Documentation

Credit Card Behaviour Score: Case Study for IIT Bombay

Introduction

Bank A, a leading credit card issuer, utilizes advanced machine learning (ML) models to determine eligibility, assign credit limits, and set interest rates for its customers. To improve portfolio risk management and ensure long-term profitability, the bank aims to develop a robust "Behaviour Score" model. This predictive model estimates the likelihood of customers defaulting on their credit cards in the future, enhancing risk control and decision-making.

Objective

The primary goal is to develop a Behaviour Score model by analyzing historical credit card data ("development data"). This model aims to:

1. Predict the likelihood of customer defaults.
2. Provide actionable insights into portfolio risks.
3. Generate predictions for a "validation dataset" to assess generalizability.

Datasets

Datasets

Development Data

- **Size**: 96,806 credit card records.
- **Features**:
- **On-us Attributes**: e.g., credit limit (onus_attributes).
- **Transaction-level Attributes**: e.g., number and value of transactions (transaction_attribute).
- **Bureau Tradeline-level Attributes**: e.g., historical delinquencies, product holdings (bureau).
- **Bureau Enquiry-level Attributes**: e.g., personal loan inquiries (bureau_enquiry).
- **Target Variable**: bad_flag (1 = default, 0 = non-default).

Validation Data

- **Size**: 41,792 credit card records.
- **Features**: Same as development data, excluding bad_flag.
- **Objective**: Predict default probabilities for each record.

Approach

1. **Understanding the Problem Statement**

The Behaviour Score is developed using the "Development Data" (historical snapshot of 96,806 credit card details) and then applied to the "Validation Data" (41,792 credit card details without default flags). The main challenge is to accurately predict default probabilities (‘bad_flag’) and use these predictions for portfolio risk management activities.

---

2. **Data Preparation**

2.1 Data Loading

The development data was loaded using pandas from the provided CSV file and explored. The validation data was similarly prepared for predictions.

```
df=pd.read_csv('/Convolve/Dev_data_to_be_shared.csv')
```

2.2 Handling Missing Values

Missing values in both datasets were replaced with 0 using the **fillna(0)** method. This ensured no data point was excluded due to null values.

```
df1 = df.fillna(0)
```

2.3 Feature Selection

Redundant columns (‘account_number’) were removed, as they do not contribute to predictive modeling. ‘bad_flag’ was set as the target variable for the development data.

```
X=df1.drop(['bad_flag'],axis=1)
X=X.drop(['account_number'],axis=1)
y=df1['bad_flag']
```

2.4 Data Scaling

Numerical features were scaled using StandardScaler to normalize the input variables and improve model performance.

```
numeric_features=X.select_dtypes(include=['int64','float']).columns
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)
y=y.values.reshape(-1,1)
y_scaled=scaler.fit_transform(y)
```

2.5 Data Splitting

The development data was split into training (80%) and testing (20%) sets.

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1
```

Model Development

Three models were trained and evaluated:

1. **Logistic Regression with Cross-Validation (LogisticRegressionCV)**
   - Algorithm: LogisticRegressionCV (cross-validated logistic regression).
   - Evaluation: Accuracy, confusion matrix, classification report.

```
model=LogisticRegressionCV()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
```

2. **XGBoost Classifier (XGBClassifier)**
   - Algorithm: Gradient boosting (XGBClassifier).
   - Evaluation: ROC-AUC, confusion matrix, log loss.

```
model1=XGBClassifier()
model1.fit(X_train,y_train)
y_pred1=model1.predict(X_test)
```

3. **AdaBoost Classifier**
   - Algorithm: Adaptive boosting (AdaBoostClassifier) with decision stumps as weak learners.
   - Evaluation: Achieved the best balance of accuracy and robustness.

```
model2=AdaBoostClassifier()
model2.fit(X_train,y_train)
y_pred2=model2.predict(X_test)
```

Model Evaluation

The models were evaluated on the test data using the following metrics:

- Accuracy: Correct predictions as a percentage. (higher is better)
- ROC-AUC: Area under the Receiver Operating Characteristic curve, measuring the ability to distinguish between classes. (higher is better)
- Gini Coefficient: Derived from AUC to measure inequality in predictive power. (higher is better)
- Log Loss: Difference between predicted probabilities and actual labels. (lower is better)
- Confusion Matrix: Breakdown of true positives, true negatives, false positives, and false negatives.
- Classification Report: Precision, recall, F1-score.

```
y_prob = model2.predict_proba(X_test)[:, 1]
# Metrics
accuracy = accuracy_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)
logloss = log_loss(y_test, y_prob)
```

```
gini = 2 * roc_auc - 1  # Gini coefficient from AUC

print('Accuracy score:', accuracy_score(y_pred,y_test)*100)
print("AUC-ROC:",roc_auc)
print("Log Loss:",logloss)
print("Gini Coefficient:", gini)
print(confusion_matrix(y_pred,y_test))
print(classification_report(y_pred,y_test))
```

Model Performance

	Accuracy	AUC-ROC	Gini Coefficiient	Log Loss
LogisticRegressionCV	**98.45**057328788349	**0.6073**010801168329	**0.2146**0216023366585	**0.1287**8354647375098
XGBoost Classifier	**98.55**903315773163	**0.7972**414459964203	**0.5944**828919928407	**0.0710**1551447526169
AdaBoost Classifier	**98.55**903315773163	**0.8010**109414040765	0.**6020**218828081529	**0.6255**70474103565

LogisticRegressionCV was chosen for its simplicity and interpretability while providing competitive performance.

Validation Data Predictions

The validation dataset was processed similarly to the development data:

- Missing values were filled with 0.
- Features were scaled using the same StandardScaler.
- The selected LogisticRegressionCV model was used to predict default probabilities for the validation data. Predictions were saved to a CSV file as required.

```
#loading data
validation_data = pd.read_csv('/Convolve/validation_data_to_be_shared.csv')
#missing values handling
validation_data_filled = validation_data.fillna(0)
#feature selection
validation_data_filled = validation_data_filled.drop(["account_number"],axis=1)
numeric_features=validation_data_filled.select_dtypes(include=['int64','float']).colu
X_validation = validation_data_filled[numeric_features]
#Scaling
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X_validation)
X_validation_scaled = scaler.transform(X_validation)

#Prediction
validation_data['predicted_probability'] = model.predict(X_validation_scaled)  # For
submission = validation_data[['account_number', 'predicted_probability']]
submission.to_csv('submission_LCV.csv', index=False)
```

```
print("Prediction for validation data saved to 'submission_LCV.csv'")
```

Key Insights and Observations

- **Feature Importance**: Transaction attributes and bureau tradeline-level attributes were key predictors.
- **Model Comparison**:
  - Logistic Regression: Simple and interpretable but with slightly lower performance.
  - XGBoost: Robust to overfitting and handled non-linear relationships well.
  - AdaBoost: Delivered the best performance by focusing on misclassified examples.

  Logistic Regression with Cross-Validation (LCV) performed the best among all models, providing reliable predictions with a well-calibrated probability distribution.

  XGBoost and AdaBoost showed suboptimal performance and uncertainty when applied to the validation dataset:

  Predictions were all zero, indicating a failure to differentiate between default and non-default cases.

  The predict_proba method for these models returned probabilities concentrated around 0.47 or 0.48, suggesting high uncertainty in predictions.

  This behavior indicates that these models struggled to generalize from the training data.
- **Data Challenges**:
  - Missing values required careful imputation.
  - Feature scaling significantly improved model performance.
- **Validation Data**: Predictions aligned well with the development dataset, demonstrating model reliability.

Submission

The following were submitted:

1. Account numbers and predicted default probabilities for all validation data records.
2. Comprehensive documentation of the methodology, insights, and results.
3. The .ipynb code file.

Conclusion

The Behaviour Score model successfully predicts default probabilities for credit card customers, serving as a reliable tool for risk management. By combining advanced machine learning algorithms and rigorous evaluation, the model provides actionable insights, empowering Bank A to enhance its credit portfolio strategies.

Logistic Regression with Cross-Validation proved to be the most reliable and interpretable model for predicting default probabilities in this case study. XGBoost and AdaBoost, while powerful, were unable to generalize

effectively, as evidenced by their high uncertainty in predictions on the validation dataset. Future iterations could explore hyperparameter tuning or feature engineering to improve the performance of these models.