Understanding Student Debt

Statistics and Machine Learning

Adrian Halarewicz
Springboard Capstone Project

Problem

- Many students blindly take on student debt.
- Students struggle to define a repayment plan.
- Will the student be able to repay the debt?

Solution

- Explore U.S. Census Income data to better understand future financial standing.
- Loan data from Lending Club can help predict if a borrower will default on a Loan.

Example - Case Study

THE CLIENT :

Age → **24**

Gender → **Male**

City → **Los Angeles, CA**

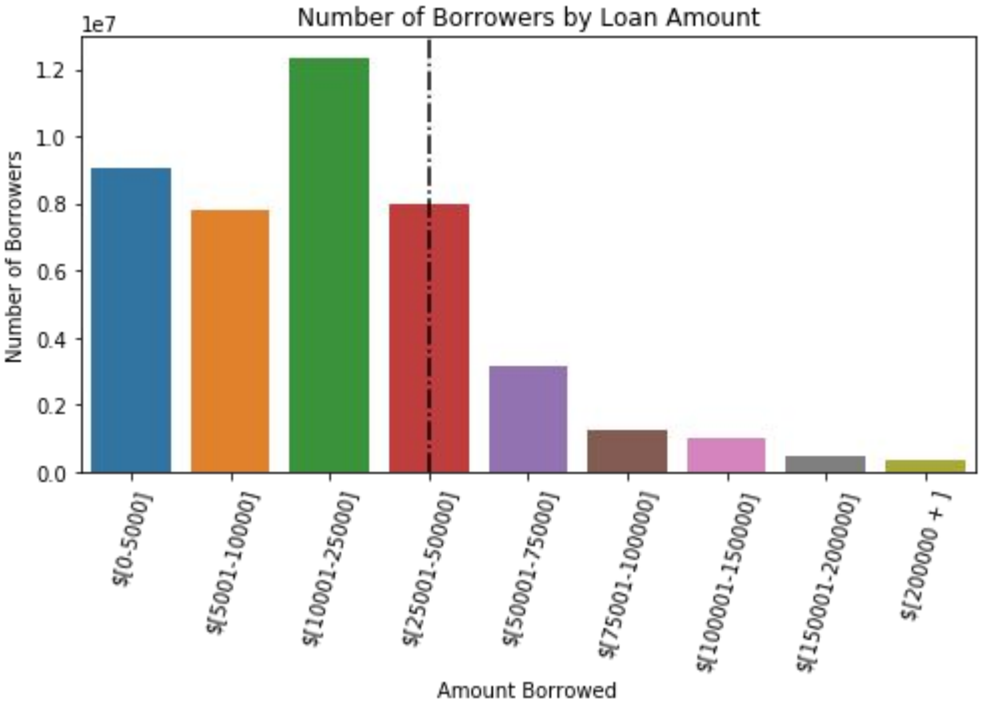
Degree/Industry → **Scientific and Technical Services**

Amount Borrowed → **$40,000**

Min. Monthly Payment → **$450**

The Student Debt Landscape

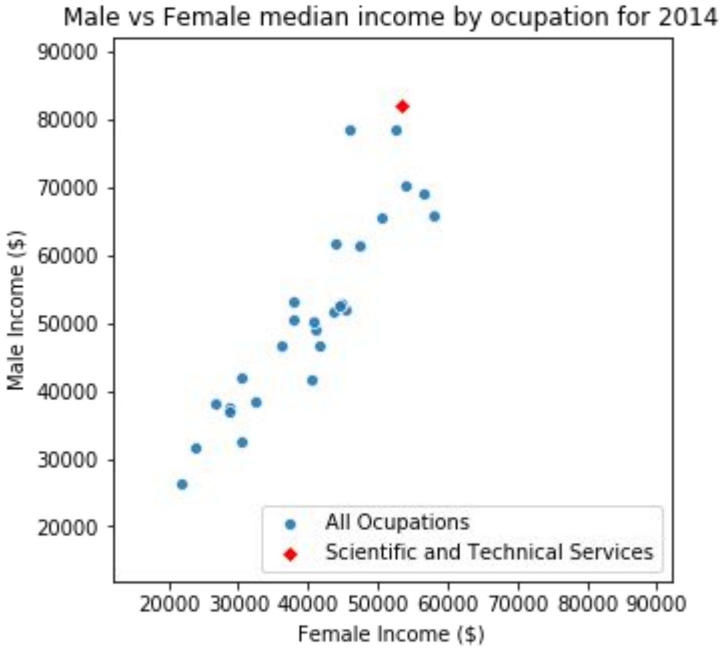
- Client Loan Balance: $40,000
- More debt than most borrowers.



Median income by occupation

- Expected Income: $83,000
- The median income for Science and Technical Services.
- Higher than most other occupational categories.
- A healthy Debt to Income Ratio:

Income : Debt → 83,000 : 40,000



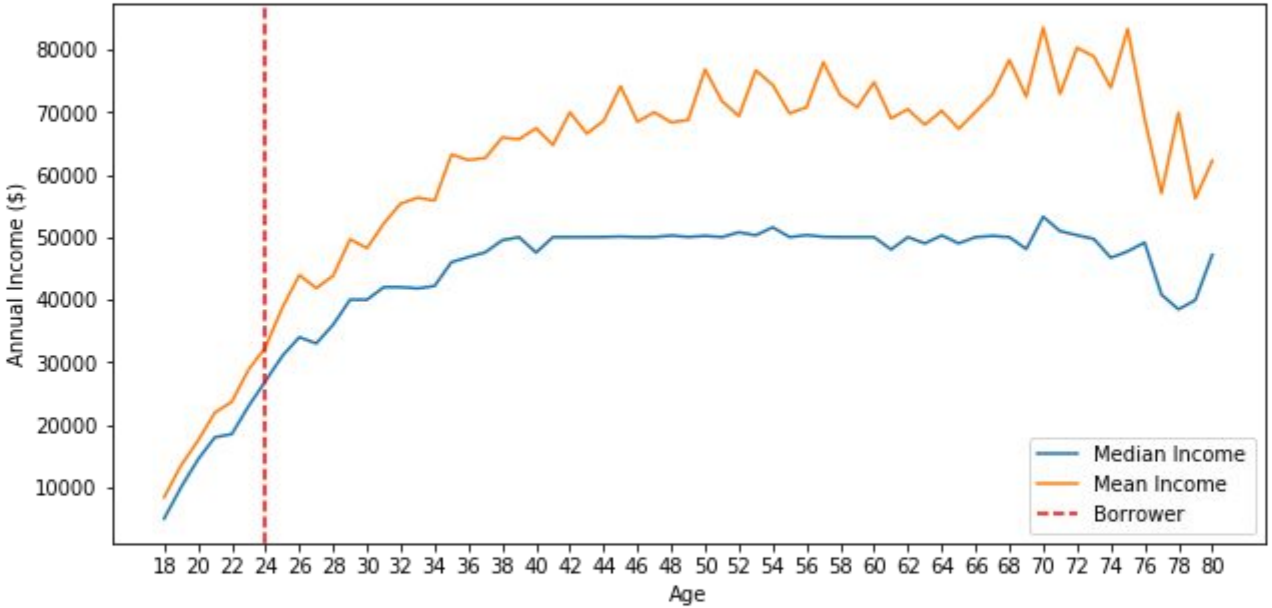
How does the client compare?

- Median Income for 24 year olds:
  - $34,000

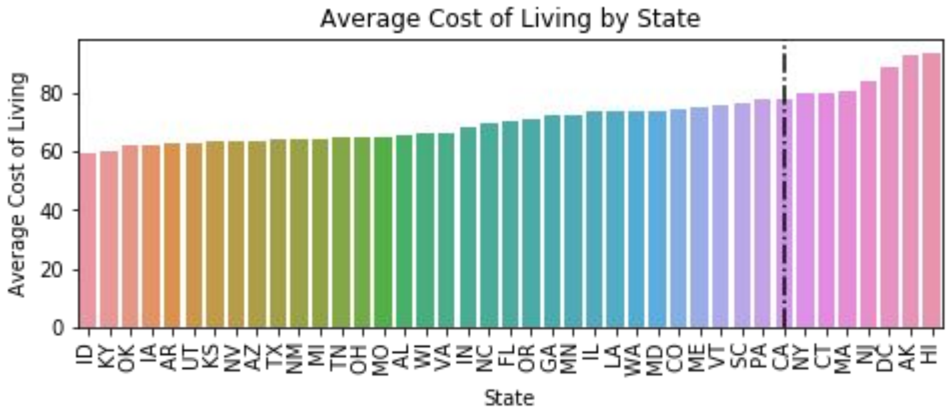
- Mean Income for 24 year olds:
  - $28,000

Client will be making significantly more

than other people his age.



Consider Cost of Living

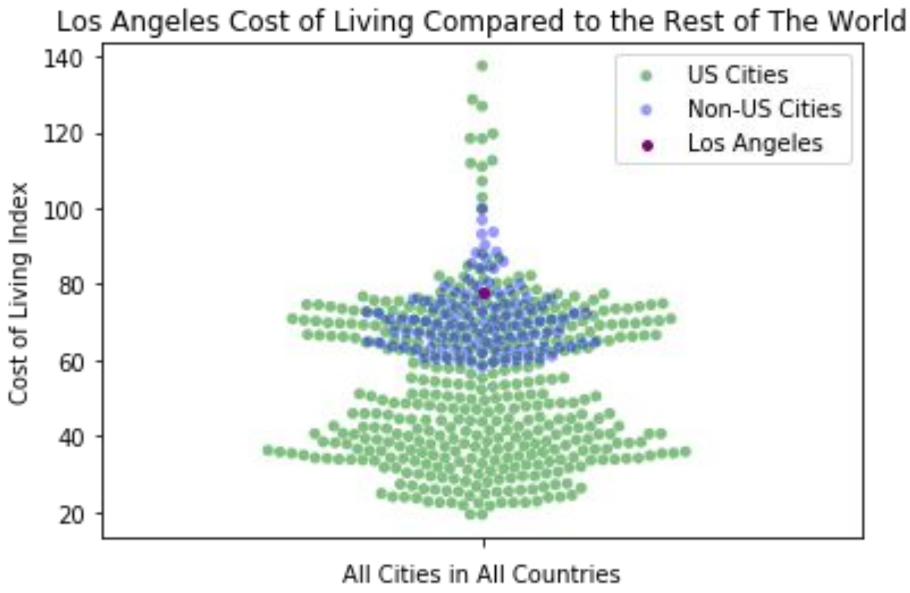


- Cost of Living in California is High!

- Should the client consider moving to a state with a Lower cost of Living?

Consider Cost of Living

- The Cost of Living in Los Angeles is among the most expensive cities in the U.S. and throughout the world.
- Fortunately, wages and salaries are typically higher in areas with a higher cost of living.
- Living in a high cost of living area is beneficial to the client's overall debt to income ratio.



Will the user Default on their Loan?

Implementing Machine Learning to Predict Loan Default

- Random Forest
  - An out of the box classifier for quickly making initial predictions with default hyperparameters.
- Support Vector Machine
  - Requires cross-validation for hyperparameter tuning.

Training Data

- Lending Club loan data: All loans from 2007 → 2019 Q3
- Tidy Data
  - Rows represent instances of loans issued by Lending Club.
  - Columns represent individual features describing each respective loan.
- 80,000+ rows of loans issued
  - a variety of purposes: Education, home, auto, debt consolidation, …
  - Loan Status is listed for each loan.

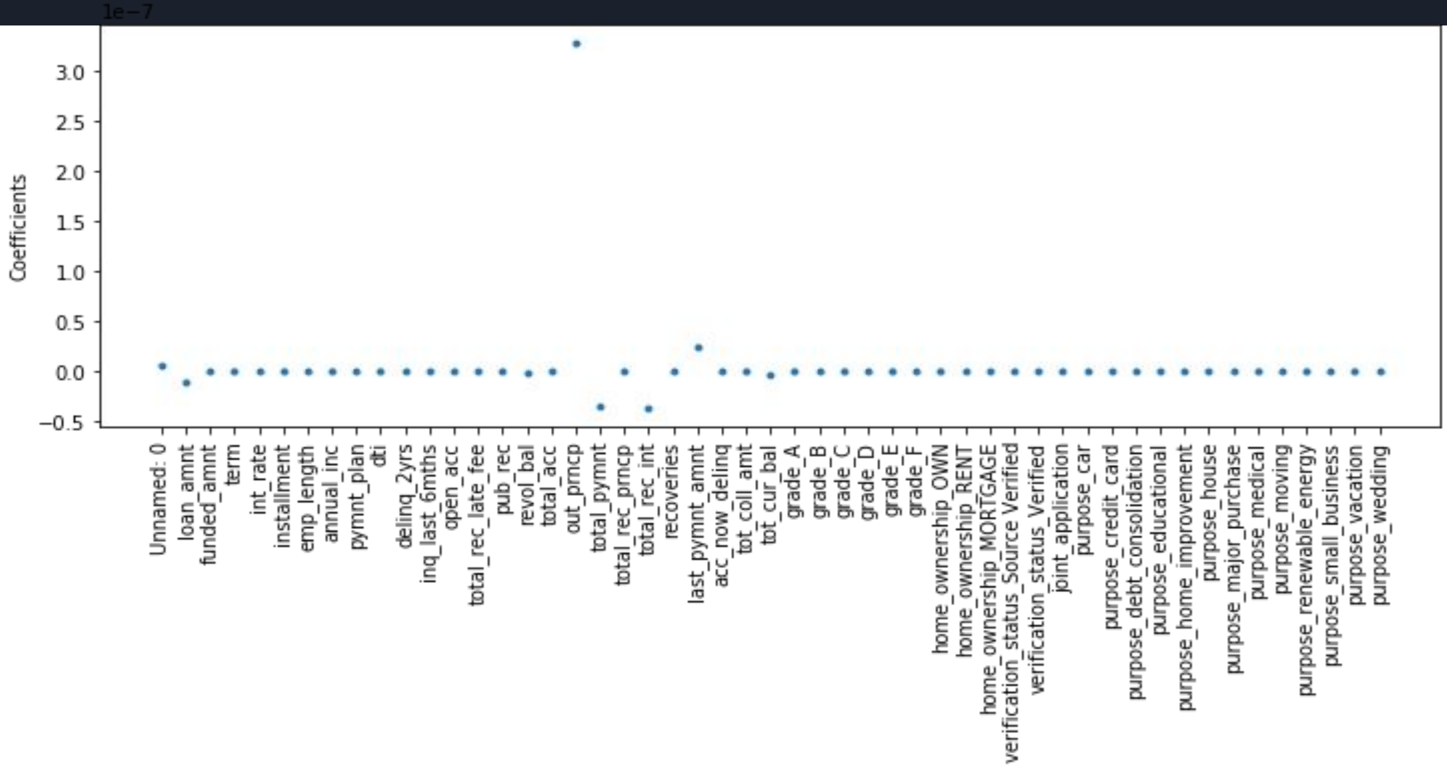
Features

- 74 unique columns describing individual rows.
- Create dummy variable columns for categorical features.
  - pd.get_dummies(column_name)
- Convert discrete features to integers.
- Lasso Regression (Elastic Net):
  - Identify important features with most predictive power.
- Drop irrelevant columns.
- Final DataFrame contains 50 columns of features + 1 column for Loan Status = 51 total columns

Lasso Regression

Predictive Power:

- \- loan_amnt
- \+ out_prncp
- \- total_pymnt
- \- total_rec_int
- \+ last_pymnt_amnt



Targets

- The Loan Status column:
    - 1 → Loan is in DEFAULT
    - 0 → Loan is NOT IN DEFAULT
- Fixing class imbalance
    - Minority class: Loans in Default
    - Majority class: Loans not in Default
    - Lending Club's best effort to prevent issuing loans that will default.

SMOTE for Class Imbalance

- Synthetic Minority Over-Sampling Technique
- Over-Sample the minority class.
- Create an equal number of instances

  for Loans in Default.

```
# inspect class imbalance for defaulted loans
unique, count = np.unique(y_train, return_counts=True)
value_counts = {k:v for (k,v) in zip(unique, count)}
value_counts
```

{0: 619995, 1: 853}

```
# Applt Synthetic Minority Over-sampling Technique (SMOTE)
sm = SMOTE(random_state=42)
X_train_bal, y_train_bal = sm.fit_sample(X_train, y_train)
```

```
# inspect balanced training data
unique, count = np.unique(y_train_bal, return_counts=True)
value_counts = {k:v for (k,v) in zip(unique, count)}
value_counts
```

{0: 619995, 1: 619995}

The Training Data and Hold-Out Set

- Drop rows with missing data.
- Extract targets and features.
- Split the targets and features into train sets and test/hold-out sets.

```
loans = loans.dropna()
loans.shape
```

```
(886927, 52)
```

```
y = loans['default'].values
X = loans.drop('default', axis=1).values
```

```
X.shape
```

```
(886927, 51)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=33)
```

Random Forest: Training the model

- Instantiate a Random Forest Classifier with Default hyperparameters.
- Fit the classifier to the training data.
- Evaluate the model using the hold-out set.

```
# instantiate random forest classifier
random_forest = RandomForestClassifier(max_depth=True)

# fit model to training data
print('fitting model to training data')
random_forest = random_forest.fit(X_train, y_train)

# test model performance
print('evaluating model performance')
score = random_forest.score(X_test, y_test)
print(score)
```

Random Forest: Evaluating the Model

- With Class Imbalance
  - Accuracy = 0.9986 but TP = 0 FP = 0
  - The model predicted that ALL LOANS WILL NOT DEFAULT
    - High accuracy due to severe class imbalance.
- After SMOTE
  - Accuracy = 0.7132 but TP = 251 FP = 76180
  - The model predicted that ALL LOANS WILL NOT DEFAULT.
    - Sacrifice Accuracy to better predict loan default.
    - **True Positive Rate = 0.685**

	Predict: NO	Predict: YES
Actual: NO	265713	0
Actual: YES	366	0

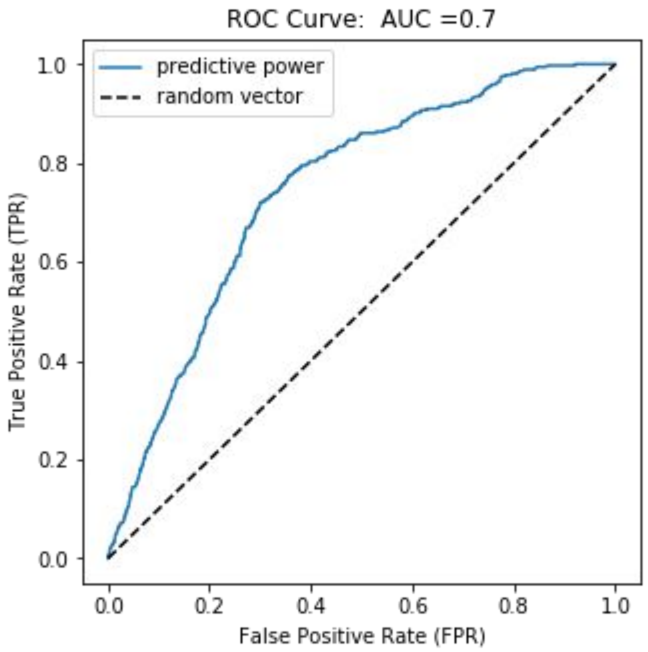
	Predict: NO	Predict: YES
Actual: NO	TN	FP
Actual: YES	FN	TP

	Predict: NO	Predict: YES
Actual: NO	189533	76180
Actual: YES	115	251

Random Forest: ROC Curve

- Area under the ROC Curve
  - AUC = 0.70

- This model has predictive power.
- Ability to make significant predictions.
- Outperforms a Random Vector.



Support Vector Machine: Training the model

- Instantiate the SVM classifier
  - with default parameters
- Fit the model to the training data.
- Predict on the hold-out set.
- Before SMOTE:
  - TP = 0 & FP = 0

```
# instantiate SVM classifier
svm = SVC()
```

```
# fit SVM to training data
svm.fit(X_train, y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
# make predictions on test features
y_pred = svm.predict(X_test)
```

- Apply smote to correct class imbalance in the training set and the model's predictions.
- NOTE: Unlike Random Forest, SVM requires the data to be scaled to prevent distance bias from occurring.

Support Vector Machine: GridSearchCV

- Hyperparameter tuning:
  - Grid Search with 5 fold Cross Validation.
  - Find best values of C and Gamma.

- **C** → the amount of slack given to outliers.

- **gamma** → the perpendicula distance of a point to the fit line in the opposite direction of **w**.

```python
# define hyperparameter space
c_values = [0.001, 0.01, 0.1, 1]
gamma_values = [0.001, 0.01, 0.1]
param_grid = {'C': c_values, 'gamma': gamma_values}

# create grid-search object
grid_search = GridSearchCV(clf, param_grid, cv=5)
```

```python
# fit apply gridsearch to SVM with training data
grid_search.fit(X_train_bal, y_train_bal)
```

```
GridSearchCV(cv=5, error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid={'C': [0.001, 0.01, 0.1, 1],
                         'gamma': [0.001, 0.01, 0.1]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

```python
# identify best C and gamma combination
best_params = grid_search.best_params_
best_params
```

```
{'C': 1, 'gamma': 0.1}
```

Support Vector Machine: Making Predictions

- GridSearchCV stores and remembers the best combination of **C** and **gamma**.
- After the cross-validation predictions are made with the model trained with the best Hyperparameters.
- Large Data → Long training time for each model
  - Means cross-validation is an extremely long process.
  - This model was trained and evaluated on a sample of the data.
  - A Pipeline was built to:
    - Perform grid search with 5-fold cross validation.
    - Train the model on the complete data set with the best hyperparameters.
    - Make better predictions based on the complete data set.

```python
# Run gridsearch cross validation
cv = GridSearchCV(pipeline, param_grid=param_grid)
cv.fit(X_train_bal, y_train_bal)

# make predictions with best params from grid search
y_pred = cv.predict(X_test)
```

Conclusion

Given the client's income and credit history, we can:

1. **Illustrate where the student borrower's financial standing.**
2. **Predict if the client will default on their loan with 68% accuracy.**

Random Forest is a great out of the box classifier.