Report: Low-Resource Fraud Detection

Approach

The project's objective is to build a machine learning model for accurate fraudulent credit card transaction detection using limited data. The primary goal is to maximize the identification of fraudulent cases (recall) while minimizing false positives (precision). This is crucial for financial institutions to minimize losses and protect customers in new and rural markets with unreliable power and internet.

Methods

The methodology involved several key steps:

- **Dataset Selection**: The MLG-ULB Credit Card Fraud Detection Dataset from Kaggle was used. The dataset has two limitations: features are anonymized with Principle Component Analysis (PCA), and it contains only two days of data.
- **Data Preprocessing**: Initial data exploration revealed 1,081 duplicate rows and no null values. The duplicates were removed, reducing the dataset from 284,807 rows to 283,726. The dataset's class distribution showed a significant imbalance, with only 492 fraudulent transactions (Class 1) compared to 284,315 legitimate ones (Class 0).
- **Model Training and Evaluation**: The project used **Logistic Regression** and **Random Forest Classifier** models. Given the severe class imbalance, the **SMOTE (Synthetic Minority Over-sampling Technique)** method was applied to oversample the minority class (fraudulent transactions) in the training data. The models were evaluated using a custom scorer that prioritizes recall over precision, defined as (precision+2*recall)/3. The baseline models were then optimized using **GridSearchCV** with **StratifiedKFold** to find the best hyperparameters.
- **Code Snippets**:
  - **Data Loading:**
    ```
    # Download latest version
    dataset_path = kagglehub.dataset_download("mlg-ulb/creditcardfraud")

    # Read the creditcard.csv file into a pandas DataFrame
    df = pd.read_csv(f'{dataset_path}/creditcard.csv')
    ```
  - **Data Cleaning:**
    ```
    # Removing duplicates
    initial_rows = df.shape[0]
    df = df.drop_duplicates().copy()
    rows_removed = initial_rows - df.shape[0]
    print(f"Number of duplicate rows removed: {rows_removed}")
    print(f"The dataset now has {df.shape[0]} rows and {df.shape[1]} columns")
    ```
  - **SMOTE Over-sampling:**

```
# Oversampling the training data using SMOTE
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
```

- **Custom Scoring for GridSearchCV:**

```
def custom_scorer(y_true, y_pred):
    precision = precision_score(y_true, y_pred)
    recall = recall_score(y_true, y_pred)
    # Prioritize recall with a higher weight
    return (precision + 2 * recall) / 3

# Make the custom scorer a scikit-learn scorer
my_scorer = make_scorer(custom_scorer)
```

Results

The models' performance was compared based on F1-score, Precision, and Recall. The optimized Logistic Regression model showed the best performance.

- **Final Performance Comparison**:
  - **Optimized Logistic Regression:**
    - F1-Score: 0.8173
    - Precision: 0.7303
    - Recall: 0.9255
  - **Optimized Random Forest Classifier:**
    - F1-Score: 0.7725
    - Precision: 0.7818
    - Recall: 0.7634

The optimized Logistic Regression model achieved a 2.38% improvement in F1-score and a 23.11% improvement in Recall compared to the baseline Logistic Regression model.

Conclusion

The optimized Logistic Regression model proved to be the most effective for low-resource fraud detection. The project successfully demonstrated that with appropriate data preprocessing and model optimization techniques, a high-performing fraud detection system can be developed even with limited data. The custom scoring metric, which prioritized recall, was a key factor in achieving a high true positive rate, which is critical for minimizing financial losses.