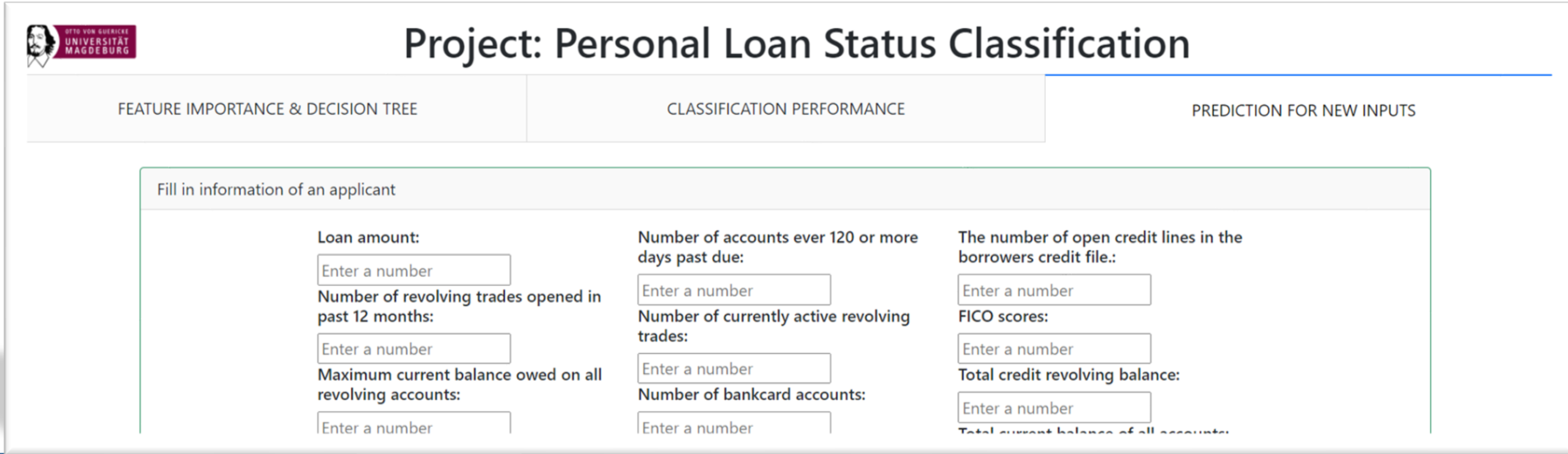
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

IFM LEHRSTUHL FÜR INNOVATIONS- UND FINANZMANAGEMENT

Financial Technology ego.-INKUBATOR



Personal Loan Status Prediction using XGBoost

SACHSEN-ANHALT

EUROPÄISCHE UNION
EFRE
Europäischer Fonds für regionale Entwicklung

Der ego.-INKUBATOR „Financial Technology (FINTECH)“ wird durch das Ministerium für Wissenschaft, Wirtschaft und Digitalisierung des Landes Sachsen-Anhalt mit Mitteln des Europäischen Fonds für regionale Entwicklung (EFRE) gefördert.

PROBLEM STATEMENT
&
UNDERSTANDING DATASET

Problem Statement

- The collection of the dataset
  - LendingClub Notes platform
  - Duration: 2007-2020 | USD | Kaggle
- Content of the data
  - (1) Demographic data such as employment status, employment length and house ownership,
  - (2) Loan Characteristics such as loan amount, term and purpose and
  - (3) Behavioral data related to historical payments
- Dimension: 2925493 rows and 141 columns
- Data choosen for this project: Sub-sample of 700000/2925493

Explanatory Data Analysis (EDA)

Univariate EDA

- Numerical data
- Categorical data

Bivariate EDA

Loan status

Loan amount

Purpose

Interest rates

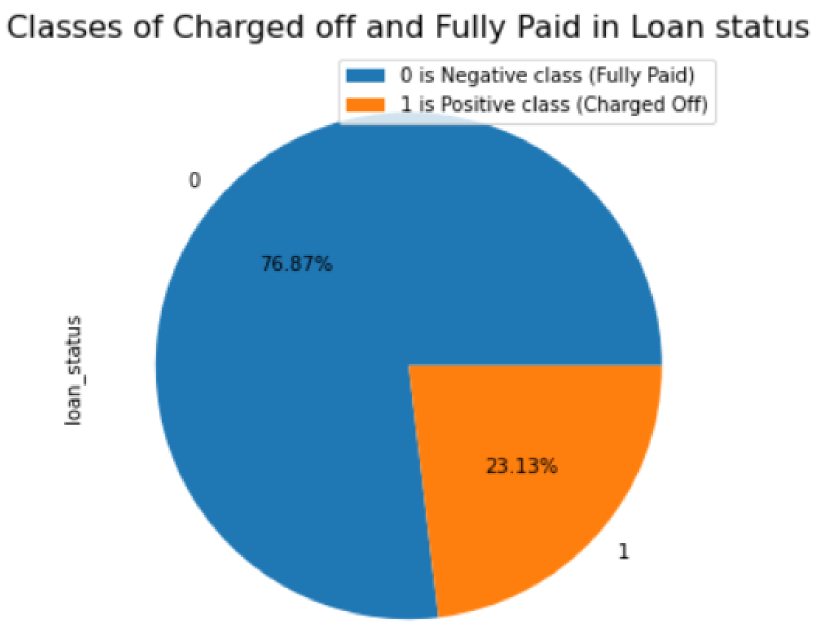
Conclusion of EDA

EDA Conclusion

- Most of accepted borrowers: a **good** profile (good average FICO scores and no public records of bankruptcies), mainly with **financial reasons** (Debt consolidation and credit card).
- Most borrowers: a shorter loan term (**36 months**, compared to 60 months) and do **not** enroll into the **hardship** plan, but the percentage of people who are Charged Off and default accounts for nearly **one third** of the data sample.
- Checking bivariance relationship **cannot** show that the features such as annual income, verified status, house ownership, credit line, etc. have a specific effect on Charged Off, Default or Late status.
- People with **lower interest rate** and **lower loan amount** likely to pay off loans successfully.

Data Preparing

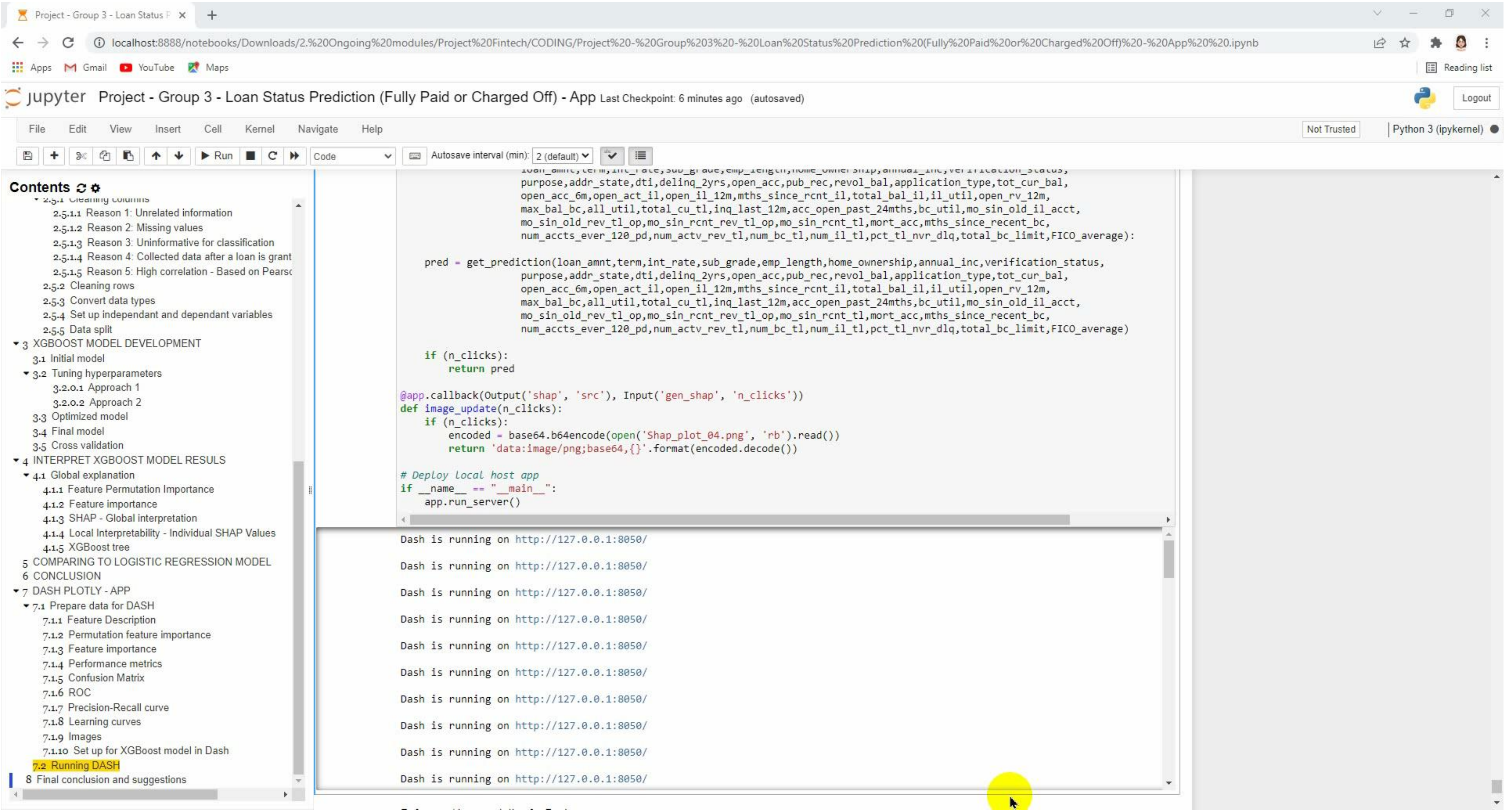
- Cleaning columns
- Cleaning rows
- Convert data types (one-hot coding for setting dummies)
- Set up predictors (43 variables) and responding variable (loan_status)
- Data split: Stratified style with proportion of train-test set (70%-30%)
- Imbalanced data:



DASH Plotly (APP)

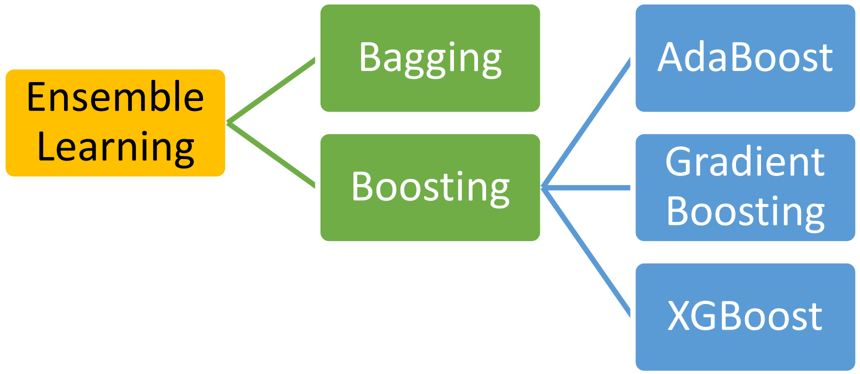
- It is a localhost web
- Conducted using DASH plot library
- Has 3 different tabs

DASH Plotly (APP) – Link



HOW XGBOOST WORKS

Overview



- Ensemble algorithm - creating a model by combining a number of baser learners
- Boosting: Sequential ensemble

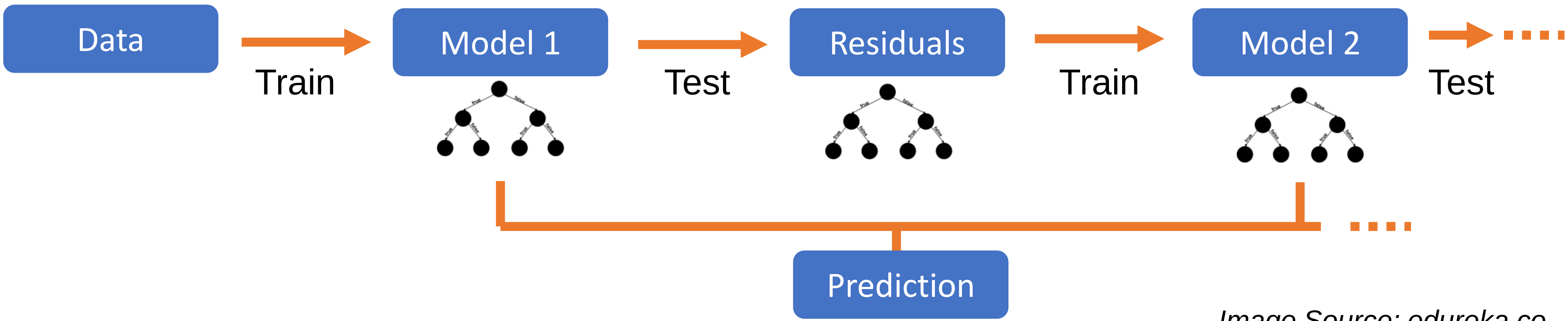


Image Source: edureka.co

XGBoost math in brief

- Extreme gradient boosting (XGBoost) is a decision-tree-based ensemble algorithm
- Log loss function: the negative log-likelihood for **Binary Classification** (Daniel & Martin(2021). Speech and Language Processing. [5.pdf (stanford.edu)](). P.152)

$$L(y_i, p_i) = -[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

- The general objective function *(Chen & Guestrin, 2016)* is:

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_K \Omega(f_k)$$

fk is an independent tree,
T is the number of leaves in a tree
w is leaf weight,
γ and λ are hyperparameters.

a loss function measuring how well model fit on the training data

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

regularization to measure the complexity of trees

How XGBoost works

Fomulas source (in this page): Tianqi Chen and Carlos Guestrin. 2016.

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_\kappa \Omega(f_k)$$

Boosting

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

Plug in

$$\mathcal{L}^{(t)} = \sum_{j=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)$$

Taylor approximation $f(x + \Delta x) \cong f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)\Delta x^2$

$$\mathcal{L}^{(t)} \cong \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

How XGBoost works

Fomulas source (in this page): Tianqi Chen and Carlos Guestrin. 2016.

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

Plug in

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}[g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$
$$= \sum_{j=1}^{T}[(\sum_{i \in I_j} g_i)w_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda)w_j^2] + \gamma T$$

Solve the quadratic function of w

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

Output, minimized objective function

$$\tilde{\mathcal{L}}^{(t)} = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T$$

How XGBoost grows trees

- XGBoost begins with a weak learner

- Loop from 1 to k:
  - Build the first tree
  - Learn the structure and use the minimized objective function
  - To avoid overfitting, learning rate and gamma is added to each tree.
  - Final model = all of the trees are combined additively

- The process stops when:
  - Gain score becomes negative (cannot gain further information from splitting)
  - fixed number of the iteration (k) is reached

Advantages of XGBoost

- Source: (Chen & Guestrin, 2016)
- For a big data sample:
  - sparsity-aware split finding algorithm to handle the problem of missing values in the data
  - For a big dataset – training speed: fast

- In practice:
  - Python package of XGBoost
  - Parallelization of tree construction
  - Out-of-Core Computing
  - Cache Optimization

Results

Model parameter:

```
%%time
xgb_model = xgb.XGBClassifier(objective='binary:logistic',
                              learning_rate=0.05,
                              scale_pos_weight=3,
                              n_estimators=200,
                              max_depth=5,
                              min_child_weight=1,
                              gamma=0.5,
                              colsample_bytree=1,
                              subsample=1,
                              use_label_encoder=False,
                              random_state=42)

xgb_model.fit(x_train, y_train,
              verbose=0,
              early_stopping_rounds=10,
              eval_metric='aucpr',
              eval_set=[(x_train, y_train), (x_test, y_test)])
```

- Development:
  - Initial mode
  - optimized model (tunning hyperparameters)
  - final mode
  - Each model is evaluated with confusion matrix, metrics scores...

- Trade-off: Bias vs Variance

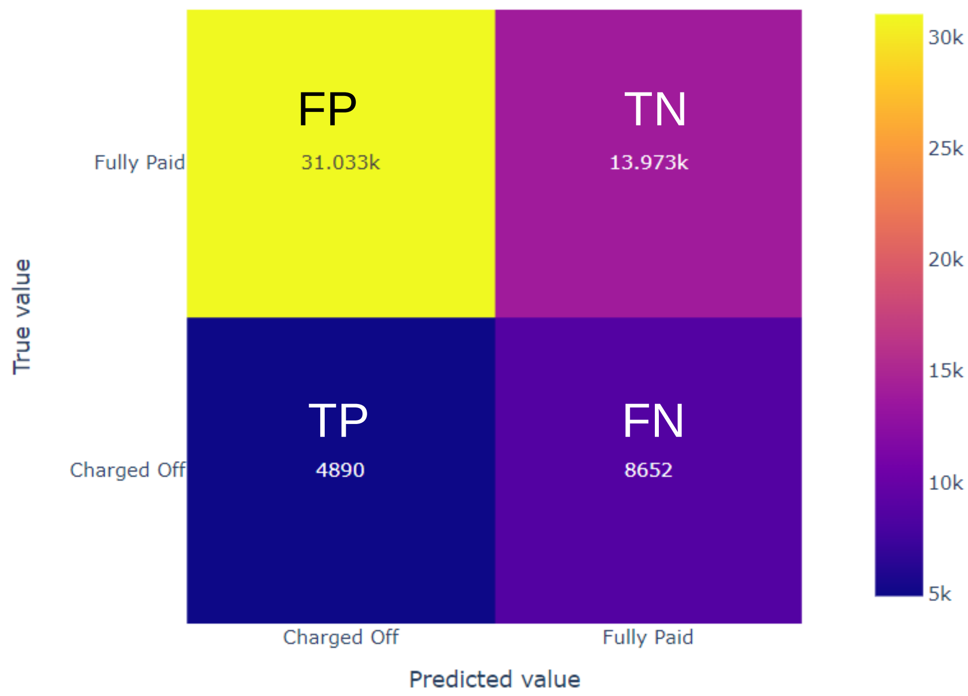
Results

1) CLASSIFICATION REPORT:

	precision	recall	f1-score	support
0	0.86	0.69	0.77	45006
1	0.38	0.64	0.48	13542
accuracy			0.68	58548
macro avg	0.62	0.66	0.62	58548
weighted avg	0.75	0.68	0.70	58548

- Target: High recall for positive class & high accuracy
- Cost of TP and FN
- Trade-off: precision vs recall score

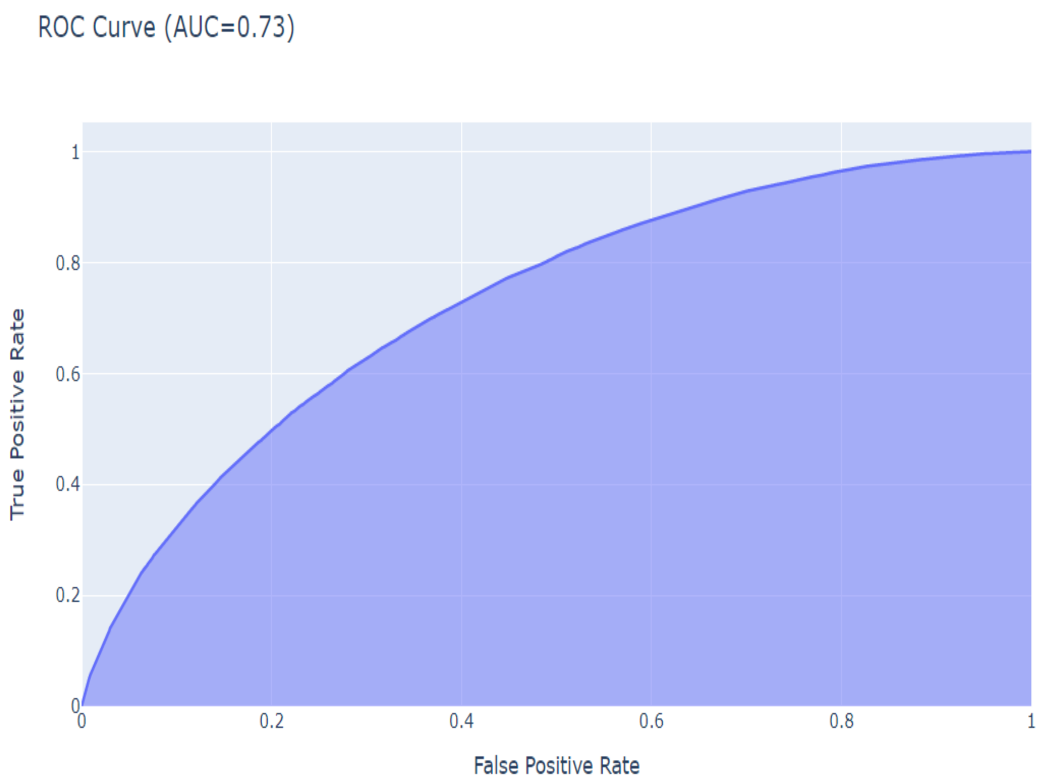
2) CONFUSION MATRIX

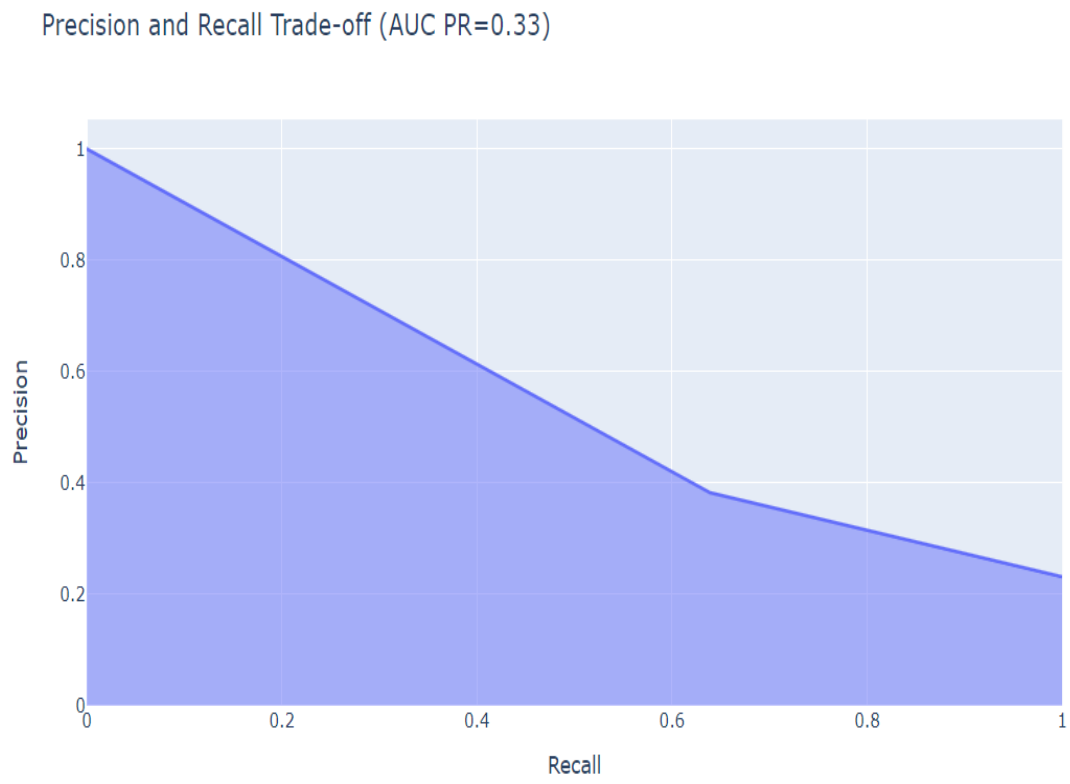


$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$precision = \frac{TP}{TP + FP}; \quad recall = \frac{TP}{TP + FN}$$

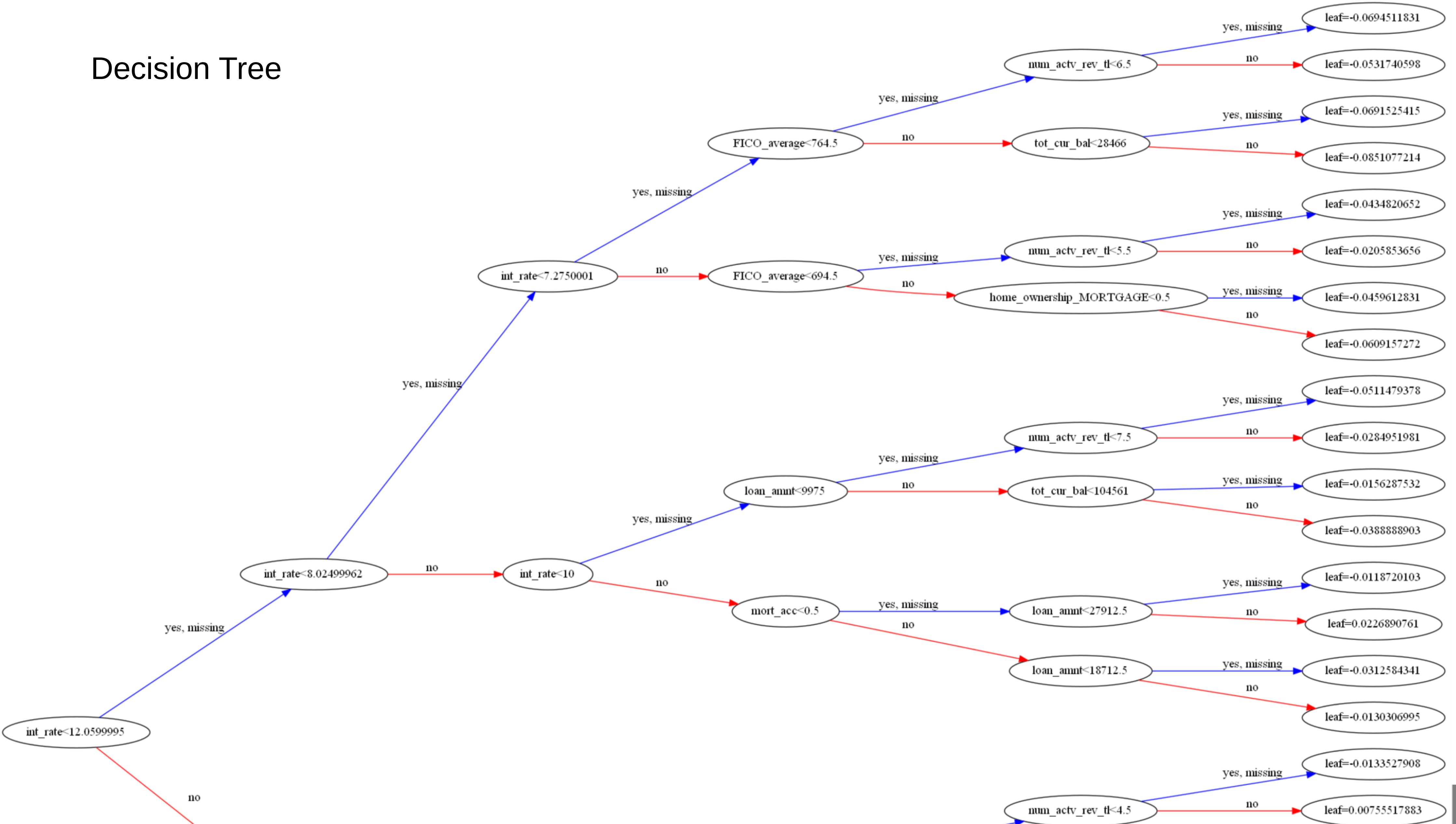
Results


ROC Curve (AUC=0.73)
True Positive Rate
False Positive Rate
0
0.2
0.4
0.6
0.8
1


Precision and Recall Trade-off (AUC PR=0.33)
Precision
Recall
0
0.2
0.4
0.6
0.8
1

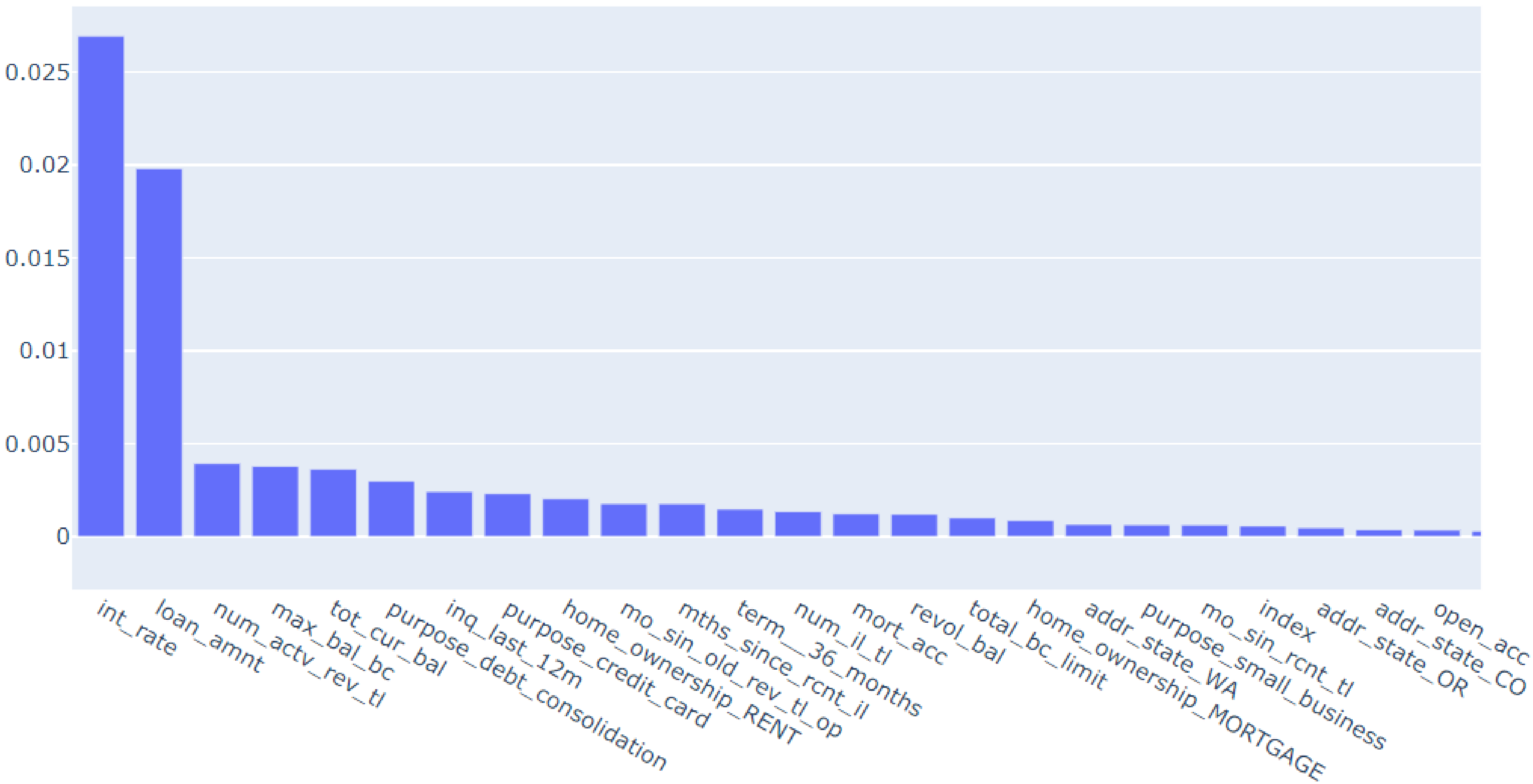
Explanation of the model

Decision Tree



Explanation of the model

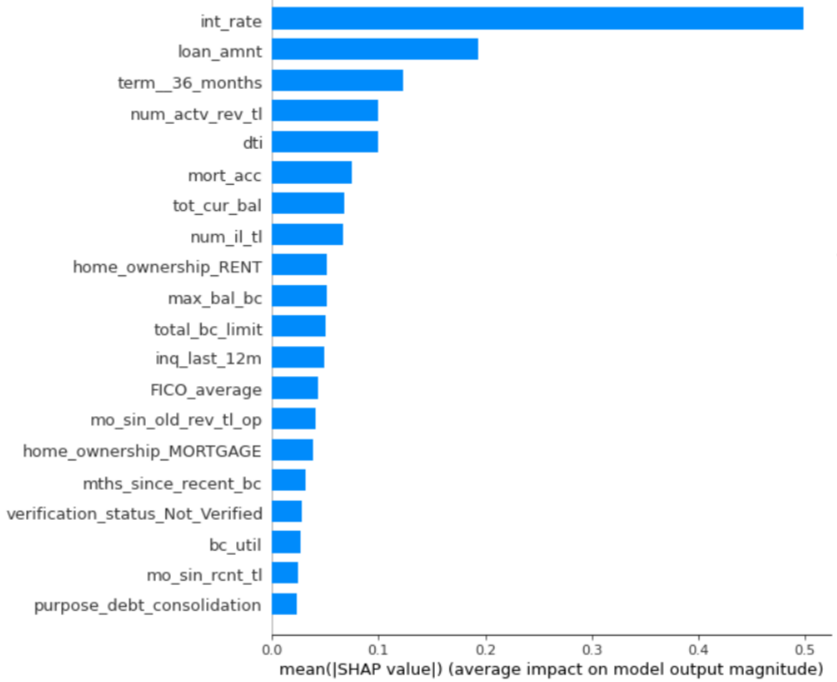
Permutation feature importance



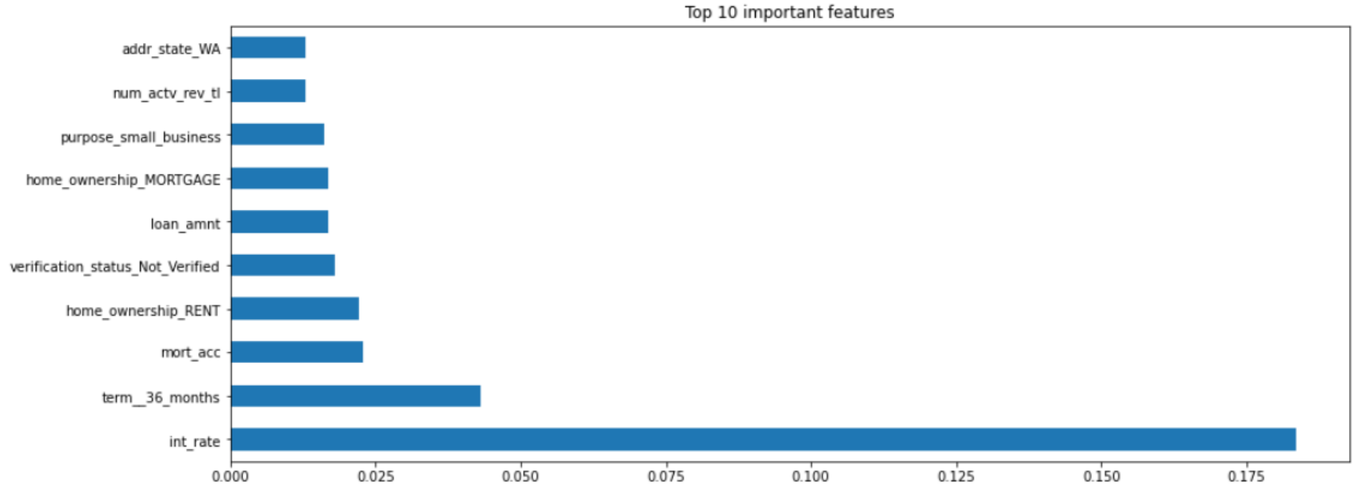
Explanation of the model

Feature Importance

a) Based on Shap values **(Lundberg & Lee, 2017)**



b) Based on Gain values



$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

Source: Tianqi Chen and Carlos Guestrin. 2016.

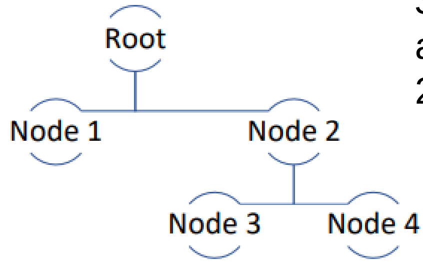
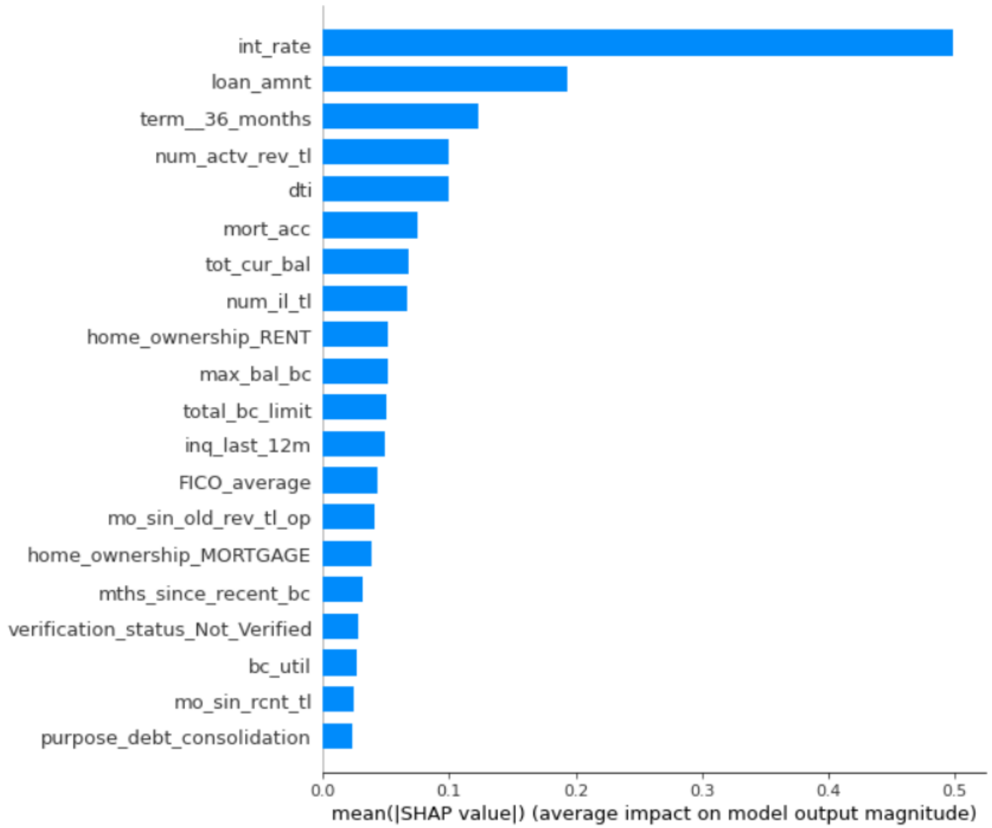


Figure 2.1: An example tree showing a node split

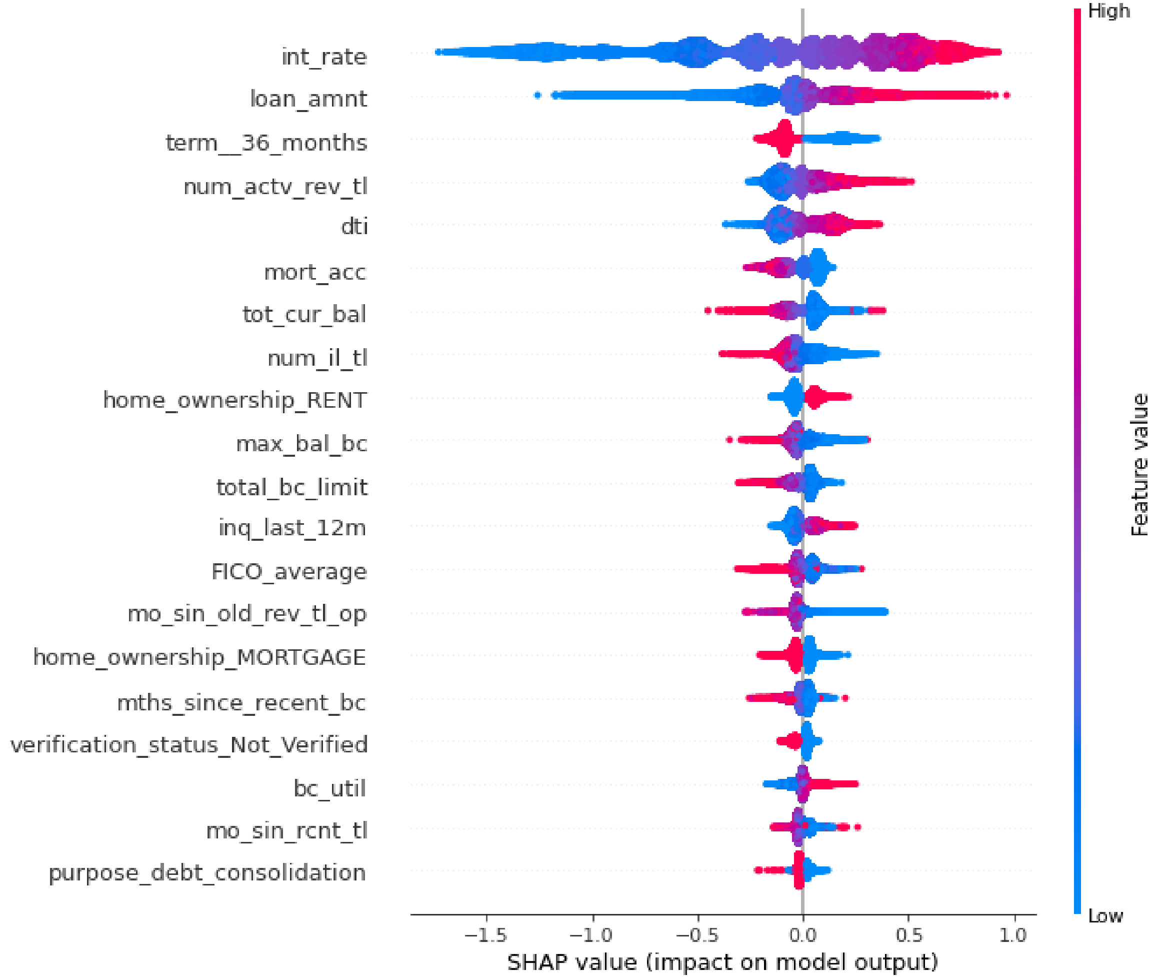
Explanation of the model

Feature Importance based on Shap values

Bar chart

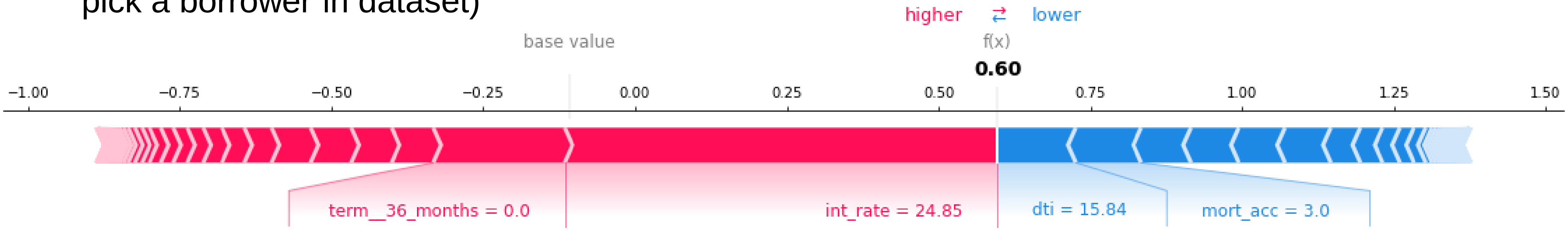


Density Scatter plot



Explanation of the model

Local explanation using SHAP values (Lundberg & Lee, 2017) of the 9th borrower (randomly pick a borrower in dataset)



- Ouput value: 0.6 – convert to probability is 0.645 - Charged Off
- Base value
- Color (Blue/Red)
- Manitude of each feature

$$\frac{1}{1+e^{-y^{\wedge}}}$$

Source: Daniel & Martin (2021). Speech and Language Processing. https://web.stanford.edu/~jurafsky/slp3/5.pdf P.3)

Improvement

- Increase model performance by:
  - Training on larger subsample of data (due to hardware capacity, only 700000 rows are used)
  - Further EDA with informative features – more distinguishing features
  - Tunning hyperparameters
- End-users: need knowledge of Machine Learning to read the dashboard

THANK YOU!

Q&A