To default or not to default? That is the question...

Presented by Steven Yan

Road Map

Data Gathering

Kaggle - UCI ML Repository

Data Cleaning / EDA

Dataset exploration

Vanilla Model

Logistic regression
Decision Tree
Naive Bayes
LDA

Feature Selection

RFECV
Random Forest
XGBoost

Evaluation Metric

F1 Score
PR AUC Score

Hyperparameter Tuning

GridSearchCV

Class Imbalance

Under vs. Over Sampling

Future Plans

Class Imbalance
Unsupervised Algorithms
Ensemble Methods
Feature Engineering
Additional Datasets

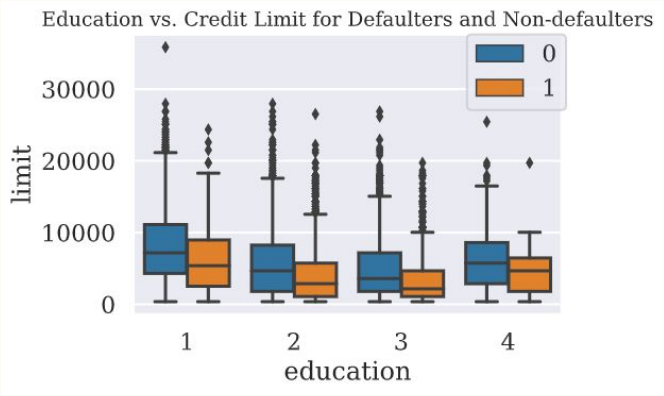
Background

- to increase market share, banks over-issued CCs to unqualified applicants
- cardholders overused CCs irrespective of ability to make payments and accumulated heavy debts
- crisis in Taiwan caused big blow to consumer finance confidence
- to mitigate damage, banks used financial information to predict customers’ credit risk

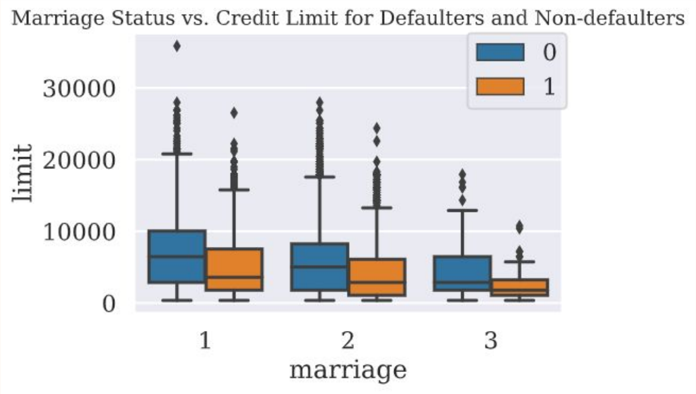
Data Overview:

- UCI Machine Learning Repository or Kaggle
- 30000 customers or observations
- 24 features
  - Credit Info: Credit Line
  - Demographics: Gender, Highest educational degree, Age, Marital Status
  - Payment History (Apr - Sept 2005): repayment status, payment amount, and monthly bill amount
- Target: Default (0 or 1)

Education Level



Marriage Status

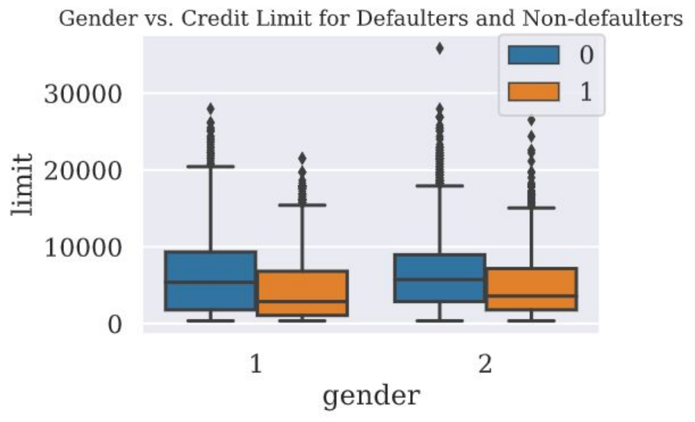


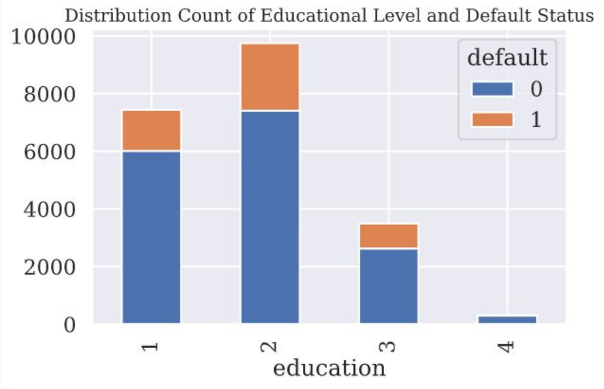
EDA

- `education`
- `marriage`
- `gender`

Hypothesize little impact on `default`

Gender

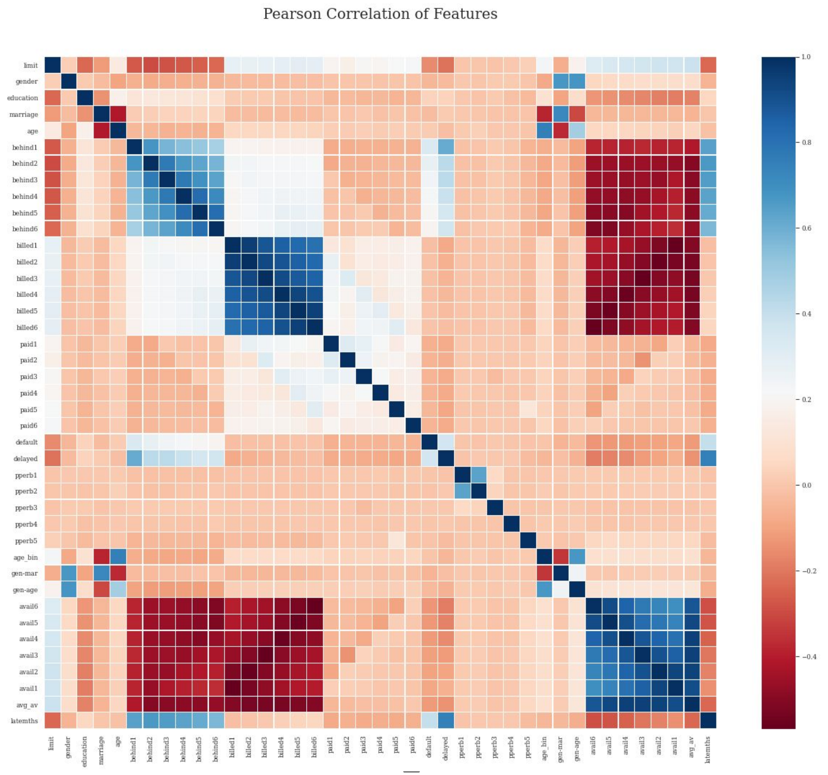




Insights from EDA

- **`default`** is correlated with:
  - **`behind1`** through **`behind6`**
  - Negatively with **`limit`**
  - With engineered features:
    - **`delayed`**
    - **`latemnths`**
    - **`avail1`** - **`avail6`**

- **`gender`** not correlated with any feature
- **`education`** slightly correlated with **`limit`** and **`age`**
- **`age`** correlated with **`marriage`** and slightly with **`education`** and **`limit`**
- **`limit`** slightly correlated with **`billed1-6`**, **`education`**, **`paid1-6`**

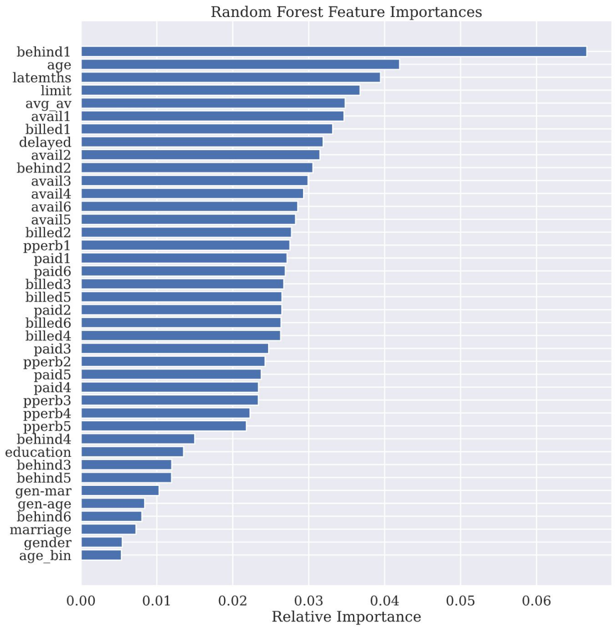


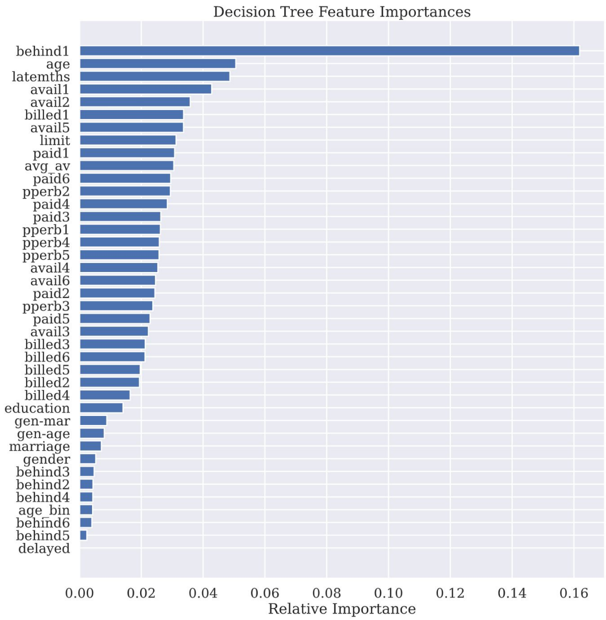
Vanilla Model

- Logistic Regression
- Random Forest
- Decision Tree
- Gaussian Naive Bayes
- Linear Discriminant Analysis
- K-Nearest Neighbors
- AdaBoost
- Gradient Boosting
- XGBoost

	Accuracy	F1 Score	ROC AUC	Recall	Precision	PR AUC
Logistic Regression	0.811500	0.360656	0.726854	0.242955	0.699561	0.486829
Random Forest Classifier	0.816167	0.460108	0.755976	0.357959	0.643836	0.512818
Decision Tree Classifier	0.730167	0.398365	0.614037	0.408225	0.388970	0.288287
K-Nearest Neighbors	0.798000	0.447080	0.704327	0.373191	0.557452	0.416605
Gaussian Naive Bayes	0.724000	0.498486	0.736553	0.626809	0.413776	0.480981
Linear Discriminant Analysis	0.810333	0.367778	0.718289	0.252094	0.679671	0.480476
AdaBoost Classifier	0.815667	0.425753	0.775158	0.312262	0.668842	0.523430
Gradient Boosting Classifier	0.821000	0.468843	0.780810	0.361005	0.668547	0.545396
XGBoost Classifier	0.816833	0.469338	0.765113	0.370145	0.641161	0.518716

Feature Selection





- behind1
- age
- latemnths
- limit
- avg_av

Hyperparameter Tuning

Tuning with GridSearchCV:

- Logistic Regression
- Random Forest
- Adaboost
- Gradient Boosting
- XGBoost

Baseline accuracy of 77%

Improved accuracy to 82%

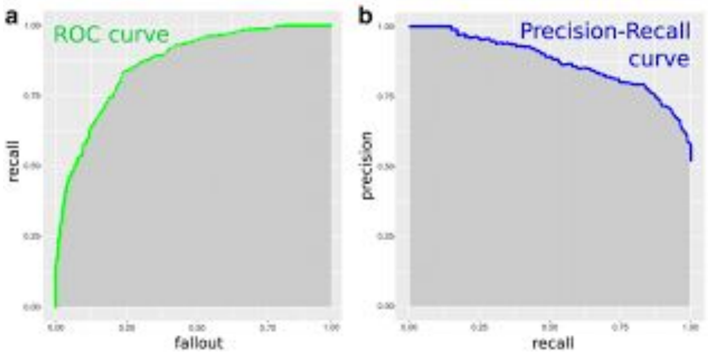
Maximizing PR AUC Score

	Accuracy	F1 Score	ROC AUC	Recall	Precision	PR AUC
Logistic Regression 3	0.807667	0.393270	0.755005	0.284844	0.634975	0.498670
Random Forest Classifier 3	0.814667	0.455436	0.755404	0.354151	0.637860	0.510753
Decision Tree Classifier 3	0.721667	0.385578	0.605986	0.399086	0.372954	0.280576
AdaBoost Classifier 3	0.818833	0.450177	0.776501	0.338919	0.670181	0.525689
Gradient Boosting Classifier 3	0.820000	0.463221	0.781312	0.354912	0.666667	0.542575
XGBoost Classifier 3	0.812500	0.451487	0.761622	0.352628	0.627371	0.515684

	Accuracy	F1 Score	ROC AUC	Recall	Precision	PR AUC
Logistic with GridSearchCV	0.816667	0.439348	0.750020	0.328256	0.664099	0.500342
Random Forest with GridSearchCV	0.817333	0.461690	0.760251	0.357959	0.650069	0.505874
Decision Tree with GridSearchCV	0.820833	0.460612	0.778612	0.349581	0.675000	0.540126
AdaBoost with GridSearchCV	0.818667	0.442051	0.772015	0.328256	0.676609	0.518960
Gradient Boosting with GridSearchCV	0.820000	0.463754	0.779051	0.355674	0.666191	0.539624
XGBoost with GridSearchCV	0.818333	0.458788	0.775866	0.351866	0.659058	0.535901

Evaluation Metrics

- **Recall:** Out of all the defaulters, how many did we get right?
  - TP and FN
- **Precision:** How correct is our model based on its own prediction
  - TP and FP
- **F1 Score:** Harmonic mean of recall and precision
  - F-score - 2 would be weighing recall more than precision
- **PR AUC Score**: average precision rate, scoring metric for GridSearchCV



Next Steps

- Exploration into undersampling and oversampling methods
    - SMOTE and Tomek
    - Ensemble Methods
    - Customer Segmentation
    - SMOTEN
- Additional datasets to fix class imbalance
- Use MinMaxScaler

Contact

Steven Yan

LinkedIn: http://ww.linkedin.com/in/examsherpa

Github: https://www.github.com/examsherpa

Email: stevenyan@uchicago.edu