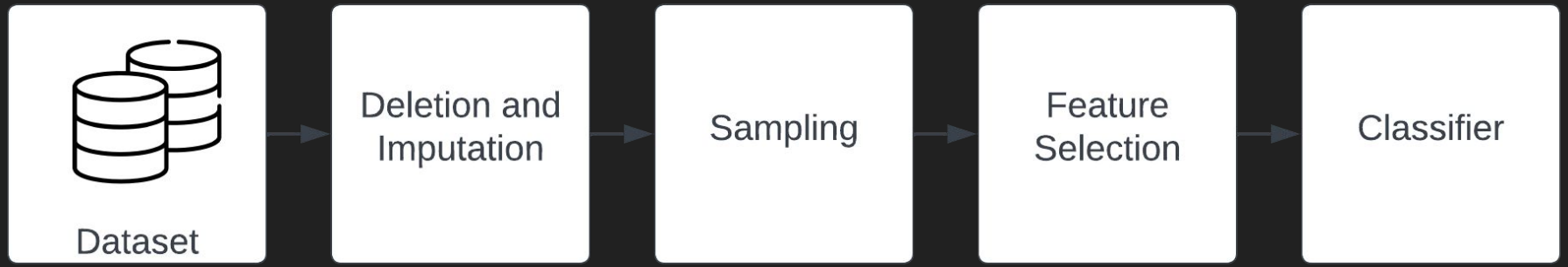
Credit Card Fraud/Worthiness Detection

Colin Blake, Kevin Wang, Eric Wenner

Background

- In 2021, 88,354 instances of credit card fraud reported to the FTC, with estimated losses of $181M
- Detecting fraudulent transactions is an important task for banks and regulators with important implications for consumers
- Data available in relation to transactions (amount, location, account information) is often incomplete or loosely related to whether fraud occurred

Model Overview

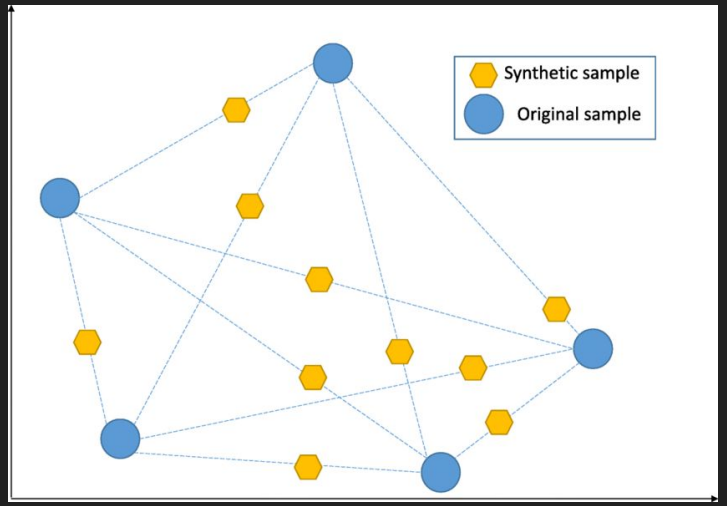


Problem 1: Missing Data

- Data can be:
  - Missing Completely at Random (MCAR): missing values do not depend on any aspect of dataset
  - Missing at Random (MAR): missing values only depend on *observable* data
  - Missing Not at Random (MNAR): missing values depend on observable and *other missing data*
- Mostly impossible to quantitatively show which category your data falls into
- Researchers have some success treating data as if it's MAR to some degree
  - middle of the spectrum
- Option chosen: regression imputation
  - Create a linear regression for each feature using every other feature as the input, use regression to predict missing values for that feature

Problem 2: Class Imbalance

- Many real world applications involve imbalanced datasets including cancer identification, credit card fraud detection, etc.
- Some ways to combat this issue include over/undersampling
- SMOTE generates new data



Metrics

Precision = TP/(TP+FP) - ability of the classifier not to label a negative sample as positive.

Recall = TP/(TP+FN) - ability of the classifier to find all the positive samples.

F1 = 2(PxR)/(P+R) - harmonic mean of both metrics that provides a balanced measure between the two

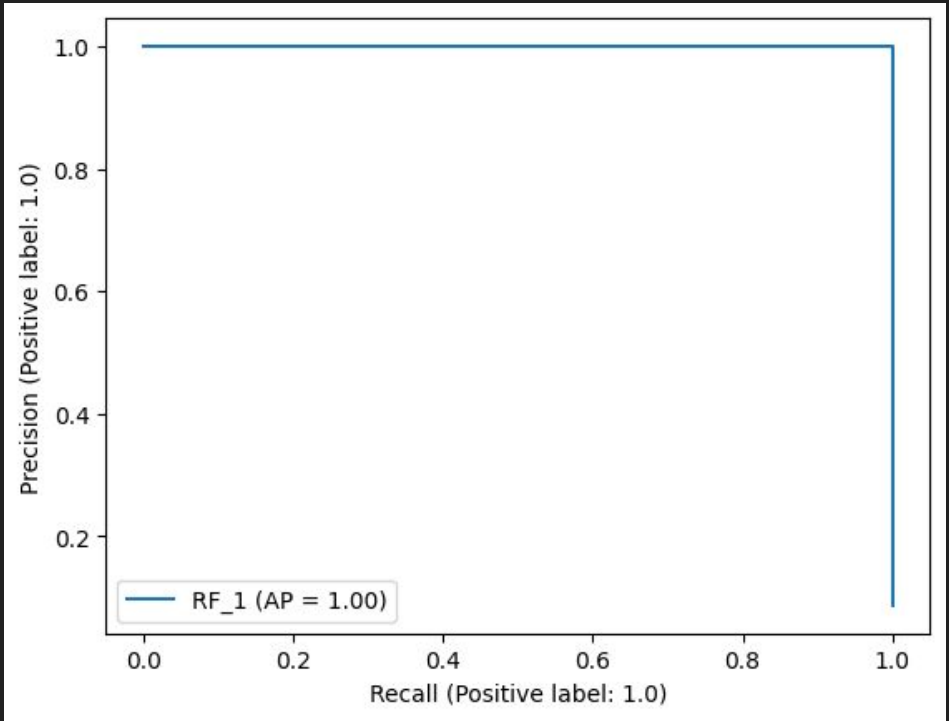
Datasets Used

- We used 2 datasets where one of them had 31 columns with unknown names due to privacy reasons and another had 122 columns
- The larger one described a lot of a person’s background such as their housing, income, assets which was more suited for credit defaults
- The other dataset with 31 columns had unknown column values for privacy reasons but was more suited for credit card fraud itself
- We had particular trouble with the larger of the two and will be investigating our attempts to create meaningful results

Classifiers

- K-nearest neighbors: Finds k-nearest data points and assigns its class to the majority.
- Random forests: Generates many different decision trees and predicts the output based on the majority prediction.
- Gradient Boosted: Models are added in order to reduce overall loss.

Simple Dataset



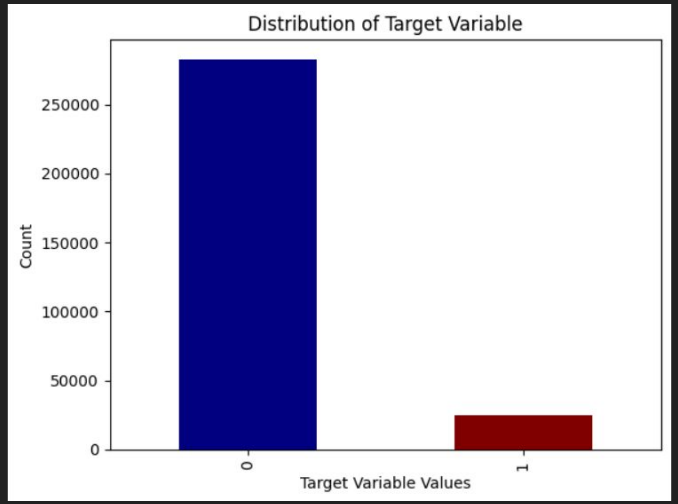
About our Problematic Dataset

- At first inspection our dataset has 122 columns that provides a high description of a client
- Some of these columns simply don’t seem to correlate much to the target class of fraud
- We will try to select some better features to reduce the computational task and remove unnecessary columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

Dataset Imbalance and Missing Values

- With this highly imbalanced dataset we will also need to perform under or over sampling in order to create a better model



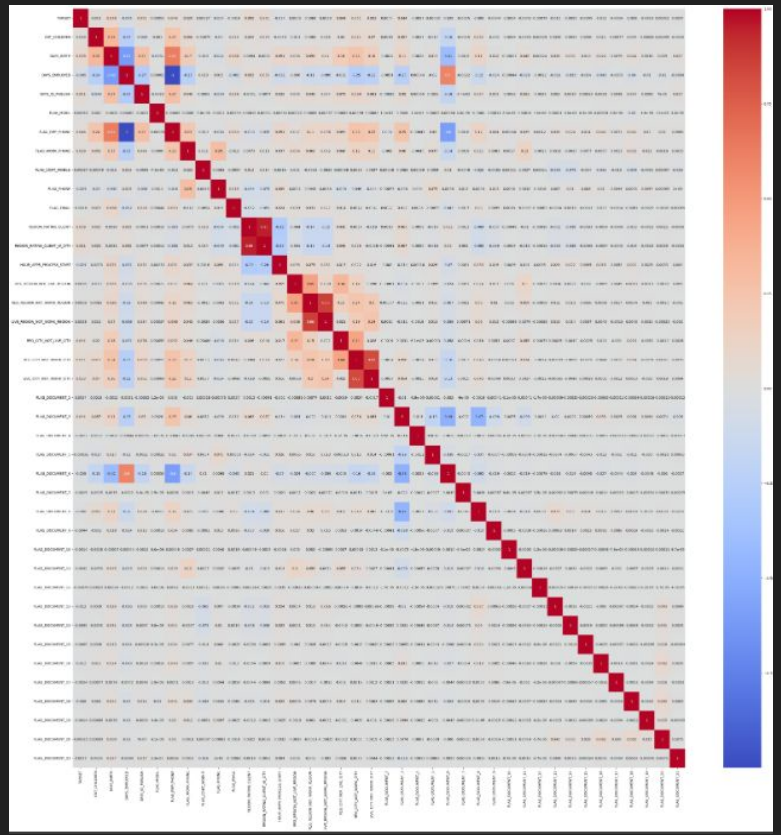
- The dataset also has many columns with high percentages of the values missing, we must also conduct imputation to fill in these values

YEARS_BEGINEXPLUATATION_AVG	48.78%
YEARS_BUILD_AVG	66.50%
COMMONAREA_AVG	69.87%
ELEVATORS_AVG	53.30%
ENTRANCES_AVG	50.35%
FLOORSMAX_AVG	49.76%
FLOORSMIN_AVG	67.85%
LANDAREA_AVG	59.38%
LIVINGAPARTMENTS_AVG	68.35%
LIVINGAREA_AVG	50.19%
NONLIVINGAPARTMENTS_AVG	69.43%
NONLIVINGAREA_AVG	55.18%
APARTMENTS_MODE	50.75%
BASEMENTAREA_MODE	58.52%

Feature Selection

- We hoped that selecting certain features can reduce the complexity of the model and improve performance and also alleviate overfitting
- Methods we used include
  - Observing correlations between the target class and different features via correlation matrix
  - SelectKBest which works by ranking features based on a statistical test and selecting the k highest scores
  - Recursive Feature Elimination which works by recursively eliminating features based on importance and training a model on the remaining features until the desired amount of features is reached
  - PCA that transforms data into its principal components

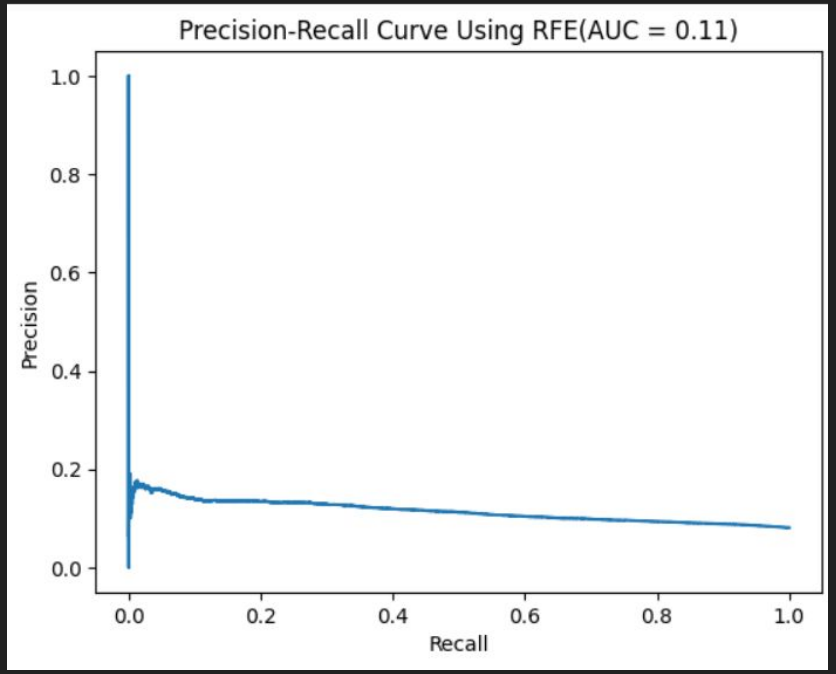
Correlation Matrix



- The correlation matrix was used to observe if any features had high correlation with the target and select the top 10

Results of Feature Reduction

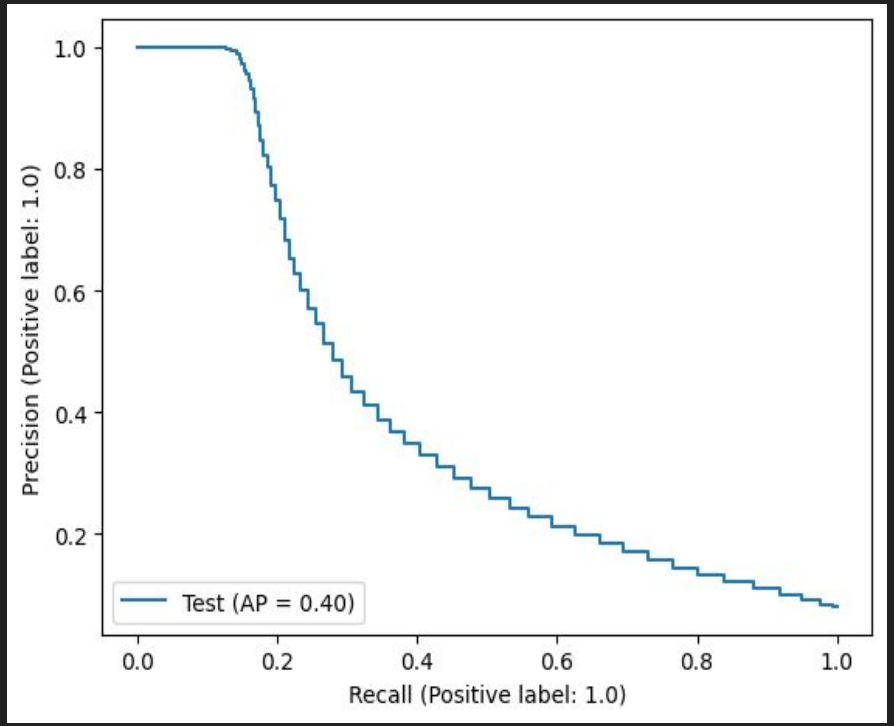
```
Precision: 0.1068417202479498
Recall: 0.5583501006036218
F1 score: 0.17936205280677375
ROC AUC score: 0.5740028499940261
```



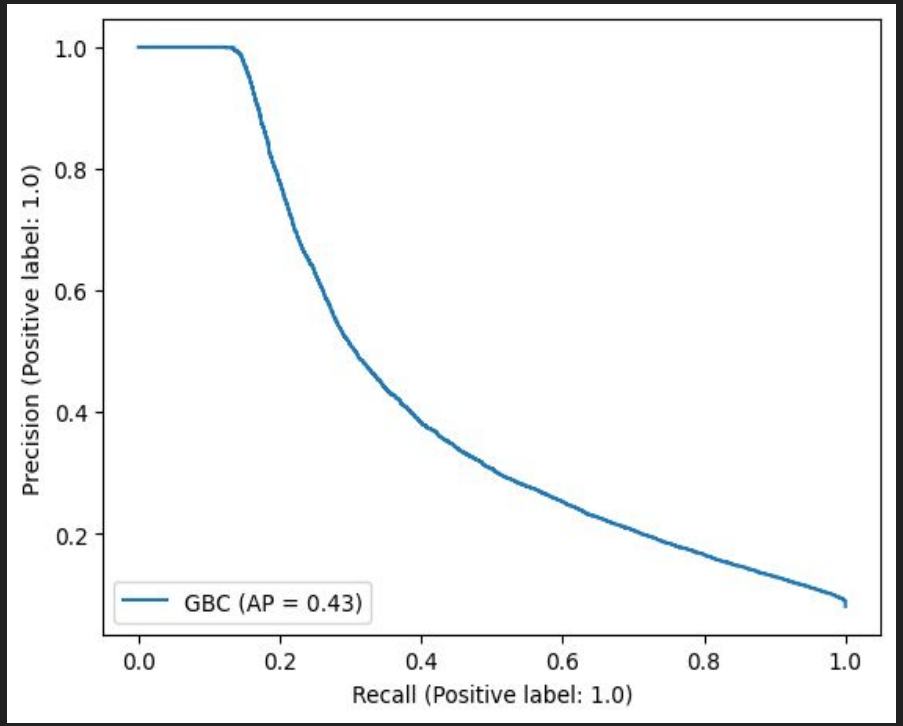
- Our results from all our feature reduction attempts look like the one here
- While it may be slightly better than randomly guessing, it's still not nearly good enough to be used in real world applications

Results on Dataset 2

Basic Random Forests Classifier

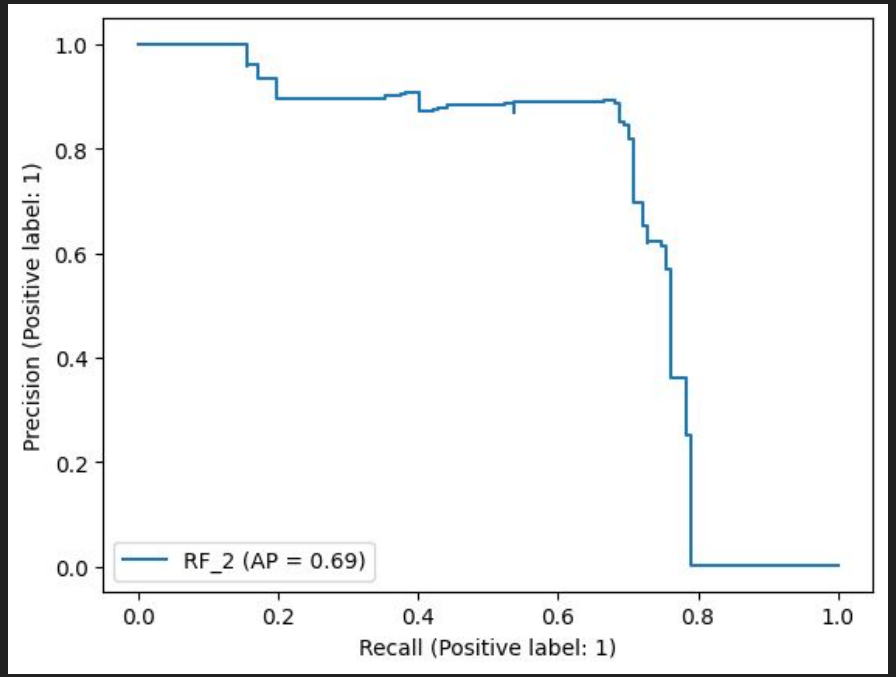


Gradient Boosted Classifier with Random Undersampling

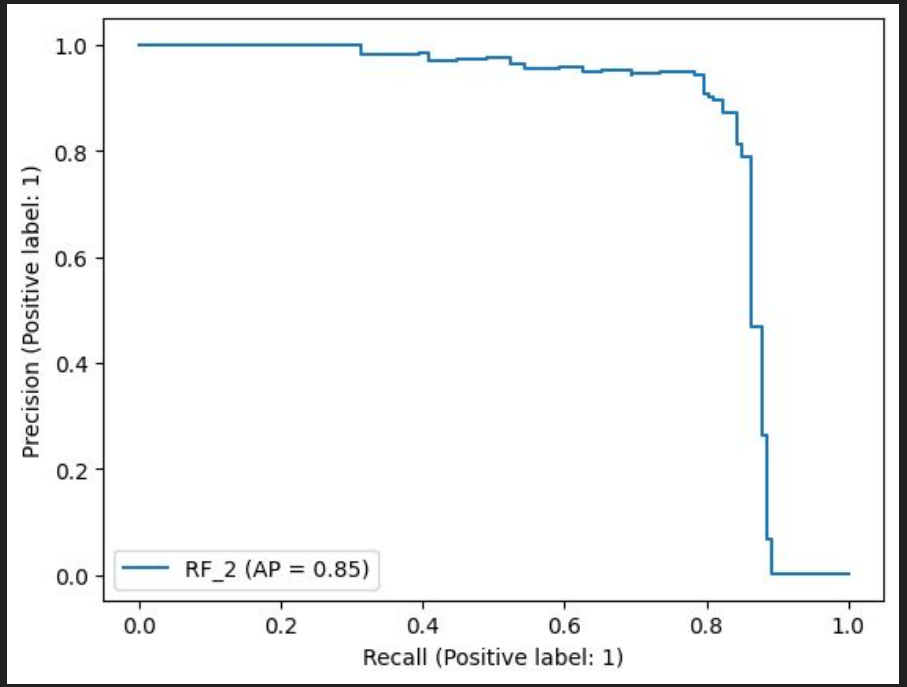


Results on Dataset 1

Basic Random Forests Classifier



Random Forests with SMOTE



Summary/Limitations

- Imputation for numeric features provides significant freedom in model selection when data has missing values
- Feature reduction can decrease the size of data while keeping the most important information
- Some datasets respond better to different approaches than others
- We believe that the large dataset we used had a lot of features that did not correlate well enough with the target class

Future research

- Unsupervised learning techniques like Isolation Forests
- Autoencoder to extract hidden information from data

Questions?