Credit Card Behaviour Score Documentation

Abstract

In today's modern world Artificial Intelligence and Machine Learning has become an integral part in various fields. From banking to healthcare, Machine Learning is used intensively to enhance efficiency and do tasks which humans can't handle.
This project aims to develop a predictive Behaviour Score model for Bank A to assess the risk of credit card default among its existing customers. Using a dataset of 96,806 historical credit card records with various attributes and a binary `bad_flag` representing default status, we designed a machine learning-based solution to forecast future defaults. The dataset includes transaction-level details, credit limits, and bureau-related features. Our approach involved comprehensive data preprocessing with K-Nearest Neighbors (KNN) imputation for missing values, feature scaling, and selection to optimize performance. We adopted an ensemble modeling strategy combining Logistic Regression, Random Forest, and Gradient Boosting Classifiers. The outputs from these base learners were merged into a meta-model to refine predictions. Evaluation metrics, including accuracy and AUC-ROC, demonstrated strong predictive capability, with Random Forest achieving perfect accuracy but showing potential overfitting. Insights from this analysis highlight the importance of feature engineering, regularization, and GPU-accelerated computing for future improvements. This Behaviour Score model supports proactive credit risk management, aligning with the bank's goal of mitigating potential losses and enhancing profitability.

Introduction

In today's rapidly evolving financial landscape, the risk of credit default poses significant challenges for banking institutions. Predicting customer behavior through advanced machine learning models is critical for effective credit risk management. Bank A aims to enhance its risk evaluation system by developing a **Behaviour Score model** to predict the likelihood of credit card defaults among existing customers. Unlike conventional credit scoring models, which often rely solely on limited variables, our model leverages a comprehensive ensemble-based framework to enhance predictive accuracy.

Our approach combines data preprocessing, feature selection, and ensamble-learning to address the complexities of real-world financial datasets. The process begins with handling missing data using mean, followed by feature scaling using Standard Scaler to normalize diverse variables. We used transformations like yeo-johnson method.We then implemented ensemble learning using **Logistic Regression**, **Random Forest, Gradient Boosting Classifiers,Decision tree, knn as base learners, combining their predictions into a meta-model for refined classification.**

While traditional frameworks face limitations such as multicollinearity and overfitting, our adaptive methodology focuses on robustness and interpretability. The ensemble strategy optimize both performance and efficiency. Our final model demonstrates superior predictive power, as reflected in high accuracy and robust evaluation metrics.

Incorporating feature engineering, regularization, and cross-validation further enhances model reliability. This predictive Behaviour Score framework offers Bank A   a proactive tool for managing portfolio risks, aligning with its strategic objectives to mitigate losses and maximize profitability. Future directions include expanding the model to integrate additional data sources and exploring deep learning techniques for more granular predictions.

Dataset

- **Development Data**: 96,806 records of credit card customers with several independent variables and `bad_flag` as the target variable.
- **Validation Data**: 41,792 records with the same independent variables but without `bad_flag`.

The features include:

- **On-us Attributes**: Variables prefixed with "onus_attributes" (e.g., credit limit).
- **Transaction-level Attributes**: Variables prefixed with "transaction_attribute" (e.g., number of transactions).
- **Bureau Tradeline Attributes**: Variables prefixed with "bureau" (e.g., historical delinquencies).
- **Bureau Enquiry Attributes**: Variables prefixed with "bureau_enquiry" (e.g., enquiries in the last 3 months).

Approach

Initial Data Exploration and Preprocessing

We thoroughly explored the dataset to identify patterns, missing values, and potential outliers. Key steps included:

1. **Filling of Missing Values**: Used mean of each attribute to fill missing data.
2. **Feature Selection and Dimensionality Reduction**: Reduced features using correlation analysis and `VarianceThreshold`.
3. **Outlier Removal**: Employed `IsolationForest` to filter out anomalies.
4. **Data Scaling** : Used `StandardScaler to normalize the data.`
5. **Transformation:** Used 'Yeo-Johnson' method for transformation as it can handle both positive and negative values.

Model Selection and Training

Model Evaluated

1.**Random Forest:** **Accuracy: 98.55%**

Confusion Matrix:

```
[[28535,   10],
 [  409,    1]]
```

Observations: High accuracy but poor performance in detecting the minority class (bad_flag = 1). Only one True Positive (TP).

2. **Logistic Regression:** **Accuracy: 77.79%**

Confusion Matrix:

```
[[22275,  6270],
 [  161,   249]]
```

Observations: Better at detecting minority class compared to Random Forest, but overall accuracy is lower.

3. XGBoost: Accuracy: 98.56%

Confusion Matrix:

```
[[28538,    7],
 [ 409,    1]]
```

Observations: Similar to Random Forest, struggles with identifying minority class.

4. SVM: Accuracy: 98.58%

Confusion Matrix:

```
[[28545,    0],
 [ 410,    0]]
```

Observations: Completely fails to identify any True Positives.

5. KNN: Accuracy: 77.85%

Confusion Matrix:

```
[[22359,  6186],
 [ 227,   183]]
```

Observations: Better at identifying the minority class than Random Forest but still has low Precision and Recall.

6. Neural Networks

Multiple configurations were tried:

Configuration 1:   Accuracy: 97.92%

Confusion Matrix:

```
[[28339,  206],
 [  395,   15]]
```

Configuration 2: Accuracy: 97.48%

Confusion Matrix:

```
[[28193,  352],
 [  379,   31]]
```

Configuration 3: Accuracy: 89.20%

Confusion Matrix:

```
[[25673,  2872],
 [  255,   155]]
```

Observations: Neural Networks achieved the best True Positive rates with further tuning.Slightly lower accuracy than tree-based models but better recall.

Class Balancing and Evaluation

SMOTE was applied to address class imbalance, and the model was evaluated using:

- **Accuracy**: Overall predictive performance.
- **Confusion Matrix**: To analyze false negatives and positives.
- **AUC-ROC Score**: For discrimination between classes.

Algorithms

Overview:

Ensemble learning combines multiple machine learning models to achieve better performance than any single model. The idea is that different models may capture different patterns in the data, and by combining them, we can create a more robust and accurate model. In this case, we are using a combination of five different classifiers to form an ensemble: Random Forest, Logistic Regression, XGBoost, K-Nearest Neighbors, and Decision Tree.

initializing Base Models:

We define five base models to be included in the ensemble. Each model has different strengths, which will complement each other when combined.

RandomForestClassifier: A forest of decision trees that are trained on random subsets of the data. It is robust and generally performs well in practice.

LogisticRegression: A linear model that works well for binary classification problems. It's simple and interpretable.

XGBClassifier: A gradient-boosted tree model known for its high performance and efficiency. It works well with structured data.

KNeighborsClassifier: A non-parametric classifier that makes decisions based on the majority class of nearby instances.

DecisionTreeClassifier: A simple, interpretable model that splits the data based on the features with the highest information gain.

Training the Models:

Each model is trained using the training dataset. By fitting these models to the training data, they learn the patterns and relationships between the features and the target. Once the models are trained, they can make predictions on the testing dataset.

Once the base models are trained and evaluated, you can combine their predictions using ensemble methods. There are two common ensemble methods:

Voting Classifier (Soft Voting):

This method combines the predictions of multiple models by averaging their probabilities (for classification) and choosing the class with the highest average probability.

Voting='soft': In soft voting, the model outputs class probabilities, and the class with the highest average probability is chosen as the final prediction.

Bagging and Boosting:

Bagging (Bootstrap Aggregating): Multiple models are trained independently on different subsets of the data (with replacement), and their predictions are averaged. Random Forest is an example of a bagging algorithm.

Boosting: Models are trained sequentially, where each new model corrects the mistakes of the previous one. XGBoost is an example of a boosting algorithm.

Conclusion

This ensemble approach combines the strengths of multiple models to improve predictive performance. Each individual classifier has different characteristics, and by combining them, the ensemble typically results in better generalization, accuracy, and robustness compared to a single model.

The main advantages of ensemble methods include:

Improved accuracy: By leveraging multiple models, the ensemble can correct errors from individual models.

Reduced overfitting: By combining several models, the risk of overfitting to the training data can be reduced.

Better generalization: Ensembles often perform better on unseen data than any single model.

Metrices

We used Confusion matrix, classification report from scikit learn metrices module to evaluate our model performance on development file after training of model. Given below are result of it:

Meta-Model Accuracy: 99.99%

Precision: 0.9998

Recall: 1.0000

F1 Score: 0.9999

ROC AUC Score: 1.0000

Confusion Matrix:

```
[[8486    2]
 [   0 8597]]
```