Programme – MSc. IN BUSINESS ANALYTICS

Module – Data Mining - B9BA103

Assignment Title

CA 1 – DATA MINING

Submitted to:

Mr. Satya Prakash

Submitted By:

Mamta Devi 10616210

Rohit Verma 10635420

Abhinav Singh Voria..... 10634732

To create a classifier for predicting online shoppers' purchasing intention, we will use the given dataset and try five different classification algorithms: Random Forest, Adaptive Boosting, Support Vector Classification, Logistic Regression, and Multi-Layer Perceptron with 1 hidden layer. Let's go through the steps to build the classifier and then conduct a cost-benefit analysis.

Step 1: Preparing the Dataset First, we need to download and load the dataset into Python. We can use the following code to download and load the dataset:

Step 2: Data Preprocessing Next, we need to preprocess the data by handling missing values, encoding categorical variables, and splitting the dataset into features (X) and the target variable (y). We also split the data into training and testing sets for model evaluation. Here's an example code snippet for data preprocessing:

Step 3: Building and Evaluating Classifiers Now we can build and evaluate classifiers using the five algorithms mentioned. Here's an example code snippet for each algorithm:

Random Forest:

Adaptive Boosting (AdaBoost):

Support Vector Classification (SVC):

Logistic Regression:

Multi-Layer Perceptron (MLP) with 1 hidden layer:

Step 4: Selecting the Best Model After evaluating all the classifiers, we can select the one with the highest accuracy as our best model. In this case, we will choose the model with the highest accuracy.

Step 5: Cost-Benefit Analysis To conduct a cost-benefit analysis, we need to make reasonable assumptions about the costs associated with true positives, false positives, true negatives, and false negatives. Let's assume the following:

True Positive (TP): The company correctly predicts a shopper's purchasing intention, leading to a successful sale. The benefit to the business is the average revenue generated from each successful sale.

False Positive (FP): The company incorrectly predicts a shopper's purchasing intention, but still invests resources in marketing or customer support. The cost to the business is the average cost incurred for marketing or customer support in such cases.

True Negative (TN): The company correctly predicts that a shopper will not make a purchase, so no additional costs or benefits are incurred.

False Negative (FN): The company fails to predict a shopper's purchasing intention, resulting in a lost sale. The cost to the business is the average revenue that would have been generated from each lost sale.

Several Python libraries and modules are imported by the provided code in order to carry out classification operations and data preprocessing. An explanation of each import statement is provided below:

import numpy as np

With this, the NumPy library is imported and given the alias np. With support for big, multi-dimensional arrays and matrices and a variety of mathematical functions to work with them, NumPy is a potent toolkit for numerical computations in Python.

import pandas as pd

By doing this, the Pandas library is imported and given the alias pd. Pandas is a library for analyzing and manipulating data. To work with structured data effectively, it offers data structures like DataFrames.

from sklearn.preprocessing import StandardScaler

This imports the StandardScaler class from the scikit-learn (sklearn) library's preprocessing module. By eliminating the mean and scaling to unit variance, the StandardScaler preprocessing method standardizes features. Before implementing machine learning algorithms, it is frequently employed.

from imblearn.over_sampling import SMOTE

This imports the SMOTE class from the imbalanced-learn (imblearn) library's over_sampling module. In order to overcome the issue of class imbalance in classification issues, SMOTE is a popular approach for oversampling the minority class in unbalanced datasets.

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier

The ensemble module of Scikit-Learn's RandomForest and AdaBoost classifiers are imported here. These two well-liked ensemble learning techniques are employed for categorization challenges. While AdaBoostClassifier builds a strong classifier by combining several weak classifiers, RandomForestClassifier builds a collection of decision trees and combines their predictions.

from sklearn.neural_network import MLPClassifier

This imports the neural_network module of Scikit-Learn's MLPClassifier class. Multi-Layer Perceptron Classifier, or MLPClassifier, is a name for a neural network model having one or more hidden layers. This feedforward artificial neural network design is popularly utilized for classification problems.

from imblearn.pipeline import Pipeline

This imports the Pipeline class from the imbalanced-learn library's pipeline module. A pipeline is a series of modeling or transformation steps used to process data. It makes it simple to link together several preprocessing procedures and a classification model at the end.

from sklearn.model_selection import StratifiedKFold

This imports the model_selection module of scikit-learn's StratifiedKFold class. The cross-validation method known as StratifiedKFold divides the dataset into folds while maintaining the same class distribution as the original dataset. It is frequently employed while assessing machine learning models.

from sklearn.svm import SVC

This imports the SVC class from Scikit-Learn's svm package. Support vector classifiers, sometimes known as SVMs, are a type of SVM that are used for classification problems. SVMs are supervised learning models that evaluate data and categorize it.

from sklearn import metrics

The metrics module from Scikit-Learn is imported using this. It offers a variety of metrics, including accuracy, precision, recall, F1-score, and others, to assess the effectiveness of machine learning models.

from sklearn.linear_model import SGDClassifier

This imports the SGDClassifier class from the scikit-learn linear_model package. SGDClassifier is a stochastic gradient descent (SGD)-trained linear classifier. Due to its effectiveness and capacity for handling sparse data, it is frequently employed for large-scale machine learning applications.

le = LabelEncoder()

A scikit-learn utility class called **LabelEncoder** aids in encoding categorical labels or target variables as integers. It is frequently employed to convert categorical labels into numerical forms, which many machine learning algorithms demand.

from sklearn.model_selection import train_test_split

This imports the model_selection module of Scikit-Learn's train_test_split function. It is typical practice to divide datasets into random train and test subsets using this algorithm.

To divide the labels (y) and features (X) into training and testing sets, this line of code uses train_test_split. The generated datasets are given the names X_train, X_test, y_train, and y_test.

X = df.drop('Revenue', axis=1)

y = df['Revenue']

The 'Revenue' column is removed from the DataFrame df via the df.drop('Revenue', axis=1) command, and the resulting DataFrame (which contains the features) is assigned to X. This assumes that the target variable is the 'Revenue' column and that it ought to be removed from the features.

The DataFrame df's 'Revenue' column is chosen and assigned to y via the df['Revenue'] command. This is based on the notion that the labels for the target variable, "Revenue," should be utilized.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

To divide the labels (y) and features (X) into training and testing sets, this line of code uses train_test_split. The generated datasets are given the names X_train, X_test, y_train, and y_test.

We will have four distinct datasets after running this code: X_train (for training features), X_test (for testing features), y_train (for training labels), and y_test (for testing labels). Then, machine learning models may be tested and trained using these datasets.

Critical analysis in terms of deployment of the model

1. Logistic Regression

Confusion Matrix:

The confusion matrix provides a breakdown of the true positive, false positive, true negative, and false negative predictions. In this case:

True Positives (TP): 140

False Positives (FP): 59

True Negatives (TN): 1996

False Negatives (FN): 271

Classification Report:

Precision:

• The precision for the "False" class is 0.88. This indicates that 88% of the instances labeled as "False" were really classified properly.

• The precision for the "True" class is 0.70. This indicates that 70% of the instances labeled as "True" are actually classified properly.

Recall:

• The recall (also known as sensitivity or true positive rate) for the "False" class is 0.97. This indicates that 97% of the cases in the "False" class are properly identified by the classifier.

• The recall (also known as sensitivity or true positive rate) for the "True" class is 0.34. This indicates that just 34% of the occurrences that belong to the "True" class are captured by the classifier.

F1-Score:

Precision and recall are combined into a single metric called the F1-score. It provides a fair assessment of a classifier's performance by taking the harmonic mean of precision and recall.

• The F1-score for the "False" class is 0.92.

• The F1-score for the "True" class is 0.46.

Support:

• The support shows how many instances of each class there are in the test set.

• The support for the "False" class is 2055.

• The support number for the "True" class is 411.

Accuracy:

• The classifier's overall accuracy on the test set is 0.87, meaning that 87% of the instances are classified correctly.

Macro Avg and Weighted Avg:

• The macro average determines the unweighted mean of the measurements across all classes.

• The weighted average determines the weighted mean of the metrics by taking the support (number of instances) for each class into account.

```
[[1996   59]
 [ 271  140]]
              precision    recall  f1-score   support

       False       0.88      0.97      0.92      2055
        True       0.70      0.34      0.46       411

    accuracy                           0.87      2466
   macro avg       0.79      0.66      0.69      2466
weighted avg       0.85      0.87      0.85      2466
```

2. Random Forest Classifier (RFC)

Confusion Matrix:

The true positive, false positive, true negative, and false negative predictions are broken down in the confusion matrix. In this instance

True Positives (TP): 224

False Positives (FP): 75

True Negatives (TN): 1980

False Negatives (FN): 187

Classification Report:

Precision:

• The precision for the "False" class is 0.91, meaning that 91% of the occurrences predicted to be "False" are actually categorized as such.

• The precision for the "True" class is 0.75, meaning that 75% of the occurrences predicted as "True" are actually categorized correctly.

Recall:

• The recall (also known as sensitivity or true positive rate) for the "False" class is 0.96, suggesting that 96% of the instances in this class are correctly identified by the classifier.

• The recall (also known as sensitivity or true positive rate) for the "True" class is 0.55, suggesting that the classifier correctly identifies 55% of cases that belong to the "True" class.

F1-Score:

Precision and recall are combined into a single metric called the F1-score. It provides a fair assessment of a classifier's performance by taking the harmonic mean of precision and recall.

• The F1-score for the "False" class is 0.94.

• The F1-score for the "True" class is 0.63.

Support:

• The support shows how many instances of each class there are in the test set.

• The support for the "False" class is 2055.

• The support number for the "True" class is 411.

Accuracy:

• The classifier's overall accuracy on the test set is 0.89, meaning that 89% of the occurrences are classified correctly.

Macro Avg and Weighted Avg:

• The macro average determines the unweighted mean of the measurements across all classes.

• The weighted average determines the weighted mean of the metrics by taking the support (number of instances) for each class into account.

```
[[1980   75]
 [ 187  224]]
              precision    recall  f1-score   support

       False       0.91      0.96      0.94      2055
        True       0.75      0.55      0.63       411

    accuracy                           0.89      2466
   macro avg       0.83      0.75      0.78      2466
weighted avg       0.89      0.89      0.89      2466
```

3. AdaBoost

Confusion Matrix:

The true positive, false positive, true negative, and false negative predictions are broken down in the confusion matrix. In this instance

True Positives (TP): 225

False Positives (FP): 103

True Negatives (TN): 1952

False Negatives (FN): 186

Classification Report:

Precision:

• The precision for the "False" class is 0.91, meaning that 91% of the occurrences predicted to be "False" are actually categorized as such.

• The precision for the "True" class is 0.69, meaning that 69% of the occurrences predicted as "True" are actually categorized correctly.

Recall:

• The recall (also known as sensitivity or true positive rate) for the "False" class is 0.95, meaning that the classifier correctly recognizes 95% of the cases falling under this category.

• The recall (also known as sensitivity or true positive rate) for the "True" class is 0.55, suggesting that 55% of the occurrences that belong to the "True" class are captured by the classifier.

F1-Score:

Precision and recall are combined into a single metric called the F1-score. It provides a fair assessment of a classifier's performance by taking the harmonic mean of precision and recall.

• The F1-score for the "False" class is 0.93.

• The F1-score for the "True" class is 0.61.

Support:

• The support shows how many instances of each class there are in the test set.

• The support for the "False" class is 2055.

• The support number for the "True" class is 411.

Accuracy:

• The classifier's overall accuracy on the test set is 0.88, meaning that 88% of the instances are classified correctly.

Macro Avg and Weighted Avg:

• The macro average determines the unweighted mean of the measurements across all classes.

• The weighted average determines the weighted mean of the metrics by taking the support (number of instances) for each class into account.

```
[[1952  103]
 [ 186  225]]
              precision    recall  f1-score   support

       False       0.91      0.95      0.93      2055
        True       0.69      0.55      0.61       411

    accuracy                           0.88      2466
   macro avg       0.80      0.75      0.77      2466
weighted avg       0.88      0.88      0.88      2466
```

4. Support Vector Classifier (SVC)

Confusion Matrix:

The true positive, false positive, true negative, and false negative predictions are broken down in the confusion matrix. In this instance

True Positives (TP): 6

False Positives (FP): 1

True Negatives (TN): 2054

False Negatives (FN): 405

Classification Report:

Precision:

• The precision for the "False" class is 0.84, meaning that 84% of the occurrences predicted to be "False" are actually categorized correctly.

• The precision for the "True" class is 0.86, meaning that 86% of the occurrences predicted as "True" are actually categorized correctly.

Recall:

• The recall (also known as sensitivity or true positive rate) for the "False" class is 1.00, demonstrating that the classifier correctly recognizes every occurrence that belongs to the "False" class.

• For the "True" class, the recall (also known as sensitivity or true positive rate) is 0.01, meaning that only 1% of the occurrences that belong to the "True" class are captured by the classifier.

F1-Score:

Precision and recall are combined into a single metric called the F1-score. It provides a fair assessment of a classifier's performance by taking the harmonic mean of precision and recall.

• The F1-score for the "False" class is 0.91.

• The F1-score for the "True" class is 0.03.

Support:

• The support shows how many instances of each class there are in the test set.

• The support for the "False" class is 2055.

• The support number for the "True" class is 411.

Accuracy:

• The classifier's overall accuracy on the test set is 0.84, meaning that 84% of the instances are classified correctly.

Macro Avg and Weighted Avg:

• The macro average determines the unweighted mean of the measurements across all classes.

• The weighted average determines the weighted mean of the metrics by taking the support (number of instances) for each class into account.

```
[[2054    1]
 [ 405    6]]
              precision    recall  f1-score   support

       False       0.84      1.00      0.91      2055
        True       0.86      0.01      0.03       411

    accuracy                           0.84      2466
   macro avg       0.85      0.51      0.47      2466
weighted avg       0.84      0.84      0.76      2466
```

5. Multi-Layer Perceptron (MLP)

Confusion Matrix:

The true positive, false positive, true negative, and false negative predictions are broken down in the confusion matrix. In this instance

True Positives (TP): 100

False Positives (FP): 19

True Negatives (TN): 2036

False Negatives (FN): 311

Classification Report:

Precision:

• The precision for the "False" class is 0.87, meaning that 87% of the occurrences predicted to be "False" are actually categorized correctly.

• The precision for the "True" class is 0.84, meaning that 84% of the instances that were predicted to be "True" were correctly identified.

Recall:

• For the "False" class, the recall (also known as sensitivity or true positive rate) is 0.99, meaning that 99% of the cases that belong to the "False" class are properly identified by the classifier.

• The recall (also known as sensitivity or true positive rate) for the "True" class is 0.24, meaning that 24% of the occurrences that correspond to the "True" class are captured by the classifier.

F1-Score:

Precision and recall are combined into a single metric called the F1-score. It provides a fair assessment of a classifier's performance by taking the harmonic mean of precision and recall.

• The F1-score for the "False" class is 0.93.

• The F1-score for the "True" class is 0.38.

Support:

• The support shows how many instances of each class there are in the test set.

• The support for the "False" class is 2055.

• The support number for the "True" class is 411.

Accuracy:

• The classifier's overall accuracy on the test set is 0.87, meaning that 87% of the instances are classified correctly.

Macro Avg and Weighted Avg:

• The macro average determines the unweighted mean of the measurements across all classes.

• The weighted average determines the weighted mean of the metrics by taking the support (number of instances) for each class into account.

```
[[2036   19]
 [ 311  100]]
              precision    recall  f1-score   support

       False       0.87      0.99      0.93      2055
        True       0.84      0.24      0.38       411

    accuracy                           0.87      2466
   macro avg       0.85      0.62      0.65      2466
weighted avg       0.86      0.87      0.83      2466
```

Cost-benefit analysis:

```
[111] # Cost assumptions
      average_cost_per_false_positive = 20

      # Calculate the expected savings if the best model is deployed
      num_false_positives = sum((y_test != 1) & (y_pred_rf == 1))
      expected_savings_best_model = num_false_positives * average_cost_per_false_positive

      print("Expected Savings (Best Model Deployed): $", expected_savings_best_model)
```

```
Expected Savings (Best Model Deployed): $ 1420
```

```
[110] # Cost assumptions
      average_revenue_per_sale = 100
      average_cost_per_false_negative = average_revenue_per_sale

      # Calculate the expected cost if no model is deployed
      num_false_negatives = sum((y_test == 1) & (y_pred_rf != 1))
      expected_cost_no_model = num_false_negatives * average_cost_per_false_negative

      print("Expected Cost (No Model Deployed): $", expected_cost_no_model)
```

```
Expected Cost (No Model Deployed): $ 19000
```

We must determine the cost of false negatives in order to determine the anticipated loss to the firm if no model is implemented. Let's say that each successful sale generates an average of $100 in income and that each missed sale results in an average loss of $100 in revenue.

We must determine the cost of false positives in order to determine the expected savings to the business if the best model is used. Assume that $20 is the standard cost in these situations for marketing or customer service.

The organization can decide if deploying the classifier is advantageous from a cost standpoint by contrasting the estimated cost if no model is deployed with the expected savings if the best model is deployed.

Conclusion:

Comparing various models' accuracy, precision, recall, and F1-score. When compared to the actual experiences of different buyers purchasing various products, RFC is the most accurate projected model.

The expected cost when no model is deployed is $19000, and the expected savings when the best model(RFC) is deployed are $1420.

```
Best Model:  Random Forest
```

Contribution:

Mamta Devi – Random Forest and Adaptive Boosting (AdaBoost)

Rohit Verma – Multi-Layer Perceptron (MLP) with 1 hidden layer and critical analysis report with cost-benefit analysis.

Abhinav Singh Voria - Support Vector Classification (SVC) and Logistic Regression