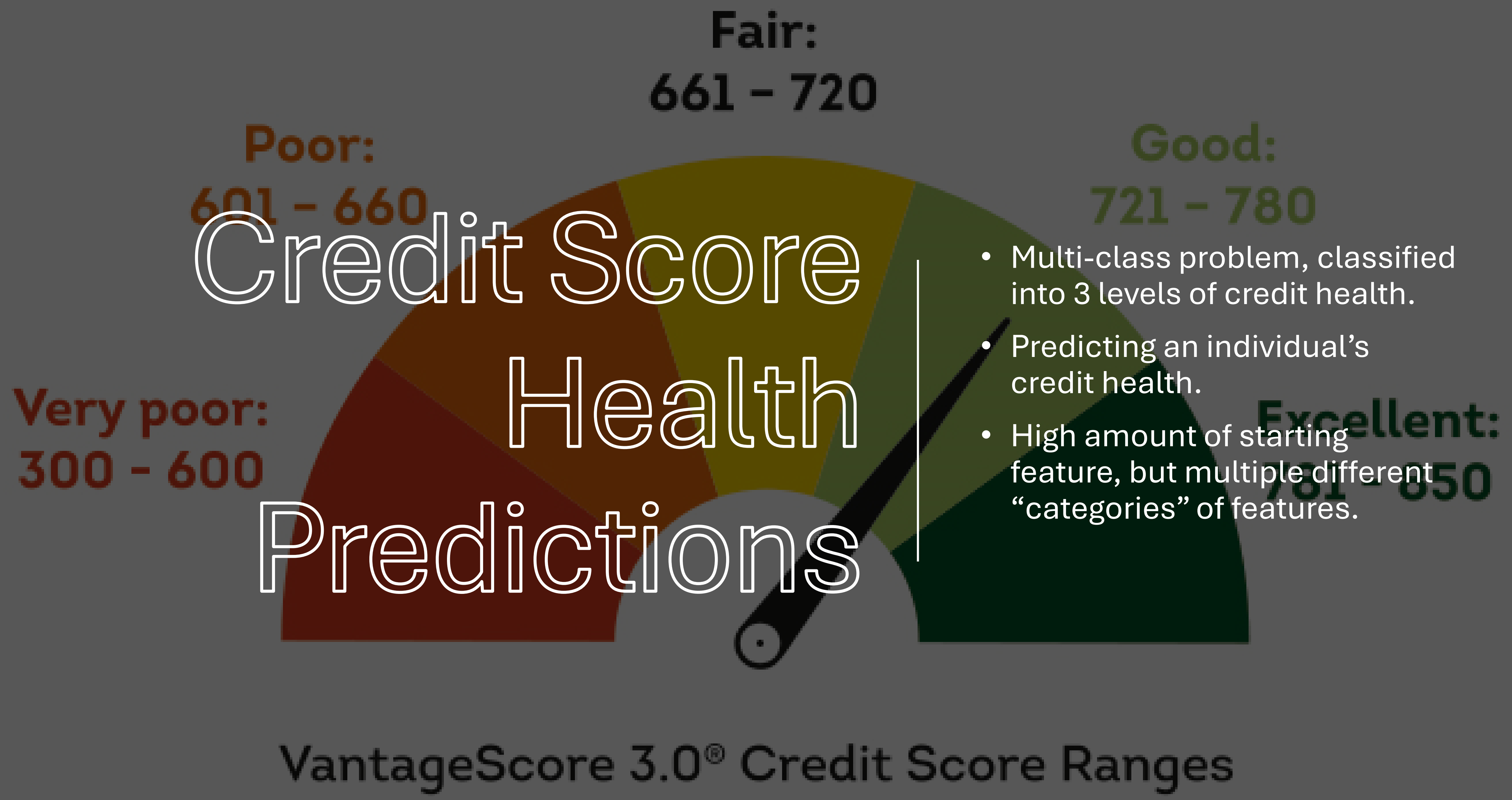
Credit Score Health Predictions

- Multi-class problem, classified into 3 levels of credit health.
- Predicting an individual's credit health.
- High amount of starting feature, but multiple different "categories" of features.



Dataset At A Glance

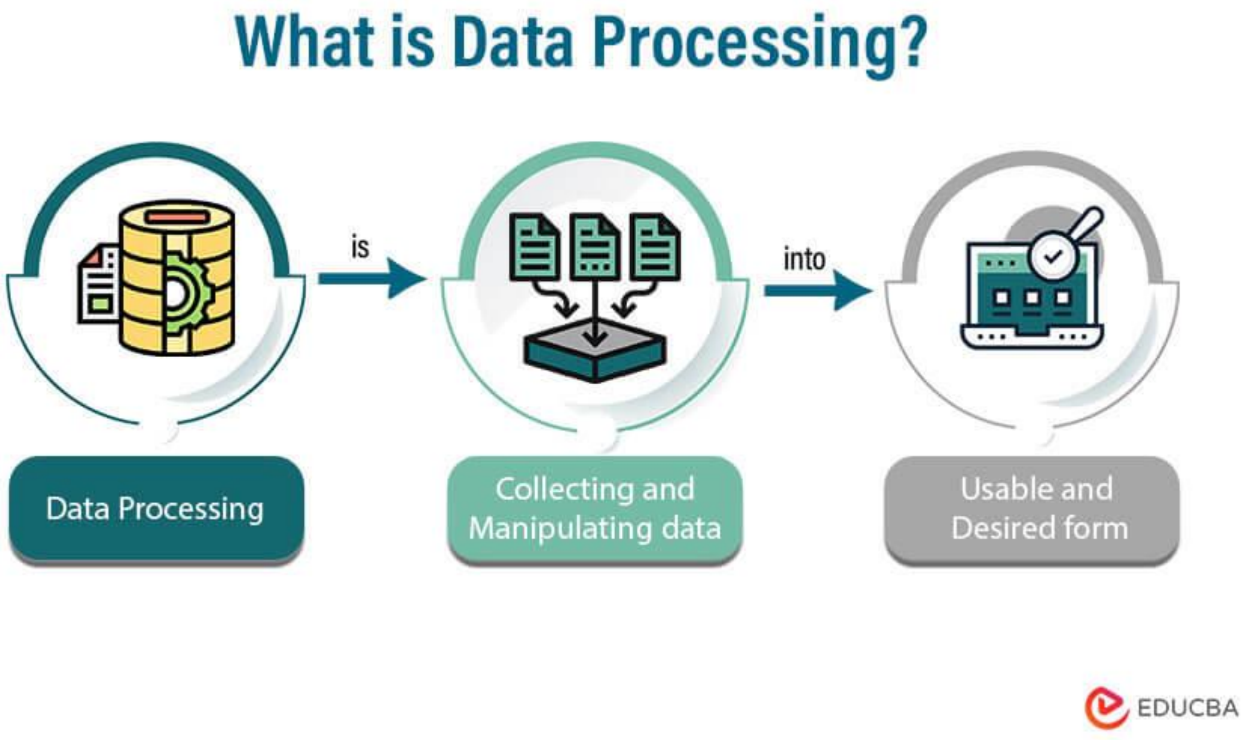
- **100,000 observations!**
- **24 different features**
- **Multiple ML Methods Used**
- **Synthetic Sampling (SMOTE)**
- **Interpretation of the model statistics through SHAP**

Feature Bundles

- **User Info**
- **# of Cards/Accounts/Loans**
- **Income by year/month & Monthly Balance**
- **Interest Rates**
- **# of delayed payments & duration**

Preprocessing

- Recasting column types
- Filling missing values & removing negative values
- Replacing/removing text in observations
- Encoding categorical variables
- Removing extreme outliers that heavily skew the data/are unrealistic (Ex. Age 1000+).



After Preprocessing

- 0 missing values!
- 31000~ observations
- More normalized distributions/lower standard deviation
- Solved column-type pains.

Engineered Features

Choices made to make a more parsimonious model, “bundling” related features.

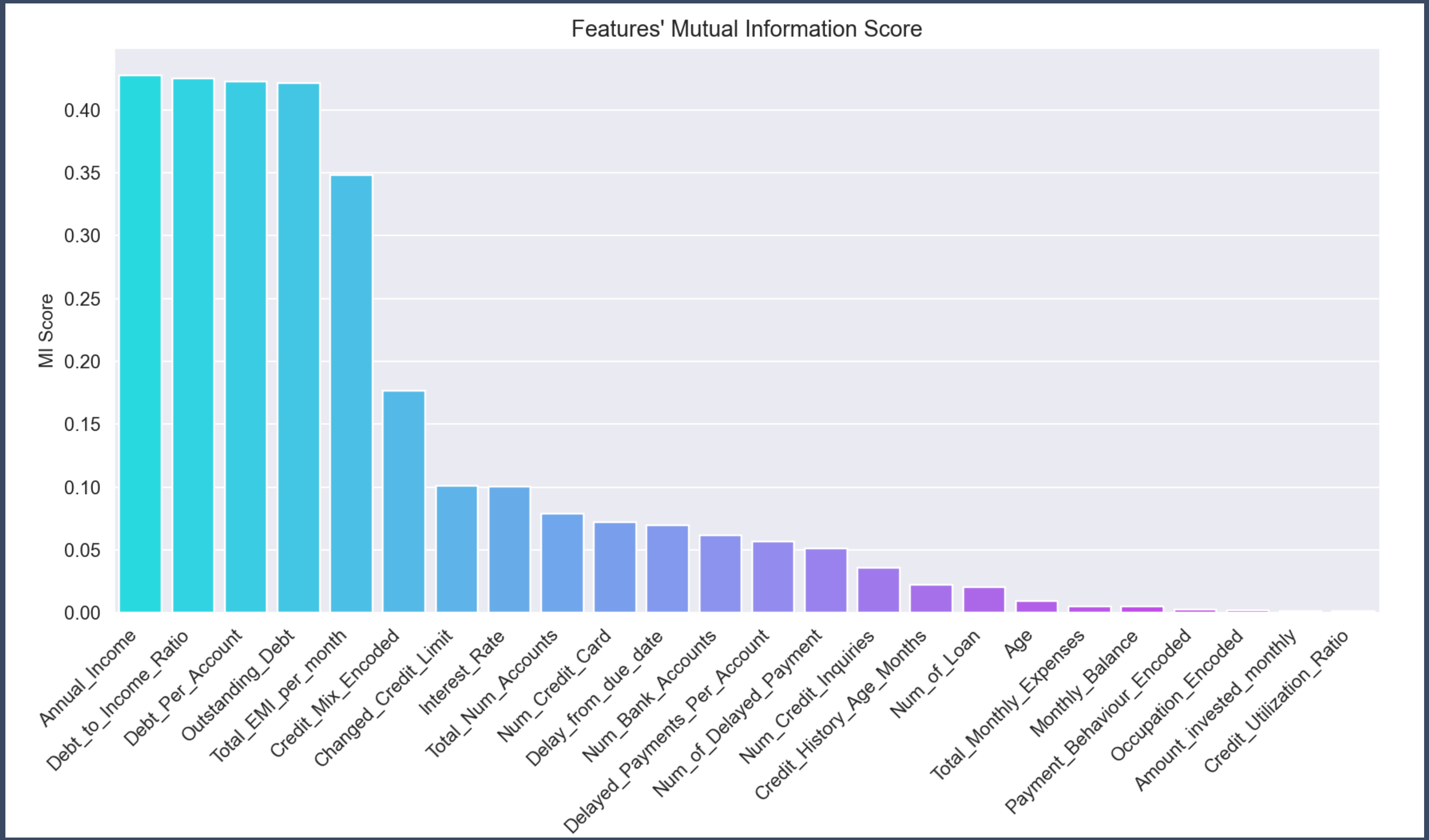
```
# Feature Engineering
##Calculate the total number of accounts (Bank Accounts + Credit Cards)
file1['Total_Num_Accounts'] = file1['Num_Bank_Accounts'] + file1['Num_Credit_Card']
##Calculate the total outstanding debt per account
file1['Debt_Per_Account'] = file1['Outstanding_Debt'] / file1['Total_Num_Accounts']
##Calculate the ratio of outstanding debt to annual income
file1['Debt_to_Income_Ratio'] = file1['Outstanding_Debt'] / file1['Annual_Income']
##Calculate the total number of delayed payments per account
file1['Delayed_Payments_Per_Account'] = file1['Num_of_Delayed_Payment'] / file1['Total_Num_Accounts']
##Calculate the total monthly expenses (EMI + Monthly Investments)
file1['Total_Monthly_Expenses'] = file1['Total_EMI_per_month'] + file1['Amount_invested_monthly']
```

Backwards Stepwise Feature Selection

- Narrowed down to 5 features
- Heavy Considerations with Runtime
- Done with random foresting as a classifier wrapper.
- Sample run below:

```
[2024-06-03 17:47:24] Features: 7/5 -- score: 0.7119267238598981
[2024-06-03 17:47:25] Features: 6/5 -- score: 0.7040572792362768
[2024-06-03 17:47:26] Features: 5/5 -- score: 0.6913823131006902
Selected Features:
('Num_Credit_Card', 'Delay_from_due_date', 'Num_of_Delayed_Payment', 'Amount_invested_monthly', 'Credit_Mix_Encoded')
Feature Selection has been applied to df object!
```

Mutual Information Score (MI Score)



- Statistic that determines how useful each feature is in the model.
- Displaying scores with Seaborn
- Feature Importance vs Feature Selection

Feature Selection

MI SCORE: HARD CUTOFF OF 0.2 USED, SIGNIFICANT FALLOFF IN MI AFTERWARDS. CONSISTENT WITH EACH RUN. EFFICIENT RUNTIME

RANDOM FOREST BACKWARD FEATURE SELECTION: HARD CUTOFF OF 5 FEATURES, EXPENSIVE IN TERMS OF RUNTIME.

Scikit Model Overviews

Multiple Classifier Options!

- Random Foresting
- Gradient Boosting (+ XGBoost)
- Neural Networks (MLPClassifier)
- Cross Validation available!

Concerns:

- ROC AUC Curves with Multiclass Classification are currently unavailable
- Computational Runtime Tradeoffs

MSE Comparisons w/ Random Foresting

- Baseline MSE used as a point of reference without cross validation.
- Significant Improvement, Lower than Gradient Boost MSE (0.17~ vs 0.22~)

```
Baseline MSE: 0.3661029100996055
MSE RFRegressor 0.17073801685446247
Comparison Runtime: 8.063578844070435
```

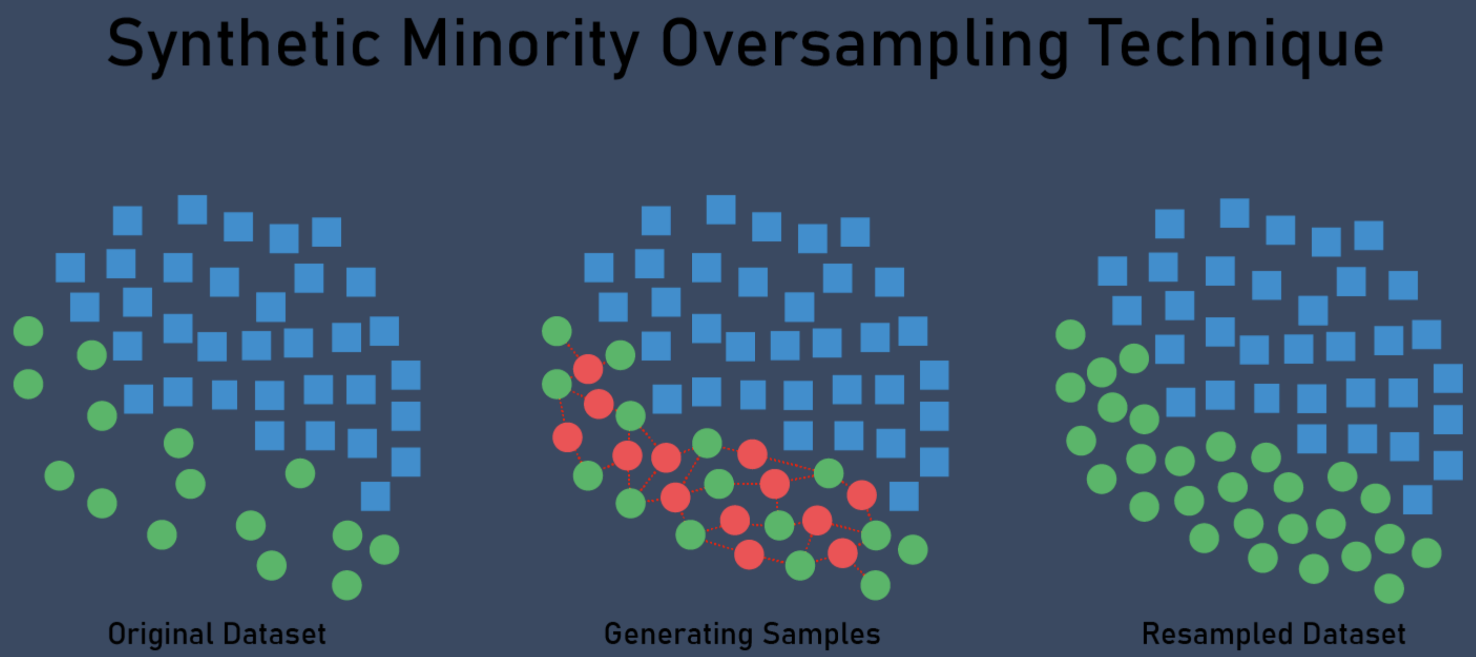
Gradient Boosting & Neural Networks

- Boosting is efficent with large datasets
- Neural Network is computationally cheap with large datasets.
- ROC AUC Curve NaN Issue w/Multiclass

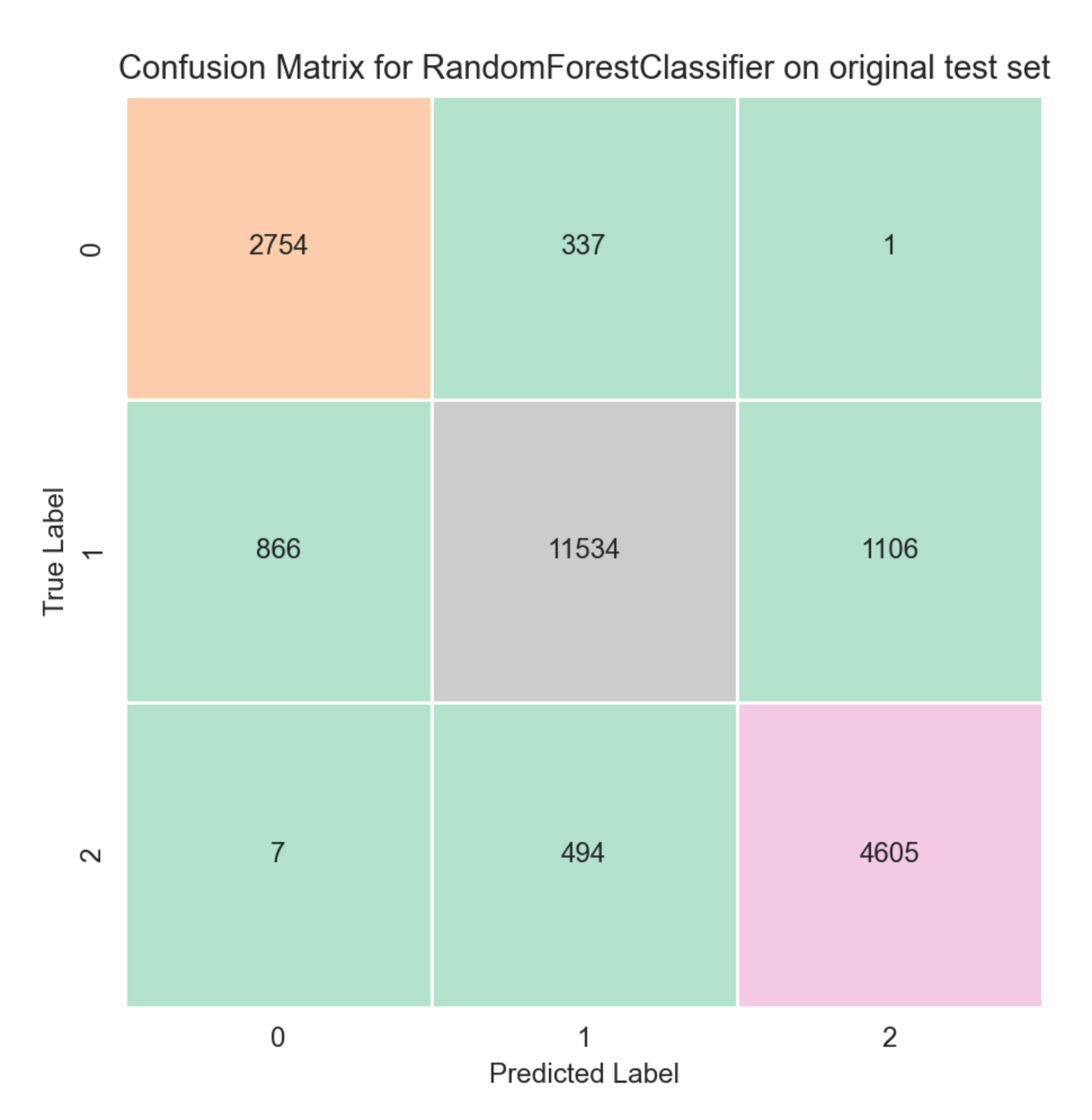
```
Gradient Boost Acc: 0.67 (+/- 0.01)
Gradient Boost AUC: nan (+/- nan)
CV Runtime: 31.037909030914307
Neural Network Acc: 0.62 (+/- 0.00)
Neural Network AUC: nan (+/- nan)
CV Runtime: 0.3123455047607422
```

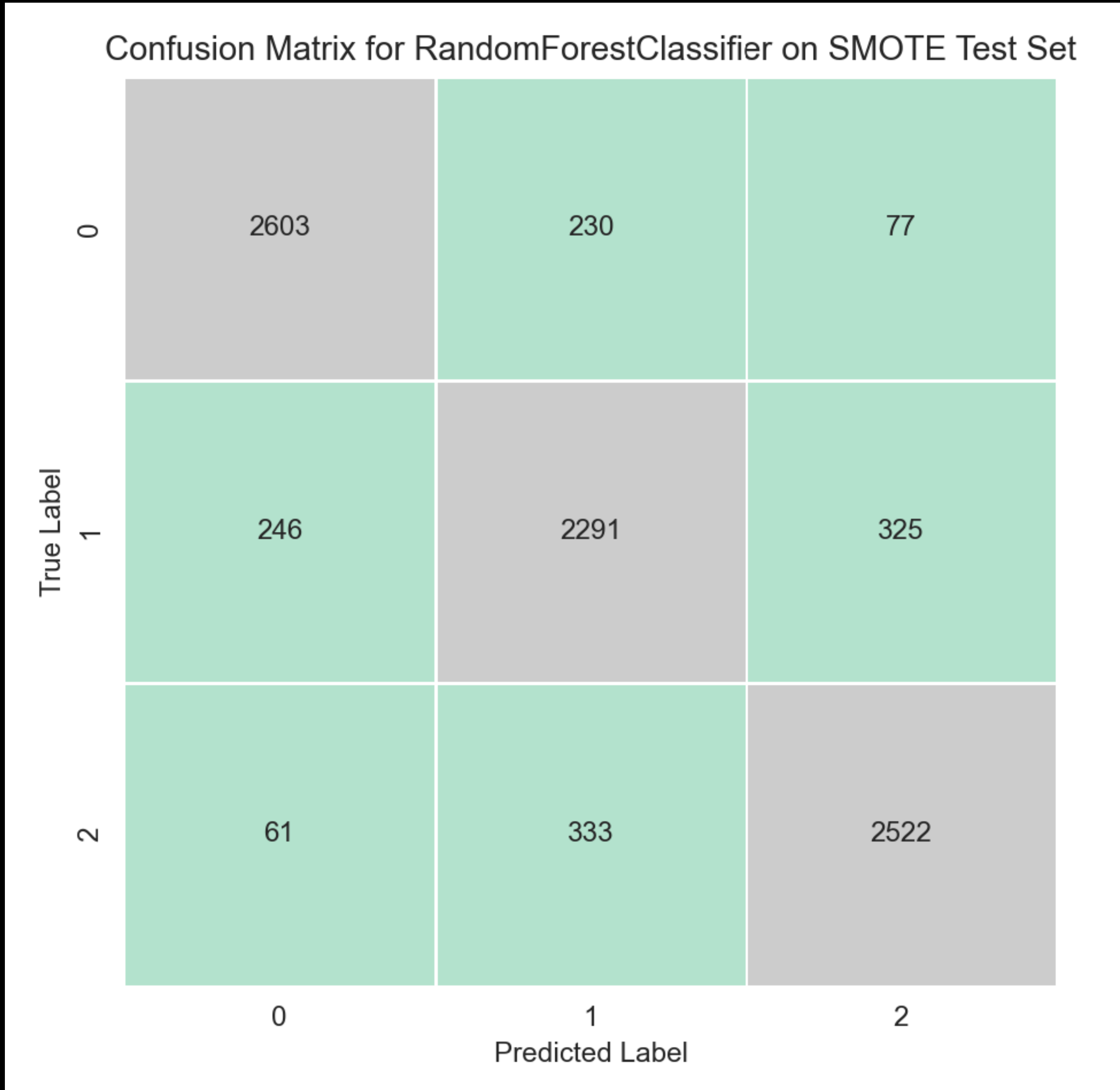
SMOTE

- Handling Class Imbalances (very useful in multiclass)!
- Lowers accuracy score, but simulates outward data (92% vs 85% model accuracy in one sample run).
- SMOTE used in upcoming statistics!



Confusion Matrix w/ RF Classifier





Classification Matrices

Some Notes:

- Lower precision but higher recall & F1 in some original set cases, struggling to predict “negative” cases.
- Support = number of obs. SMOTE set is far more balanced.
- Smote set has the same scores across statistics, hence “negative” case issues.

Classification report for SMOTE test set:

	precision	recall	f1-score	support
0.0	0.89	0.89	0.89	2910
1.0	0.80	0.80	0.80	2862
2.0	0.86	0.86	0.86	2916
accuracy			0.85	8688
macro avg	0.85	0.85	0.85	8688
weighted avg	0.85	0.85	0.85	8688

Classification report for original test set:

	precision	recall	f1-score	support
0.0	0.76	0.89	0.82	3092
1.0	0.93	0.85	0.89	13506
2.0	0.80	0.90	0.85	5106
accuracy			0.87	21704
macro avg	0.83	0.88	0.85	21704
weighted avg	0.88	0.87	0.87	21704

Classification Matrix Runtime: 1.860008716583252

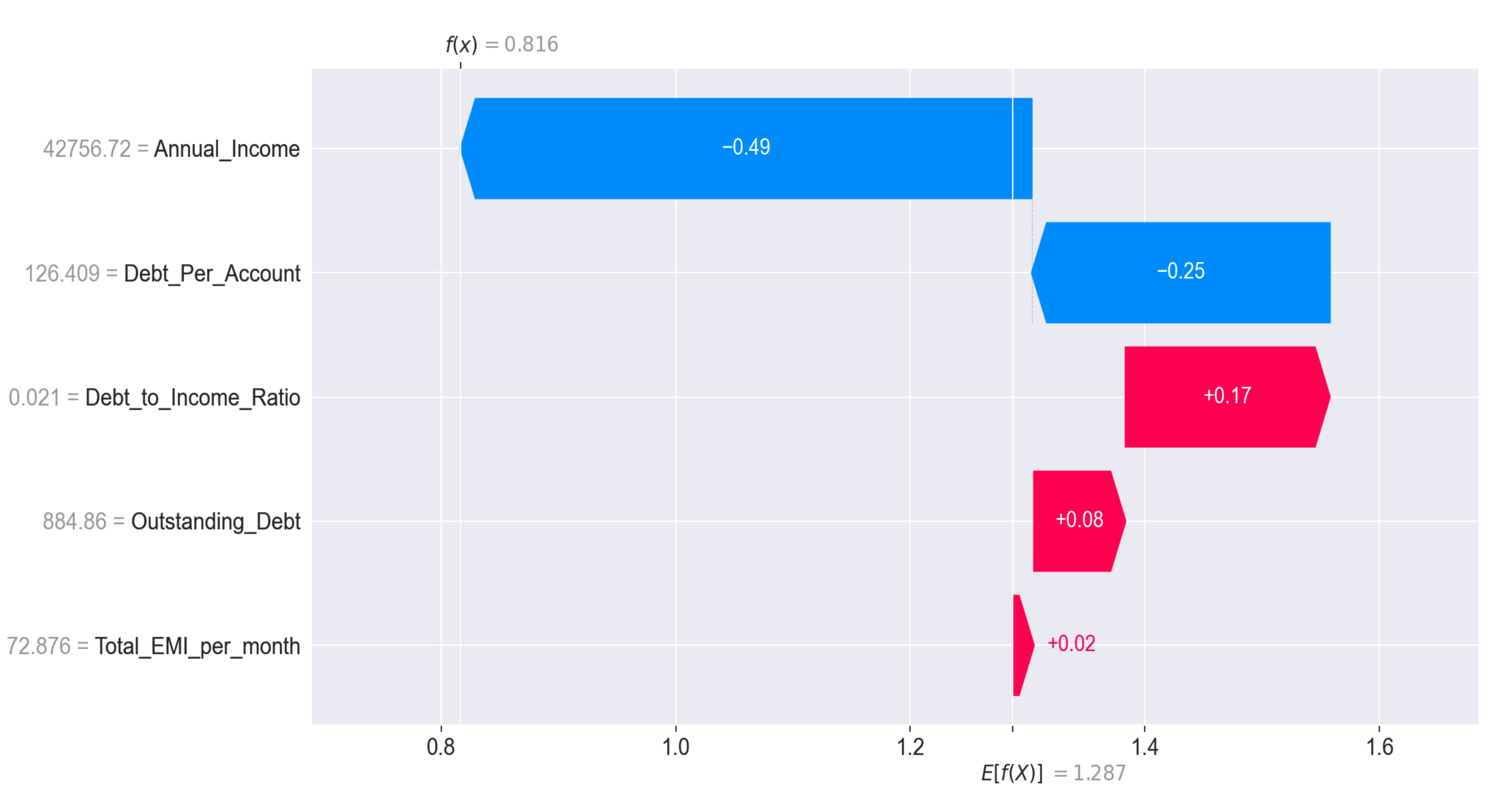
Random Foresting Sticks Out

- High accuracy and low MSE.
- Solid confusion and classification results.
- SMOTE results are promising but small split.

SHAP Show

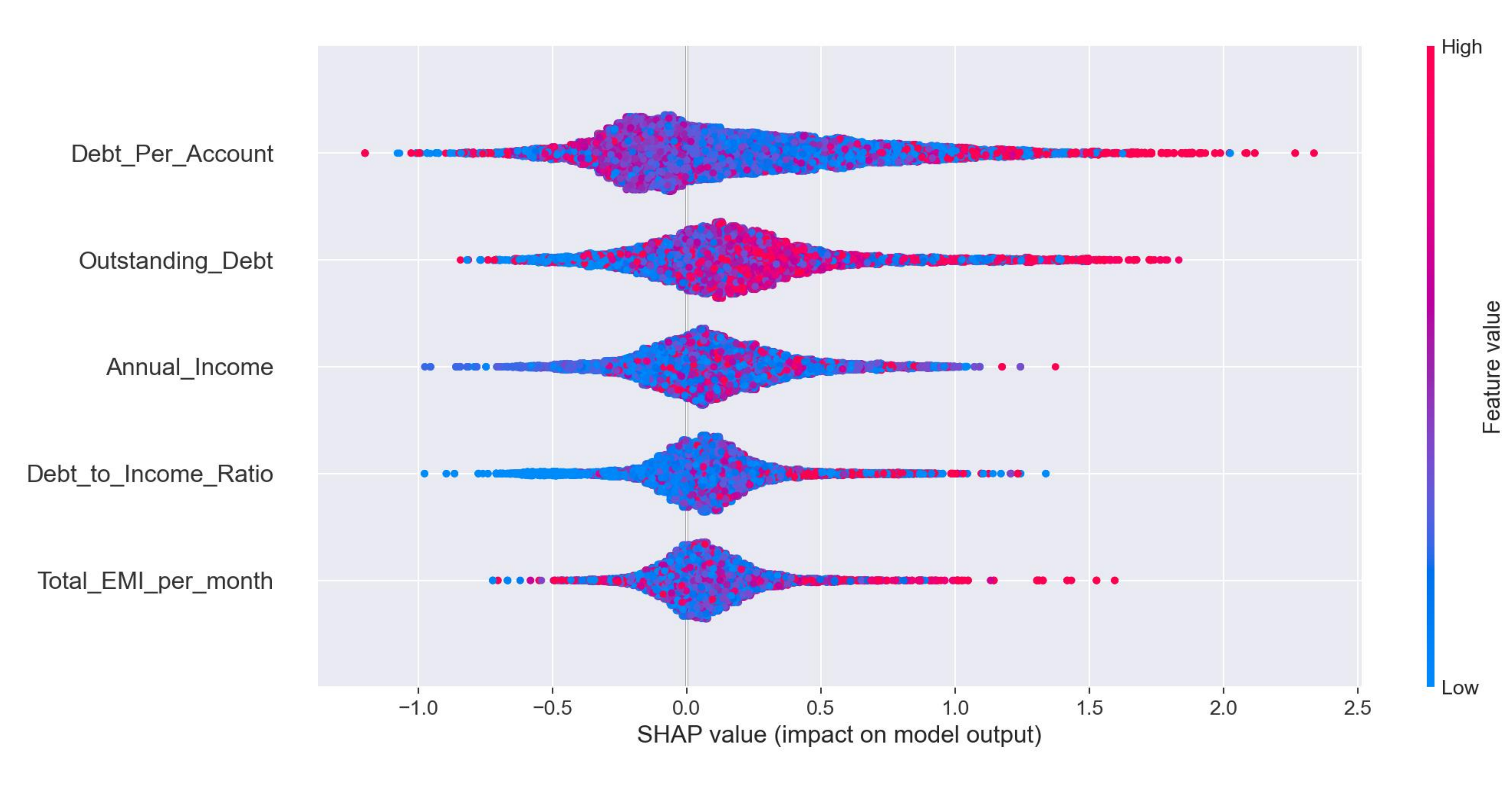
What is SHAP?

- Showcases Features' Impacts
- Extremely powerful interpretation tool.
- Using Multiclass XGBoost Classification.

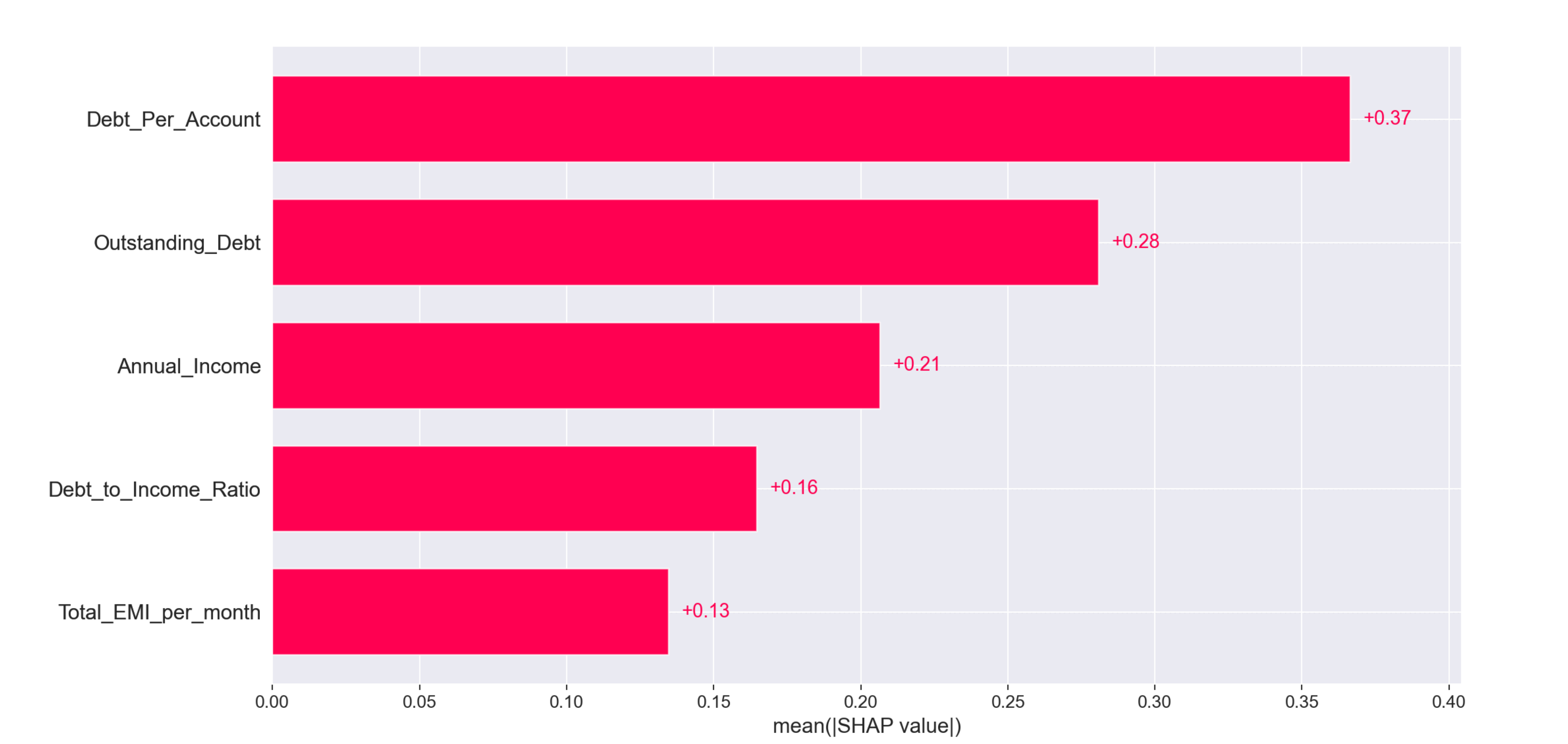


SHAP Show: Beeswarm Plot

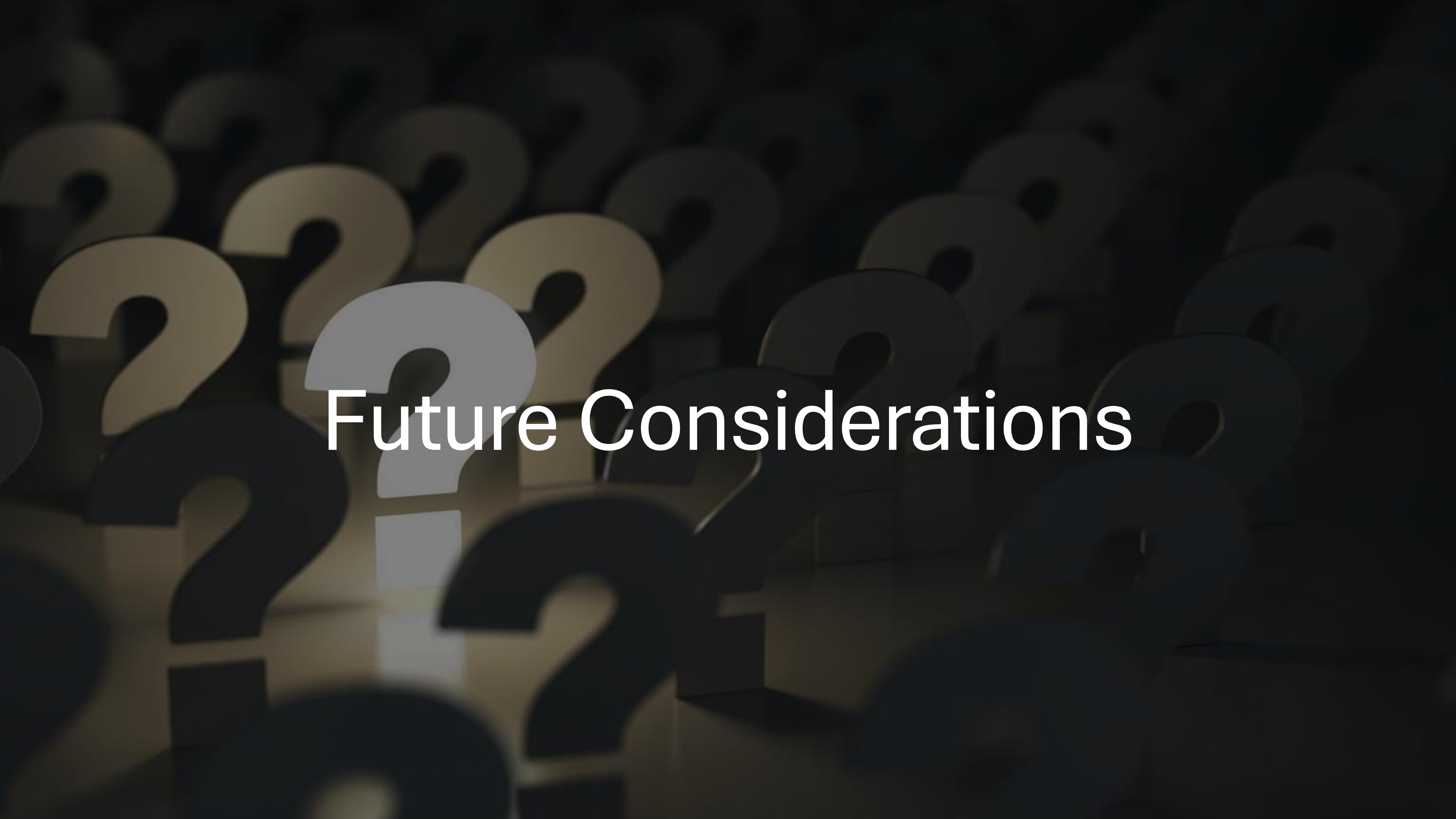
- Pay attention to Debt_Per_Account and Outstanding_Debt's swarm relation!



SHAP Show: Mean Contribution Bar Plot



Wrap-Up



Future Considerations