Predicting Credit Card Default

By: Mike McIntire

Introduction

Background: When credit companies issue credit cards to consumers, those companies are taking on risk that the customer may not pay back the loan. Frequently, lenders use customer information and history to determine credit risk. That information can then be used to determine card approval, credit limits, and risk of default. This dataset was collected over several months in 2005 from credit customers in Taiwan. The variables collected are both categorical and numerical. They include demographic information like age, sex, education level, marriage status. The study also includes credit information about the customers including bill and payment amount, payment status, an credit limit.

Target Audience: Risk management department managers and technical staff at banks and credit companies

Project Goal

Predict if a customer will default on the next month’s payment (target variable), given demographic information in the dataset.

Data

- This study will use a dataset from the UCI machine learning repository. A link to the data can be found in the cited sources section.
- The data contains 25 variables and 30,000 rows of data.

Source: https://archive.ics.uci.edu/ml/index.php

```
[2]   1 credit_df.shape

      (30001, 25)
```

Project Workflow

Binary classification of whether a customer will default on a credit payment.

Models to evaluate: **Logistic Regression, K-Nearest Neighbors, Random Forest, Gradient Boost**

Implementing initial models with default inputs for each model.

Models showed little improvement even using hyperparameter tuning.

Researched ways to improve models with class imbalance.

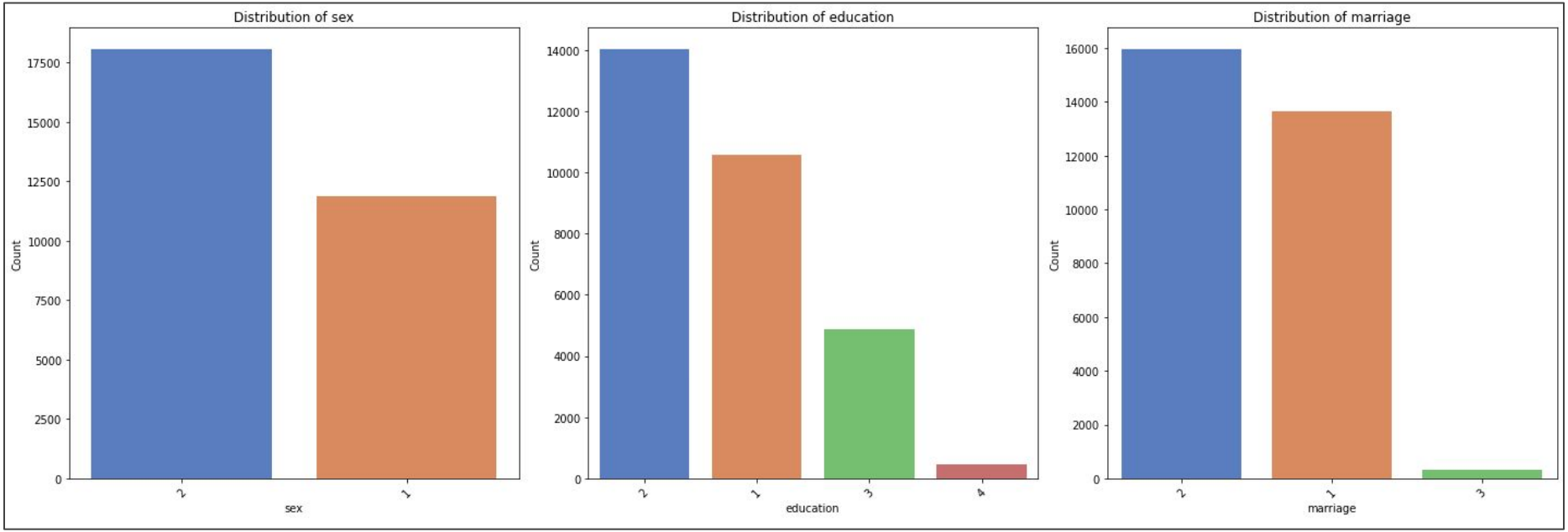
Implement Synthetic Minority Oversampling Technique (SMOTE)

RandomizedSearchCV then GridSearchCV to identify optimal parameters for each model. This takes a really long time to run.

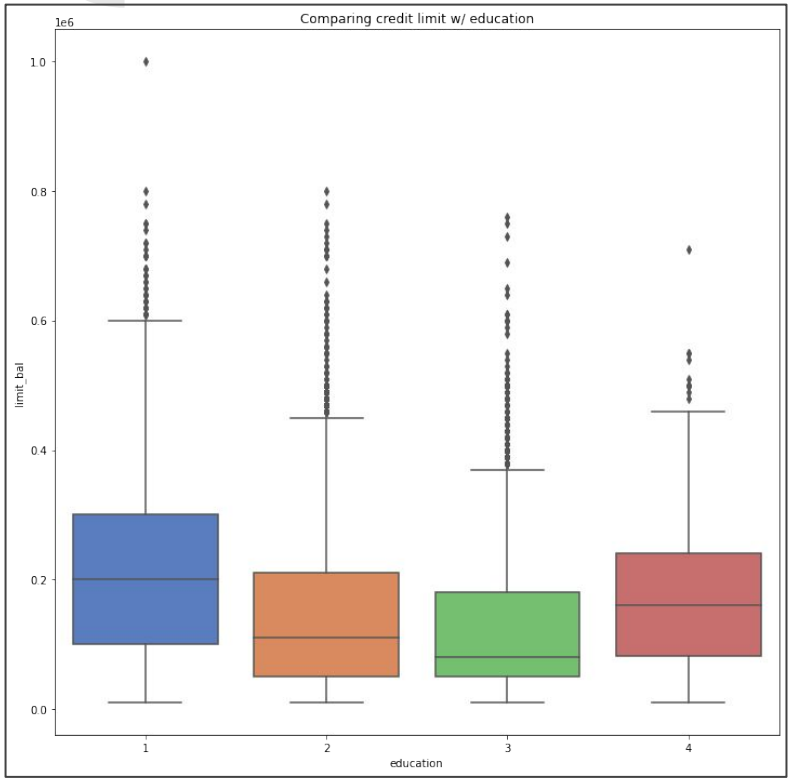
Use updated parameters from GridSearchCV to update models.

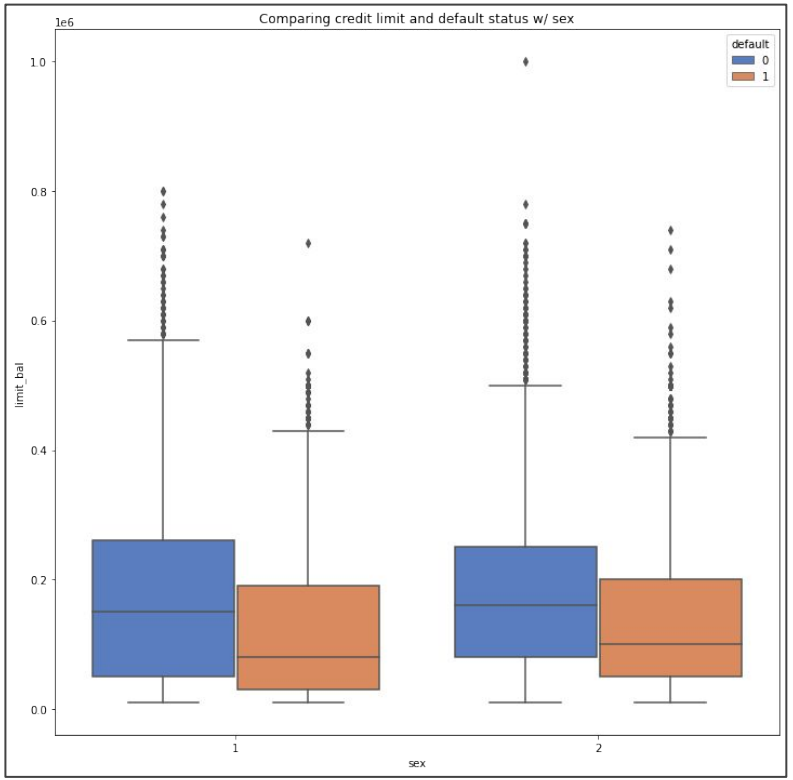
Use precision recall curve plots and Confusion matrices to compare model results.

Data Exploration



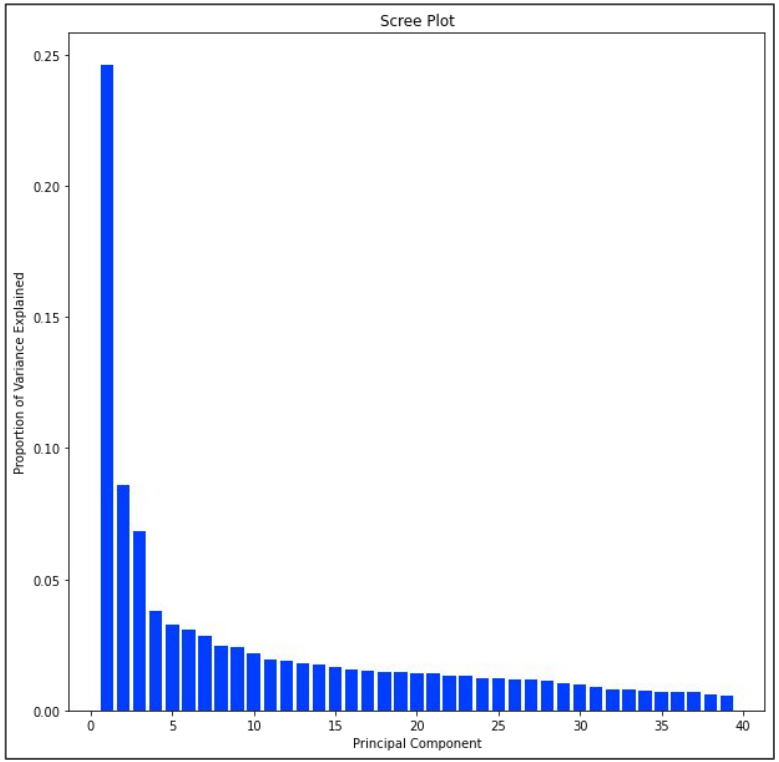
Data Exploration (cont.)





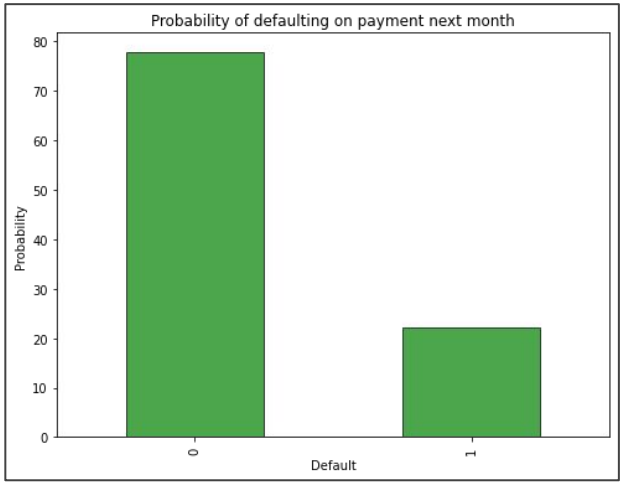
Data Cleaning, Feature Engineering & PCA

- **Reclassified multiple ‘Other’ categories into one ‘Other’ category per variable. This will help reduce the number of features.**
- **<u>Created 5 new features</u> : Percent Bill Paid per Month, Total of Bills, Total of Payments, Total Percent Bills Paid, and Credit Utilization**
- **Created dummy variables for categorical data.**
- **Used Principal Components Analysis to reduce features from 68 to 39**

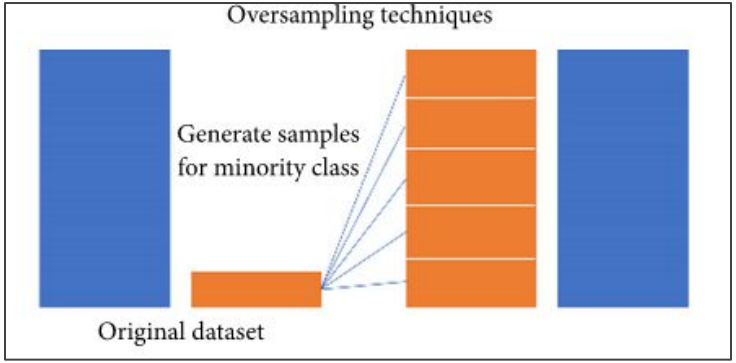


Class imbalance

- Major issue with this dataset is the target variable is imbalanced.
- What does imbalanced mean? 78% of the data indicate customers who have not defaulted, the remaining 22% are customers who have default.
- If we applied no ML algorithms for prediction and randomly chose a customer, 78% of the time we would choose not default.
- The models implemented without adjusting for the imbalance show poor performance correctly identifying defaults, even using hyperparameter tuning.
- Applied SMOTE to oversample minority class



What is SMOTE?

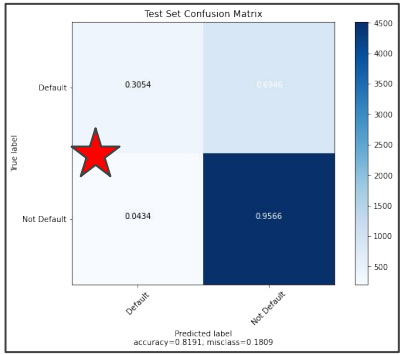


Source: https://www.hindawi.com/journals/complexity/2019/8460934/

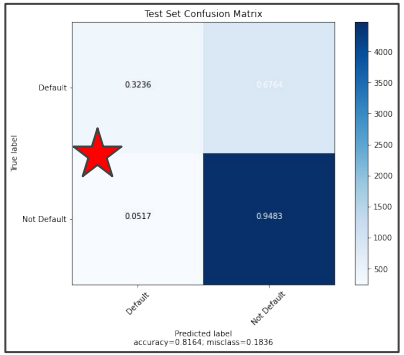
- Synthetic Minority Oversampling Technique
- SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line

No SMOTE

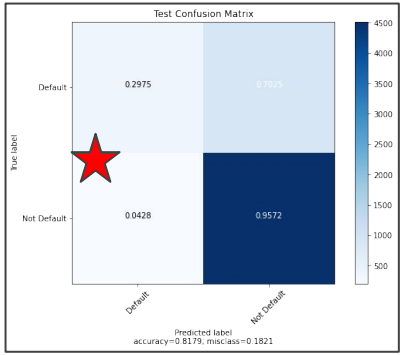
Logistic Regression



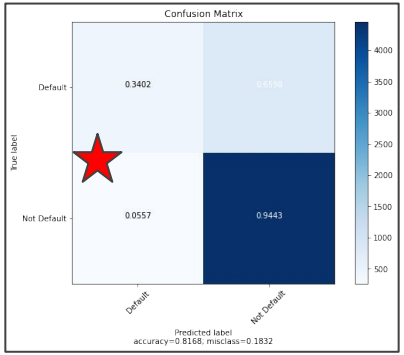
KNN



Random Forest

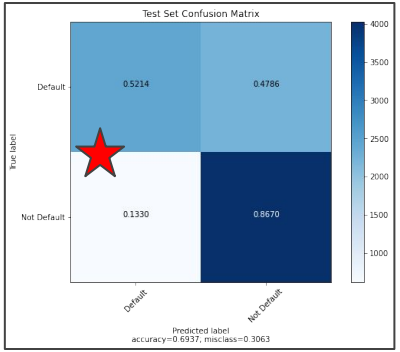


Gradient Boost

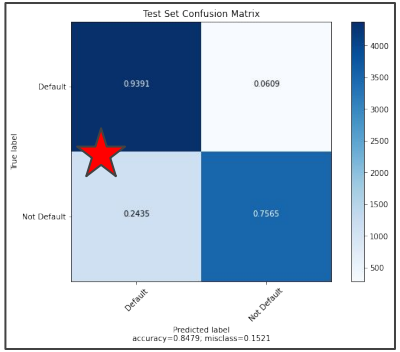


SMOTE

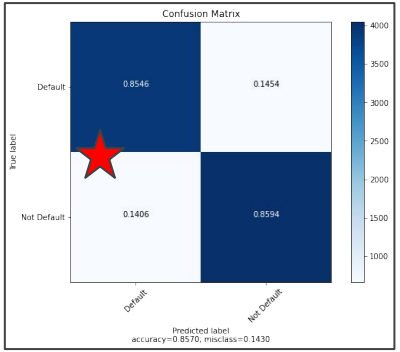
Logistic Regression



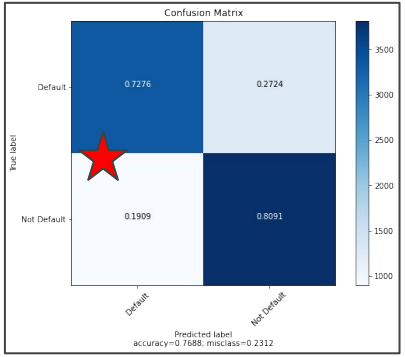
KNN



Random Forest



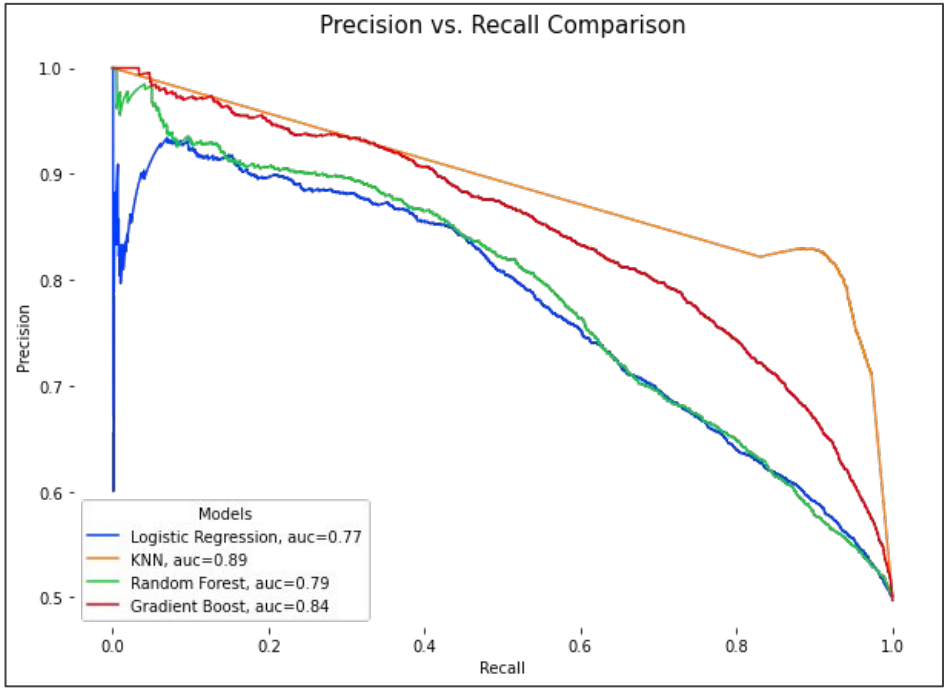
Gradient Boost



Note: Random forest with SMOTE performed better with default parameters

Model Results

Model	Default Precision	Default Recall	AUC
Training LogReg	.78	.57	0.77
Test LogReg	.77	.56	0.77
Training KNN	1.0	1.0	1.0
Test KNN	.79	.94	0.89
Training RFC	.79	.57	0.80
Test RFC	.79	.57	.79
Training GBC	.84	.77	.90
Test GBC	.79	.73	.84



Conclusions

- Evaluated 4 models to predict credit card default (Logistic Regression, KNN, Random Forest, Gradient Boost). Used both RandomizedSearchCV and GridSearchCV to tune hyperparameters.
- Class imbalance is a major issue with this dataset. I applied SMOTE to oversample the minority dataset. This technique may not be the proper way to handle in the real world.
- Primary goal is to correctly identify people who default. The key comparison metric used was Default Recall. Recall improved significantly with SMOTE implementation, but may not be representative of real world observations.
- KNN and Random Forest models were the most successful at predicting default with recall values of .94 and .85, respectively, for the final test set.
- The KNN and Random Forest models indicate overfitting on the training dataset, which is a reason for concern, but the test set is still a significant improvement on data before oversampling.
- Future work: Hybrid oversampling/undersampling, continue to improve hyperparameters (spend more time with Scikit-learn documentation)

Sources Cited

- Raw data: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients
- Credit limits: https://www.nerdwallet.com/article/finance/30-percent-ideal-credit-utilization-ratio-rule
- Exceeding credit limits: https://www.cnbc.com/select/exceeding-credit-limit/
- Credit risk: https://www.investopedia.com/terms/c/creditrisk.asp
- Kaggle variable discussion: https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset/discussion/34608
- Precision-Recall vs. ROC AUC discussion: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/
- SMOTE Discussion: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/
- Hybrid sampling. Possibly a next step: https://www.hindawi.com/journals/complexity/2019/8460934/