Application scoring model

Churkin Artem, Ushkov Lev, Zagoskin Egor

Problem Statement and Key Challenges

Project Objective

The main objective is to develop and implement a machine learning model capable of accurately predicting the probability of customer loan defaults. This will enable the bank to minimize risks and optimize the decision-making process.

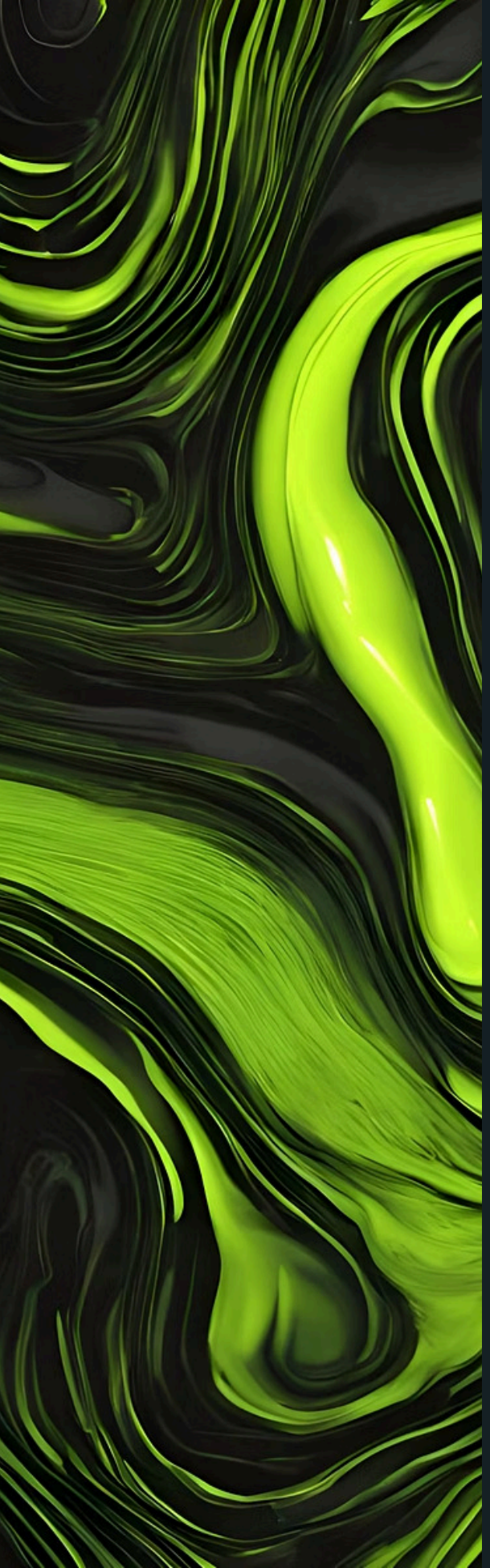
Key Challenges

Categorical Features: Effective processing of non-numerical data, such as employment type or marital status.

Class Imbalance: Far fewer defaults than successful payments, requiring special techniques for training the model.

Outliers: Unusual data values that can distort model training and reduce its accuracy.

Each of these factors will be carefully analyzed and taken into account during the development process.



Search and Model Development Strategy

Multiple Models

Our team will develop different combinations of data processing methods and machine learning algorithms. This will ensure broad coverage and enable us to find the most effective solution.

Variety of Approaches

We will experiment with various techniques for processing categorical features (e.g., One-Hot Encoding, Target Encoding), methods for addressing class imbalance (SMOTE, Oversampling, Undersampling), and strategies for handling outliers.

Comprehensive Solutions

Each “model” will represent not just an algorithm, but an entire chain: from data preprocessing to final forecasting, including specific methods for each of the identified problems.

This approach guarantees the creation of a reliable model that is robust against real data.

Project Stages: From Data to Implementation

Import and Preprocessing

We will start with data import and basic cleaning, including handling missing values.

EDA and Visualization

We will conduct a comprehensive exploratory data analysis (EDA) to identify patterns, anomalies, and relationships between features.

Model Building

Implementation of various combinations of preprocessing methods and machine learning algorithms for default prediction.

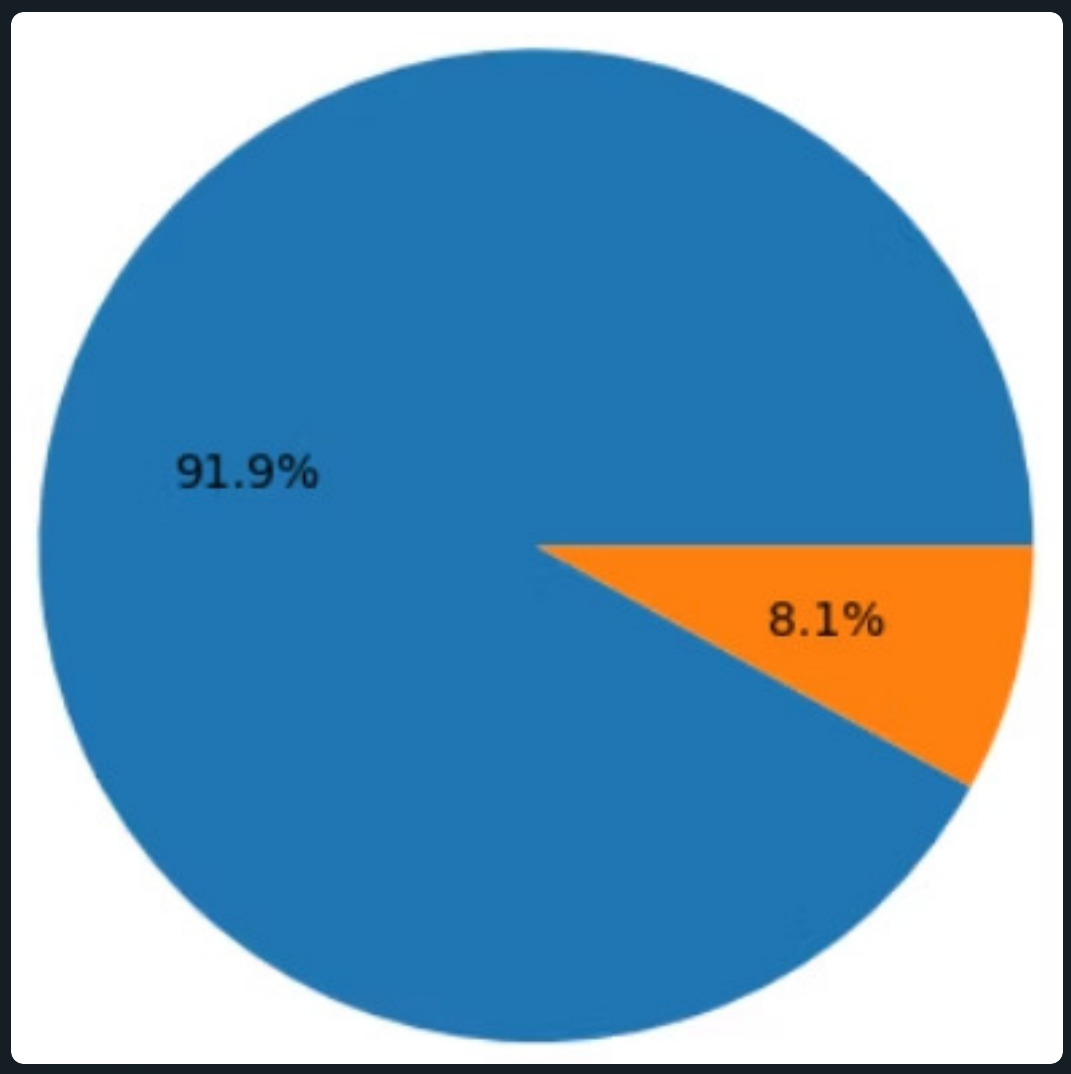
Model Interpretation (SHAP)

We will ensure model transparency using SHAP analysis, which is critical for the banking sector.

Application Interface (Streamlit/Gradio)

We will develop an interactive web interface to demonstrate and apply the model to new applications.

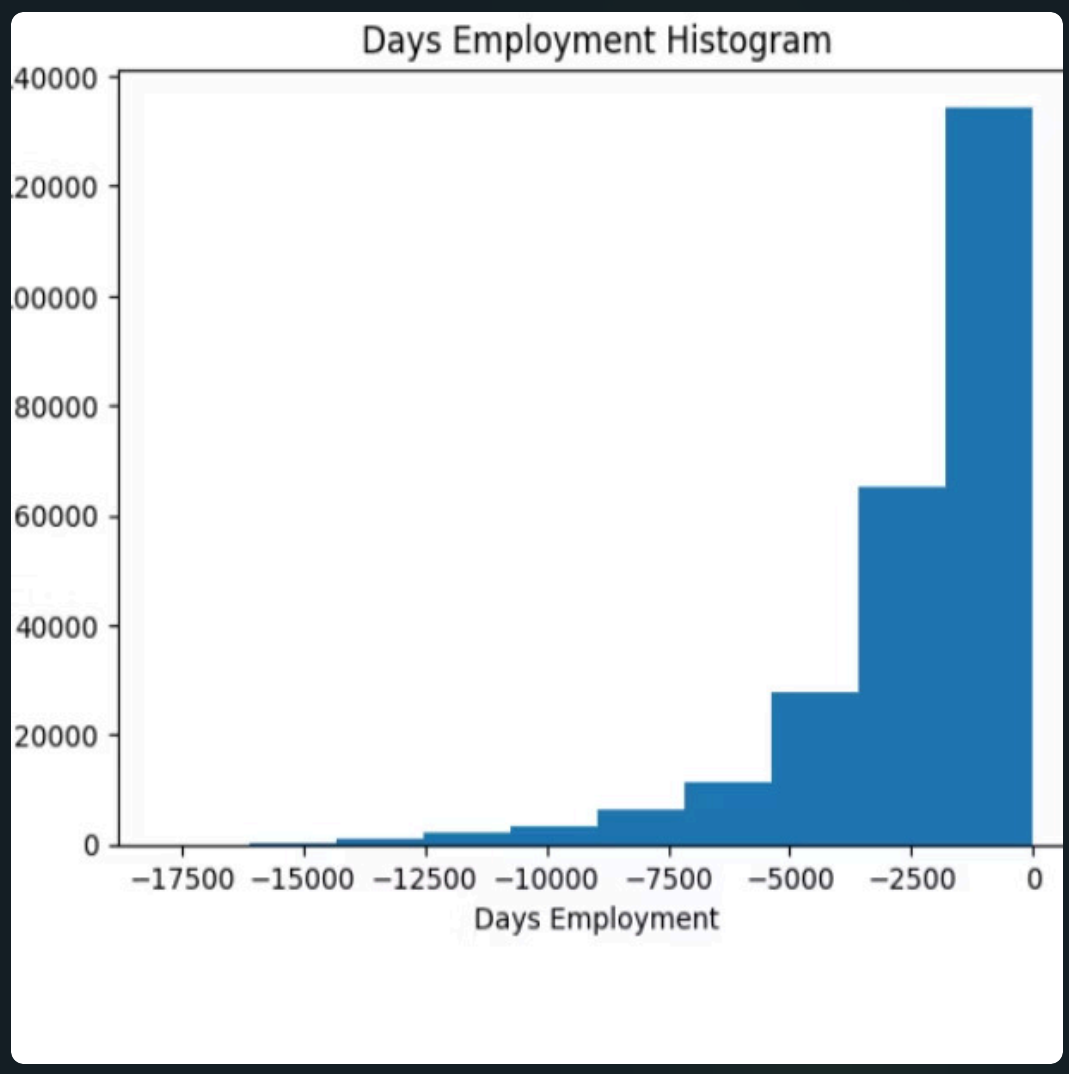
Model Search: EDA



Target Distribution

Orange - Default

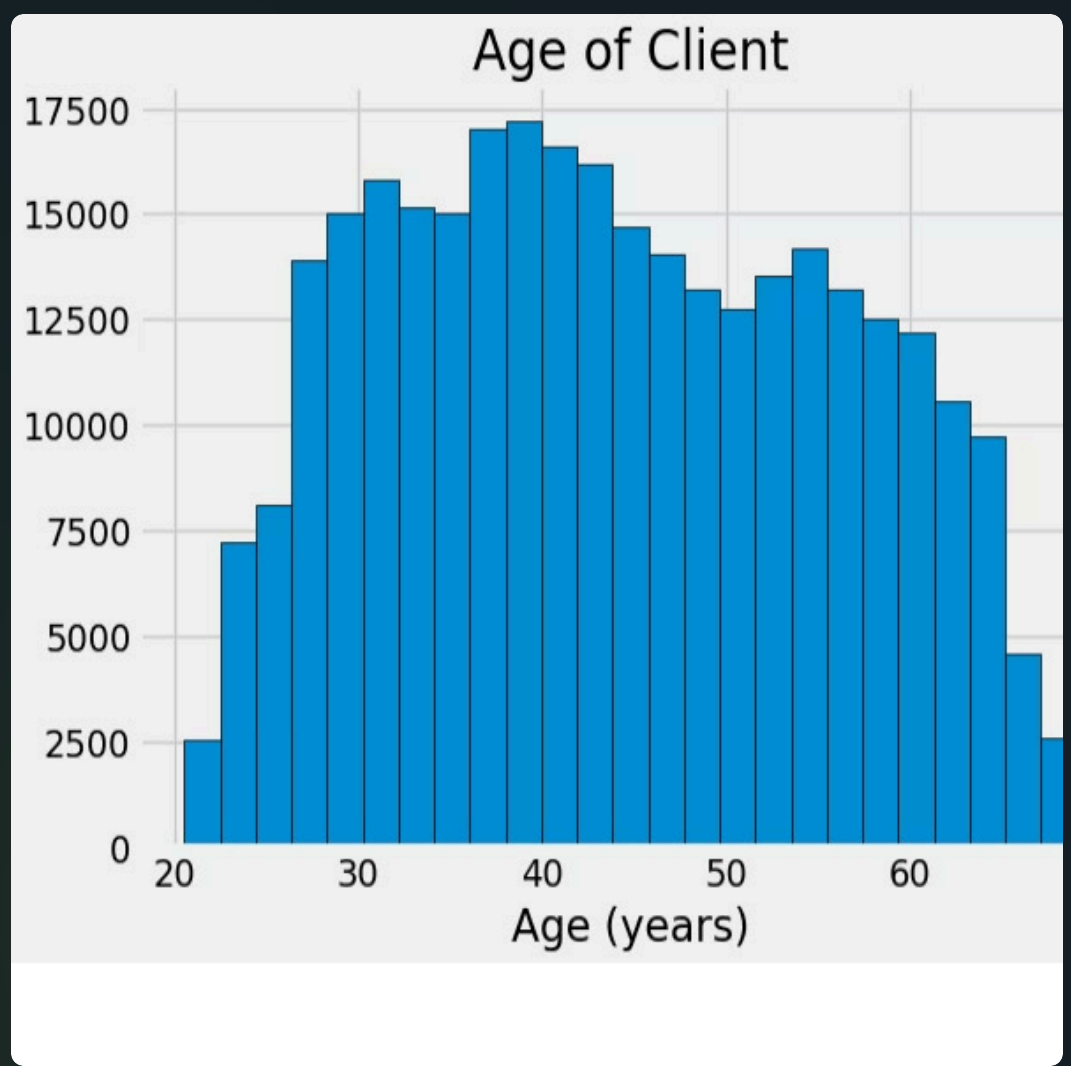
Blue - No_Default



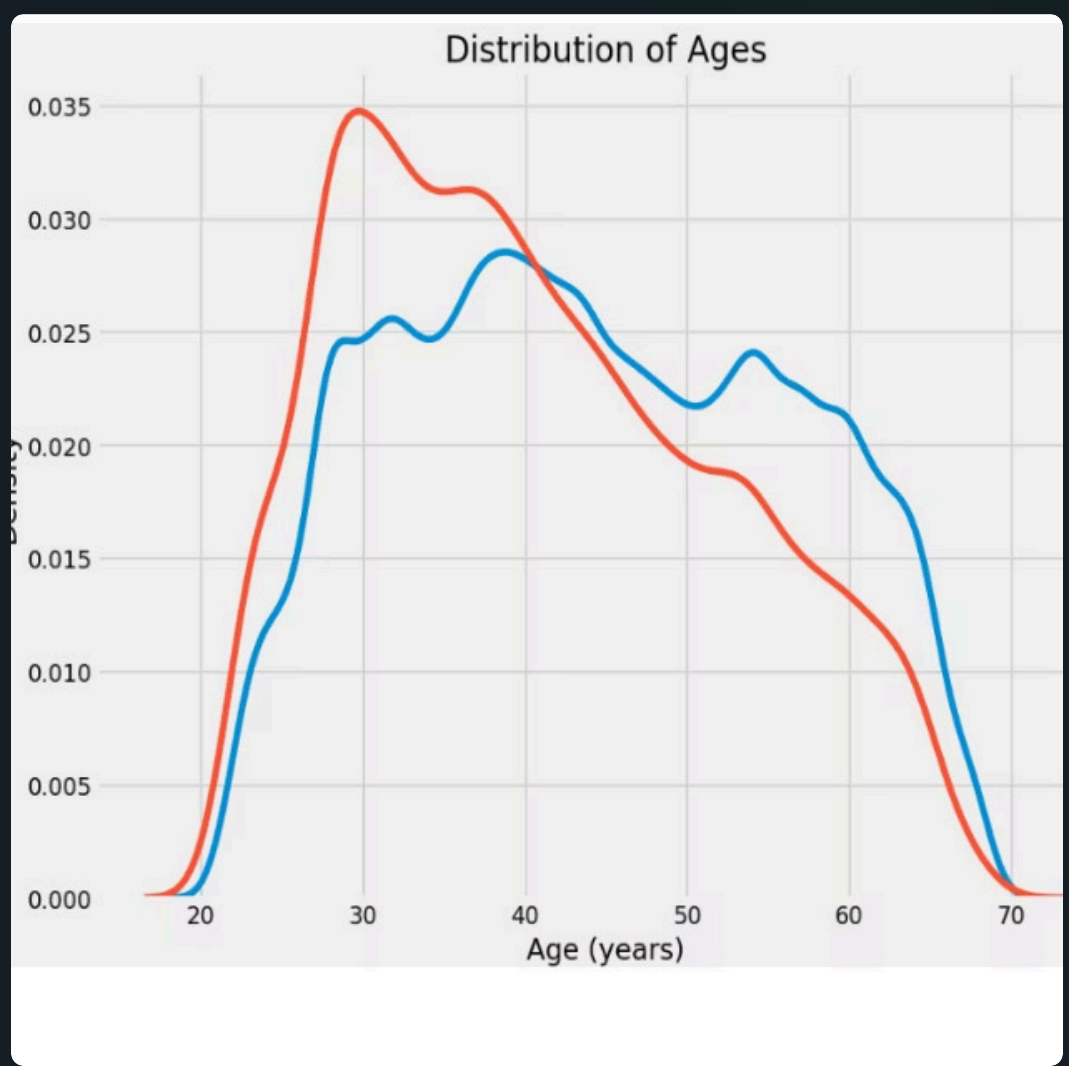
Days Employment Histogram

Frequency/Days Employment

After dealing with outliers

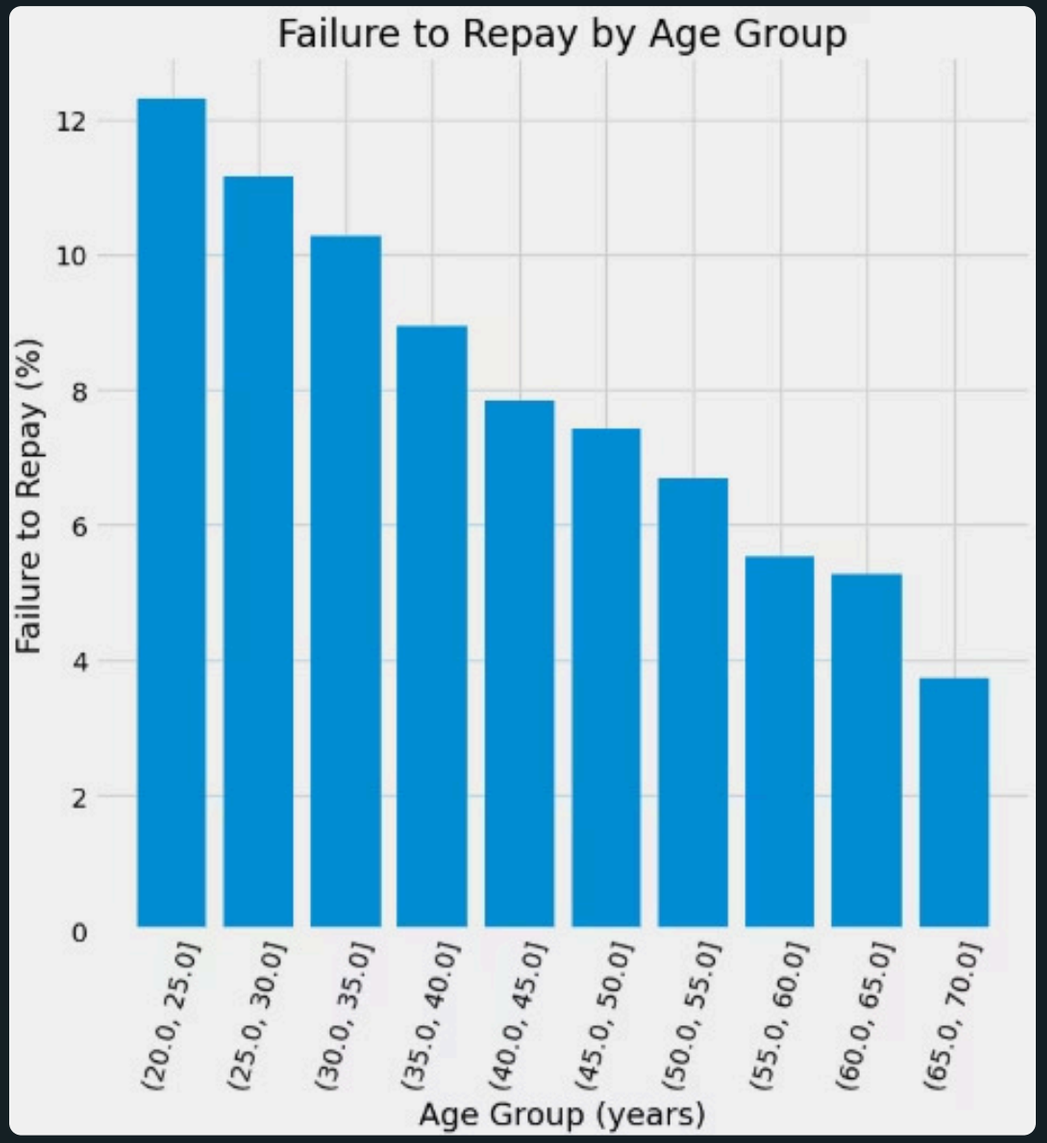


Age Distribution

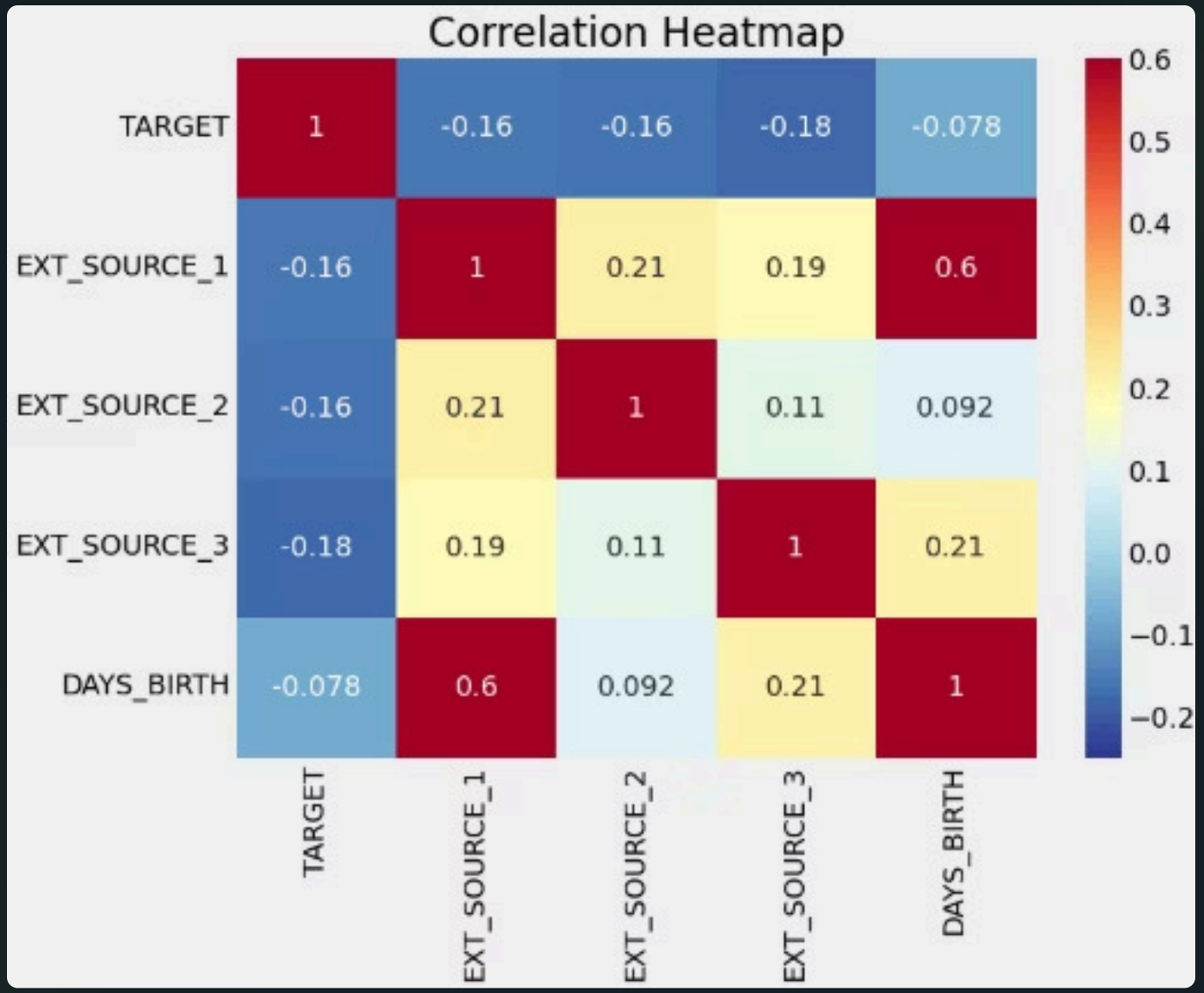


Age Distribution

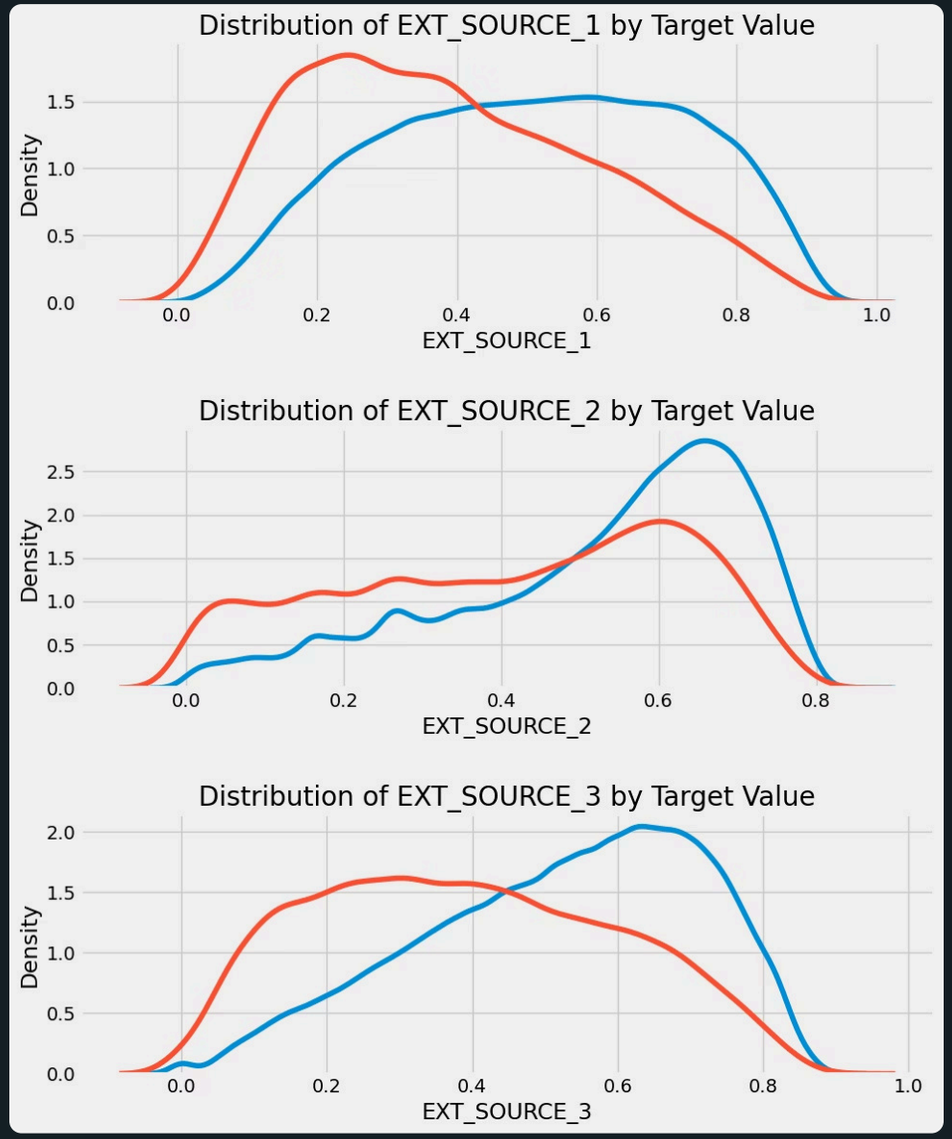
Model Search: EDA



Model Search: EDA



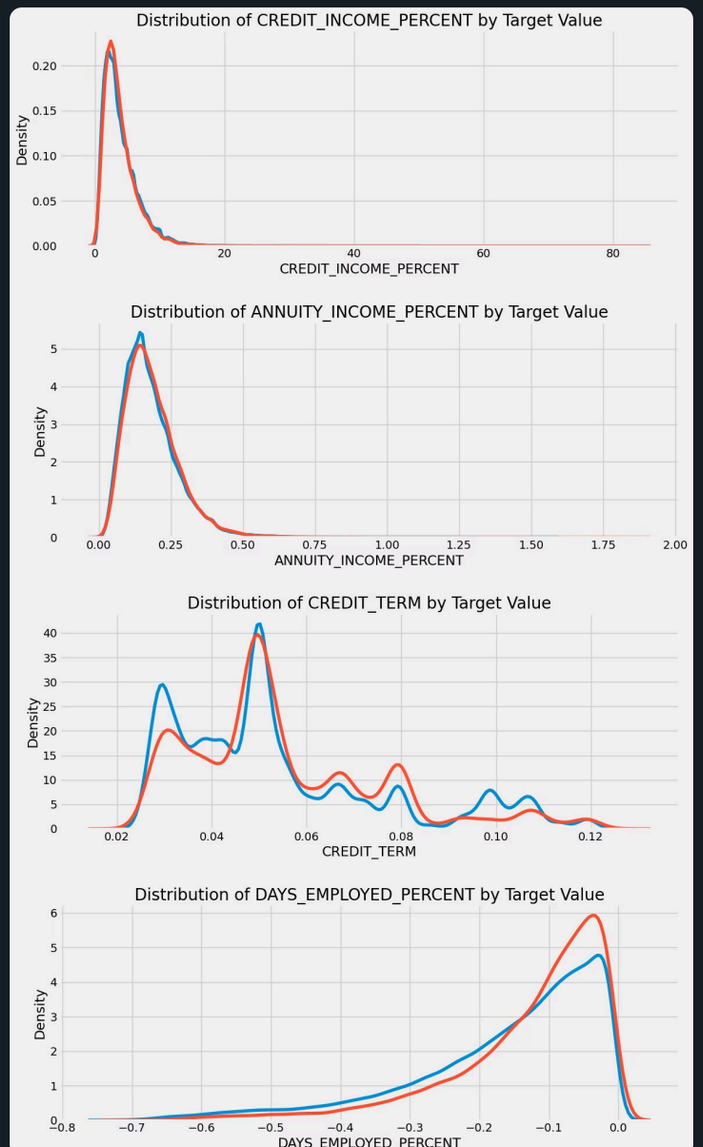
Model Search: EDA



Overview of data transformation and training

1. **Data Loading**
   - `application_train.csv` and `application_test.csv` are read using `pd.read_csv()`.
2. **Categorical Encoding**
   - Object-type columns (`dtype == 'object'`) are selected.
   - `OneHotEncoder` (with `drop='first'`) is used:
     - Fitted on `app_train`, transformed both `app_train` and `app_test`.
     - Result is concatenated back, replacing original categorical columns.
3. **DAYS_EMPLOYED Anomaly Handling**
   - Anomalous value `365243` identified.
   - Created binary anomaly flag column: `DAYS_EMPLOYED_ANOM`.
   - Replaced anomalous values with `NaN`.
4. **Polynomial Feature Engineering**
   - Highest correlation features: `EXT_SOURCE_1/2/3`, `DAYS_BIRTH`, and `TARGET`.
   - Imputation (median strategy) using `SimpleImputer`.
   - `PolynomialFeatures(degree=3)` generated all interaction/combination terms.
   - Transformed data was turned back into DataFrames with named columns.
   - Merged with original train/test by `SK_ID_CURR`.
5. **Manual Domain Features Added**
   - `CREDIT_INCOME_PERCENT = AMT_CREDIT / AMT_INCOME_TOTAL`
   - `ANNUITY_INCOME_PERCENT = AMT_ANNUITY / AMT_INCOME_TOTAL`
   - `CREDIT_TERM = AMT_ANNUITY / AMT_CREDIT`
   - `DAYS_EMPLOYED_PERCENT = DAYS_EMPLOYED / DAYS_BIRTH`
6. **Final Preprocessing**
   - `TARGET` removed from training features.
   - `SimpleImputer` (median) applied to missing values.
   - `MinMaxScaler` scaled all features into `[0, 1]`.
7. **Model Training 1**
   - Models trained on preprocessed data without polynomial or manually engineered features:
     - `LogisticRegression`:
       - Base model
       - With `class_weight='balanced'`
       - With manually defined `weights = {0: 8, 1: 92}`
     - `RandomForestClassifier`:
       - Base model
       - With `class_weight='balanced'`
       - With `class_weight='balanced_subsample'`
       - With manually defined `weights = {0: 8, 1: 92}`
     - `LGBMClassifier`:
       - Base model
       - With is_unbalance=True
       - With manually defined `weights = {0: 8, 1: 92}`
8. **Model Training 2**
   - Best variants from Model Training 1 are trained on data with polynomial and manually engineered features.

Model Search: EDA



LogReg

Base code:

```
log_reg = LogisticRegression(C = 0.0001, class_weight=weights)
log_reg.fit(train, train_labels)

log_reg_pred = log_reg.predict_proba(test)[:, 1]
submit = app_test[['SK_ID_CURR']]
submit['TARGET'] = log_reg_pred
submit.to_csv('log_reg_3.csv', index = False)
```

With default features

class_weight=None: 0.67901

class_weight='balanced': 0.70464

class_weight={0: 8, 1: 92}: 0.73176

With polynomial features

ROC-AUC: 0.73369

With additional features

ROC-AUC: 0.73235

RandomForestClassifier

Base code:

```
random_forest = RandomForestClassifier(n_estimators = 100, random_state = 50, verbose = 1, n_jobs = -1,
class_weight="balanced_subsample")
random_forest.fit(train, train_labels)

predictions = random_forest.predict_proba(test)[:, 1]
submit = app_test[['SK_ID_CURR']]
submit['TARGET'] = predictions
submit.to_csv('random_forest_3.csv', index = False)
# 0.71332
```

With default features

class_weight=None: 0.69308

class_weight="balanced": 0.71198

class_weight="balanced_subsample": 0.71332

class_weight={0: 8, 1: 92}: 0.70707

With polynomial features

ROC-AUC: 0.70100

With additional features

ROC-AUC: 0.71130

LGBM

Base code:

```
model = lgb.LGBMClassifier(class_weight=weights)
model.fit(train, train_labels)

preds = model.predict_proba(test)[:, 1]
sub = app_test[['SK_ID_CURR']]
sub['TARGET'] = preds
sub.to_csv('lgbm_3.csv', index = False)
```

With default features

class_weight=None: 0.74433

is_unbalance=True: 0.74117

class_weight={0: 8, 1: 92}: 0.74496

With polynomial features

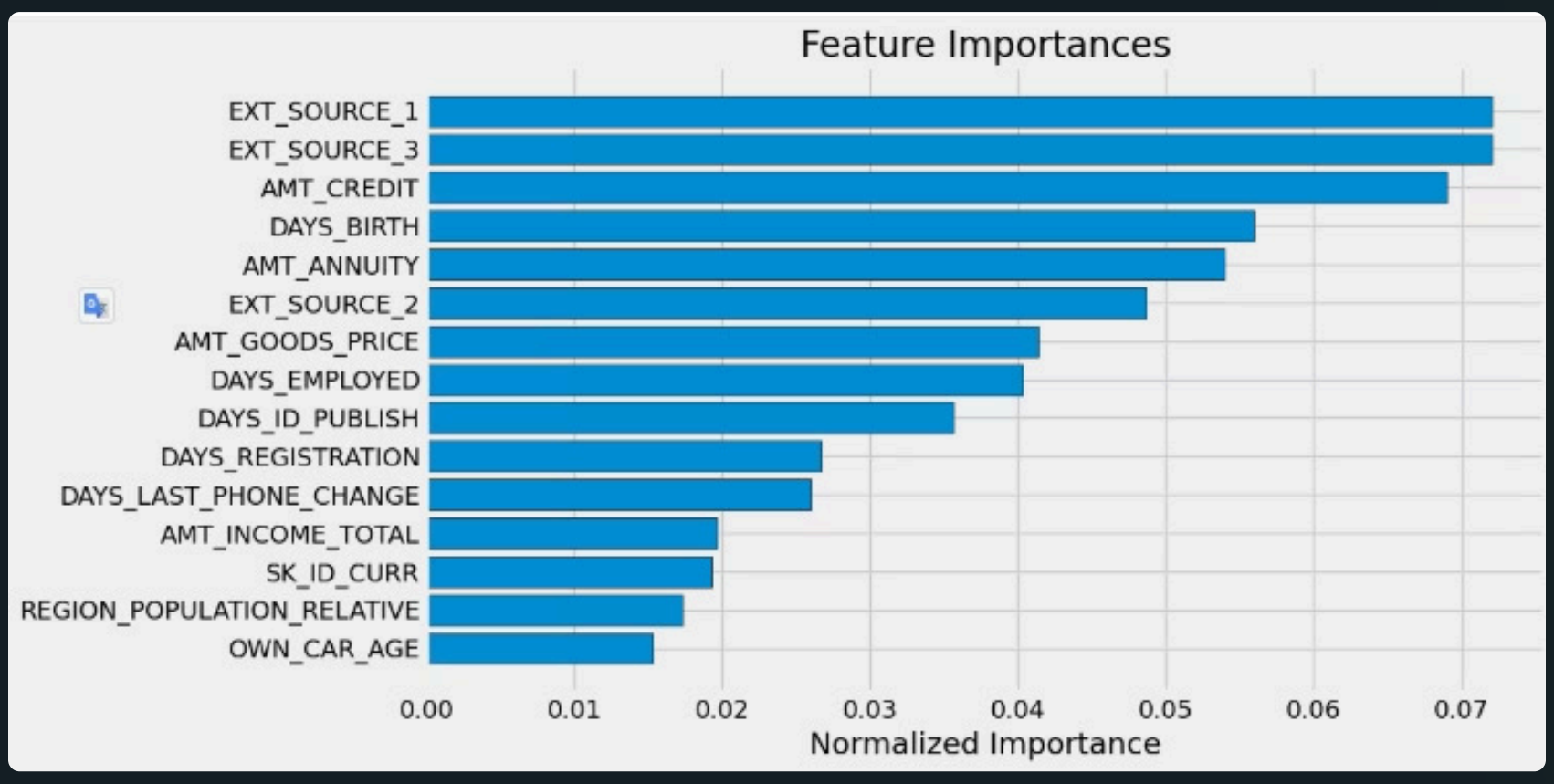
ROC-AUC: 0.74075

With additional features

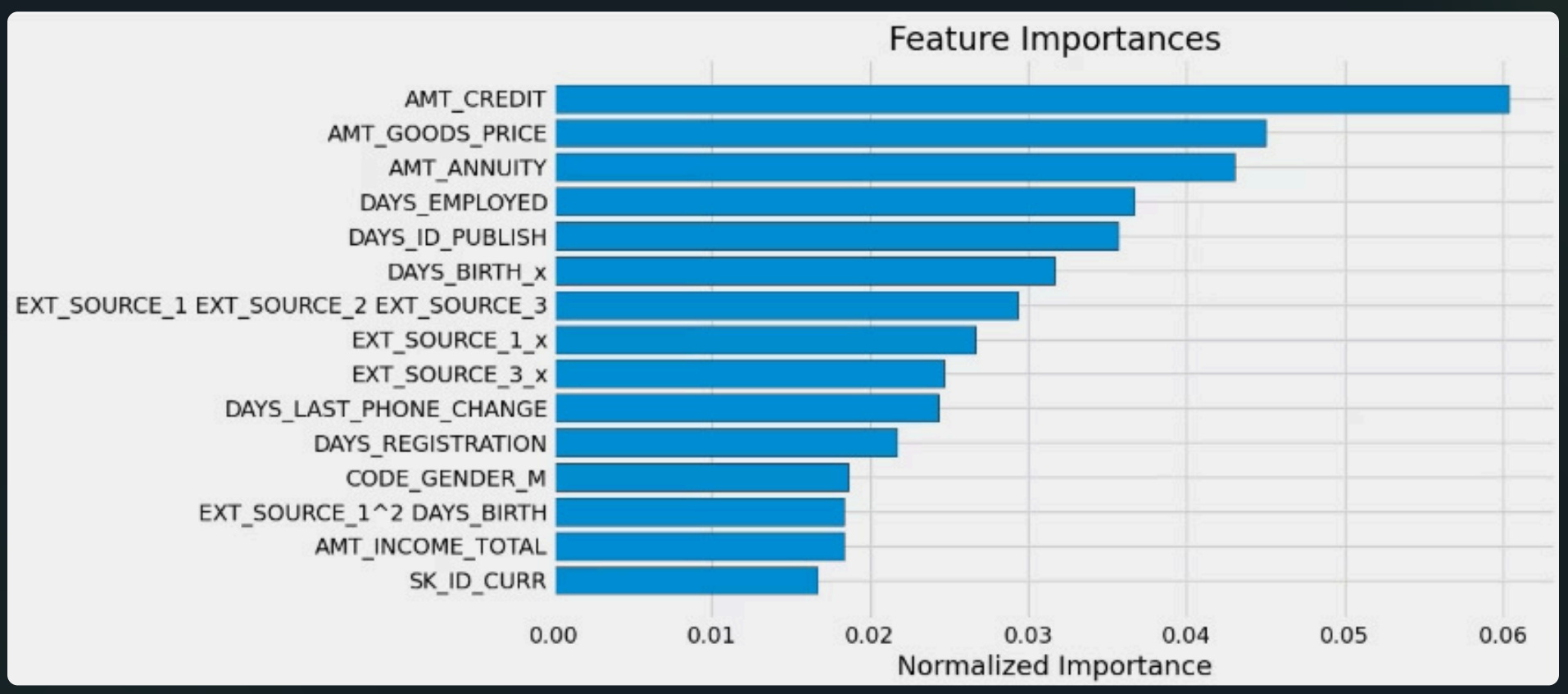
ROC-AUC: 0.76270

Model Selection: Interpretation

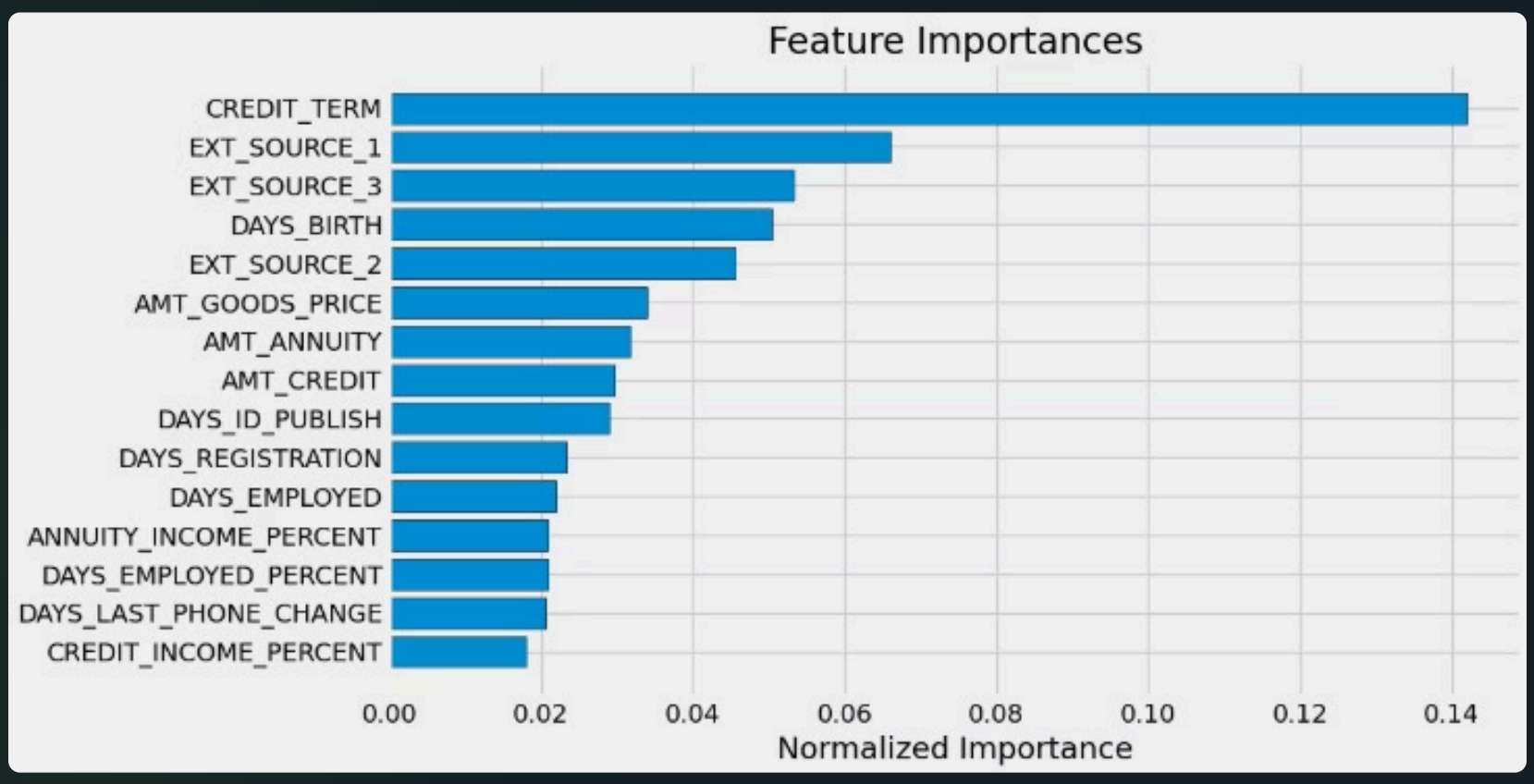
Interpretation for the best model (LGBMClassifier)



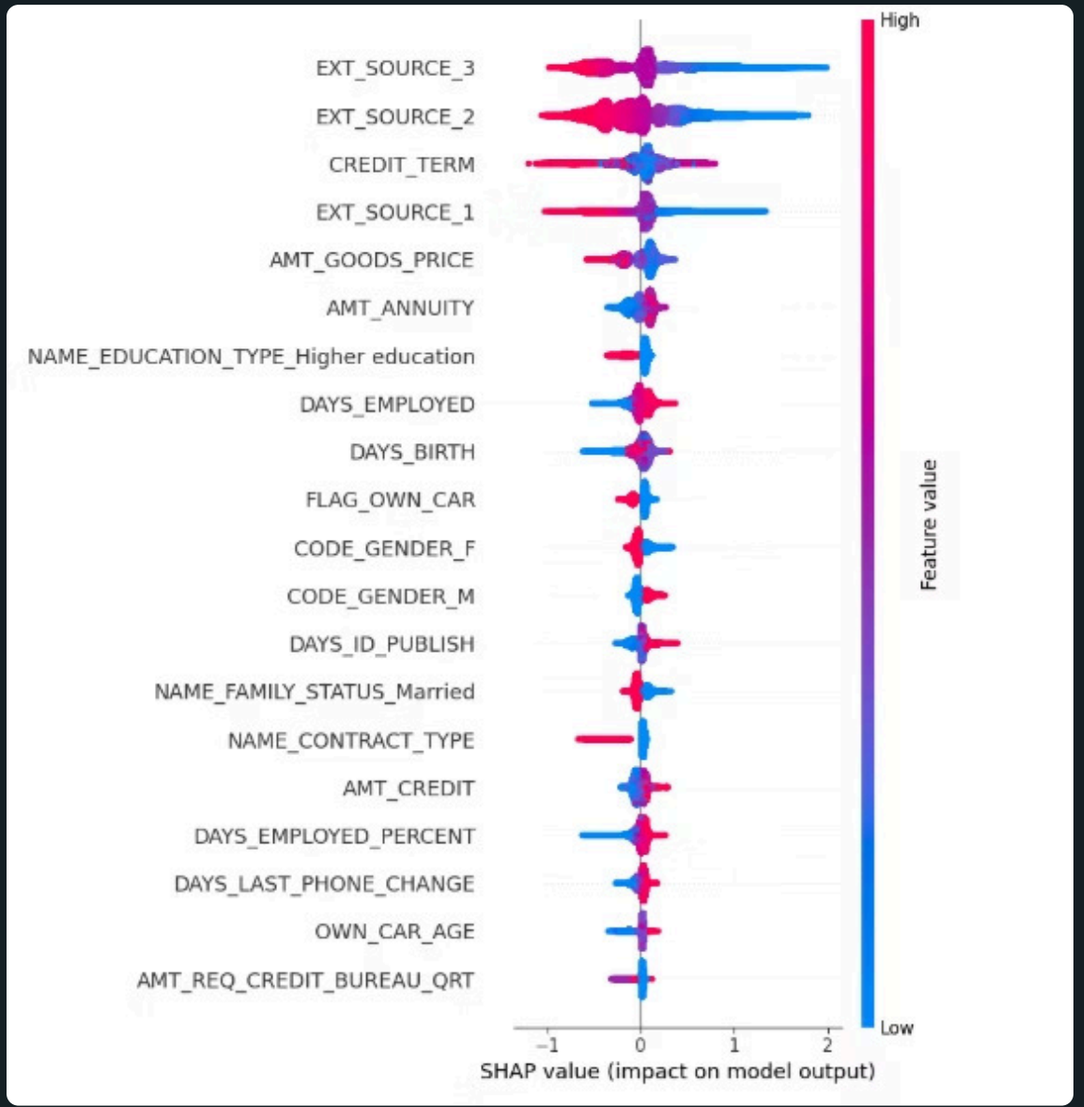
For polynomial features:



Feature importances for the additional features

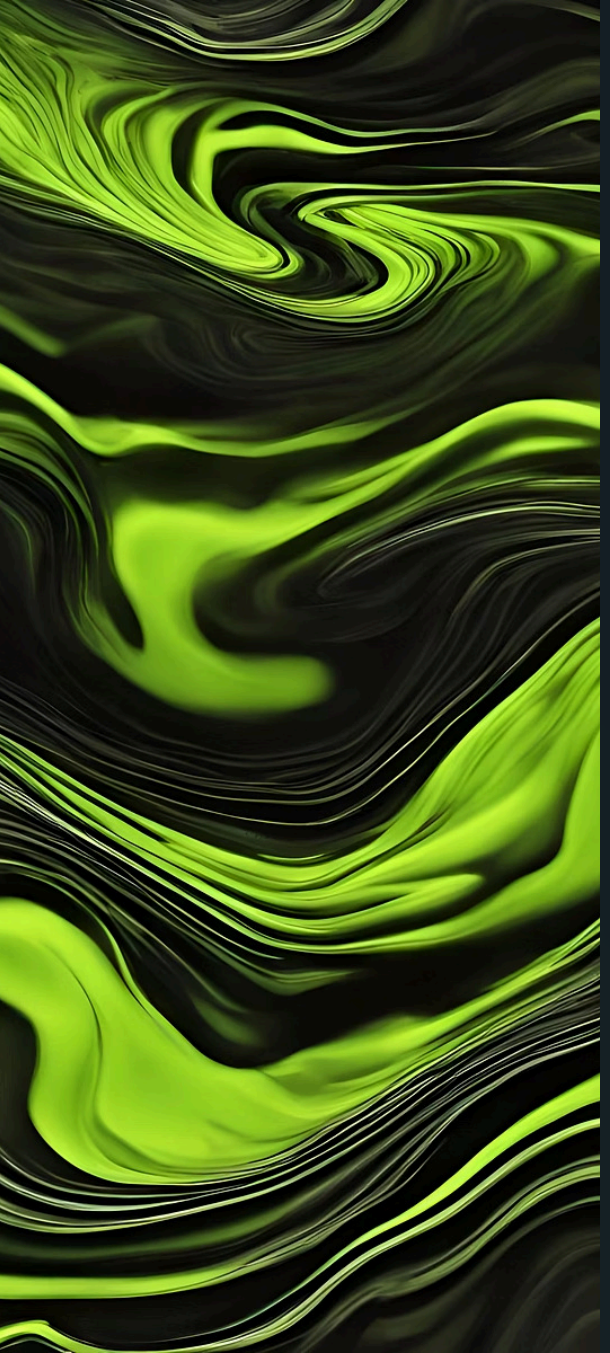


SHAP



One Client Prediction | Streamlit

https://default-prediction.streamlit.app/



Conclusion

Key Findings

We presented a comprehensive plan for creating and implementing a default prediction model, taking into account all the complexities: from data processing to ensuring interpretability and usability.

Thank you for your attention!