

Project: Personal Loan Status Classification

FEATURE IMPORTANCE & DECISION TREE
CLASSIFICATION PERFORMANCE
PREDICTION FOR NEW INPUTS

Fill in information of an applicant

Loan amount: <input type="text" value="Enter a number"/>	Number of accounts ever 120 or more days past due: <input type="text" value="Enter a number"/>	The number of open credit lines in the borrowers credit file.: <input type="text" value="Enter a number"/>
Number of revolving trades opened in past 12 months: <input type="text" value="Enter a number"/>	Number of currently active revolving trades: <input type="text" value="Enter a number"/>	FICO scores: <input type="text" value="Enter a number"/>
Maximum current balance owed on all revolving accounts: <input type="text" value="Enter a number"/>	Number of bankcard accounts: <input type="text" value="Enter a number"/>	Total credit revolving balance: <input type="text" value="Enter a number"/>
		Total current balance of all accounts: <input type="text"/>

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 chosen 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

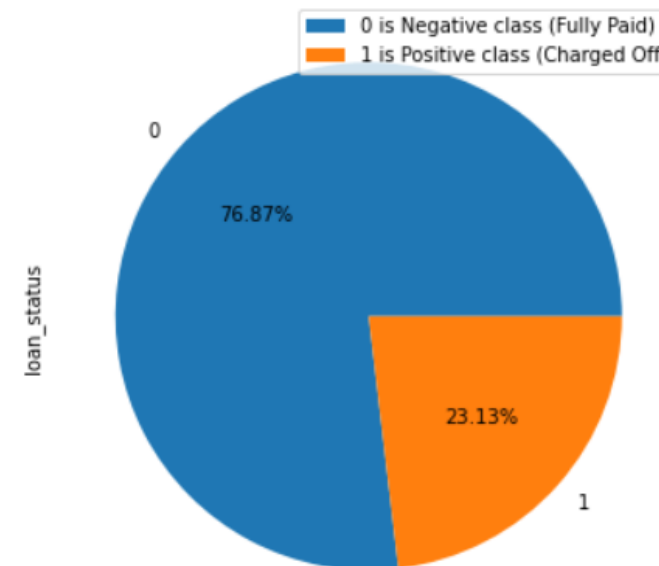
EDA Conclusion

- 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 bivariate 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:

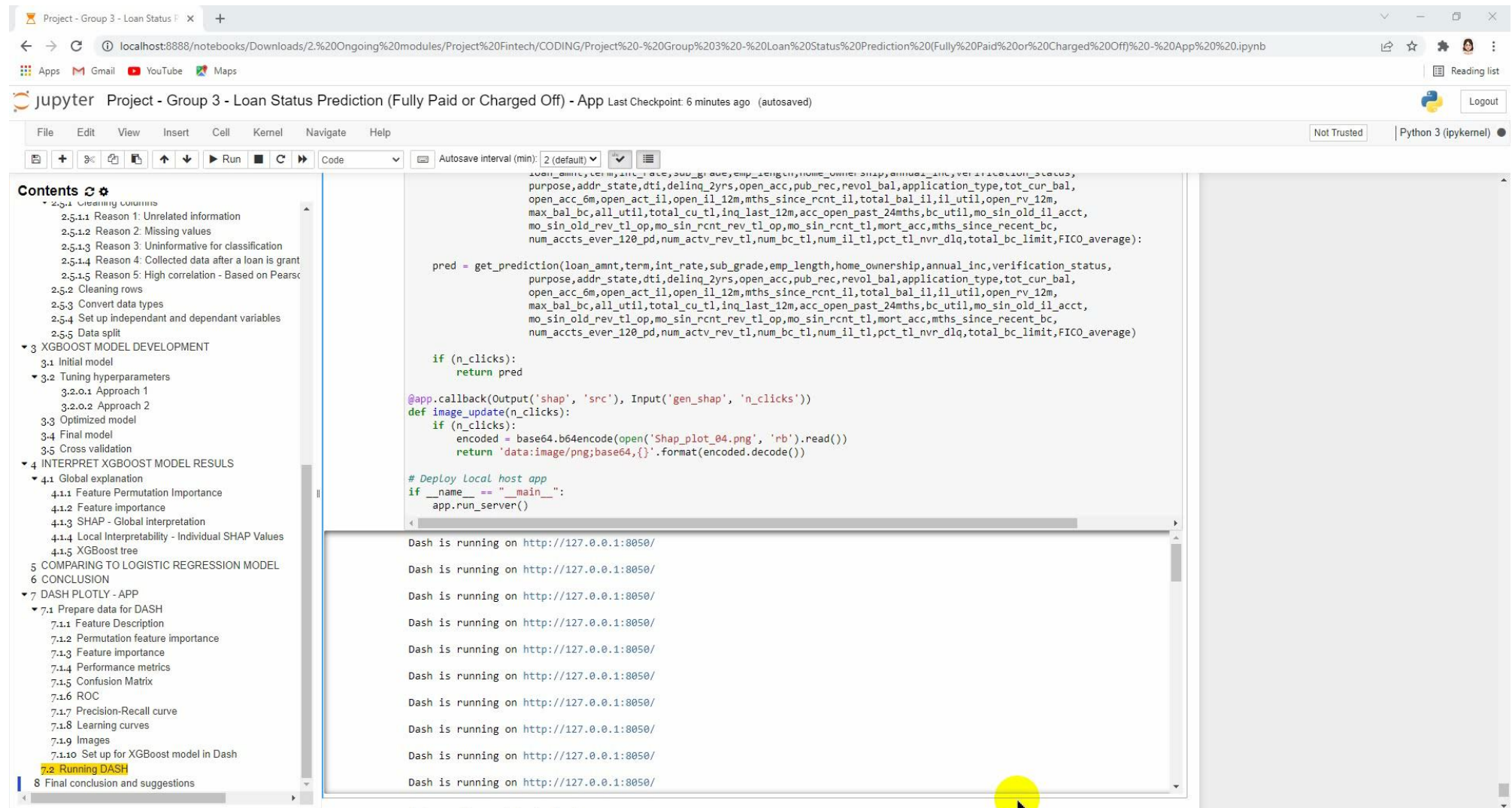
Classes of Charged off and Fully Paid in Loan status



DASH Plotly (APP)

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

DASH Plotly (APP) – [Link](#)



Project - Group 3 - Loan Status Prediction (Fully Paid or Charged Off) - App Last Checkpoint: 6 minutes ago (autosaved)

File Edit View Insert Cell Kernel Navigate Help

Autosave interval (min): 2 (default)

Not Trusted Python 3 (ipykernel)

Contents

- 2.5.1.1 Reason 1: Unrelated information
- 2.5.1.2 Reason 2: Missing values
- 2.5.1.3 Reason 3: Uninformative for classification
- 2.5.1.4 Reason 4: Collected data after a loan is grant
- 2.5.1.5 Reason 5: High correlation - Based on Pearsc
- 2.5.2 Cleaning rows
- 2.5.3 Convert data types
- 2.5.4 Set up independent and dependant variables
- 2.5.5 Data split
- 3 XGBOOST MODEL DEVELOPMENT
 - 3.1 Initial model
 - 3.2 Tuning hyperparameters
 - 3.2.0.1 Approach 1
 - 3.2.0.2 Approach 2
 - 3.3 Optimized model
 - 3.4 Final model
 - 3.5 Cross validation
- 4 INTERPRET XGBOOST MODEL RESULTS
 - 4.1 Global explanation
 - 4.1.1 Feature Permutation Importance
 - 4.1.2 Feature importance
 - 4.1.3 SHAP - Global interpretation
 - 4.1.4 Local Interpretability - Individual SHAP Values
 - 4.1.5 XGBoost tree
- 5 COMPARING TO LOGISTIC REGRESSION MODEL
- 6 CONCLUSION
- 7 DASH PLOTLY - APP
 - 7.1 Prepare data for DASH
 - 7.1.1 Feature Description
 - 7.1.2 Permutation feature importance
 - 7.1.3 Feature importance
 - 7.1.4 Performance metrics
 - 7.1.5 Confusion Matrix
 - 7.1.6 ROC
 - 7.1.7 Precision-Recall curve
 - 7.1.8 Learning curves
 - 7.1.9 Images
 - 7.1.10 Set up for XGBoost model in Dash
 - 7.2 Running DASH
- 8 Final conclusion and suggestions

```

loan_amnt, term, int_rate, sub_grade, emp_length, home_ownership, annual_inc, verification_status,
purpose, addr_state, dti, delinq_2yrs, open_acc, pub_rec, revol_bal, application_type, tot_cur_bal,
open_acc_6m, open_act_il, open_il_12m, mths_since_rcnt_il, total_bal_il, il_util, open_rv_12m,
max_bal_bc, all_util, total_cu_tl, inq_last_12m, acc_open_past_24mths, bc_util, mo_sin_old_il_acct,
mo_sin_old_rev_tl_op, mo_sin_rcnt_rev_tl_op, mo_sin_rcnt_tl, mort_acc, mths_since_recent_bc,
num_accts_ever_120_pd, num_actv_rev_tl, num_bc_tl, num_il_tl, pct_tl_nvr_dlq, total_bc_limit, FICO_average):

pred = get_prediction(loan_amnt, term, int_rate, sub_grade, emp_length, home_ownership, annual_inc, verification_status,
purpose, addr_state, dti, delinq_2yrs, open_acc, pub_rec, revol_bal, application_type, tot_cur_bal,
open_acc_6m, open_act_il, open_il_12m, mths_since_rcnt_il, total_bal_il, il_util, open_rv_12m,
max_bal_bc, all_util, total_cu_tl, inq_last_12m, acc_open_past_24mths, bc_util, mo_sin_old_il_acct,
mo_sin_old_rev_tl_op, mo_sin_rcnt_rev_tl_op, mo_sin_rcnt_tl, mort_acc, mths_since_recent_bc,
num_accts_ever_120_pd, num_actv_rev_tl, num_bc_tl, num_il_tl, pct_tl_nvr_dlq, total_bc_limit, FICO_average)

if (n_clicks):
    return pred

@app.callback(Output('shap', 'src'), Input('gen_shap', 'n_clicks'))
def image_update(n_clicks):
    if (n_clicks):
        encoded = base64.encode(open('Shap_plot_04.png', 'rb').read())
        return 'data:image/png;base64,{}'.format(encoded.decode())

# Deploy Local host app
if __name__ == "__main__":
    app.run_server()

```

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

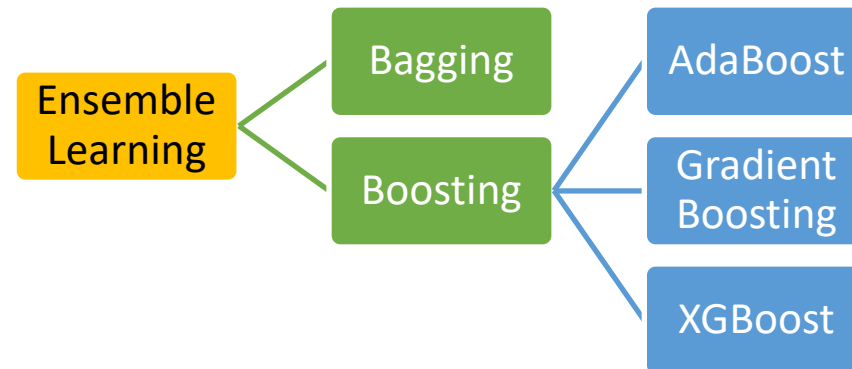
Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

HOW XGBOOST WORKS

Overview



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

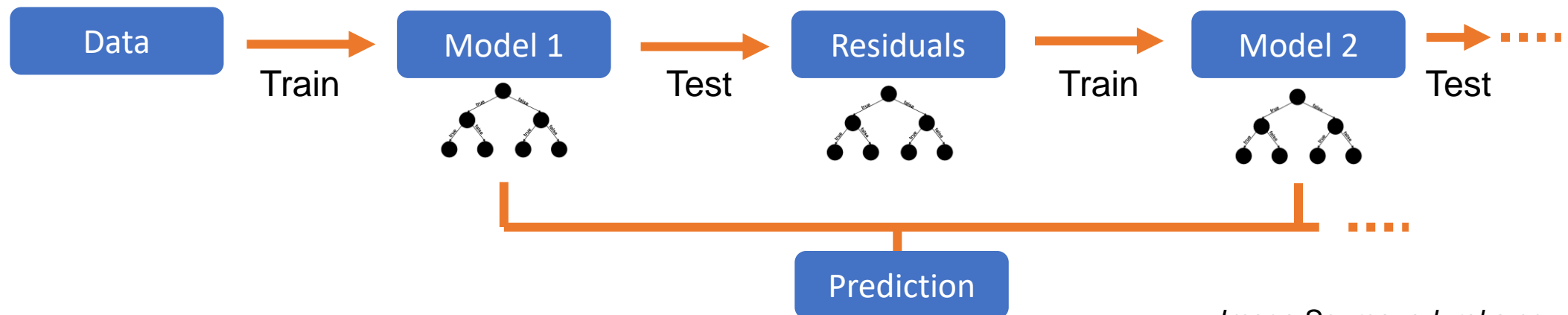


Image Source: edureka.co



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:

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

f_k 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

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

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 $h_i = \partial_{\hat{y}^{(t-1)}}^2 l(y_i, \hat{y}^{(t-1)})$

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



How XGBoost works

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

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

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

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

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

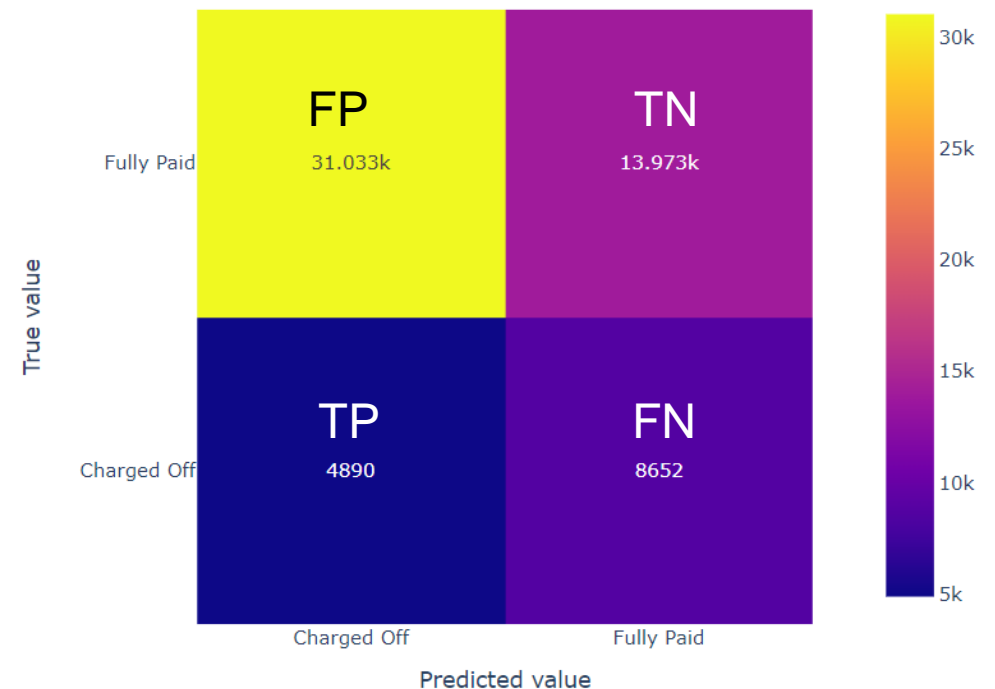
Results

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



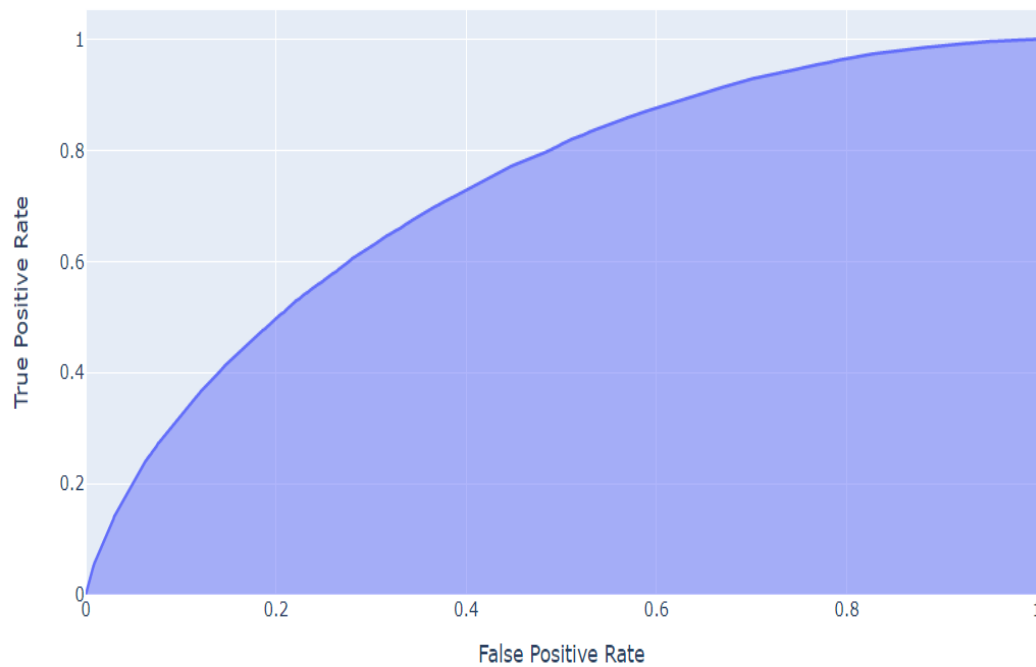
$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

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

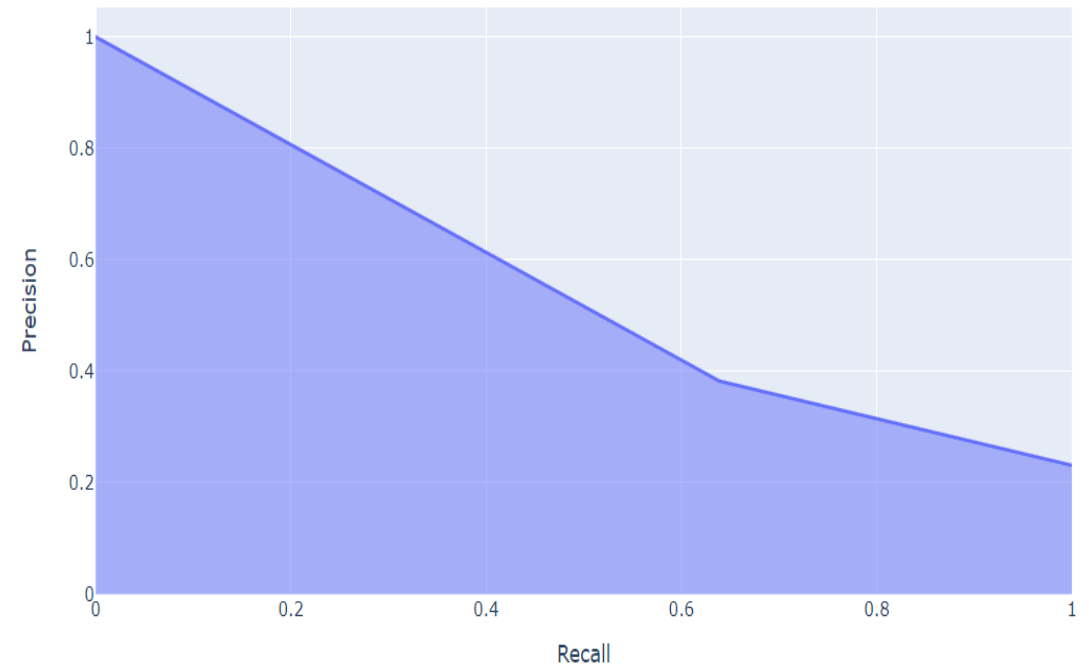


Results

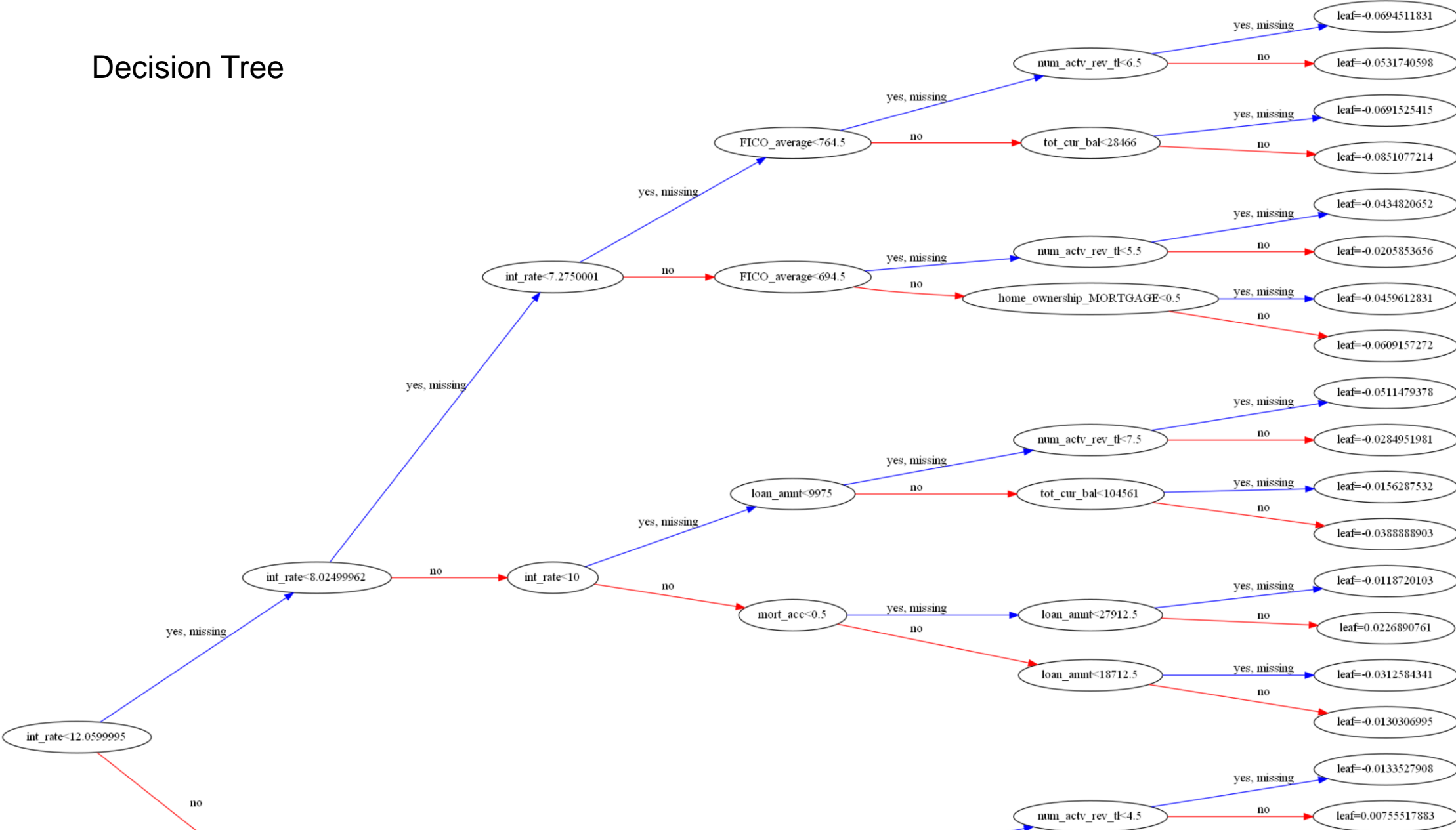
ROC Curve (AUC=0.73)



Precision and Recall Trade-off (AUC PR=0.33)

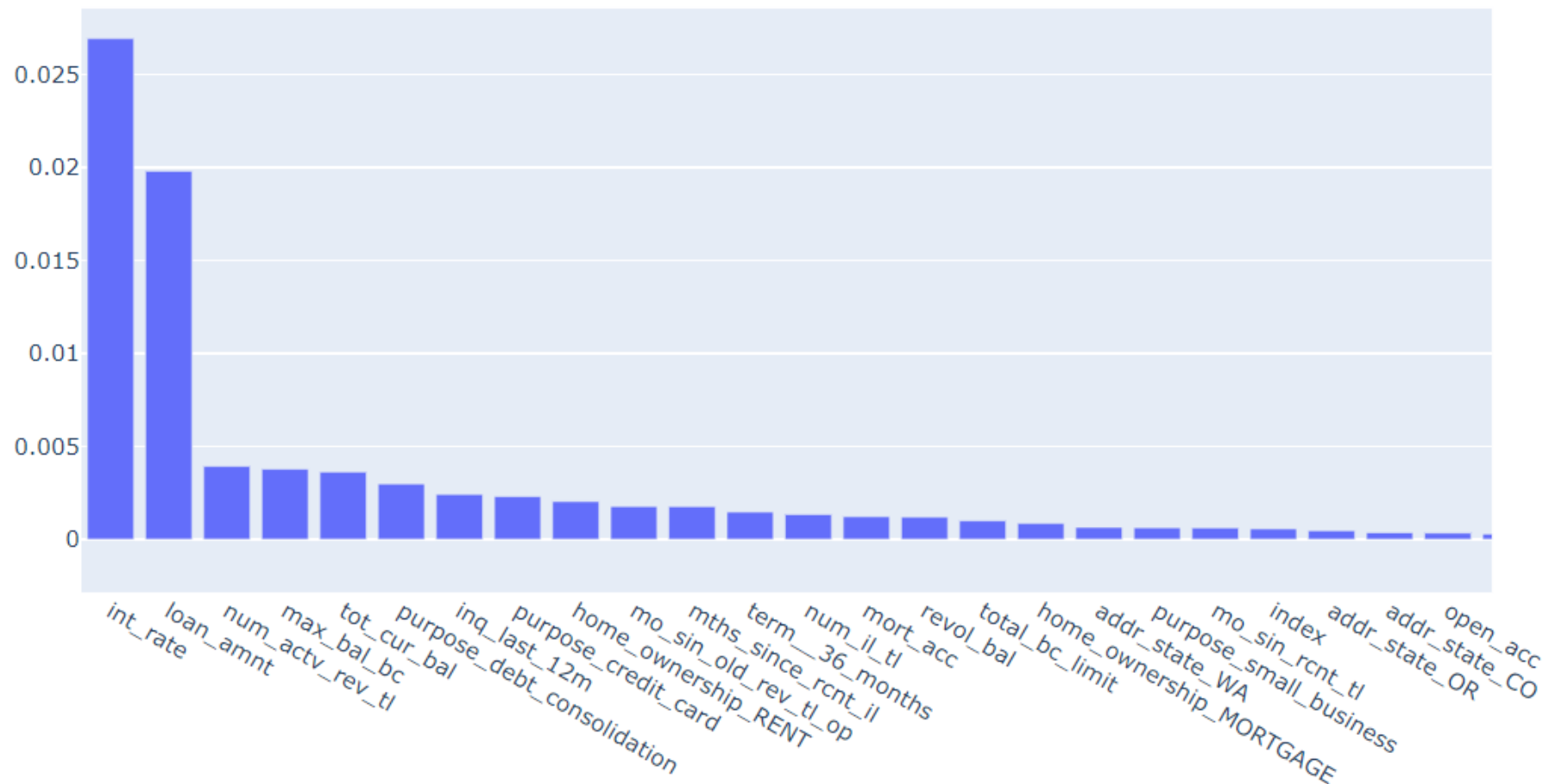


Decision Tree



Explanation of the model

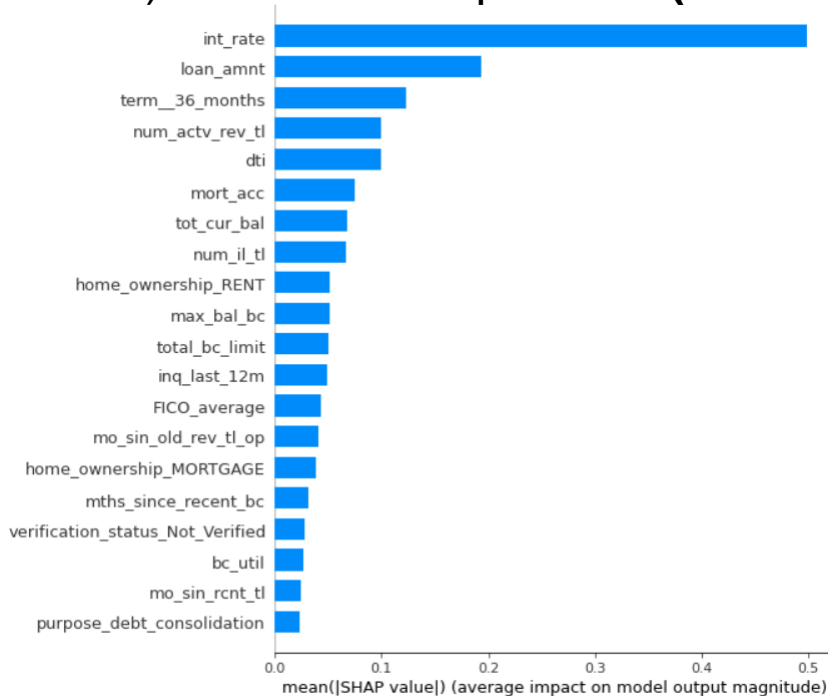
Permutation feature importance



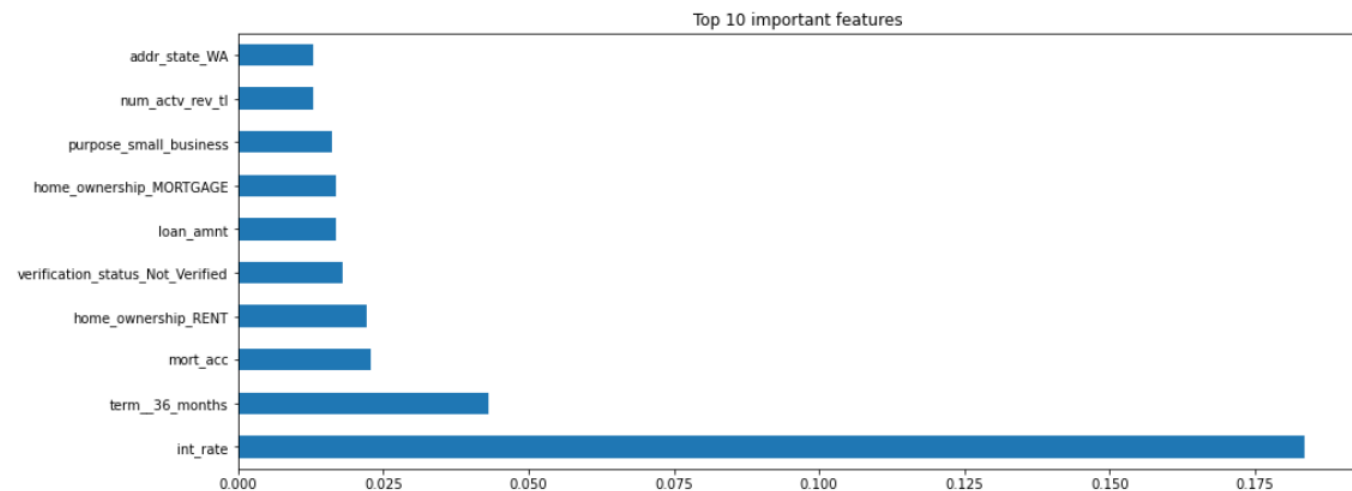
Explanation of the model

Feature Importance

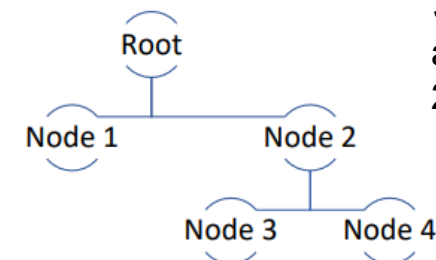
a) Based on Shap values (Lundberg & Lee, 2017)



b) Based on Gain values



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



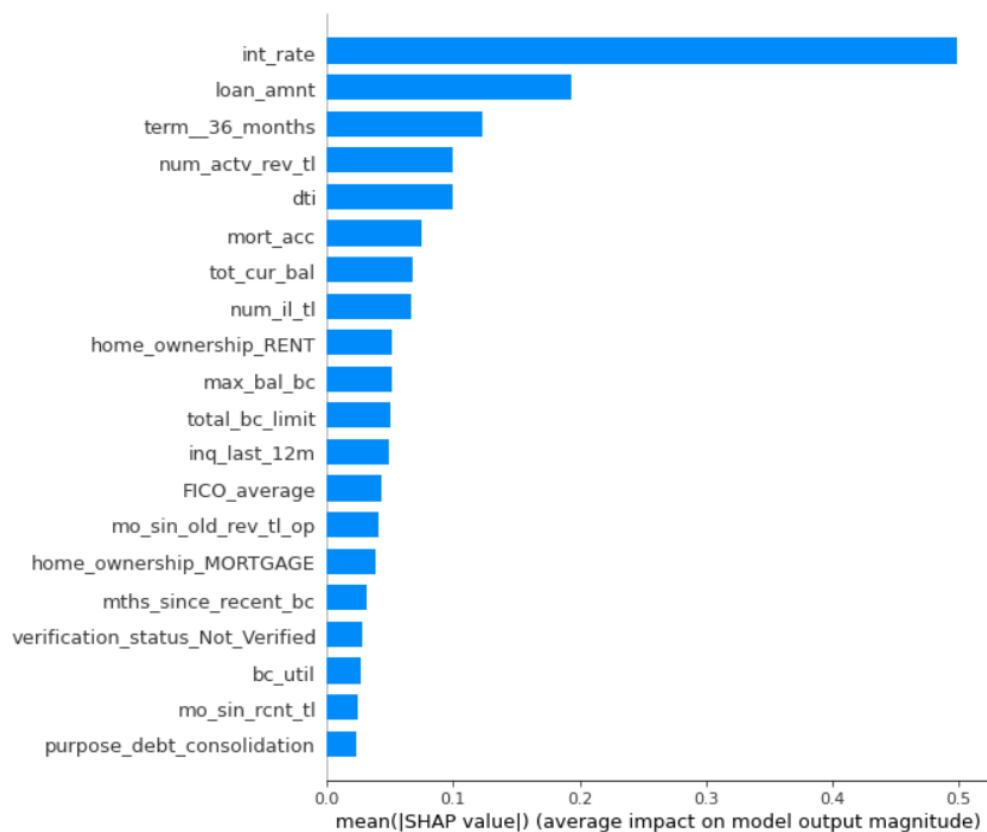
Source: Tianqi Chen
and Carlos Guestrin.
2016.

Figure 2.1: An example tree showing a node split

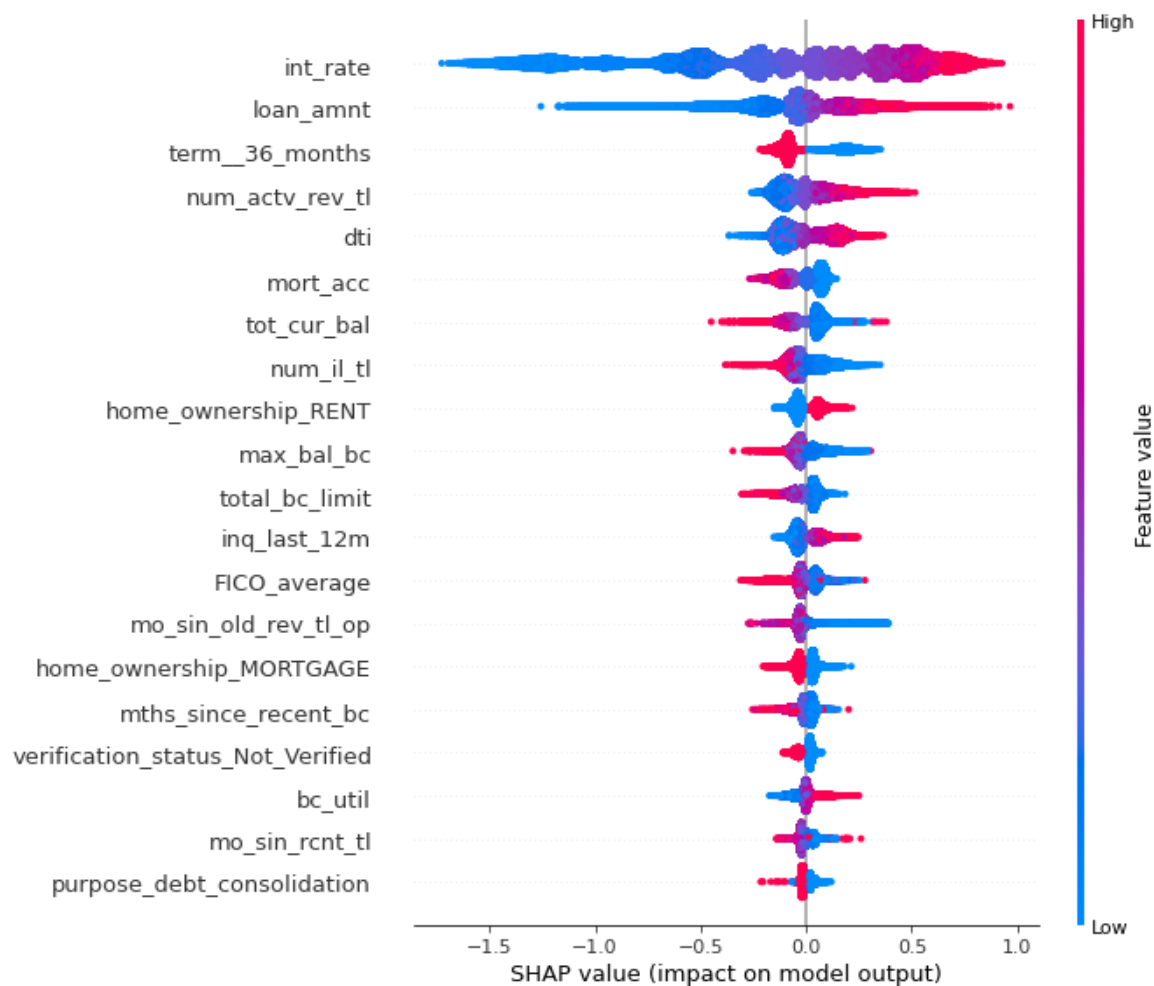


Explanation of the model

Feature Importance based on Shap values
Bar chart



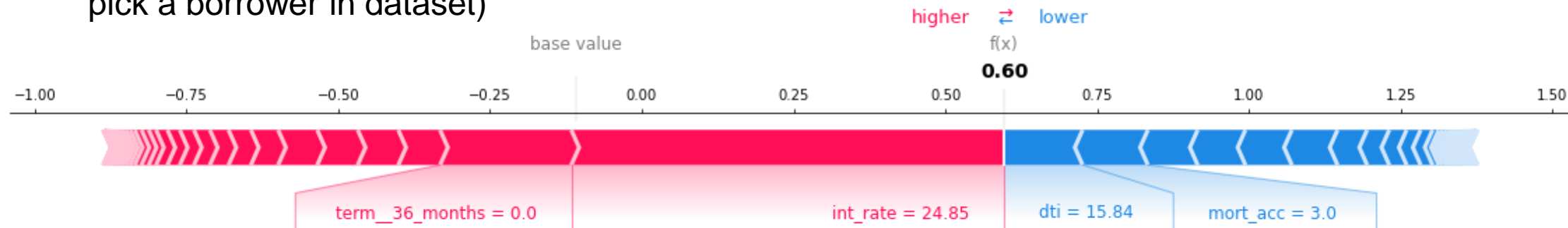
Density Scatter plot





Explanation of the model

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



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

1

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

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
 - Tuning hyperparameters
- End-users: need knowledge of Machine Learning to read the dashboard

THANK YOU!

Q&A