Machine Learning Report

Contents

1 Introduction

1.1 Team Details

- **Name and Roll Numbers:**
  - Shashank Tippanavar - IMT2022014
  - Vaibhav Bajoriya - IMT2022574
  - Soma Datta - IMT2022077

1.2 Theme Details

Theme: Lend or Lose
Description: Financial loan services are leveraged by companies across many industries, from big banks to financial institutions to government loans. One of the primary objectives of companies with financial loan services is to decrease payment defaults and ensure that individuals are paying back their loans as expected. In order to do this efficiently and systematically, many companies employ machine learning to predict which individuals are at the highest risk of defaulting on their loans, so that proper interventions can be effectively deployed to the right audience.

2 Exploratory Data Analysis

2.1 Overview of the Dataset

Dataset	Columns	Rows
Train.csv	18	204,277
Test.csv	18	51,070

Table 1: Dataset Information

Information on the train and test dataset (dropping the `label_id` column):

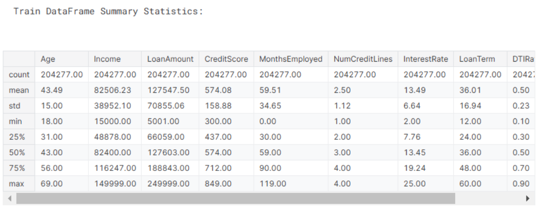
```
Train DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 204277 entries, 0 to 204276
Data columns (total 17 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Age             204277 non-null  int64
 1   Income          204277 non-null  int64
 2   LoanAmount      204277 non-null  int64
 3   CreditScore     204277 non-null  int64
 4   MonthsEmployed  204277 non-null  int64
 5   NumCreditLines  204277 non-null  int64
 6   InterestRate    204277 non-null  float64
 7   LoanTerm        204277 non-null  int64
 8   DTIRatio        204277 non-null  float64
 9   Education       204277 non-null  object
 10  EmploymentType  204277 non-null  object
 11  MaritalStatus   204277 non-null  object
 12  HasMortgage     204277 non-null  object
 13  HasDependents   204277 non-null  object
 14  LoanPurpose     204277 non-null  object
 15  HasCoSigner     204277 non-null  object
 16  Default         204277 non-null  int64
dtypes: float64(2), int64(8), object(7)
memory usage: 26.5+ MB
None
```

Figure 1: The info from the train dataset.

```
Test DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51070 entries, 0 to 51069
Data columns (total 17 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   LoanID          51070 non-null   object
 1   Age             51070 non-null   int64
 2   Income          51070 non-null   int64
 3   LoanAmount      51070 non-null   int64
 4   CreditScore     51070 non-null   int64
 5   MonthsEmployed  51070 non-null   int64
 6   NumCreditLines  51070 non-null   int64
 7   InterestRate    51070 non-null   float64
 8   LoanTerm        51070 non-null   int64
 9   DTIRatio        51070 non-null   float64
 10  Education       51070 non-null   object
 11  EmploymentType  51070 non-null   object
 12  MaritalStatus   51070 non-null   object
 13  HasMortgage     51070 non-null   object
 14  HasDependents   51070 non-null   object
 15  LoanPurpose     51070 non-null   object
 16  HasCoSigner     51070 non-null   object
dtypes: float64(2), int64(7), object(8)
memory usage: 6.6+ MB
None
```

Figure 2: The info from the test dataset.

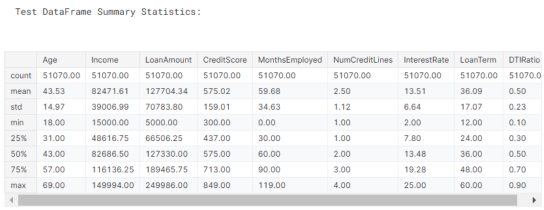
Statistics of the dataset columns:



Train DataFrame Summary Statistics:

	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTIRa
count	204277.00	204277.00	204277.00	204277.00	204277.00	204277.00	204277.00	204277.00	20427
mean	43.49	82506.23	127547.50	574.08	59.51	2.50	13.49	36.01	0.50
std	15.00	38952.10	70855.06	158.88	34.65	1.12	6.64	16.94	0.23
min	18.00	15000.00	5001.00	300.00	0.00	1.00	2.00	12.00	0.10
25%	31.00	48878.00	66059.00	437.00	30.00	2.00	7.76	24.00	0.30
50%	43.00	82400.00	127603.00	574.00	59.00	3.00	13.45	36.00	0.50
75%	56.00	116247.00	188843.00	712.00	90.00	4.00	19.24	48.00	0.70
max	69.00	149999.00	249999.00	849.00	119.00	4.00	25.00	60.00	0.90

Figure 3: The stats from the train dataset.



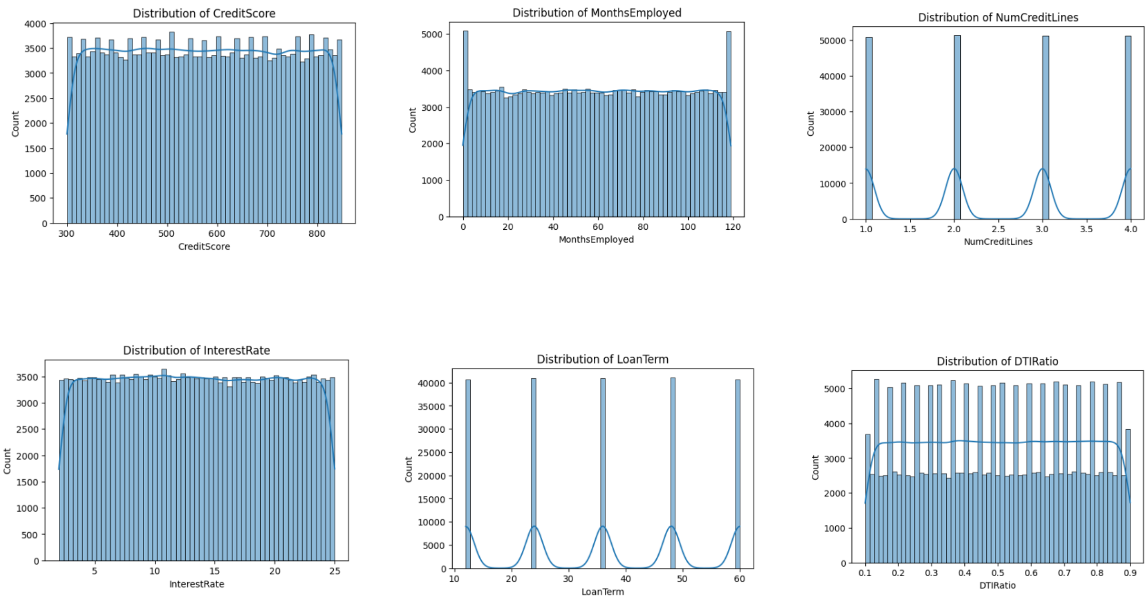
Test DataFrame Summary Statistics:

	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTIRatio
count	51070.00	51070.00	51070.00	51070.00	51070.00	51070.00	51070.00	51070.00	51070.0
mean	43.53	82471.61	127704.34	575.02	59.68	2.50	13.51	36.09	0.50
std	14.97	39006.99	70783.80	159.01	34.63	1.12	6.64	17.07	0.23
min	18.00	15000.00	5000.00	300.00	0.00	1.00	2.00	12.00	0.10
25%	31.00	48616.75	66506.25	437.00	30.00	1.00	7.80	24.00	0.30
50%	43.00	82686.50	127330.00	575.00	60.00	2.00	13.48	36.00	0.50
75%	57.00	116136.25	189465.75	713.00	90.00	3.00	19.28	48.00	0.70
max	69.00	149994.00	249986.00	849.00	119.00	4.00	25.00	60.00	0.90

Figure 4: The stats from the test dataset.

2.2 Features Overview

Numeric Features: Age, Income, LoanAmount, CreditScore, MonthsEmployed, NumcreditLines, InterestRate, LoanTerm, DTIRatio.
Categorical Features: Education, EmploymentType, Marital Status, HasMortgage, HasDependents, LoanPurpose, HasCosigner.



2.3 Correlation Analysis

This heatmap shows the correlation matrix of various features in the dataset. The values range from -1 to 1, where values near 0 indicate a weak or no correlation, and values closer to ± 1 indicate a strong correlation. Most feature correlations are very close to zero, suggesting minimal interdependence, which could imply that each feature independently contributes to lending decisions, reducing the risk of overfitting in predictive models for lending or credit analysis.

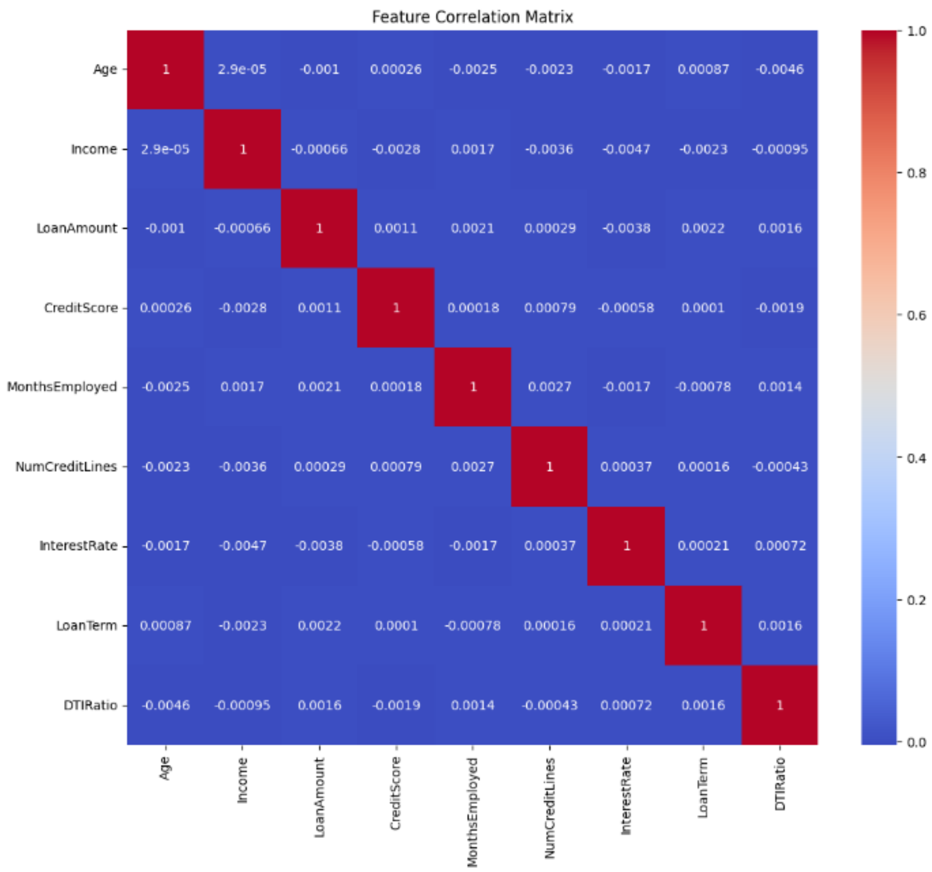


Figure 5: Correlation Matrix

2.4 Distribution of Categorical Features

- **Education**: The dataset has a fairly balanced distribution of educational backgrounds, with Bachelor's degrees slightly more common.
- **Employment Type**: Part-time employment is the most common, followed closely by unemployed, full-time, and self-employed categories, indicating diverse employment backgrounds.
- **Marital Status**: Married individuals make up the largest group, but there's a nearly equal distribution among married, divorced, and single individuals.
- **Has Mortgage**: There is an almost equal distribution between those with and without mortgages, suggesting a balanced dataset in terms of mortgage ownership.
- **Has Dependents**: There is a nearly equal split between individuals with and without dependents.
- **Loan Purpose**: Loan purposes are evenly distributed, with "Business" as the most common purpose and "Auto" as the least common.
- **Has Co-Signer**: A nearly equal split exists between individuals with and without co-signers.

Table 2: Value Counts of Key Features

Feature	Category	Count
Education	Bachelor's	51,483
	High School	51,046
	PhD	50,980
	Master's	50,768
Employment Type	Part-time	51,460
	Unemployed	50,994
	Full-time	50,921
	Self-employed	50,902
Marital Status	Married	68,217
	Divorced	68,137
	Single	67,923
Has Mortgage	Yes	102,145
	No	102,132
Has Dependents	Yes	102,180
	No	102,097
Loan Purpose	Business	40,984
	Home	40,878
	Education	40,855
	Other	40,829
	Auto	40,731
Has Co-Signer	Yes	102,196
	No	102,081

3 Preprocessing Steps

3.1 Columns Used or Omitted

- **Target Column (`Default`)**: The target variable representing whether a loan is defaulted. It was excluded from the features (`X`) to prevent data leakage during training.
- **Numerical Features**: All numerical features detected using `X.select_dtypes(include=np.numb` were included. These features are critical for prediction but were scaled using `StandardScaler` to ensure consistent distribution and scale.
- **Categorical Features**: All categorical features detected using `X.select_dtypes(include='obje` were included. They were encoded using `OneHotEncoder` to convert them into a format suitable for machine learning models.
- **Newly Created Features (Binning)**:
  - `Age_bin`: Binned into four categories (young, middle-aged, older, senior) to capture meaningful groupings.
  - `Income_bin`: Binned to simplify financial capacity analysis.
  - `LoanAmount_bin`: Categorized loan amounts into bins for simplified modeling.

  - CreditScore_bin: Grouped credit scores into risk categories to assess creditworthiness.
  - DTIRatio_bin: Binned DTI ratios into risk levels.

- **Excluded Columns**: Irrelevant columns, such as identifiers or features with excessive missing values, were omitted to improve model performance.

3.2 Reasoning Behind Experimental Design

- **Binning**: Reduces variability in continuous variables and focuses on meaningful groupings, such as age demographics or income tiers, making models less prone to overfitting by introducing bins or grouping.
- **Scaling Numerical Features**: Ensures numerical features like LoanAmount and Income have comparable influence by normalizing their scale.
- **Encoding Categorical Features**: Converts categorical variables into numeric representations using OneHotEncoder, avoiding the assumption of ordinal relationships.
- **Pipeline Modularity**: Combines preprocessing steps into reusable pipelines, ensuring reproducibility and minimizing manual intervention.
- **Train-Test Split**: Splits the data into training (70%) and validation (30%) subsets to evaluate the model's generalizability.

4 Code Implementation

4.1 Feature Binning

```
train_df['Age_bin'] = pd.cut(train_df['Age'], bins=[18, 31, 43, 56, 69], labels=[0, 1
train_df['Income_bin'] = pd.cut(train_df['Income'], bins=[15000, 48878, 82400, 116247
```

This step simplifies continuous variables like Age and Income into meaningful categories based on the distribution of each feature(like their min,max,ranges,quartiles).

4.2 Preprocessing Pipelines

```
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
```

The numerical pipeline applies standardization, while the categorical pipeline encodes textual features.

4.3 Column Transformer

```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ]
)
```

Combines both pipelines into a unified step for efficient preprocessing.

4.4 Train-Test Split

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3, random_state=4
```

Splits the data for training and validation.

4.5 Applying Transformations

```
X_train = preprocessor.fit_transform(X_train)
X_val = preprocessor.transform(X_val)
```

Fits the preprocessing pipeline on training data and applies it to both training and validation sets.

4.6 Outlier Handling

There were no outlier, we checked using the boxplot, therefore we don't require to remove any outlier for the training dataset

5 Learning Models

5.1 Approach

We defined a set of hyperparameter grids for each of the following learning methods:

1. **Decision Tree:** `dt_params`
2. **Random Forest:** `rf_params`
3. **XGBoost:** `xgb_params`
4. **CatBoost:** `catboost_params`
5. **AdaBoost:** `ada_params`

We then instantiated the classifiers for each of these models and encapsulated them in a dictionary called `models`:

1. **Decision Tree:** `DecisionTreeClassifier()`
2. **Random Forest:** `RandomForestClassifier()`
3. **XGBoost:** `XGBClassifier()`
4. **CatBoost:** `CatBoostClassifier()`
5. **AdaBoost:** `AdaBoostClassifier()`

HyperParameter Tunning:-
We then used `RandomizedSearchCV` on each of these models by iterating through the set of parameter we have defined for each models, after that we will get the best paramters for each model. That we are using for getting the final and best accuracy for each model

5.2 Explanation of Metrics

- **Precision:** The proportion of positive predictions that are actually positive.
- **Recall:** The proportion of actual positive cases that are correctly identified as positive.
- **F1-Score:** The harmonic mean of precision and recall, providing a balanced measure of both.
- **Support:** The number of samples for each class.
- **Accuracy:** The overall proportion of correct predictions.
- **Macro Average:** The average of the metric for each class, giving equal weight to each class.
- **Weighted Average:** The average of the metric for each class, weighted by the number of samples in each class.

After obtaining the best hyperparameters for each model, we instantiated the models and plotted their confusion matrices. The results for each model are as follows:

5.3 Decision Tree

5.3.1 Results

We achieved an **accuracy of 88.529%** on the training data.

submission_SingleTree.csv
Complete · 13d ago
0.88529

5.3.2 Model Overview

In a Decision Tree model, the algorithm splits the data into smaller subsets based on feature values to facilitate decision-making. Each decision is represented as a node, while branches correspond to different feature thresholds. The tree grows by recursively partitioning the data, with the process culminating in leaf nodes that represent final predictions.
We achieved an **accuracy of 82.5%** on the validation data.

```
Decision Tree Classifier Accuracy: 0.8253377716859213
Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       0.89      0.91      0.90     54106
           1       0.22      0.19      0.20      7178

    accuracy                           0.83     61284
   macro avg       0.56      0.55      0.55     61284
weighted avg       0.82      0.83      0.82     61284
```

5.3.3 Classification Report

Analysis of the Report:

1. **Class Imbalance:**
   - The dataset exhibits significant class imbalance, with many more samples in class 0 compared to class 1.
   - This imbalance impacts the model's performance and affects the interpretation of metrics.
2. **Low Recall for Class 1:**
   - The model struggles to correctly identify positive instances of class 1, resulting in low recall.
   - Many actual positive cases of class 1 are misclassified as class 0.
3. **Low F1-Score for Class 1:**
   - The low F1-score for class 1 indicates the model's difficulty in accurately predicting this class.

5.3.4 Learnings

1. **Overfitting:**

   - Decision Trees tend to overfit, especially when the tree grows too deep.
   - We mitigated overfitting by limiting the maximum depth and requiring a minimum number of samples for node splitting. Despite these measures, this was the best performance we could achieve.

2. **Categorical Feature Encoding:**

   - We experimented with both one-hot encoding and ordinal encoding for categorical features, but neither approach made a significant difference in performance.

3. **Model Robustness:**

   - Although Decision Trees are inherently good at handling non-linear data, the model proved less robust compared to ensemble methods like Random Forests or XGBoost.
   - It showed a tendency to overfit and was sensitive to variations in the dataset.

5.4 Random Forest

5.4.1 Results

We achieved an **accuracy of 88.662%** on the training data.

submission_randomForest.csv
Complete · 13d ago
0.88662

We achieved an accuracy of **88.49%** on the validation dataset.

The Random Forest algorithm is an ensemble method that constructs multiple decision trees on random subsets of the dataset. Each tree learns from different parts of the feature space. The final prediction is obtained by aggregating the outputs of all the trees, which helps reduce variance and improves overall accuracy.

```
Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     54106
           1       0.62      0.04      0.08      7178

    accuracy                           0.88     61284
   macro avg       0.76      0.52      0.51     61284
weighted avg       0.86      0.88      0.84     61284
```

5.4.2 Report Analysis

1. **Class Imbalance:**
   - The dataset has a significant class imbalance, with **54,106 instances** in class 0 and **7,178 instances** in class 1.
   - This imbalance can influence model performance and skew the interpretation of metrics, especially for class 1.
2. **Low F1-score for Class 1:**
   - The model exhibits difficulty in accurately predicting class 1, leading to a low F1-score for this class.
3. **Low Recall for Class 1:**
   - The model struggles to identify all positive instances of class 1 (low recall).
   - This indicates that many actual positive cases in class 1 are misclassified as class 0.

5.4.3 Learnings

1. **Categorical Encoding:**
   - Both label encoding and one-hot encoding techniques were applied for categorical variables.
   - However, Random Forests are relatively robust to preprocessing and can handle categorical variables effectively without requiring extensive encoding.
2. **Feature Importance:**
   - Random Forests provide valuable insights into feature importance.
   - This allows us to identify which attributes of the loan application (e.g., applicant income, loan type) are most predictive of loan defaults.

5.5 XGBoost

5.5.1 Results

We achieved an **accuracy of 88.662%** on the training data.

XGBoost (Extreme Gradient Boosting) is an optimized gradient boosting algorithm that builds decision trees sequentially, where each tree focuses on correcting the errors made by the previous one. It is renowned for its speed and accuracy, particularly for tabular datasets with intricate patterns.

We achieved an accuracy of **88.5141%** on the validation dataset.

```
XGBoost Classifier Accuracy: 0.8851413093140135
Classification Report for XGBoost:
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     54106
           1       0.58      0.07      0.12      7178

    accuracy                           0.89     61284
   macro avg       0.74      0.53      0.53     61284
weighted avg       0.85      0.89      0.84     61284
```

5.5.2 Classification Report for XGBoost

Observations:

1. **Strong Performance on Class 0 (Lend):**
   - The model performs exceptionally well for class 0, with high precision, recall, and F1-score.
   - This indicates that the model is effective at identifying loans that will be successfully repaid.
2. **Struggles with Class 1 (Lose):**
   - The model demonstrates significant difficulty with class 1, as evidenced by a very low recall (7%) and F1-score (0.12).
   - This suggests that most loans predicted to default (class 1) are being misclassified as repaid (class 0).
3. **Impact of Class Imbalance:**
   - The dataset's severe class imbalance is likely a contributing factor to the poor performance on class 1.
   - With far fewer instances in class 1 compared to class 0, the model prioritizes the majority class (class 0), achieving high recall and precision for that class but failing to accurately identify the minority class.

5.6 CatBoost

5.6.1 Results

We achieved an **accuracy of 88.811%** on the training data.

loan_default_predictions_be.csv
Complete · 9h ago
0.88811

CatBoost is a gradient boosting algorithm designed specifically for handling categorical data, which is commonly encountered in datasets like loan-default prediction. It performs exceptionally well with minimal data preprocessing, making it particularly effective for features like loan type and borrower region.

We achieved an accuracy of **88.5255%** on the validation dataset.

```
CatBoost Classifier Accuracy: 0.8852555316232622
Classification Report for CatBoost:
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     54106
           1       0.61      0.06      0.11      7178

    accuracy                           0.89     61284
   macro avg       0.75      0.53      0.52     61284
weighted avg       0.86      0.89      0.84     61284
```

5.6.2 Classification Report for CatBoost

Observations

1. **Class Imbalance:**
   - The model exhibits a significant imbalance in its classification performance between the two classes.
   - Class 0 demonstrates high precision and recall, indicating strong performance on the majority class.
   - However, class 1 shows poor performance, with a low recall (0.06) and a very low F1-score (0.11), highlighting the model's difficulty in correctly predicting the minority class.
2. **Accuracy vs Recall:**
   - While the overall accuracy of 0.885 is high, the model performs poorly on the minority class (class 1).
   - The low recall for class 1 suggests that although the model effectively identifies class 0, it fails to capture many instances of class 1.
3. **Macro vs Weighted Averaging:**
   - The macro averages for precision, recall, and F1-score are significantly lower than their weighted counterparts.
   - This discrepancy indicates that the model's performance is heavily influenced by class 0, which dominates the dataset in terms of sample size.

5.6.3 Learnings

1. **Ordered Boosting:**
   - CatBoost employs an ordered boosting technique to reduce overfitting by ensuring the model generalizes better.
   - This approach prevents the model from memorizing the data, improving its ability to handle unseen data.
2. **Feature Importance:**

- CatBoost automatically identifies the most relevant features, enhancing overall model performance.
- However, despite these capabilities, the model struggles with accurately classifying the minority class in an imbalanced dataset.

5.7 4.7 AdaBoost

5.7.1 Results

We achieved an **accuracy of 88.662%** on the validation data.

loan_default_predictions_he.csv
Complete · 9h ago
0.88740

AdaBoost (Adaptive Boosting) is an ensemble technique that sequentially builds a series of weak learners (typically decision trees). Each new model focuses on correcting the errors made by the previous ones. The final prediction is made by combining the outputs of all individual models, giving more weight to models that made fewer mistakes.

We achieved an accuracy of **88.740%** on the testing data.

```
AdaBoost Classifier Accuracy: 0.8851413093140135
Classification Report for AdaBoost:
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     54106
           1       0.58      0.07      0.12      7178

    accuracy                           0.89     61284
   macro avg       0.74      0.53      0.53     61284
weighted avg       0.85      0.89      0.84     61284
```

5.7.2 Classification Report for AdaBoost

Observations:

- **Precision, Recall, and F1-Score:**
  - For the non-default class (class 0), the precision is 0.89, and the recall is 0.99, indicating that the model is effective at identifying non-default loans, with very few false positives.
  - For the default class (class 1), the precision is 0.58, and the recall is only 0.07. This shows a significant imbalance; while the model identifies some defaults, it misses a large number of them (low recall), resulting in a high number of false negatives.
  - The F1-score for class 1 is 0.12, reflecting poor performance in predicting defaults. The model struggles to effectively classify the minority class, likely due to the class imbalance in the dataset.
- **Class Imbalance:** AdaBoost is more sensitive to the majority class (non-default), resulting in high accuracy but poor performance for the minority class (defaults). This is evident from the disparity between the precision and recall for class 1.

5.7.3 Learnings:

- AdaBoost's strength lies in its iterative learning process, where it focuses more on instances misclassified by earlier models. However, in the presence of significant class imbalance, as seen in this dataset, it can become biased towards the majority class.
- In this case, while the model performs well on the majority class (non-default), it fails to generalize effectively to the minority class (defaults). This is evident from the poor recall and F1-score for class 1.

6 Conclusion

The machine learning model development process for predicting loan defaults addressed key challenges such as class imbalance, low recall for the minority class (class 1), and improving overall performance metrics.

From the models evaluated, the **CatBoost Classifier** was identified as the best-performing model, achieving the highest accuracy and effectively handling the dataset's unique characteristics. After applying preprocessing and feature engineering steps, the final predictions were generated using the test dataset. These predictions were structured into a submission file for evaluation.

This process highlights the model's capability to provide actionable insights, enabling financial institutions to mitigate loan default risks effectively. The submission file has been successfully created and is ready for deployment or further analysis.