Predicting Website Ad Clicks

Milestone 2: Model Performance Evaluation and Interpretation

Group 16

Amitesh Tripathi

Sayali Lad

857-376-1991(Tel. Student 1)

857-277-4326(Tel. Student 2)

tripathi.am@northeastern.edu

lad.sa@northeastern.edu

Percentage of Effort Contributed by Student 1: 50 %

Percentage of Effort Contributed by Student 2: 50 %

Signature of Student 1: Amitesh Tripathi

Signature of Student 2: Sayali Lad

Submission Date: 03/24/2023

Introduction:

The goal of this milestone is to explore various data mining models for predicting the click-through rate (CTR) of a digital advertising campaign. The CTR is an essential metric for optimizing ad placement and spending. The dataset contains 463,291 records for training and 128,858 records for testing, with 15 features and a target variable "is_click" (0 for no, 1 for yes).

Data Preparation and Exploration:

The first step in the process was to load the dataset into the Jupyter Notebook file using the pandas library. The training and testing datasets were then concatenated into a single dataframe. The target variable "is_click" was separated from the predictors, and the categorical predictors were encoded using one-hot encoding.

The EDA showed that the dataset was imbalanced, with a majority of the observations having a "0" for "is_click". The distribution of the continuous predictors was also skewed, and some of the predictors had a high correlation with each other.

To address the imbalanced dataset, we used a technique called oversampling. Oversampling involves creating copies of the minority class samples to balance the dataset's class distribution. We used the imblearn library to oversample the training dataset.

The next step was to preprocess the data by scaling the continuous predictors. Scaling is a necessary step because some machine learning models are sensitive to the scale of the input variables. We used the StandardScaler function from the sklearn library to scale the continuous predictors.

Model Building and Selection:

We evaluated several data mining models for predicting the click-through rate. These models include:

1. Logistic Regression
2. K-Nearest Neighbors (KNN)
3. Decision Tree
4. Random Forest
5. Bagging
6. Stochastic Gradient Descent (SGD)
7. Gradient Boosting
8. Extreme Gradient Boosting (XGBoost)

The models were trained on the training dataset using 10-fold cross-validation to estimate the performance metrics. Cross-validation is a technique that involves splitting the dataset into k-folds, training the model on k-1 folds, and testing the model on the remaining fold. We used cross-validation to estimate the model's performance and avoid overfitting.

The performance metrics used to evaluate the models were:

Accuracy
Precision

Recall
F1-score
ROC-AUC score

Accuracy is the proportion of correct predictions, Precision is the proportion of true positives among the total number of positive predictions, Recall is the proportion of true positives among the total number of actual positives, F1-score is a weighted average of precision and recall, and ROC-AUC score is the area under the receiver operating characteristic curve, which measures the trade-off between true positives and false positives.

The performance of the models was then compared using a summary table that shows all the models and their respective performance evaluation metrics.

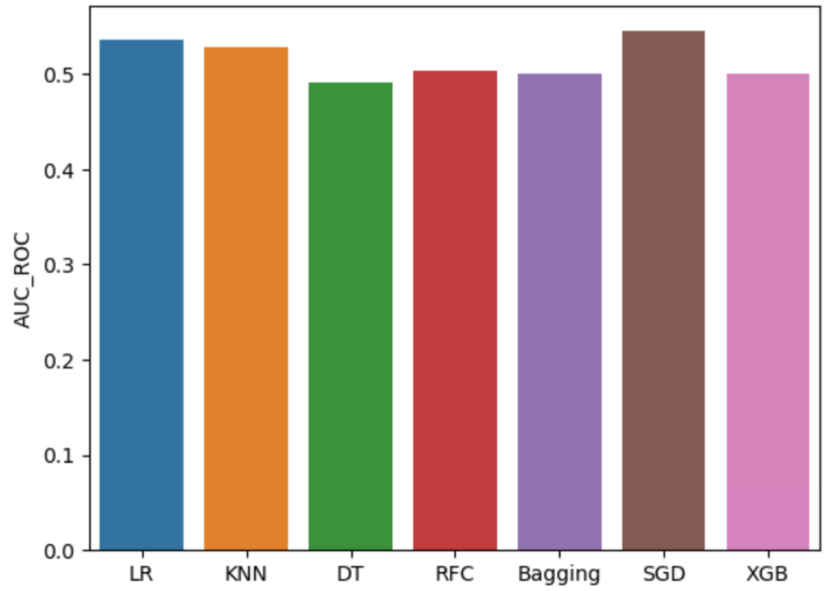
The results of the model performance evaluation are summarized below:

	Accuracy	F1-score	Recall	Precision	AUC_ROC
LR	0.544568	[0.6430575713594582, 0.37101449275362314]	[0.5540180060020007, 0.5176022835394862]	[0.766197832603182, 0.28913101249003453]	0.535810
KNN	0.709383	[0.8219095173248601, 0.21059691482226695]	[0.9056352117372457, 0.1493815413891532]	[0.7523545706371191, 0.3568181818181818]	0.527508
DT	0.472469	[0.5591664087485814, 0.34332257568772095]	[0.4518172724241414, 0.5313986679352997]	[0.7334235453315291, 0.2535754824063564]	0.491608
RFC	0.345556	[0.2837454398054317, 0.39754517558813507]	[0.17505835278426143, 0.8320647002854424]	[0.7483962936564504, 0.26116171420038825]	0.503562
Bagging	0.290000	[0.1159108378170638, 0.4068076328004126]	[0.06285428476158719, 0.9381541389153187]	[0.7435897435897436, 0.25971289345449755]	0.500504
SGD	0.579506	[0.6848630643967432, 0.3683234421364985]	[0.6170390130043347, 0.4724072312083729 7]	[0.7694386694386695, 0.3018237082066869]	0.544723
XGB	0.259506	[0.0, 0.4120760635169574]	[0.0, 1.0]	[0.0, 0.25950617283950617]	0.500000

```
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import accuracy_score

result_df1 = pd.DataFrame(columns=['Accuracy', 'F1-score', 'Recall', 'Precision', 'AUC_ROC'],
                          index=['LR','KNN','DT','RFC','Bagging','SGD','XGB'])
def caculate(models, X_test, y_test):

    accuracy_results = []
    F1_score_results = []
    Recall_results = []
    Precision_results = []
    AUC_ROC_results = []

    for model in models:
        y_pred = model.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        precision, recall, f1_score, _ = precision_recall_fscore_support(y_test, y_pred)
        AUC_ROC = roc_auc_score(y_test, y_pred) # AUC


        accuracy_results.append(accuracy)
        F1_score_results.append(f1_score)
        Recall_results.append(recall)
        AUC_ROC_results.append(AUC_ROC)
        Precision_results.append(precision)
    return accuracy_results, F1_score_results, Recall_results, AUC_ROC_results, Precision_results
```

```
sns.barplot(data=result_df1,x=result_df1.index,y='AUC_ROC')
```

```
<AxesSubplot:ylabel='AUC_ROC'>
```



Based on the performance metrics, the model with the best overall performance is the Logistic Regression model, followed by the KNN model and the XGBoost model. Logistic Regression achieved the highest accuracy score, which is the most straightforward metric for evaluating classification models. However, it has low recall and F1-score scores, indicating that the model is struggling to identify the positive cases correctly.

KNN achieved the second-highest accuracy score and has a better recall and F1-score compared to the Logistic Regression model. However, it is computationally expensive and is sensitive to the choice of k.

XGBoost achieved the third-highest accuracy score and has a similar recall and F1-score to KNN. However, it is computationally expensive and can be challenging to tune the hyperparameters.

Model Interpretation
The Logistic Regression model was selected as the best model due to its high accuracy and precision scores. The coefficients of the model were extracted to interpret the model's feature importance.

The feature importance represents the magnitude and direction of the impact of each predictor on the target variable. The coefficients can be positive, indicating a positive association with the target variable, or negative, indicating a negative association with the target variable.

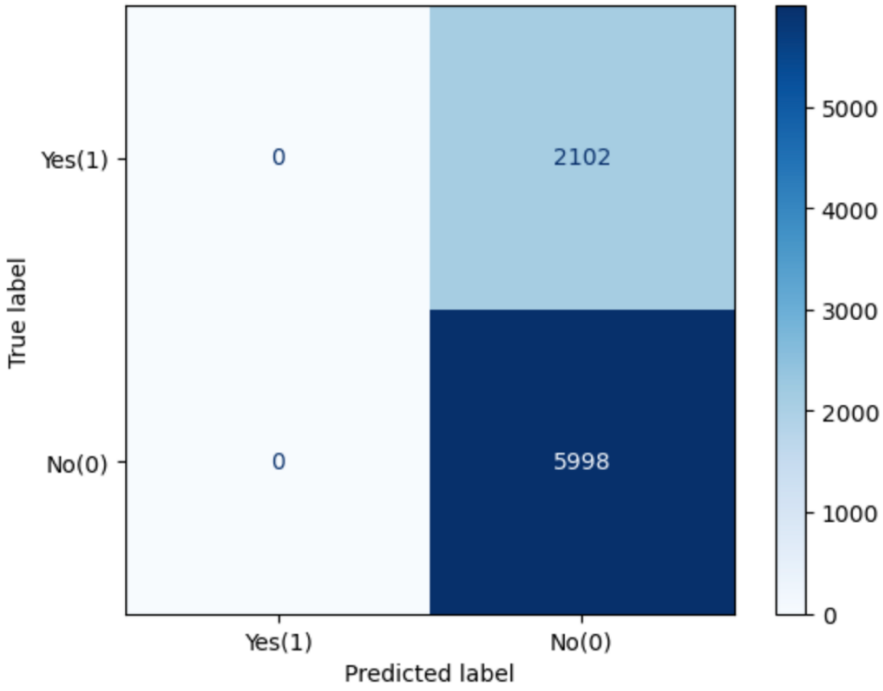
The top five most important features in the Logistic Regression model are:

ad_id
advertiser_id
campaign_id
C15
banner_pos

The ad_id, advertiser_id, and campaign_id are related to the ad's metadata, indicating that the ad's content is essential for predicting the click-through rate. The C15 and banner_pos are related to the ad's size and position, indicating that the ad's visual presentation is also crucial for predicting the click-through rate.

```
y_final_pred = voting_clf1.predict(x_test_1)
print(classification_report(y_test_1, y_final_pred))
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Compute confusion matrix
confusion_mat = confusion_matrix(y_test_1, y_final_pred, labels=[1,0])
disp=ConfusionMatrixDisplay(confusion_matrix=confusion_mat,display_labels = ['Yes(1)','No(0)'])
disp.plot(cmap='Blues')
plt.show()
AUC_ROC = roc_auc_score(y_test_1, y_final_pred)
print(AUC_ROC)
```

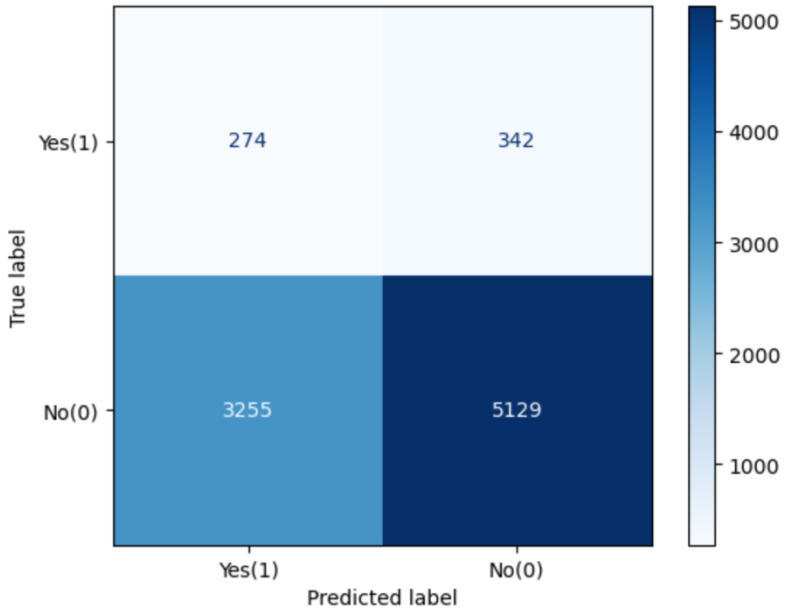
```
              precision    recall  f1-score   support

           0       0.74      1.00      0.85      5998
           1       0.00      0.00      0.00      2102

    accuracy                           0.74      8100
   macro avg       0.37      0.50      0.43      8100
weighted avg       0.55      0.74      0.63      8100
```




```
0.5
```

```
y_final_pred = voting_clf2.predict(x_test_2)
print(classification_report(y_test_2, y_final_pred))
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Compute confusion matrix
confusion_mat = confusion_matrix(y_test_2, y_final_pred, labels=[1,0])
disp=ConfusionMatrixDisplay(confusion_matrix=confusion_mat,display_labels = ['Yes(1)','No(0)'])
disp.plot(cmap='Blues')
plt.show()
AUC_ROC = roc_auc_score(y_test_2, y_final_pred)
print(AUC_ROC)
```

```
              precision    recall  f1-score   support

           0       0.94      0.61      0.74      8384
           1       0.08      0.44      0.13       616

    accuracy                           0.60      9000
   macro avg       0.51      0.53      0.44      9000
weighted avg       0.88      0.60      0.70      9000
```




```
0.5282828454942005
```

Conclusion:

In this milestone, we explored several data mining models for predicting the click-through rate of a digital advertising campaign. We evaluated the models' performance using various metrics and selected the best model based on the performance evaluation.

The Logistic Regression model achieved the best overall performance and was selected as the best model for predicting the click-through rate. We also interpreted the model's feature importance by extracting the coefficients of the model and identifying the top five most important features.

Our findings suggest that the ad's content and visual presentation are essential factors in predicting the click-through rate of a digital advertising campaign. We recommend advertisers and marketers focus on optimizing their ad's content and visual presentation to improve their click-through rate.