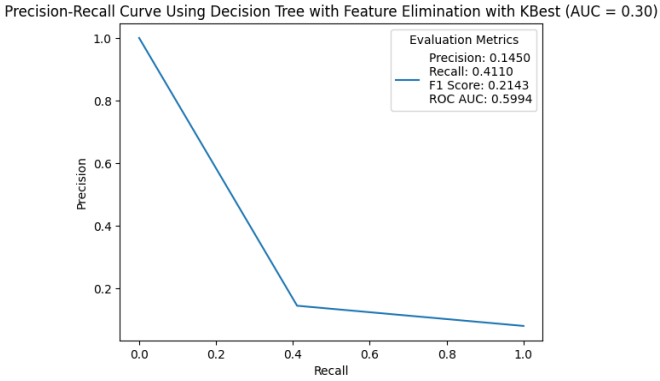


Figure 4.7: Decision Tree w/ KBest

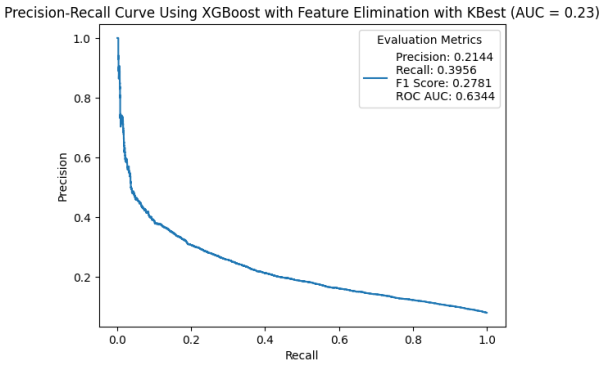


Figure 4.8 XGBoost w/KBest

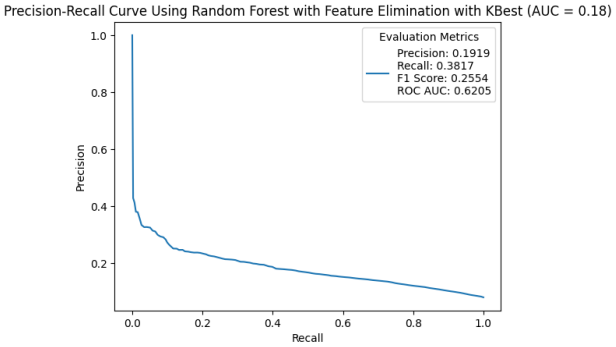


Figure 4.9: Random Forest w/KBest

From examining figures 4, we can observe that the overall shape of the curves still remain similar to the ones shown in figures 3, which is not always the case when applying feature reduction. Although the F1 scores for all models did decrease, the reduction was only marginal. It's worth noting that the ROC AUC scores for the XGBoost and Random Forest classifiers actually increased slightly after feature reduction.

These results suggest that the KBest feature reduction technique we applied was effective in removing redundant or irrelevant features from the models, without significantly sacrificing their overall predictive performance. The small increase in ROC AUC score for some models is particularly promising, as it suggests that feature reduction may help improve the models' ability to differentiate between positive and negative samples.

Another method of feature reduction we performed is Recursive Feature Elimination (RFE), which is an iterative process that selects the most important features based on their contribution to the model's predictive accuracy. RFE works by training the model on subsets of the input features, and ranking the features based on their importance. The least important features are then eliminated, and the process is repeated until the desired number of features is reached.

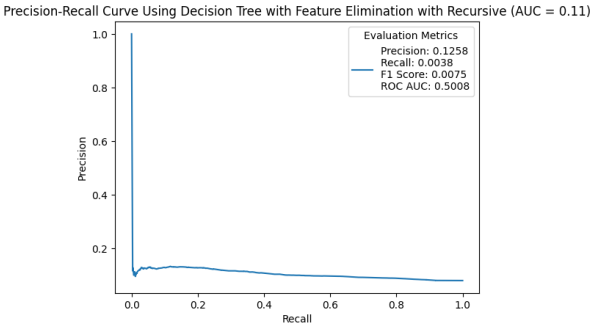


Figure 4.10: Decision Tree w/RFE

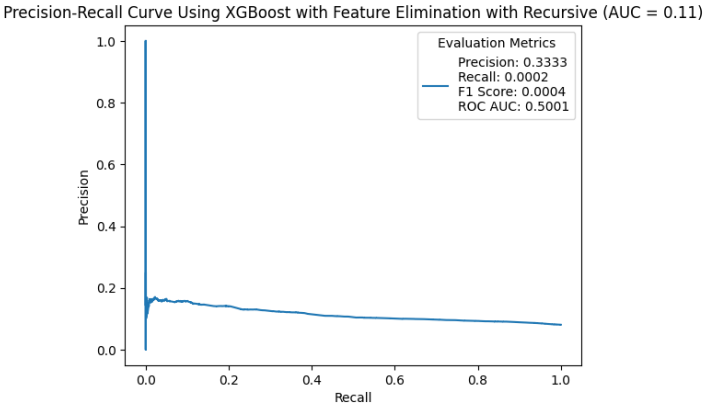


Figure 4.11: XGBoost w/RFE

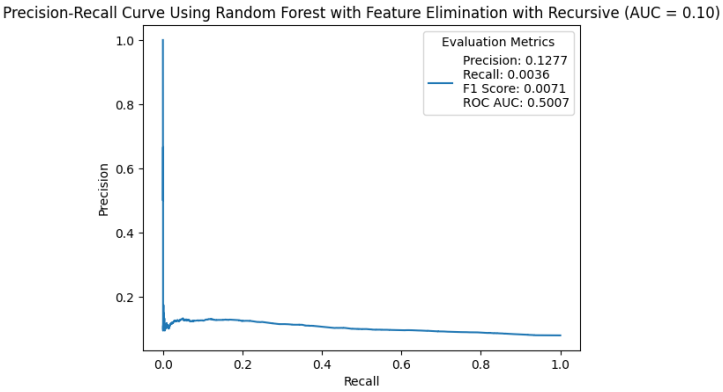


Figure 4.12: Random Forest w/RFE

The new graph clearly demonstrates that Recursive Feature Elimination did not perform well on our dataset, as evidenced by the altered shape of the curve. Our results show that the ROC AUC score for all three models after applying Recursive Feature Elimination is essentially equivalent to random guessing. This suggests that this particular feature reduction technique was not effective in retaining the most informative features and removing the least important ones, which negatively impacted the performance of our models. We finally tested one last feature reduction method, PCA.

One of the most popular methods of feature reduction is Principal Component Analysis (PCA), which is a mathematical technique that can be used to transform a set of correlated features into a set of uncorrelated features, while retaining the most important information about the data. PCA works by identifying the directions of maximum variance in the data, and projecting the data onto a lower-dimensional space that captures the most important features.

Precision: 0.28807303
Recall: 0.43467693
F1: 0.34650628

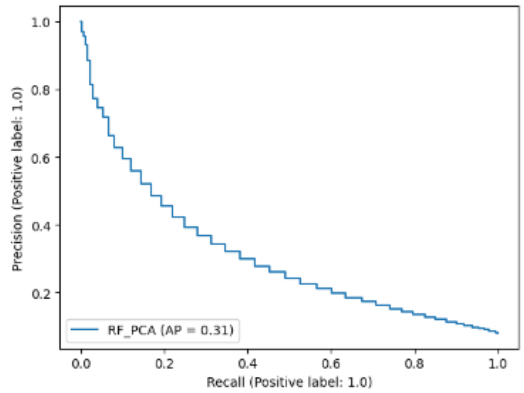


Figure 4.13: PCA w/20 components using Random Forests

Precision: 0.30038358
Recall: 0.48379583
F1: 0.37064049

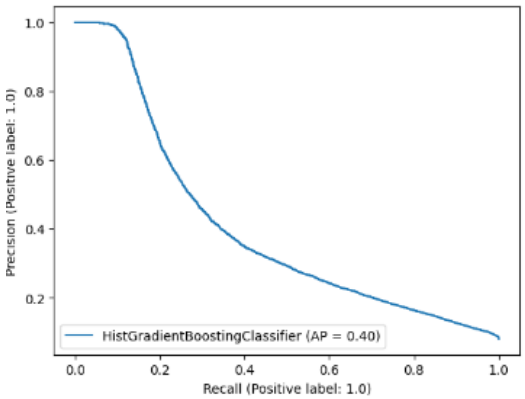


Figure 4.14: PCA w/20 components using Gradient Boosting Classifier

Precision: 0.24620893
Recall: 0.47356694
F1: 0.32397977

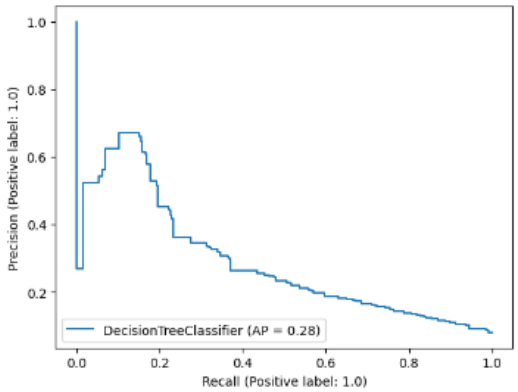


Figure 4.15: PCA w/20 components using Decision Tree

Following our initial testing of PCA with 10 components, we found that its performance was unsatisfactory. In an effort to preserve more of the crucial information while achieving better or equal performance, we made the decision to increase the number of components to 20. This allowed us to retain a greater amount of important information while still reducing the dimensionality of the input data. After applying PCA, the structure of the classifier was preserved with only minimal decline in performance. This indicates that PCA was successful in selecting a subset of features that retained most of the important information needed for classification, while reducing the dimensionality of the input data.

### 4.4 Applying models to different datasets

Applying the same oversampling techniques to dataset 2 led to a much better model. Precision is still extremely high when recall approaches 85%, indicating that the model was able to correctly identify a large proportion of positive samples without an increase of false positives. This high performance can be attributed to the fact that this dataset did not have any missing values and was more robust and clean in general.

Precision: 0.96907216
Recall: 0.75806452
F1: 0.85067873

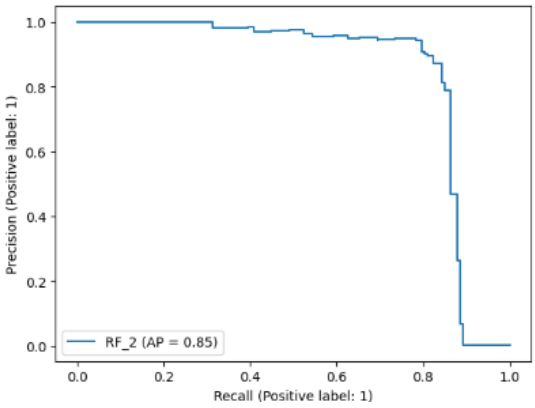


Figure 4.16: Random Forests classifier on dataset 2

## 5. CONCLUSION

From Figure 5 the best results came from PCA using 20 components. This suggests that retaining more input features allowed for our models to better capture patterns in the data. The best models were the models that utilized gradient boosting which allowed for complex relationships to be captured. Overall the figure demonstrates how PCA effectively helped increase our performance on a troublesome and difficult dataset to work with.

Unfortunately we were only able to achieve a small improvement in model performance for the 122 column dataset. We believe that this is due to the lack of correlation

between the input features, which made it difficult for the models to accurately capture the underlying patterns in the data.

Despite only having small improvements, we were able to successfully apply our models and procedures on to other datasets, including the 31 column dataset. For this dataset, we were able to create a model with a F1 score of 0.85.

These findings highlight the importance of carefully selecting and preprocessing input features, as well as adapting machine learning models to the specific characteristics of each dataset. While some datasets may be inherently difficult to work with, careful consideration and adaptation of machine learning techniques can still yield successful results for other datasets.

| Model | Just Imputation | Imputation + SMOTENC | SelectKBest | Recursive Feature Elimination | PCA w/20 Components |
|---|---|---|---|---|---|
| Decision Tree | F1: 0.291 | F1: 0.271 | F1: 0.214 | F1: 0.008 | F1: 0.324 |
| Random Forest | F1: 0.261 | F1: 0.273 | F1: 0.255 | F1: 0.007 | F1: 0.347 |
| Boosting: XGBoost and GradientBoost | F1: 0.320 | F1: 0.329 | F1: 0.278 | F1: 0.0004 | F1: 0.371 |

Figure 5: Conclusion of our findings using F1 score

## 6. REFERENCES

[1] Federal Trade Commission. 2021. Consumer Sentinel Network Data Book 2021. https://www.ftc.gov/system/files/ftc_gov/pdf/CSN%20Annual%20Data%20Book%202021%20Final%20PDF.pdf

[2] Emmanuel, T., Maupong, T., Mpoeleng, D. et al. A survey on missing data in machine learning. J Big Data 8, 140 (2021). https://doi.org/10.1186/s40537-021-00516-9

[3] Liu, Y.; Gopalakrishnan, V. An Overview and Evaluation of Recent Machine Learning Imputation Methods Using Cardiac Imaging Data. Data 2017, 2, 8. https://doi.org/10.3390/data2010008

[4] Chen, RC., Dewi, C., Huang, SW. et al. Selecting critical features for data classification based on machine learning methods. J Big Data 7, 52 (2020). https://doi.org/10.1186/s40537-020-00327-4