RMIT University Vietnam, School of Science and Technology

Title	Assessment 3: Data modeling for online shoppers purchasing intention dataset	
Author Information	Wonjun Lee	s3773920
	Nguyen Dang Nhat	s3878292
	Vo Tuong Minh	s3877562
Contact Details	Wonjun Lee	s3773920@rmit.edu.vn
	Nguyen Dang Nhat	s3878292@rmit.edu.vn
	Vo Tuong Minh	s3877562@rmit.edu.vn
Date of report	19 Jan 2024	

Table of Contents

1. Abstract

In this project, we investigate the Online Shoppers Purchasing Intention Dataset, in which online shopping demographic, behavioral, and technical data are recorded as feature vectors corresponding to browsing session on the company's website. The target class "Revenue" contains 10,422 negative samples indicating that the shopper did not end up shopping on the website, and the rest 1,908 positive samples indicating the shopper did end up shopping on the website [1]. Prepared with thorough data cleaning and feature engineering, we focused on developing clustering models and classifications model to gain key insights into the shopper's behavior on the company's website, as well as building a reliable predictor for future unseen data. For clustering, we utilized K-Means and DBSCAN to explore the dataset and found that "Revenue" is proportional to page value, but inversely proportional to bounce rate and exit rate, suggesting the importance of a good UI/UX design for maximum retention, especially for the home page of the website. Our classification models utilize Stratification, SMOTE (Synthetic Minority Oversampling Technique), k-Fold cross validation, and Grid Search hyperparameter tuning to find that the best machine learning algorithm for predicting shopper behavior is Random Forest, with consistently higher F1 score of 94% for the positive class and MMC (Matthews correlation coefficient) of 64% comparing to that of Decision Tree and Extra Trees. Our findings support the feasibility of using such data to reliably predict and improve the company's website to optimize revenue through shopper retention.

2. Introduction

2.1. Introduction of dataset

The dataset chosen for this project is Online Shoppers Purchasing Intention Dataset [1]. The dataset includes 12,330 rows of data that contains a collection of feature vectors of individual sessions on the company's website. The dataset consists of 10 numerical and 8 categorical attributes. The introduction pf each attribute is described below.

Attribute type	Attribute	Description	Notes
Numerical	Administrative	Number of different types of pages visited by visitor in that session	URL information of the pages visited by user
	Informational		
	Product Related		
	Administrative Duration	Total spent in each of these page categories	
	Informational Duration		
	Product Related Duration		
	Bounce Rate	Percentage of visitors who enter the site from that page and then leave without triggering another request	Metrics calculated by Google Analytics
	Exit Rate	Percentage that pages was last in the session	
	Page Value	Average value for web page that user visited before completing an e-commerce transaction	
Categorical	Special Day	Closeness of site visiting time to specific day	Value as 0, 0.2, 0.4, 0.6, 0.8, and 1.0
	operating system	Information of software/hardware information of users	
	browser		
	region		
	traffic type		
	visitor type	New/Returning Visitor, Others	
	weekend	Is day weekend or not	
	month	Month of year	
Label	Revenue	Positive: Ended with shopping Negative: Not ended with shopping	Imbalance dataset where 84.5% is negative and rest is positive

Table 1. Information of attributes of dataset.

As shown in the table above, the label column is "Revenue", which shows if that user is useful for revenue of the company.

2.2. Goal of project

The primary object of this project is to examine dataset to get insights to boost the revenue of the company. By viewing dataset of online data collected from the website to find relationships, patterns and correlations associated with users who are potential buyers.

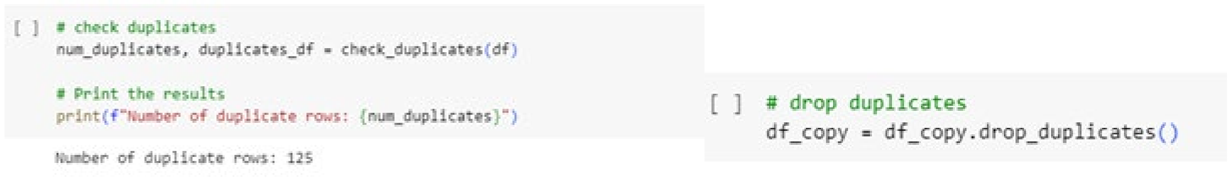
For this goal, our project will use analytical techniques of machine learning, which are clustering and classification. Clustering uses unlabeled data to find structure or intrinsic grouping which are groups of data similar in some way. Although the dataset has label column, it will be dropped for the model. Classification uses label data split into training, evaluation, and testing data for the model that can predict label of unseen instances [2].

Based on these machine learning models, it is aimed to empower the company with actionable insights that can lead to building business strategies that can improve the website and result in increasing revenue of the company.

3. Methodology

3.1. Feature Engineering

The feature engineering will be done cleaning and processing data commonly needed for clustering and classification models. For the first step of feature engineering, duplicates will be removed because they cause inaccuracy of machine learning models. By inspection, there are 125 rows of duplicates found.

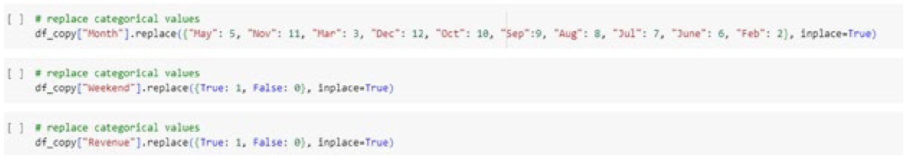


```
[ ] # check duplicates
    num_duplicates, duplicates_df = check_duplicates(df)

    # Print the results
    print(f"Number of duplicate rows: {num_duplicates}")

    Number of duplicate rows: 125
```

```
[ ] # drop duplicates
    df_copy = df_copy.drop_duplicates()
```

Figure 1: Inspecting & Removing duplicates.

Next, changing values in categorical columns and replacing Boolean values to integers will be done.



```
[ ] # replace categorical values
    df_copy["Month"].replace({"May": 5, "Nov": 11, "Mar": 3, "Dec": 12, "Oct": 10, "Sep":9, "Aug": 8, "Jul": 7, "June": 6, "Feb": 2}, inplace=True)
```

```
[ ] # replace categorical values
    df_copy["Weekend"].replace({True: 1, False: 0}, inplace=True)
```

```
[ ] # replace categorical values
    df_copy["Revenue"].replace({True: 1, False: 0}, inplace=True)
```

Figure 2: Replacing values.

For the next step, outliers will be removed with the IQR method. It is effective because it considers the natural spread of the data, not just a single threshold. It's robust to outliers and works well for non-normal distributions, making it a versatile tool for data cleaning. However, eliminating all outliers is dangerous because it may lead to data loss with too many rows deleted. Therefore, only the outliers for “Administrative”, “Region”, and “OperatingSystems” are removed.

For processing categorical features, one-hot encoding was used. One-hot encoding is a technique used in machine learning and data analysis to represent categorical variables as binary vectors. It is a way to convert categorical data into a numerical format that can be used by machine learning algorithms. In one-hot encoding, each category or class in a categorical variable is transformed into a binary vector of zeros and ones. The length of the binary vector is equal to the number of unique categories in the variable.

3.2. Clustering

3.2.1. Goal of model:

Clustering is one of the unsupervised learning approaches. Unlike supervised learning approach as classification, unsupervised learning models are usually used for discovering patterns, finding structure, and grouping data. Since it doesn’t have label data, it cannot make predictions. Therefore, clustering will be used to find patterns in dataset and see what type of users having the higher likelihood of generating revenue for the company.

3.2.2. Type of model selected:

For clustering approach, two models will be selected. First is K-Means, and second is DBSCAN, each from scikit-learn.

K-Means model aims to partition observations into k clusters which each observation belongs to the clusters with the nearest mean based on Euclidean distance. K-Means has advantages of being fast, robust, and easy to understand while being efficient. It also gives the best result when datasets are distinct or well separated. However, it has disadvantages when data are highly overlapping. Also, the performance is sensitive to the form of transformations applied to unput data. It is only applicable when the mean is defined as numeric data. It is also sensitive to outliers [2].

DBSCAN model is density-based clustering. It recognized outlier points on low density regions as noise and excluded them from the data. It only has 2 parameters, eps and MinPts, where eps mean maximum radius of the neighborhood, and MinPts mean minimum number of points required in the neighborhood. It has the advantages of not requiring number of clusters and being robust to outliers but has some disadvantages of being problematic in high dimensional data and cannot cluster dataset well with large difference in densities.

3.2.3. Pre-processing before model training

Although feature engineering was done before, extra data cleaning as scaling will be done. Clustering models that use distance measures between data points are sensitive to range of data of each dimension because features with larger scales may create biased results. Therefore, scaling is needed to put all data in standard scale to prevent specific column overshadowing other features. There are various scaling techniques available such as MinMax scaling, Robust scaling, standard scaling, and power transformation. Among those techniques, MinMax scaling was chosen because it showed the best result in clustering to find patterns of users.

3.2.4. Model training:

3.2.4.1. Training K-means clustering model:

For K-means model, only numeric columns are used since k-means clustering uses distance measures, and only columns with information that distance can be measured as numeric values can be used. 2 models of K-means model will be tested to view the dataset: K-means clustering and K-means clustering with PCA. Inertia for each model of k clusters of k being range of 1 to 10 will be recorded, and the info will be used to find the best value of cluster use for the actual model.

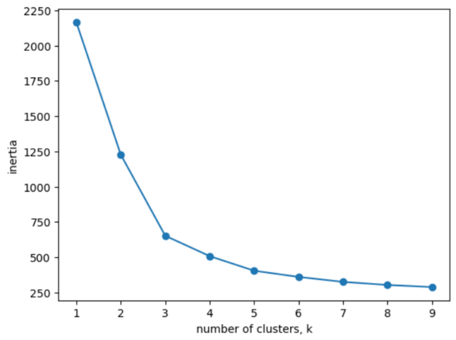


Figure 3: Graph of inertia value for each number of clusters.

Additionally, n_init and max_iter will be tuned showing the best clustering of data. The parameter of n_init and max_iter is parameter that decides number of time algorithms runs, n_init being running algorithm to find result with lowest inertia and max_iter being maximum number of iterations algorithm will perform on single run. These parameters increase quality of clustering but increase computational efficiency since algorithms run more. The value of these parameters is chosen based on computing power and result of clustering. PCA is used for reducing the dimensions since it is known as K-means works better when there are less dimensions.

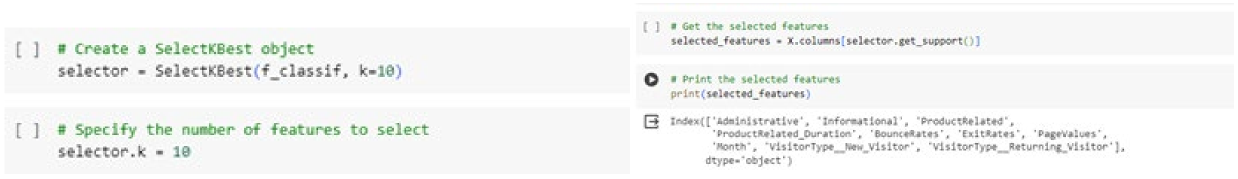
3.2.4.2. Training DBSCAN clustering model:

DBSCAN clustering model is easier to train than K-means since most of the important attributes are auto-selected by the model, in addition to it being less sensitive than K-means model. It can also use categorical column unlike K-means clustering. Attributes of eps and min_samples are tuned for this model,(explanation of attributions). The prediction is done with values of -1 to maximum number of clusters, where -1 is considered noise of the result.

3.3. Classification

3.3.1. Feature extraction:

Next is feature extraction. Feature selection is an essential step in machine learning to identify the most relevant features from a given dataset. One commonly used technique for feature selection is KBest, which selects K number of features based on their statistical scores. KBest evaluates the relevance of each feature independently and assigns a score to measure its significance in predicting the target variable. The KBest algorithm then selects the K features with the highest scores as the most informative and relevant features for the model. By using KBest feature selection, the dimensionality of the dataset can be reduced, removing irrelevant or redundant features.



```
[ ] # Create a SelectKBest object
    selector = SelectKBest(f_classif, k=10)

[ ] # Specify the number of features to select
    selector.k = 10
```

```
[ ] # Get the selected features
    selected_features = X.columns[selector.get_support()]

    # Print the selected features
    print(selected_features)

    Index(['Administrative', 'Informational', 'ProductRelated',
           'ProductRelated_Duration', 'BounceRates', 'ExitRates', 'PageValues',
           'Month', 'VisitorType__New_Visitor', 'VisitorType__Returning_Visitor'],
          dtype='object')
```

Figure 4: Selecting KBest object (left) and features selected by Kbest (right).

3.3.2. Model Selection:

For model training, we would use tree-based models (such as decision trees, random forests, extra trees and gradient boosting machines) rather than linear models (Support Vector Machine, Logistic Regression) for several reasons:

- **Non-linearity:** Tree-based models can capture non-linear relationships between features and the target variable. Unlike linear models, which assume a linear relationship, tree-based models recursively split the data based on feature thresholds, enabling them to capture complex patterns and interactions among features.
- **Handling Mixed Data Types:** Tree-based models can handle both numerical and categorical features without requiring extensive pre-processing. They automatically handle categorical features by splitting them into different branches based on their categories.
- **Robustness to Outliers:** Tree-based models are robust to outliers and noisy data. Outliers have minimal impact on the model's performance since the splitting criterion is based on relative values within each node. This robustness makes tree-based models suitable for datasets that contain noisy or inconsistent data.
- **Handling Missing Values:** Tree-based models can handle missing values without requiring imputation.

Another reason why we choose tree-based models is that linear models are sensitive to outliers due to their reliance on mean and variance. Skewed data often contains outliers that can distort these measures, which can significantly influence the model's decision boundary, leading to suboptimal performance.

We can handle skewness by using Log transformation. Log transformation is a versatile and valuable tool for dealing with skewed data in statistical analysis. It works by applying the logarithm (usually natural logarithm, base e) to each data point in the skewed feature. It reduces the skewness of a distribution, making it more symmetrical and closer to a normal distribution and improves the performance of linear models that are sensitive to skewness. But when attempting to take the logarithm of zero, the algorithm encounters an undefined value, resulting in errors or NaN (Not a Number) values, potentially crashing the machine learning algorithm.

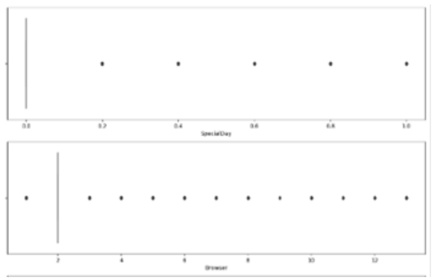


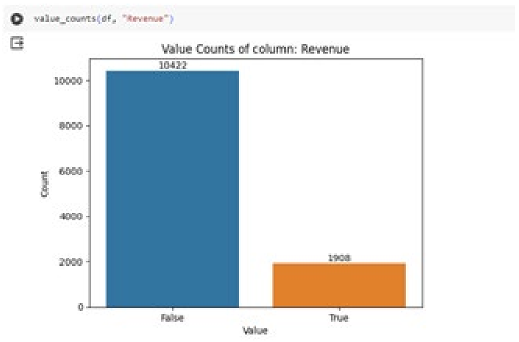
Figure 5: Outliers inspection of column ‘specialDay’ & ‘Browser’.

Model selected for training:

- Random Forest Classification with Hyperparameter Tunning, Stratification and Oversampling
- Decision Tree Classification with Hyperparameter Tunning, Stratification and Oversampling
- Extra Tree Classification with Hyperparameter Tunning, Stratification and Oversampling

3.3.3. Model Training Technique:
For all 3 of the above models, the same training technique is applied. Following is the explanation for technique used for training:

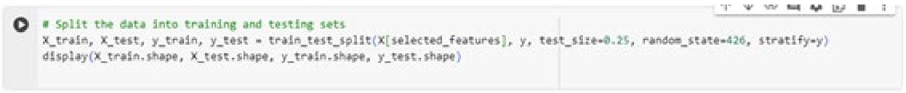
Stratification: Stratification is a technique commonly used when splitting a dataset into a training set and a test set. It ensures that the class distribution of the target variable remains consistent across both sets. Additionally, it preserves the proportion of different classes within the dataset when creating the train-test split. This is particularly important when dealing with imbalanced datasets. It also helps prevent bias in the model evaluation process. Without stratification, there is a risk of having a highly imbalanced distribution of classes in either the training or test set.



```
value_counts(df, "Revenue")
```

Figure 6: Inspect target variable.

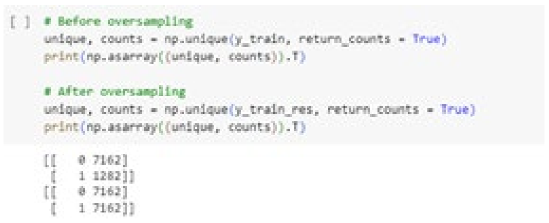
The figure above shows that the “False” value is about 5 times more than the “True” value, which causes significant imbalance in the dataset. If we don’t apply stratification, there is a chance that there will be no “True” value in the test set.



```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X[selected_features], y, test_size=0.25, random_state=426, stratify=y)
display(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

Figure 7: Applying stratification.

SMOTE (Synthetic Minority Oversampling Technique): a popular technique used to address class imbalance in machine learning datasets. SMOTE is specifically designed to oversample the minority class by generating synthetic samples. It works by analyzing the feature space of the minority class and creating synthetic examples by interpolating between existing minority class instances. It selects a minority class instance and identifies its “k” nearest neighbors. It then randomly selects one or more of these neighbors and creates synthetic samples along the line segments.



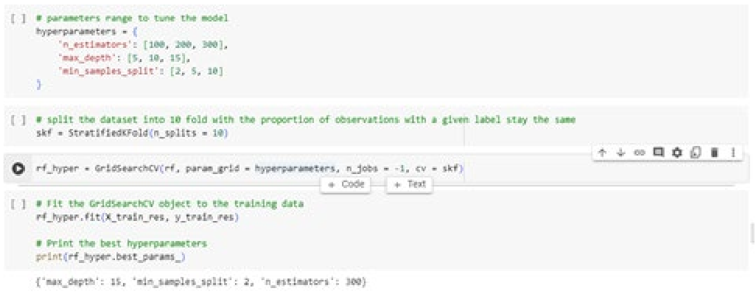
```
# Before oversampling
unique, counts = np.unique(y_train, return_counts = True)
print(np.asarray((unique, counts)).T)

# After oversampling
unique, counts = np.unique(y_train_res, return_counts = True)
print(np.asarray((unique, counts)).T)
```

```
[[   0 7162]
 [   1 1282]]
[[   0 7162]
 [   1 7162]]
```

Figure 8: Shape of train and test set before and after applying SMOTE.

Hyperparameters Tunning with GridSearchCV: Hyperparameter tuning is an important step in optimizing the performance of machine learning models while GridSearchCV is a technique provided by scikit-learn to search through a list of hyperparameters and identify the ones that gives the best model performance using cross-validation. It splits the data into multiple subsets, trains the model on one subset, and evaluates the model performance on the rest. This helps us to obtain a more accurate estimate of the model's performance and reduces the risk of overfitting.

StratifiedKFold: StratifiedKFold is a cross-validation technique that addresses the challenge of class imbalance in machine learning datasets. It is an extension of the K-Fold cross-validation method. In traditional K-Fold cross-validation, the data is randomly divided into K folds without considering the class distribution, which can lead to some folds having a significantly different class distribution than the original dataset. StratifiedKFold ensures that each fold has a proportional representation of each class, maintaining the same proportion of class instances as the original dataset, which is a more reliable evaluation of the model's performance across all classes.



```
# parameters range to tune the model
hyperparameters = {
    'n_estimators': [100, 200, 300],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5, 10]
}
```

```
# split the dataset into 10 fold with the proportion of observations with a given label stay the same
skf = StratifiedKFold(n_splits = 10)
```

```
rf_hyper = GridSearchCV(rf, param_grid = hyperparameters, n_jobs = -1, cv = skf)
```

```
# Fit the GridSearchCV object to the training data
rf_hyper.fit(X_train_res, y_train_res)

# Print the best hyperparameters
print(rf_hyper.best_params_)
```

```
{'max_depth': 15, 'min_samples_split': 2, 'n_estimators': 300}
```

Figure 9: Applying StratifiedKFold and GridSearchCV in hyperparametes tuning.

4. Results

4.1. Clustering

4.1.1. K-Mean model

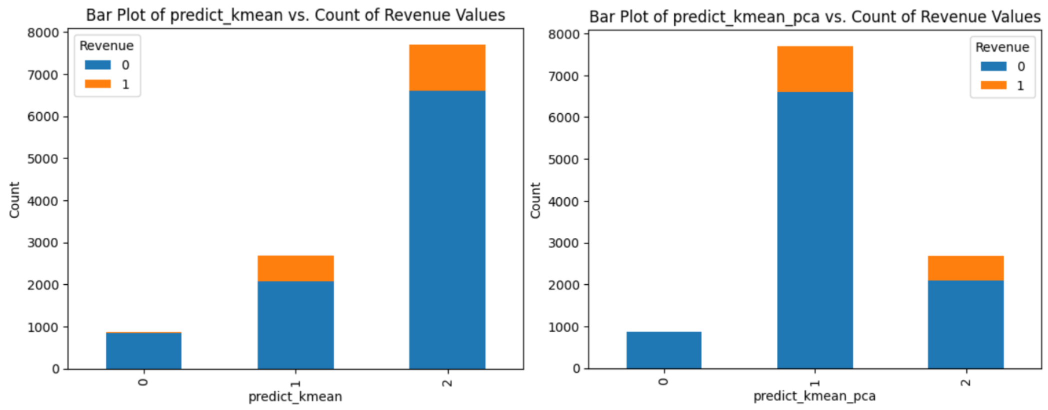


Figure 10: Bar plot of clustering of k-mean model (left) and k-mean model with PCA (right).

By viewing bar plot of predicted values and count of revenue for each cluster, K-mean model was able to cluster group of users mostly with 0 value of revenue (Cluster 0). For the rest of clusters, one has more data, but percentage of revenue is lower, and other one has less data, but percentage of revenue is higher. Therefore, this result can be viewed to see feature of users with low profit compared to normal users.

Surprisingly, K-mean model with PCA shows similar result to K-mean model, with some changes on order of clusters. This may be caused by various reasons such as the dimension of dataset not that high, the model working well, or dimension reduction doesn't attribute much in clustering quality. Since two models show similar clustering, only result for K_means model will be viewed for analysis.

Before diving into analysis, some definition of key columns will be defined below again:

Name	Formula	Description
Bounce Rate	$(\frac{Number\ of\ Single\text{-}Page\ Sessions\ on\ the\ page}{Number\ of\ Entrance\ to\ the\ Page})$ x 100	Percentage of visitors who enter the site from that page and then leave without triggering another request
Exit Rate	$(\frac{Number\ of\ Exits\ from\ the\ page}{Number\ of\ Pageviews\ for\ the\ Page})$ x 100	Percentage that a page was last in the session
Page Value	$\frac{Total\ Revenue}{Number\ of\ Unique\ Pageviews}$	Average value for web page that user visited before completing an e-commerce transaction

Table 2: Columns for analysis.

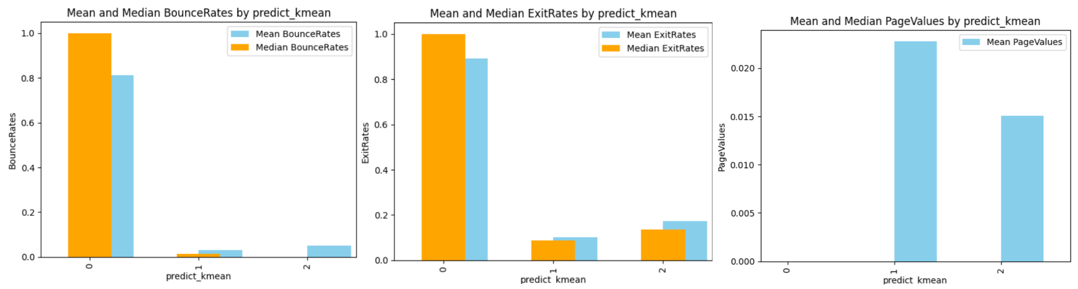


Figure 11: Bar chart of mean and median values of BounceRates(left), ExitRates(center), and PageValues(right) of K-means model.

For K-means model, there can be comparison made between Cluster 0 and other clusters. As shown on bar charts above, bounce rate and exit rate of Cluster 0 is way higher than other clusters. Also, the mean of page values is close to 0. This means users that didn't make any purchase have a higher chance of leaving the website after the first visit and exits on specific pages with low page values even after some surfing.

4.1.2. DBSCAN model

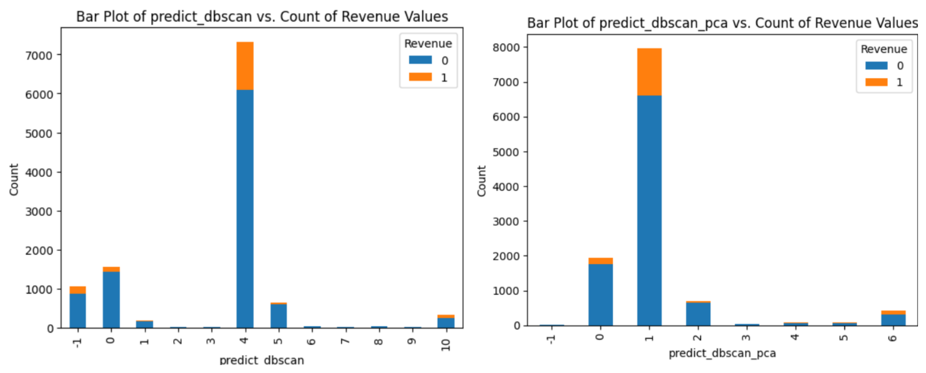


Figure 12: Bar chart of revenue count for each cluster for DBSCAN model(left) and DBSCAN model with PCA(right).

The bar plot of DBSCAN model shows data clustering one specific cluster and rest of dataset distributed on other clusters. The bar plot of DBSCAN model with PCA (DBSCAN-PCA) shows similar tendency as DBSCAN model except there is almost no noise for DBSCAN-PCA.

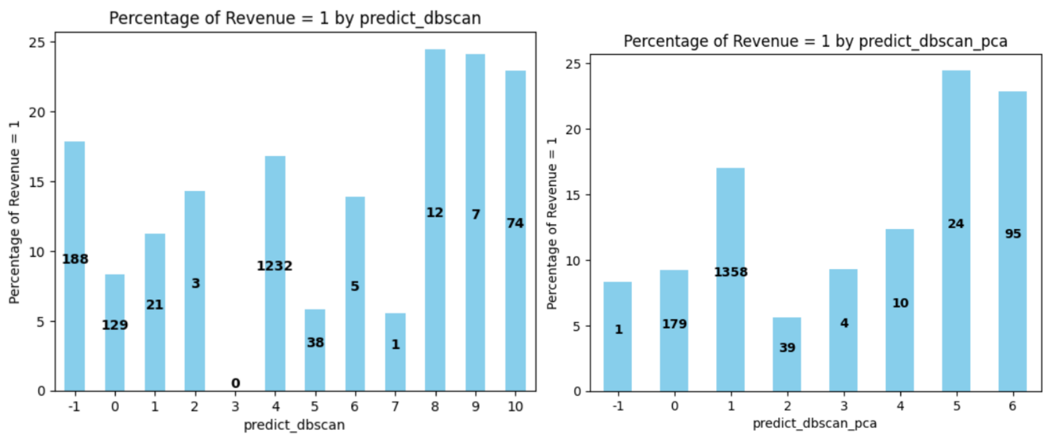


Figure 13: Bar chart of percent and count of positive revenue values for each cluster for DBSCAN model (left) and DBSCAN model with PCA (right).

By viewing percentage and count of positive revenue values, plan of viewing bar charts can be set. For DBSCAN model, cluster 5 can be used as representative of users not likely to be positive in revenue, and cluster 10 as representative of users likely to be positive in revenue. For DBSCAN-PCA model, cluster 2 can be representative of users not likely to be positive in revenue, and cluster 6 as representative of users likely to be positive in revenue. It is hard to define those clusters as representatives, but the difference between those columns may give insight of difference between users being positive in revenue and not.

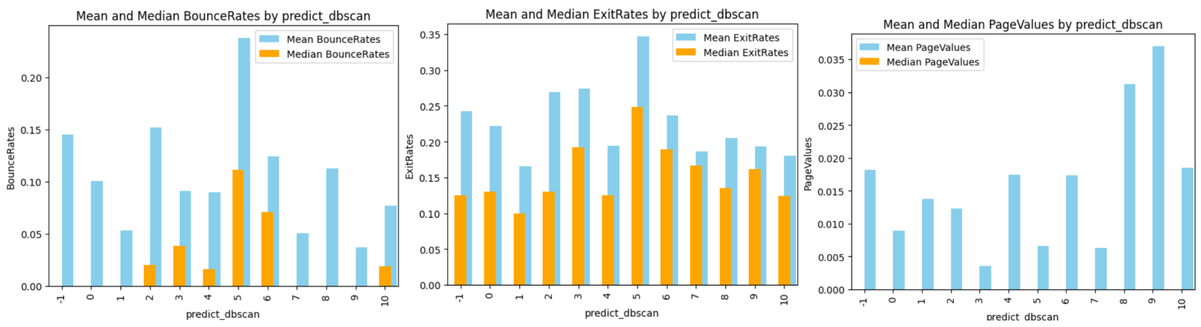


Figure 14: Bar chart of mean and median values of BounceRates(left), ExitRates(center), and PageValues(right) of DBSCAN model.

As discussed above, cluster 5 and 10 will be compared mostly to view tendencies of DBSCAN model. For bounce rates, cluster 5 has the highest bounce rates, and cluster 10 has lower bounce rates. For exit rates, cluster 5 also shows the highest rate while cluster 10 shows lower bounce rates. For page values, cluster 5 shows lowest page values while cluster 10 shows higher value. Considering cluster 8 and 9 also having a high percentage of positive value in revenue, it can be said higher page value can lead to higher percentage of positive value in revenue.

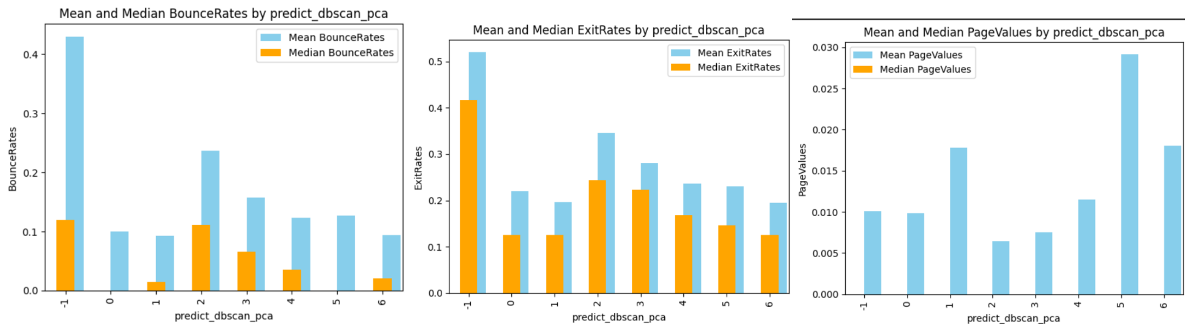


Figure 15: Bar chart of mean and median values of BounceRates (left), ExitRates (center), and PageValues (right) of DBSCAM model with PCA.

For DBSCAN-PCA model, cluster 2 and 6 will be determined mainly for analysis. Generally, tendencies are similar to that of the DBSCAN model. Based on these results, it can be said that lower bounce rate and exit rates and higher page values increase the probability of users having positive value in revenue.

4.2. Classification

The metrics used to evaluate classification models are as below:

Name	**Calculation method**	**Description**
Accuracy (acc)	$\frac{TP+TN}{TP+FP+TN+FN}$	Ratio of correct predictions over the total number of instances evaluated [3].
Precision (p)	$\frac{TP}{TP+FP}$	Measure of positive patterns that are correctly predicted from the total predicted patterns in a positive class [3].
Recall (r)	$\frac{TP}{TP+FN}$	Measure of fraction of positive patterns that are correctly classified [3].
F1 Score (F_1)	$\frac{2 \times p \times r}{p+r}$	Harmonic mean between recall and precision [3].
Matthews correlation coefficient (MMC)	$\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$	An alternative measure unaffected by the unbalanced datasets issue. MCC generates a high score only if the binary predictor can correctly predict the majority of

		both positive data instances as well as negative data instances. It ranges in the interval [−1,+1], where −1 is perfect misclassification and +1 is perfect classification, while MCC = 0 is the expected value for the random "coin tossing" classifier [4].
AUC-ROC curve	$\int_0^1 TPR\ d(FPR)$	Graphical representation of effectiveness of the binary classification model that plots true positive rate ($TPR = \frac{TP}{TP+FN}$) vs the false positive rate ($FPR = \frac{FP}{FP+TN}$) at different classification thresholds [5].

Table 3: Evaluation metrics for classification model.

4.2.1. Model 1: Random Forest Classification with Hyperparameter Tunning, Stratification and Oversampling

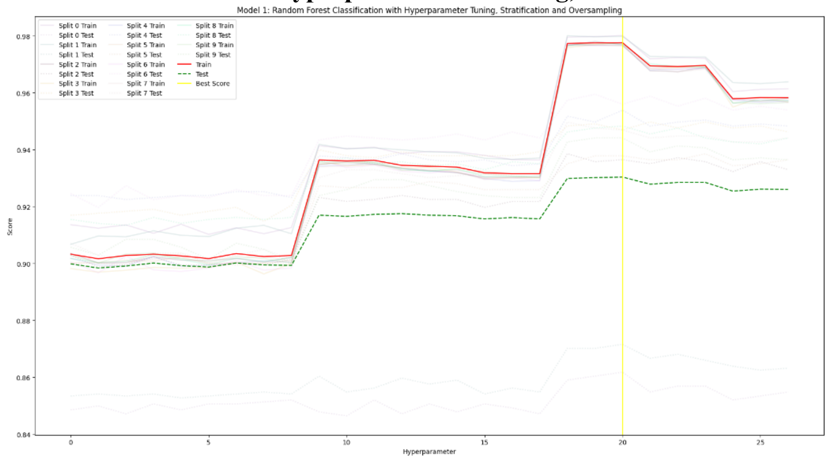
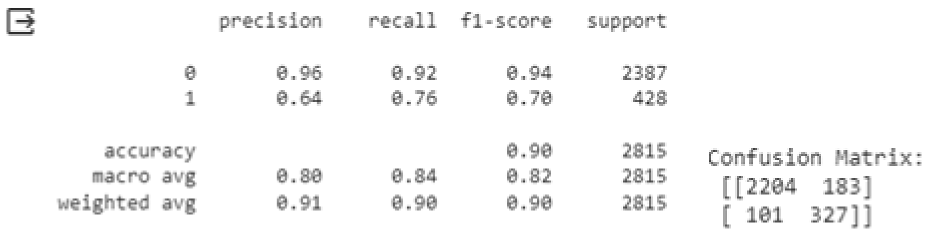


Figure 16: Random Forest Classification training result.



```
              precision    recall  f1-score   support

           0       0.96      0.92      0.94      2387
           1       0.64      0.76      0.70       428

    accuracy                           0.90      2815      Confusion Matrix:
   macro avg       0.80      0.84      0.82      2815       [[2204  183]
weighted avg       0.91      0.90      0.90      2815       [ 101  327]]
```

Figure 17: Random Forest Classification result (left) and confusion matrix (right).

For model 1, the results show that the model has achieved a high accuracy of 90% and MMC score of 64%; with a precision of 96% (positive class) and 64% (negative class), recall of 92% (positive class) and 76% (negative class), and F1-score of 94% (positive class) and 70% (negative class).These results suggest the model can make accurate predictions for both classes and is not biased towards either.

This model correctly classified 2531 out of 2815 instances, resulting in an accuracy of approximately 89.9%.

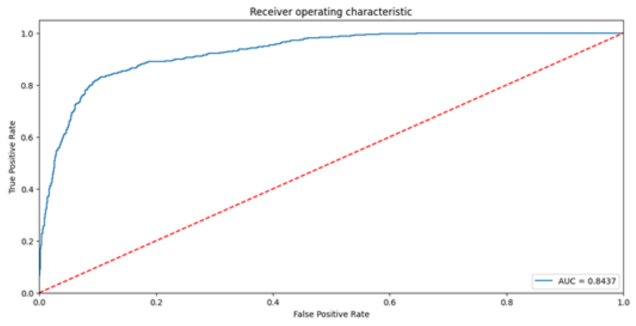


Figure 18: Random Forest ROC-AUC.

The AUC value of 0.8437 indicates that this model is performing well at discriminating between the positive and negative classes.

However, we must note that there is evidence that this model shows slight, but insignificant overfitting (Figure 16).

4.2.2. Model 2: Decision Tree Classification with Hyperparameter Tunning, Stratification and Oversampling

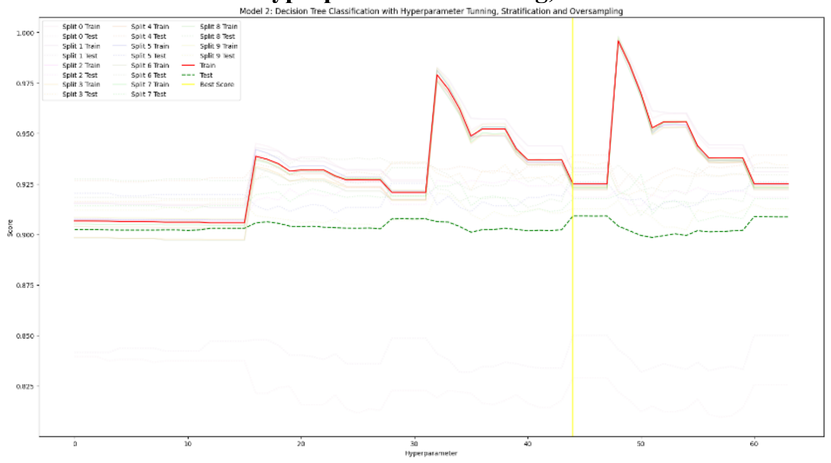
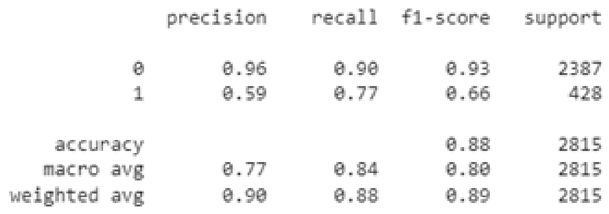
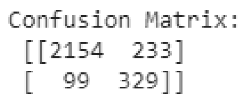


Figure 19: Decision Tree Classification training result.



```
              precision    recall  f1-score   support

           0       0.96      0.90      0.93      2387
           1       0.59      0.77      0.66       428

    accuracy                           0.88      2815
   macro avg       0.77      0.84      0.80      2815
weighted avg       0.90      0.88      0.89      2815
```



```
Confusion Matrix:
 [[2154  233]
 [  99  329]]
```

Figure 20: Decision Tree Classification result (left) and Confusion Matrix (right).

This model achieves a promising accuracy of 88% and MMC score of 60%; with a precision of 96% (positive class) and 59% (negative class), recall of 90% (positive class) and 77% (negative class), and F1-score of 93% (positive class) and 66% (negative class); demonstrating its ability to distinguish between the two classes. It performs well with 2453 correct classification out of 2815.

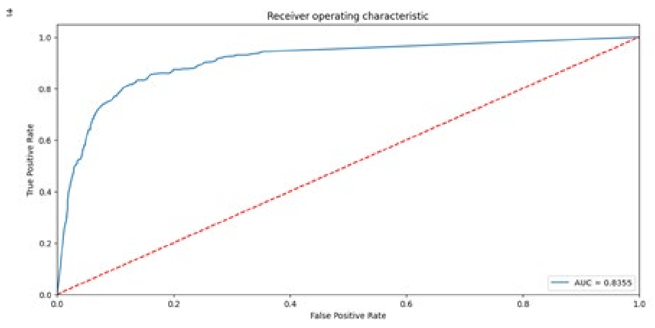


Figure 21: Decision Tree ROC-AUC.

The AUC value of 0.83555 indicates that this model is performing well at classifying between classes.

The training result (Figure 19) also shows that this model would generalize well, without overfitting nor underfitting.

4.2.3. Model 3: Extra Tree Classification with Hyperparameter Tunning, Stratification and Oversampling

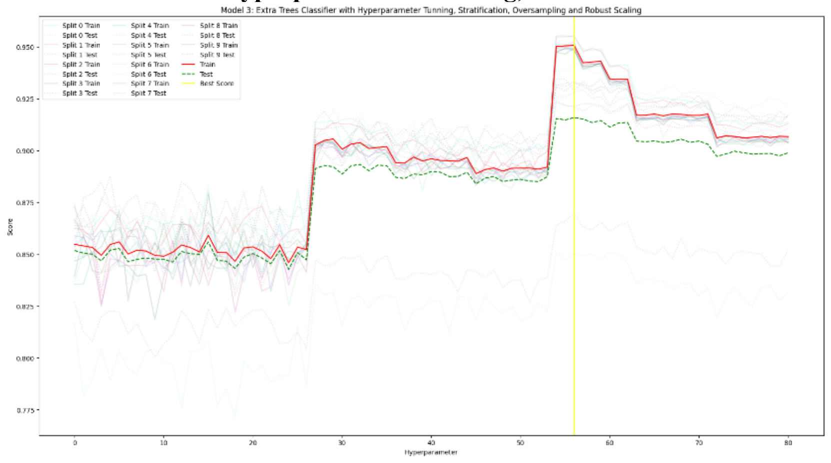
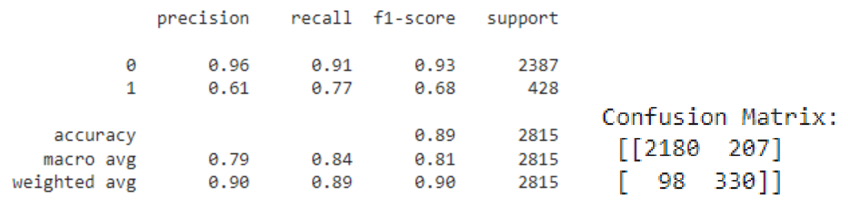


Figure 22: Extra Tree Classification training result.



```
              precision    recall  f1-score   support

           0       0.96      0.91      0.93      2387
           1       0.61      0.77      0.68       428

    accuracy                           0.89      2815
   macro avg       0.79      0.84      0.81      2815
weighted avg       0.90      0.89      0.90      2815
```

```
Confusion Matrix:
 [[2180  207]
 [  98  330]]
```

Figure 23: Extra Tree Classification result (left) and Confusion Matrix (right).

This model achieves an accuracy of 89% and MMC score of 63%; with a precision of 96% (positive class) and 61% (negative class), recall of 91% (positive class) and 77% (negative class), and F1-score of 93% (positive class) and 68% (negative class).

This model has a good performance of 89% in accuracy, with 2510 correct classification out of 2815.

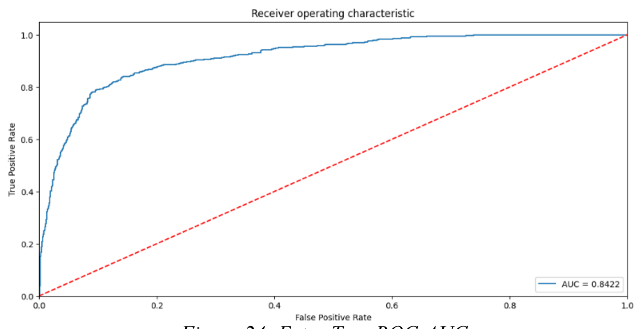


Figure 24: Extra Tree ROC-AUC.

The AUC value of 0.8422 indicates that the model is performing well.

Like model 2, this model is also showing promising results of not underfitting nor overfitting (Figure 22).

5. Discussion

5.1. Clustering

The purpose of clustering was finding patterns and tendencies of data to gain insight of data and solution, which is increasing revenue of the company. Therefore, there cannot be any type of evaluation of result of accuracy since the model isn't built to cluster based on revenue of user.

By examining the clustering result, it can be said revenue is related to bounce rate, exit rate, and page value. Clusters of high percentage of users with positive revenue values tended to have lower bounce rate and exit rate with higher page value compared to other clusters. Therefore, the company should focus on upgrading website by creating attractive home page to lower bounce rate and update pages with low page values as improving user interface and user experience to reduce exit rate and increase page value to increase duration of users and ultimately increasing users with positive value on revenue.

5.2. Classification

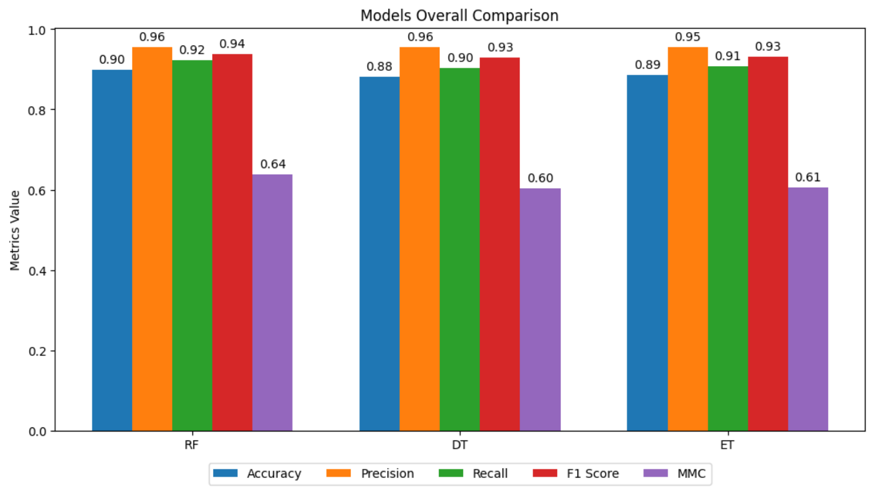


Figure 25: Overall comparison between model 1 (RF), model 2 (DT) and model 3 (ET). Note that for precision, recall, and F1, we are focusing on the metric for positive classification.

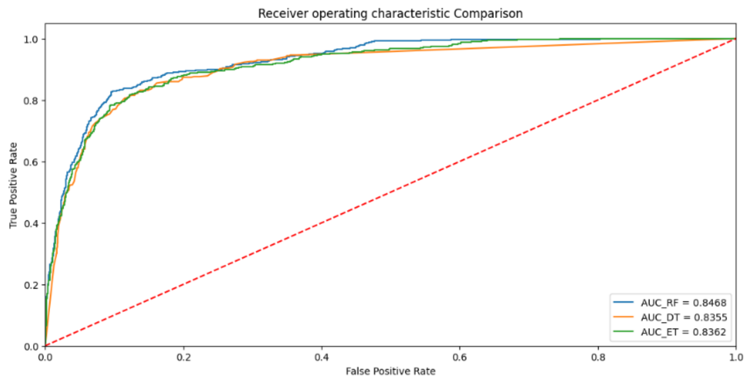


Figure 26: ROC-AUC comparison between model 1 (AUC_RF), model 2 (AUC_DT) and model 3 (AUC_ET).

Our classification effort showed promising results for all 3 machine learning algorithms, in which Model 1 - Random Forest Classification with Hyperparameter Tunning, Stratification and Oversampling showing the best results comparing to that of the other two models. Its performance metrics are the highest (Figure 25) in all metrics and its ROC-AUC curve is the closest to a perfect classifier (Figure 26), while its training result also showed that this model would generalize well to unseen data, as it is not overfitting nor underfitting (Figure 16). As such, Model 1 is our final model of choice for the task of classification.

Our confidence in this result is based on our robust choice of evaluation metrics, most importantly the F1 and MMC score, along with ROC-AUC. Despite the imbalance in our dataset, F1 and MMC, especially MMC, are unsusceptible to this, and are good indicators of our model's generalization performance.

6. Conclusion

After thorough data cleaning and feature engineering, and based on our machine learning methodology, our project has discovered the following actionable insights via our clustering models:

- Pages with low page values should be found and be upgraded to improve user retention.
- Lower bounce rate and exit rates along with higher page values increases the probability of users having positive value in revenue.
- The home page's attractiveness plays a key role in increasing revenue.

For positive/negative revenue prediction, we have concluded that our binary classifier algorithm of Random Forest Classification with Hyperparameter Tunning, Stratification and Oversampling (Model 1 – Section 4.2.1) is a robust and strong candidate for commercial use by the company, with a high accuracy of 90% and MMC score of 64%; with precision of 96% (positive class) and 64% (negative class), recall of 92% (positive class) and 76% (negative class), and F1-score of 94% (positive class) and 70% (negative class).

This finding supports the feasibility of using online user shopping demographic, behavioral, and technical data to predict and improve shopper retention.

References

[1] C. Sakar and Y. Kastro. *Online Shoppers Purchasing Intention Dataset*, doi: https://doi.org/10.24432/C5F88Q.
[2] T. Nguyen, "COSC2789 Practical Data Science - Week 6 Classification," RMIT University Vietnam, Lecture Note, 2023.
[3] T. Nguyen, "COSC2789 Practical Data Science - Week 8 Clustering," RMIT University Vietnam, Lecture Note, 2023.
[4] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics,* vol. 21, no. 1, p. 6, 2020/01/02 2020, doi: 10.1186/s12864-019-6413-7.
[5] R. Shamar. "AUC ROC Curve in Machine Learning." GeeksforGeeks. https://www.geeksforgeeks.org/auc-roc-curve/ (accessed Jan. 10, 2024).

Appendix A. Member Contribution

Member name	**Contribution score (100% total)**	**Role**
Wonjun Lee	35%	Development of clustering model
Nguyen Dang Nhat	35%	Feature engineering, Development of classification model
Vo Tuong Minh	30%	Model performance and results analysis