Report on Predicting User Ad Clicks Using Machine Learning

Objective

The objective of this project was to build a predictive model to determine whether a user will click on an online ad based on their demographics, browsing behavior, the context of the ad's display, and the time of day. My goal was to clean and preprocess the data, address class imbalance, engineer useful features, and apply machine learning models to optimize performance. This analysis can be used to improve ad targeting strategies and optimize user interaction with online ads.

Data Preprocessing and Feature Engineering

1. **Dropped Unnecessary Columns**:

- I dropped columns such as id and full_name as they did not contribute any useful information for prediction.

2. **Created Interaction Terms**:

   I created interaction terms between:

- "device_type and time_of_day" & "ad_position and browsing_history"
- This was done to capture potential interactions that could influence ad click behavior. For example, a certain type of device may lead to higher click rates at specific times of the day.

3. **Handled Age Binning**:

- Initially, I binned the age column into categorical age groups (e.g., Teen, Young Adult, etc.). However, due to missing values in the resulting age_group column, I decided to drop this feature.

4. **One-Hot Encoding**:

- I applied one-hot encoding to categorical columns such as gender, device_type, ad_position, browsing_history, and time_of_day to transform them into a format suitable for machine learning models.

Initial Model Training

I began by training models on the original dataset before addressing the class imbalance issue. This helped establish baseline performance.

1. **Logistic Regression**:

- I started with Logistic Regression, a simple linear model for binary classification, to establish a baseline performance. The model showed reasonable accuracy but struggled to capture non-linear relationships and complex interactions.

2. **Random Forest**:

- Next, I trained a Random Forest model, which improved performance over Logistic Regression by using multiple decision trees. However, due to class imbalance, the model had difficulty predicting the minority class (click = 1).

3. **XGBoost**:

- I also trained an XGBoost model, known for its robustness in handling structured/tabular data. However, the initial results were limited due to the imbalance in the dataset. I noted the need for hyperparameter tuning and addressing class imbalance.

Handling Class Imbalance with SMOTE

After observing that the initial models were affected by the imbalance in the target class (click), I implemented **SMOTE (Synthetic Minority Over-sampling Technique)** to improve model performance.

1. **Class Imbalance in Target Variable (click)**:

- The click variable was highly imbalanced, with far more "non-clicks" than "clicks." This imbalance skewed the model's ability to predict the minority class.

2. **Applied SMOTE**:

- To address the class imbalance, I applied SMOTE, which generated synthetic samples of the minority class. This balanced the dataset, ensuring that the models could learn effectively from both classes.

Model Training After SMOTE

With the balanced dataset, I retrained the models to assess their improved performance.

1. **Logistic Regression**:

- Logistic Regression showed improved recall for the minority class after SMOTE, though its overall performance remained limited due to the model's linear nature.

2. **Random Forest**:

- Random Forest demonstrated significant improvement in predicting the minority class and achieved higher accuracy, precision, and recall. The model was able to capture complex feature interactions that Logistic Regression missed.

3. **XGBoost**:

- XGBoost, benefiting from SMOTE, provided the best results. I further optimized the model using **RandomizedSearchCV** to fine-tune hyperparameters such as n_estimators, max_depth, and learning_rate, which improved both accuracy and recall.

Hyperparameter Tuning for XGBoost

To maximize XGBoost's performance, I conducted hyperparameter tuning using **RandomizedSearchCV**. The key parameters I optimized were:

- n_estimators (number of trees)
- max_depth (maximum depth of each tree)
- learning_rate (step size for updating model parameters)
- subsample (proportion of data used for fitting individual trees)

This tuning process improved XGBoost's ability to capture complex patterns, resulting in the highest performance among all models.

Model Performance Evaluation

1. **Metrics Used**:

- **Accuracy**: The proportion of correct predictions.
- **Precision**: The proportion of positive identifications (clicks) that were correct.
- **Recall**: The proportion of actual positive cases (clicks) that were correctly identified.

2. **Model Performance**:

- **Logistic Regression**: While this model provided a solid baseline, its performance was limited by its linear assumptions.
- **Random Forest**: After SMOTE, Random Forest performed well, capturing non-linear interactions and achieving good recall for the minority class.
- **XGBoost**: With hyperparameter tuning, XGBoost achieved the best overall performance in terms of accuracy, precision, and recall.

Visualizations

1. **Class Distribution Before and After SMOTE**:

- I visualized the class distribution of the click variable before and after SMOTE, which showed a significant improvement in balance.

2. **Confusion Matrices**:

- Confusion matrices for each model were displayed to examine the number of true positives, false positives, true negatives, and false negatives. This helped assess model performance in detail.

3. **ROC Curves**:

- ROC curves were plotted to compare how well each model distinguished between the clicked and non-clicked ads. XGBoost had the highest AUC, indicating superior performance.

4. **Bar Charts for Accuracy, Precision, and Recall**:

- I visualized the comparison of accuracy, precision, and recall across all models, with XGBoost standing out in terms of overall performance.

Conclusion and Recommendations

- **Best Model**: XGBoost emerged as the best-performing model after tuning and balancing the dataset with SMOTE. It outperformed Logistic Regression and Random Forest in terms of accuracy, precision, and recall.
- **Improving Ad Targeting**: The results of this model can be applied to optimize ad targeting strategies by focusing on key features such as device_type, time_of_day, and ad_position to increase click-through rates.
- **Further Improvements**:
    1. Additional feature engineering, such as incorporating user-specific browsing habits or time zone data, may further enhance model performance.
    2. Real-time predictions could be deployed to adjust ad placements dynamically based on user behavior.

By leveraging the results of these models, businesses can better understand which factors drive user interaction with ads and adjust their ad placement strategies accordingly.