DO YOU KNOW WHAT GOES INTO A CREDIT SCORE?!?!

Credit Score Classification

Jenny Shin, Hannah Hodek, & Louis Gonzalez

Overview

Industry: Finance

Problem Statement

We are working as data scientists at a global finance company. The company has collected basic bank details and gathered a lot of credit-related information. The management wants to build an automated system to classify the records into credit score brackets.

Our Task

Given the customers' credit-related information, build a machine learning model that can classify the credit score.

Dataset and Technologies

- Kaggle Dataset - Credit score classification

  https://www.kaggle.com/datasets/parisrohan/credit-score-classification/data

- Credit related information for 10,139 people during 8 month period
- ***Google Colab***
- ***Pandas*** and ***PySpark*** for data wrangling and analysis
- ***Scikit-learn*** and ***Tensorflow*** for machine learning library
  - K-nearest Neighbors
  - Random forests
  - Boosts
  - Decision Tree
  - Neural Network
- ***Tableau*** for data visualization
- ***Keras-Tuner*** for NN model optimization

Data Preprocessing

- CreditScore (target) had 3 categories (Good, Standard, Poor) > > 2 labels
- Eliminated unnecessary columns
- Converted Credit History Age to FLOAT
- Features:
    - AnnualIncome
    - NumBankAccounts
    - NumCreditCard
    - NumofLoan
    - Delayfromduedate
    - NumofDelayedPayment
    - NumCreditInquiries
    - CreditMix
    - OutstandingDebt
    - CreditUtilizationRatio
    - CreditHistoryAge
    - MonthlyBalance
    - Credit Score

Project Breakdown 1: *Spark*

Understanding our Dataset

PySpark
- Used to manipulate and filter the dataframe data
- Used temporary table to run queries

Analysis
- Credit Score
  - Good/Standard 71%
  - Poor 29%
- Outstanding Debt (Avg)
  - Good/Standard 1161
  - Poor 2082
- Credit Utilization Ratio (Min / Avg / Max) *
  - Good/Standard 20.0 % / 32.4% / 50%
  - Poor 20.2% / 32% / 48%

CreditUtilizationRatio	CreditScore
50.00000000000001	Good/Standard
49.5223242978724З	Good/Standard
48.17659890246224б	Good/Standard
48.48985172844354	Poor
49.5645193473869 9	Good/Standard

Project Breakdown 2: *Classification Models*

K- nearest neighbors algorithm

- ★ k=3
- ★ Labels: Good/Standard, Poor
- ★ Scaled necessary columns
- ★ Classification Report
- ★ Accuracy Score: 0.78
- ★ Precision, Recall for Poor is low
- ★ Model is sensitive
  - Missing values
  - Dimensionality
  - Outliers

Confusion matrix

```
array([[12215,  2475],
       [ 1987,  3275]])
```

Classification report

	precision	recall	f1-score	support
Good/Standard	0.86	0.83	0.85	14690
Poor	0.57	0.62	0.59	5262
accuracy			0.78	19952
macro avg	0.71	0.73	0.72	19952
weighted avg	0.78	0.78	0.78	19952

Project Breakdown 2: *Machine Learning Models*

Random Forest, Decision Tree, AdaBoost, Gradient Boost MLMs.

- ★ RandomForest n_estimator = 100, random_state = 42
- ★ AdaBoost n_estimator = 100
- ★ Gradient Boost n_estimator = 100
- ★ Labels = Good/Standard, Poor
- ★ Random Forest Accuracy = 86%
    - Precision for Poor is OK
    - Recall for Poor is OK
- ★ Decision Tree Accuracy = 81%
    - Precision for Poor is Low
    - Recall for Poor is Low
- ★ Gradient Boost Accuracy = 81%
    - Precision for Poor is OK
    - Recall for Poor is Low
- ★ AdaBoost Accuracy = 78%
    - Precision for Poor is Low
    - Recall for Poor is Low

```
Confusion Matrix Random Forest:
[[13002  1146]
 [ 1701  4103]]
Classification Report Random Forest:
               precision    recall  f1-score   support

Good/Standard       0.88      0.92      0.90     14148
         Poor       0.78      0.71      0.74      5804

     accuracy                           0.86     19952
    macro avg       0.83      0.81      0.82     19952
 weighted avg       0.85      0.86      0.86     19952
```

```
Confusion Matrix Decision Tree:
[[12234  1914]
 [ 1903  3901]]
Classification Report Decision Tree:
               precision    recall  f1-score   support

Good/Standard       0.87      0.86      0.87     14148
         Poor       0.67      0.67      0.67      5804

     accuracy                           0.81     19952
    macro avg       0.77      0.77      0.77     19952
 weighted avg       0.81      0.81      0.81     19952
```

```
Confusion Matrix Gradient Boost:
[[12736  1412]
 [ 2434  3370]]
Classification Report Gradient Boost:
               precision    recall  f1-score   support

Good/Standard       0.84      0.90      0.87     14148
         Poor       0.70      0.58      0.64      5804

     accuracy                           0.81     19952
    macro avg       0.77      0.74      0.75     19952
 weighted avg       0.80      0.81      0.80     19952
```

```
Confusion Matrix AdaBoost:
[[12558  1590]
 [ 2729  3075]]
Classification Report AdaBoost:
               precision    recall  f1-score   support

Good/Standard       0.82      0.89      0.85     14148
         Poor       0.66      0.53      0.59      5804

     accuracy                           0.78     19952
    macro avg       0.74      0.71      0.72     19952
 weighted avg       0.77      0.78      0.78     19952
```

Project Breakdown 3: *Neural Network Model*

Summary:
The objective of this model is to classify data into one of two categories based on the 'target' column, which represents simplified credit score categories ('Good/Standard' or 'Poor').

★

- **Label Encoding**:
  - The 'target' column is label-encoded using scikit-learn's LabelEncoder.
- **Neural Network Model Structure**:
  - Consists of an input layer with 80 units ReLU activation
  - A hidden layer with 30 units and ReLU activation
  - Output layer with 1 unit and sigmoid activation.

Project Breakdown 3: *Deep Learning Model*

Initial Results

```
624/624 - 1s - loss: 0.4285 - accuracy: 0.8019 - 903ms/epoch - 1ms/step
Loss: 0.42845645546913147, Accuracy: 0.8018745183944702
624/624 [==============================] - 1s 1ms/step
Confusion Matrix:
[[12712  1436]
 [ 2517  3287]]

Accuracy Score: 0.801874498797113

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.90      0.87     14148
           1       0.70      0.57      0.62      5804

    accuracy                           0.80     19952
   macro avg       0.77      0.73      0.74     19952
weighted avg       0.79      0.80      0.80     19952
```

- **Accuracy**:
  - Correctly classified about 80.19% of the samples in the test data.
- **Confusion Matrix**:
  - True Positives (TP): 12712 - The model correctly predicted "Good/Standard" credit scores.
  - True Negatives (TN): 3287 - The model correctly predicted "Poor" credit scores.
  - False Positives (FP): 2517 - The model incorrectly predicted "Good/Standard" when it was actually "Poor."
  - False Negatives (FN): 1436 - The model incorrectly predicted "Poor" when it was actually "Good/Standard"
- **Precision**:
  - Predicts "Good/Standard" as correct about 83% of the time.
  - Predicts "Poor" as correct about 70% of the time.
- **Recall**:
  - Correctly identifies 90% of the actual "Good/Standard" cases.
  - Correctly identifies 57% of the actual "Poor" cases.
- **F1-Score**:
  - For "Good/Standard," the F1-score is approximately 87%
  - For "Poor" the F1-score is approximately 62%

Project Breakdown 3:
Deep Learning Model

★ ★ ★

Optimization Process

- ★ Ran Keras tuner
- ★ Ran model with tuner recommendations
- ★ Played with adding/dropping columns
- ★ Binned columns with different amount of bins
- ★ Tried different amounts of epochs
- ★ Tried different amounts of hidden layers and activations (relu, tanh, & sigmoid)
- ★ Only found noteable increase in scores when “ChangedCredit Limit” wasn’t dropped

Project Breakdown 3: *Neural Network Model*

Optimization

Results

```
624/624 - 2s - loss: 0.4232 - accuracy: 0.8183 - 2s/epoch - 3ms/step
Loss: 0.42324042320251465, Accuracy: 0.818263828754425
624/624 [==============================] - 3s 4ms/step
Confusion Matrix:
[[12805  1343]
 [ 2283  3521]]

Accuracy Score: 0.8182638331996792

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.91      0.88     14148
           1       0.72      0.61      0.66      5804

    accuracy                           0.82     19952
   macro avg       0.79      0.76      0.77     19952
weighted avg       0.81      0.82      0.81     19952
```

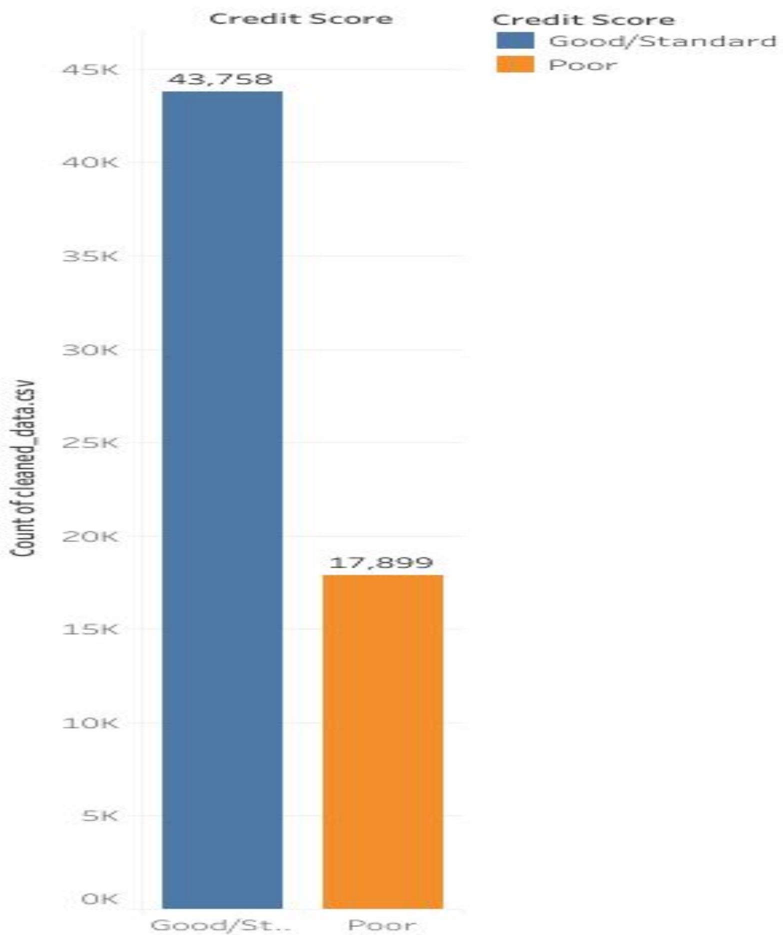
- ★ **Accuracy**:
  - Correctly classified about 81.83% of the samples in the test data.
- ★ **Confusion Matrix**:
  - True Positives (TP): 12805 - Correctly predicted "Good/Standard" credit scores.
  - True Negatives (TN): 3521 - Correctly predicted "Poor" credit scores.
  - False Positives (FP): 2283 - Incorrectly predicted "Good/Standard" when it was actually "Poor."
  - False Negatives (FN): 1343 - Incorrectly predicted "Poor" when it was actually "Good/Standard."
- ★ **Precision**:
  - Predicts "Good/Standard," as correct about 85% of the time.
  - Predicts "Poor" as correct about 72% of the time.
- ★ **Recall**:
  - Correctly identifies 91% of the actual "Good/Standard" cases.
  - Correctly identifies 61% of the actual "Poor" cases.
- ★ **F1-Score**:
  - "Good Standard" F1 score is approximately 88%
  - "Poor" F1 score is approximately 66%

Project Breakdown 4: *Visualizing Data with Tableau*

Project Breakdown 4: *Visualizing Data with Tableau*

We started off by looking at all the data and seeing how many people in our data have good or poor credit scores

Comparing the number of Good/Standard Credit Scores VS Poor Credit Scores

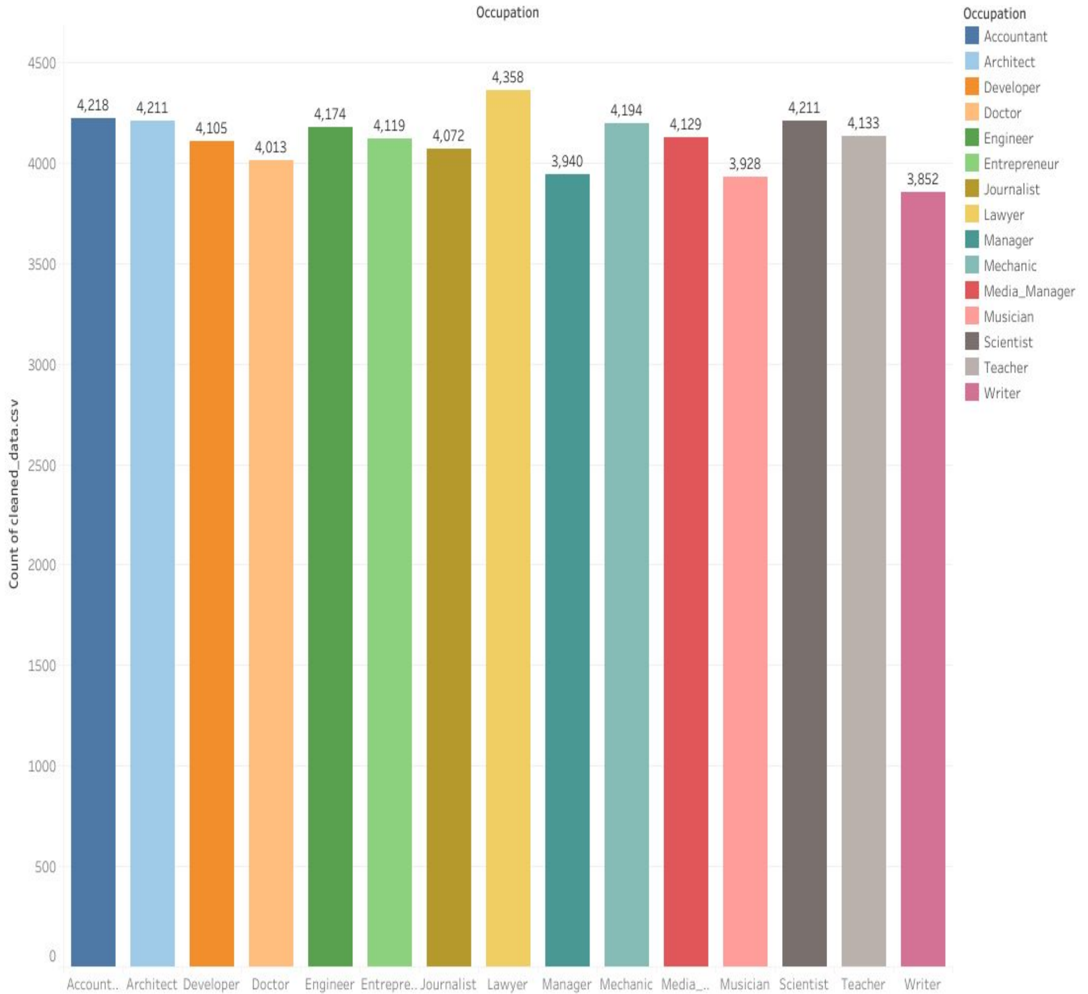


Project Breakdown 4: *Visualizing Data with Tableau*

★ ★ ★

With this chart, we wanted to visualize the types of occupation and how many of our data points fall into each job

Count of Occupations

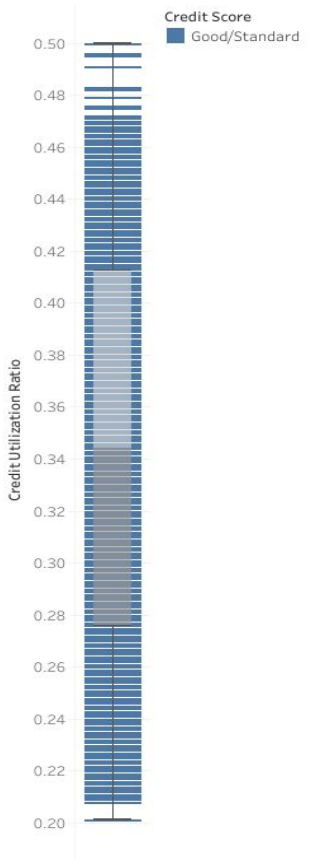


Project Breakdown 4: *Visualizing Data with Tableau*

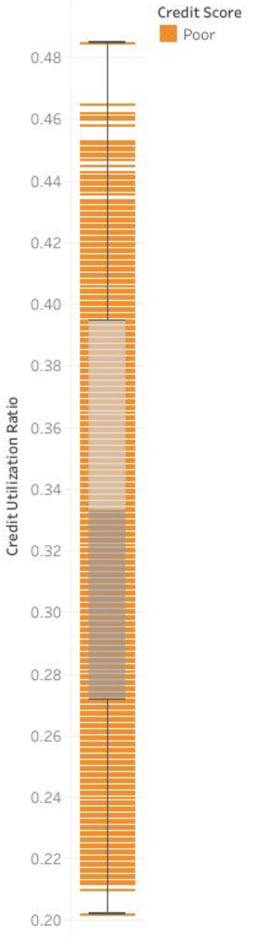
★ ★ ★

We thought we saw some outliers within our dataset and we wanted to focus on the Credit Utilization Ratio. We Used a Box Plot to better understand how our data is spread out. While the data is spread out it shows no significant outliers within the Credit Utilization Ratio Column

Box Plot of Credit Utilization Good/Standard Credit Score



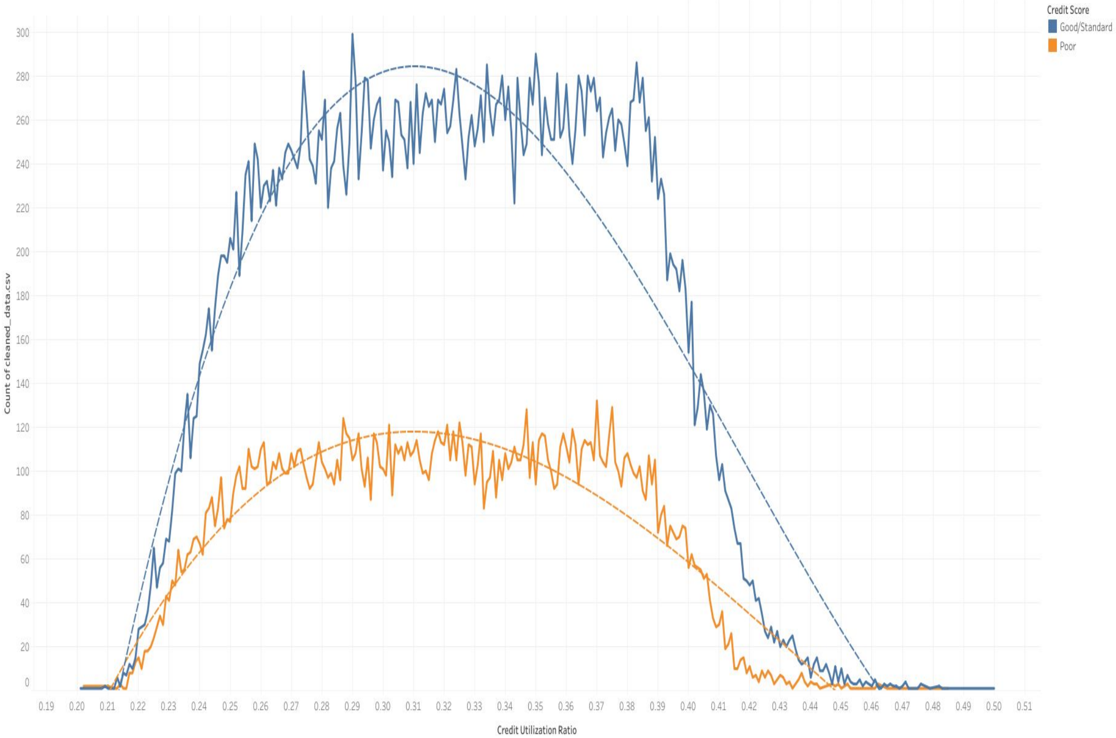
Box Plot of Credit Utilization Poor Credit Score



Project Breakdown 4: *Visualizing Data with Tableau*

Comparing Good/Standard and Poor Credit Scores along their Credit Utilization Ratio. We saw that there isn't a huge difference between the distribution of Good Vs Poor

Credit Utilization Count with Trendline

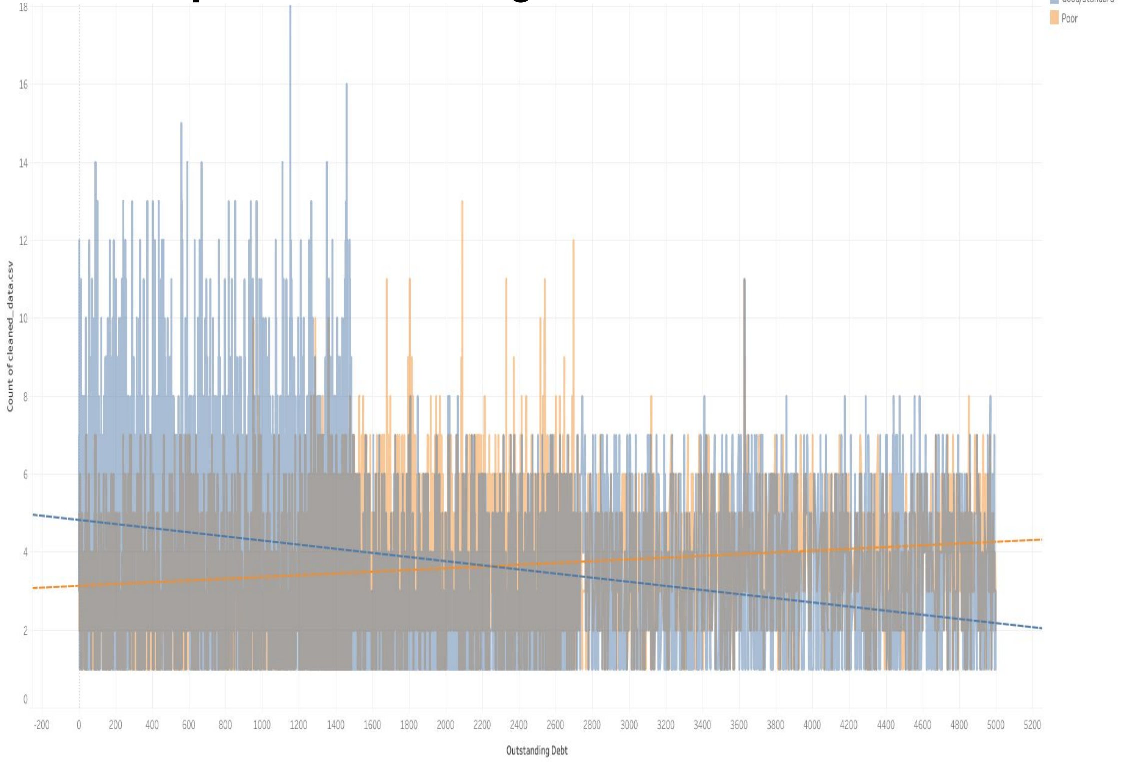


Project Breakdown 4: *Visualizing Data with Tableau*

★ ★ ★

Here we wanted to see how the outstanding debt compares between people with Good/Standard and Poor credit scores. We can see that a lot more of the people with Good Credit score have less than $1,600 of debt whereas those with Poor Credit Score are more clustered between $1,000-$3,000 of debt

Line Graph of Outstanding debt with Trendline



Good/Standard R-Value = 0.085, P-Value = <0.0001

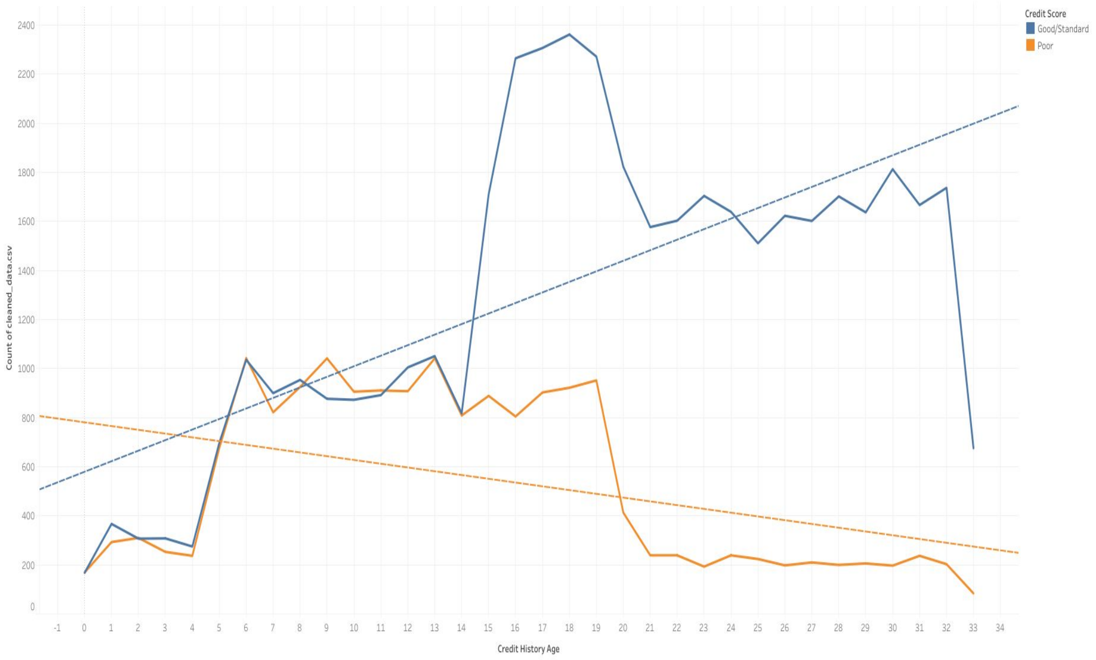
Poor R-Value = 0.025, P-Value = <0.0001

Project Breakdown 4: *Visualizing Data with Tableau*

Here we are comparing the Credit Scores and the Credit History Age.

We see that there is a larger number of those with Good Credit having a longer Credit History, and we notice a decline on the Poor Credit line.

Line Graph Credit Age By Credit Score



*Good/Standard R-Value = 0.459, P-Value = <0.0001

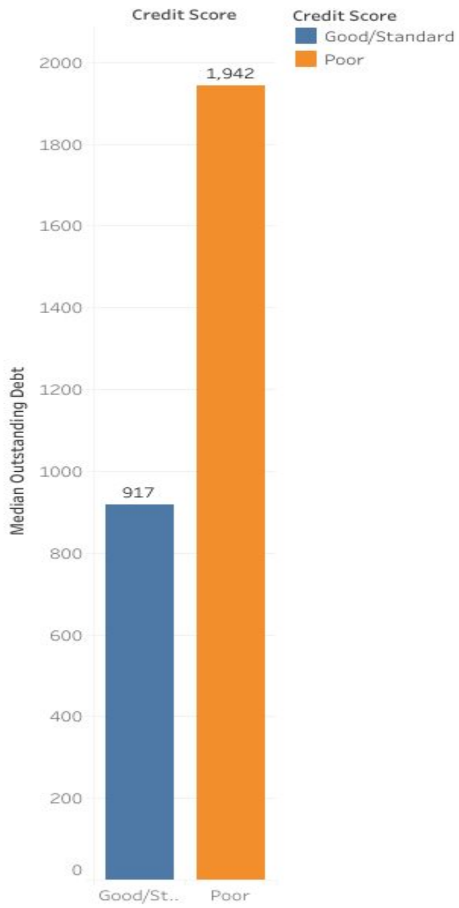
*Poor R-Value = .0191, P-Value = 0.0097

Project Breakdown 4: *Visualizing Data with Tableau*

Comparing the Median Outstanding Debt and Credit Scores

Those with Poor Credit have a higher Median Outstanding Debt than those with Good Credit

Median Outstanding Debt for Good/Standard Credit



Final Results

```
# Get the feature importance array
importances = rf_model.feature_importances_
# List the top 10 most important features
importances_sorted = sorted(zip(rf_model.feature_importances_, X.columns), reverse=True)
importances_sorted[:5]
```

```
[(0.23636865430945517, 'OutstandingDebt'),
 (0.09700193074513488, 'Delayfromduedate'),
 (0.09010244526785605, 'MonthlyBalance'),
 (0.08650588758496071, 'AnnualIncome'),
 (0.08584078274452349, 'CreditUtilizationRatio')]
```

- ★ **K-nearest neighbors Model**
  - For a starter model, this performed decently at 78% accuracy, but the precision/recall values for Poor credit score category could be better.
  - The model is sensitive to dimensionality and outliers so these are potential areas of improvement.
- ★ **RandomForest, Decision Tree, AdaBoost, Gradient Boost**
  - RF performed very well at 86% accuracy, the precision and recall values are very good to start
  - Decision Tree and Gradient Boost performed well at 81% accuracy, the precision and recall are low
  - AdaBoost performed well at 78% accuracy, the precision and recall are low
- ★ **Neural Network Model**
  - The NN model appears to perform reasonably well with an accuracy of about 81.83%. It is better at classifying "Good/Standard" credit scores (higher precision and recall) compared to "Poor" credit scores. However, there is still room for improvement.

Conclusion

In summary, the objective of our project was to build a machine learning model that accurately classifies credit scores when given customers' credit information. We achieved this by building a variety of different models and comparing their results. Our final recommendation would be to use the Random Forest model which attained an accuracy score of about 86%.

Possible recommendations:

- Reduce number of features
- Address outliers (This could be attributed to the use of a Kaggle dataset)

THANK YOU RYAN AND ANDREW!