RIGA TECHNICAL UNIVERSITY

Faculty of Computer Science and Information Technology Institute of Applied Computer Systems

INDIVIDUAL TASK WORK

"Intelektuālo lietišķo datorsistēmu uzbūves metodes"

Accomplished: Anthony LAFARGUE

Student Card No: 240AEM029

2024./2025

Shopping Behavior Prediction: Final Report

Introduction

The objective of this project was to develop a classifier capable of predicting whether a user would make a purchase based on their online session data. Using a dataset containing information about user sessions, we explored different approaches and techniques to optimize the classification task.

The primary metrics for evaluation were:

1. **Sensitivity**: Proportion of positive examples (users who made a purchase) correctly identified.
2. **Specificity**: Proportion of negative examples (users who did not make a purchase) correctly identified.

Dataset and Preprocessing

The dataset consisted of the following features for each user session:

- **Numerical**: Page visits, duration on pages, bounce rates, exit rates, etc.
- **Categorical**: Month, visitor type (returning or not), weekend status (true/false).
- **Target**: Whether a user made a purchase (1) or not (0).

Steps Taken:

1. **Exploratory Data Analysis (EDA):**
    - Checked for missing values and inconsistencies.
    - Explored correlations between features and the target variable.
    - Identified class imbalance, where users who did not make a purchase significantly outnumbered those who did.
2. **Feature Engineering:**
    - Encoded categorical features (e.g., converting months to integers and visitor type to binary values).
    - Normalized numerical features to improve model performance.
3. **Data Splitting:**
    - Divided the dataset into training (60%) and testing (40%) sets.

Models Evaluated

We tested several models to identify the best approach for this task:

1. Baseline K-Nearest Neighbors (KNN):

- Implemented a basic KNN classifier with .
- Observed high specificity but low sensitivity due to class imbalance.

```
Best Parameters: {'metric': 'euclidean', 'n_neighbors': 3, 'weights': 'uniform'}
Correct: 4297
Incorrect: 635
True Positive Rate: 39.50%
True Negative Rate: 95.83%
```

2. KNN with Class Weights:

- Adjusted the model to weigh classes inversely proportional to their frequency.
- Achieved improved sensitivity but at the cost of slight specificity degradation.

```
Best Parameters: {'metric': 'euclidean', 'n_neighbors': 3, 'weights': 'uniform'}
Correct: 4297
Incorrect: 635
True Positive Rate: 39.50%
True Negative Rate: 95.83%
```

3. KNN with SMOTE (Synthetic Minority Oversampling Technique):

- Applied SMOTE to generate synthetic samples for the minority class (users who made a purchase).
- Observed a balanced improvement in both sensitivity and specificity, making this the preferred approach.

Implementation in Google Colab

To provide a flexible and interactive environment for this project, we implemented the classifier using Google Colab. This allows easy access to the dataset, integration of additional libraries like SMOTE, and visualization tools. Key steps include:

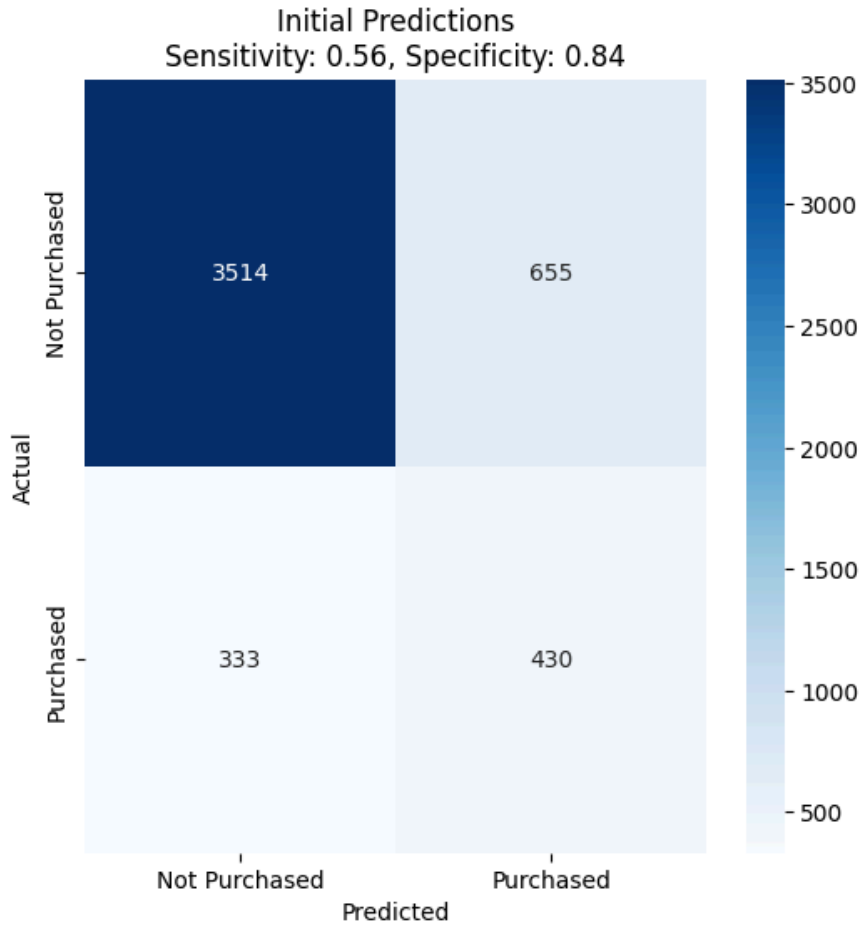
1. **Uploading the Dataset:**
   - The dataset is uploaded directly into the Colab environment using `google.colab.files.upload()`.
2. **Data Preprocessing:**
   - Handled directly in Colab with features like StandardScaler for normalization and SMOTE for oversampling.
3. **Model Training with GridSearchCV:**
   - Optimized hyperparameters for KNN using Stratified K-Fold cross-validation.
4. **Visualization:**
   - Confusion matrix and other visualizations are generated using `matplotlib` and `seaborn`.

The Colab implementation provides flexibility for experimenting with the dataset and models in an interactive notebook format. The code is included in the `shopping_colab.py` script.

Results

When tested on the dataset, the final model achieved:

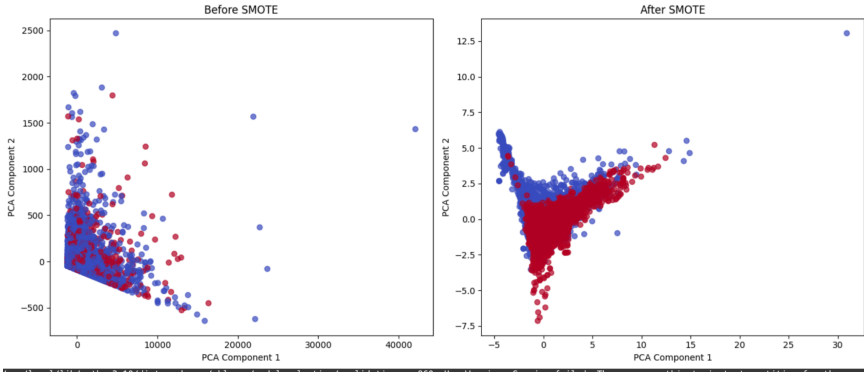
- **Sensitivity (True Positive Rate):** 56.36%
- **Specificity (True Negative Rate):** 84.29%
- **Accuracy:** 79.99%



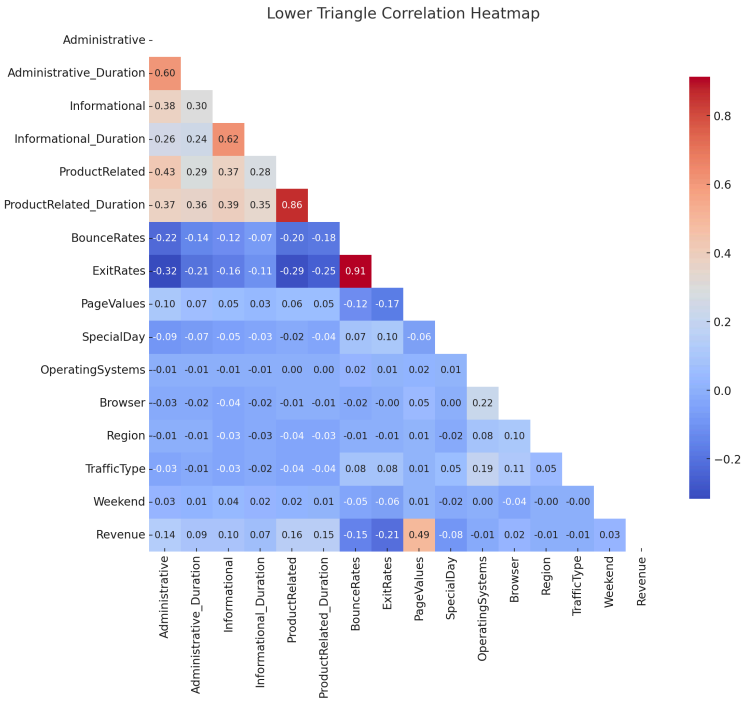
The combination of KNN with SMOTE provided the best balance between sensitivity and specificity, outperforming the baseline model.

Challenges and Insights

1. **Class Imbalance:**
    - Without addressing imbalance, the classifier heavily favored the majority class, leading to poor sensitivity.



2. **Feature Importance:**
    - Not all features contributed equally; exit rates and page values had the highest predictive power.



3. **Model Complexity:**
    - Simpler models like KNN performed well, emphasizing that added complexity (e.g., deeper models) was unnecessary for this dataset.

Conclusion

The final model provides a robust solution for predicting purchasing behavior. By leveraging SMOTE and careful preprocessing, we ensured a balanced and interpretable classifier that performs well across key metrics.

Code and Usage

The implementation is provided in `shopping.py` for local execution and as a Colab-compatible script in `shopping_colab.py`. Execute the Colab implementation as follows:

1. Open the Colab notebook.
2. Upload the dataset using the upload button.
3. Run all cells to preprocess the data, train the model, and visualize results.

To execute locally:

python shopping.py shopping.csv

This project highlights the importance of addressing class imbalance and the effectiveness of simple, interpretable models for practical use cases.