**UCCD2063 Artificial Intelligence Techniques Group Assignment**

Classification Prediction for In-Vehicle Radio Advertising Effectiveness

|  |  |  |  |
| --- | --- | --- | --- |
| Student Name: | Ha Lik Quan | Sim Rui En | Tan Kiong Peng |
| Student ID: | 2105296 | 2104630 | 2004059 |
| Contribution: | 34% | 34% | 32% |
| Signature: | A blue signature on a white background  Description automatically generated | SIM |  |

**1. Introduction**

**Background**

In the ever-evolving area of marketing and advertising, the efficacy of promotional strategies has become a focal point of interest. In today's fast-paced world, advertisers are constantly seeking innovative ways to capture the attention of their target audience. One of the examples is in-vehicle radio advertising where drivers may hear promotions from local stores while commuting.

The dataset provided offers a unique opportunity to dig into this matter. It comprises diverse information such as driver demographic information, driving conditions, preferences, and store promotion details. These variables collectively offer a rich source of data to explore and analyze to predict whether a car driver will stop by a local store when exposed to a radio advertisement while driving.

The predictive modeling techniques will be applied to predict the result such as data exploration, visualization, preprocessing, model training, validation, tuning, and testing. Firstly, the dataset will be described to clearly understand the dataset's structure and composition which helps in choosing appropriate machine learning models.

Moreover, the distribution of data, correlations between features, and any potential class imbalances in the target variable are gained through data visualization. This exploratory phase guides feature selection and informs decisions regarding model complexity.

Furthermore, data preprocessing is essential to ensure data quality and model performance as it involves handling missing data, encoding categorical variables, and scaling numerical features. Pre-processing decisions are justified based on the nature of the dataset and the potential impact on model stability and accuracy.

Besides that, the model training and validation occurred to select suitable machine learning algorithms that can effectively capture the underlying patterns in the data. The validation procedures included such as cross-validation to help assess model generalization and prevent overfitting or underfitting.

Lastly, the model tuning and testing aim to optimize model performance. Hyperparameter tuning ensures that the chosen models are operating at their best, and final testing provides an unbiased assessment of predictive accuracy.

**Objective**

The primary objective of this project is to implement machine learning techniques to develop predictive models that can accurately determine whether a driver will make a stop at a local store after encountering a promotional advertisement on the car radio.

In pursuit of this objective, we will undertake the following key steps:

* Thoroughly explore the dataset, addressing complexities and handling missing data through preprocessing.
* Experiment with various classification algorithms with examinate the impact of hyperparameter settings, training strategies, and regularization techniques on model performance.
* Investigate potential underfitting or overfitting issues and assess the significance of feature selection.
* Apply rigorous validation and testing procedures to validate the reliability of our predictive models.

**2. Methods**

**Dataset description:**

The 'dataset.csv' dataset is composed of 21 features that encompass diverse aspects, including driver's demographic information, driving conditions, preferences, and store promotion details, among others. Additionally, the dataset includes one prediction output variable. Notably, the dataset exhibits some complexity due to the presence of missing data. It comprises a total of 12,372 rows and 22 columns, making it a valuable resource for conducting analyses and modelling tasks in various domains. The columns are defined as follows: 'gender,' 'age,' 'maritalStatus,' 'children,' 'education,' 'income,' 'destination,' 'passenger,' 'weather,' 'temperature,' 'time,' 'visitPub,' 'visitCoffeeHouse,' 'visitFastFood,' 'visitFoodCourt,' 'visitRestaurant,' 'storeType,' 'offerEndSoon,' 'moreThan5km,' 'moreThan15km,' 'sameDirection,' and 'stopBy.' Of these, 'stopBy' takes the spotlight as the prediction output variable. Among these columns, 14 of them is the object data type, 3 of them are numerical columns (age, weather, time), and 5 binary columns (children, moreThan5km, moreThan15km, sameDirection, and stopBy).

There are columns with null values. For instance,the columns 'visitPub,' 'visitCoffeeHouse,' 'visitFastFood,' 'visitFoodCourt,' and 'visitRestaurant' have 105, 214, 148, 128, and 186 null rows, respectively. Based on data observations, these five columns have only five distinct value ranges: 'never,' 'rarely,' 'sometimes,' 'often,' and 'very often.

A computer screen shot of a computer code

Description automatically generated with medium confidence

Figure 1 Count of null value in the column

**Data exploration and visualization:**

In the 'age' column, ages are categorized into eight distinct groups, ranging from 18 to 55 years old. Specifically, the age groups are as follows: 18 years old, 21 years old, 26 years old, 31 years old, 36 years old, 41 years old, 46 years old, and 55 years old.

A pie chart with different colored circles

Description automatically generated

**Figure 2** Age pie chart

In the "maritalStatus" column, the dataset classifies individuals into five distinct marital status categories. Analyzing the data, we observe that the most prevalent marital status is "Married," with 4,976 occurrences, followed by "Single" with 4,641 occurrences. "Partner" is the third most common category, appearing 2,126 times. Meanwhile, "Divorced" and "Widowed" are less frequent, with 500 and 129 occurrences, respectively.

A pie chart on a computer screen

Description automatically generated

**Figure 3** Marital status pie chart

The 'education' column categorizes individuals' educational backgrounds. The most prevalent category is 'College,' with 4,253 instances, closely followed by 'Bachelors degree,' which accounts for 4,227 individuals. 'Graduate degree' is the third most common category, representing 1,797 individuals. 'Associates degree' is chosen by 1,119 individuals, while 'High School Graduate' and 'High School' are selected by 888 and 88 individuals, respectively. This distribution suggests that a substantial portion of the dataset consists of individuals with higher education, particularly those with college and bachelor's degrees.

A pie chart with different colored circles

Description automatically generated

**Figure 4** Education pie chart

The 'income' column in the dataset captures the income levels of individuals, and it reveals a diverse range of income brackets among the surveyed population. The largest income bracket is '$25,000 - $37,499,' comprising 1,963 individuals, closely followed by '$12,500 - $24,999' with 1,786 individuals. ' $37,500 - $49,999' is the third most common category, encompassing 1,762 individuals. Notably, a substantial portion of respondents falls into the higher income brackets, with ' $100,000 or More' accounting for 1,695 individuals, and ' $50,000 - $62,499' including 1,619 individuals. On the lower end of the income spectrum, 'Less than $12,500' is reported by 1,009 individuals. The categories '$87,500 - $99,999,' '$75,000 - $87,499,' and '$62,500 - $74,999' are represented by 876, 837, and 825 individuals, respectively. This income distribution indicates a diverse economic profile among the dataset's subjects, with a notable presence of both lower and higher-income individuals.

A pie chart with different colored circles

Description automatically generated

**Figure 5** Income pie chart

The 'storeType' column in the dataset classifies different types of stores that individuals frequent, shedding light on their preferences. Among these, 'Coffee House' emerges as the most popular store type, with 3,905 instances, suggesting a strong inclination towards coffee shops or cafes among respondents. 'Food Court' follows closely with 2,708 instances, signifying a substantial interest in dining at food courts. 'Fast Food' outlets also have a notable presence, with 2,344 instances. 'Pub' and 'Restaurant' lag slightly behind, with 1,961 and 1,454 instances, respectively. This distribution reflects a diverse range of store preferences, with a particularly high interest in coffee houses and food courts.

A pie chart with different colored sections

Description automatically generated

**Figure 6** Store type pie chart

The Chi-Square Statistic and p-value are fundamental elements of a statistical examination known as the Chi-Square test, employed to assess whether there is a notable relationship or independence between categorical variables within a dataset. The p-value, which is short for probability value, is a critical outcome of the Chi-Square test. In simpler terms, when the p-value is small (usually less than a predefined significance level like 0.05), it suggests that the observed relationship between the variables is probably not a result of random chance.

Put differently, if the p-value is greater than 0.05, it indicates that the observed connection between the variables is likely due to random chance and is not statistically meaningful. Looking at the figure below, we can see that for the "same direction" column, the p-value is >0.05, so we can conclude that this column is not statistically significant for predicting the outcome.

A screenshot of a computer

Description automatically generated

**Figure 7** Chi-Square test results

A heatmap is a graphical representation of data that uses color-coding to visualize the values in a matrix or a table. It is often used to show patterns, relationships, or variations in data. From the heatmap of the dataset, we can conclude that the relationship between temperature and stopby is the most intimate.

A screenshot of a computer

Description automatically generated

**Figure 8** Heatmap

**Data pre-processing:**

First, we will drop the "stopby" column because it represents our target variable, which is denoted as "column y." Then, we will randomly split the dataset into a training set and a test set, with an 80:20 ratio. So, the training set contains 9,897 rows, and the test set contains 2,475 rows.

From the observation of the "storetype" values, we notice that they are as follows: Coffee House, Food Court, Fast Food, Pub, and Restaurant. In order to give the "storetype" data more influence, we will expand it into five additional binary columns, each named after one of the five original values. Then, in the corresponding column for each of the original values, we will set the value to 1, while the values in the other four columns will be set to 0. For example, if the original value was "Coffee House” in that row, the "Coffee House" column value would be 1, and the values in the other four columns would be 0.

A screenshot of a computer

Description automatically generated

**Figure 9** storetype values

First of all, we removed all columns with the 'object' data type, including 'gender,' 'maritalStatus,' 'education,' 'destination,' 'passenger,' 'weather,' 'storeType,' and 'offerEndSoon”for the value transformation purpose. Additionally, we excluded the 'SameDirection' column from our analysis due to its p-value being greater than 0.5, indicating low statistical significance.

To process these columns, we applied LabelBinarizer, which converted these categorical variables into a binary format. However, we preserved the 'income' column as it contains ordinal data, and applying one-hot encoding to ordinal data would not be appropriate.

We used an ordinal encoder to transform the 'income' column. After the transformation, the values in this column were mapped to nine distinct categories, ranging from 0 to 8, corresponding to the income ranges: 'Less than $12500,' '$12500 - $24999,' '$25000 - $37499,' '$37500 - $49999,' '$50000 - $62499,' '$62500 - $74999,' '$75000 - $87499,' '$87500 - $99999,' and '$100000 or More,' respectively."

We also applied the same method to map the values in the 'visitPub,' 'visitCoffeeHouse,' 'visitFastFood,' 'visitFoodCourt,' and 'visitRestaurant' columns. These columns originally had five categories: 'never,' 'rarely,' 'sometimes,' 'often,' and 'very often.' After the transformation, their values were assigned integers ranging from 0 to 4.

A screenshot of a computer

Description automatically generated

**Figure 10** Result after the transformation

A screenshot of a computer

Description automatically generatedTo weight the transformed data, we multiplied the converted values by the columns that we initially split from the 'storeType.' As a result, the output consisted of a single column in each row with a value, while the rest of the columns were filled with zeros.

**Figure 11** Result after the multiplication

The above steps were carried out separately for both the train set and the test set. Eventually, the train and test set contains 44 columns for the model training.

**Model training and validation：**

Our project primarily involves binary classification, and the model proposed by our team is Random Forest, SVM, and XGBoost.

**Random Forest:**

Random Forest is a versatile and robust ensemble learning method recognized for its exceptional performance across a diverse set of classification tasks. It excels in handling both categorical and numerical data, making it a suitable choice for our dataset, which contains a mixture of data types. As part of our date processing process, we transformed the entire dataset into a numerical format for convenience and compatibility. Random Forest's ability to mitigate overfitting by averaging predictions from multiple decision trees distinguishes it from a single decision tree model.

**Underlying mechanism:**

Random Forest is a powerful ensemble learning technique that operates by constructing a multitude of decision trees during the training phase. In this ensemble, each decision tree is built on a different random subset of the dataset, a process known as bagging, which helps reduce the risk of overfitting and increases model generalization. Furthermore, each tree is trained on random subsets of features, a technique known as feature bagging, which adds an additional layer of diversity to the individual trees.

Since we're working on a binary classification task to predict whether a driver will make a stop ('1') or not ('0'). Random Forest builds, say, 100 decision trees. Each tree is trained on a different random subset of our dataset, which means they see different portions of the data. Additionally, when considering features for splitting at each node, they only have access to a random subset of the available features. This diversity in data subsets and feature subsets helps to capture different aspects of the underlying patterns in your data.

For example, one of the decision trees predicts that a particular driver will make a purchase based on their age and the weather conditions. Another tree might focus on income and store type, while another looks at education level and marital status. In the end, when the model wants to make a prediction for a new driver, each of the 100 trees casts its vote on whether the driver will make a stop or not. The final prediction is determined by the majority vote – whichever outcome (stop or no stop) gets the most votes from the individual trees becomes the model's prediction. This ensemble approach helps improve accuracy and reduces the likelihood of making decisions based on noisy or outlier data points, making Random Forest a robust choice for classification tasks.

**Support Vector Machine (SVM):**

SVM is a powerful algorithm for binary classification tasks, and it's effective when dealing with high-dimensional data, which can be the case in many real-world datasets since our dataset eventually contains 44 columns after the data processing stage. SVM aims to find the best hyperplane that separates the classes, making it suitable for your binary classification problem.

**Underlying mechanism:**

SVM seeks to find the optimal hyperplane that maximizes the margin between two classes in the feature space. The margin, in this context, represents the distance between the decision boundary (the hyperplane) and the nearest data points of each class. By maximizing this margin, SVM aims to achieve the best separation between the two classes, making it a robust choice for classification problems.

In cases where the data is not linearly separable in its original feature space, SVM employs a clever technique known as kernelization. It transforms the original data into a higher-dimensional space using a kernel function. This transformation can make the data linearly separable in the new space, allowing SVM to find an optimal hyperplane that can effectively separate the classes. Common kernel functions include the polynomial kernel and the radial basis function (RBF) kernel.

Since we are working on a binary classification task to predict whether a driver will make a stop product ('1') or not ('0'). Your dataset contains features like age, income, and store type. SVM, in this scenario, will aim to find the hyperplane that best separates the customers who make a stop from those who don't. If the data points in the original feature space aren't clearly separable with a straight line (hyperplane), SVM can apply a kernel function to map the data into a higher-dimensional space. In this transformed space, the customers who make a stop and those who don't might become more distinctly separated, allowing SVM to find an optimal hyperplane that maximizes the margin between the two groups.

The final decision boundary, the hyperplane, is determined by not only maximizing the margin but also minimizing classification errors. SVM strives to balance the desire for a wide margin with the need to correctly classify data points. This results in a robust classifier that can handle complex decision boundaries and perform well on a variety of classification tasks.

**XGBoost (Extreme Gradient Boosting):**

XGBoost, or Extreme Gradient Boosting, presents a compelling choice for our binary classification project for several key reasons. First and foremost, it boasts exceptional predictive accuracy, often surpassing other machine learning algorithms. In a binary classification task where precision and recall are crucial, XGBoost's performance can translate into more reliable and effective predictions. Furthermore, XGBoost is adept at handling complex relationships within your data, making it well-suited for real-world datasets that frequently exhibit intricate dependencies between features and the target variable. Its ability to model non-linear relationships is a significant advantage.

XGBoost provides valuable insights through its feature importance scores, enabling us to identify the most influential factors in your dataset. This feature analysis aids in understanding which variables are driving your binary classification outcomes, facilitating informed decision-making. Moreover, XGBoost offers regularization techniques, such as L1 and L2 regularization, to mitigate overfitting, thereby ensuring the model's generalization to unseen data. This is especially important for maintaining model performance.

**Underlying mechanism:**

At its core, XGBoost is an ensemble of decision trees, but what sets it apart is its sequential and gradient-driven approach. It constructs a series of decision trees sequentially, where each tree is meticulously designed to correct the errors made by the preceding ones. This iterative process allows XGBoost to incrementally improve its predictive accuracy.

Since we are using XGBoost to predict whether a driver will make a stop ('1') or not ('0'). XGBoost starts by building its first decision tree based on your data's features and target variable. This initial tree will likely make some errors in predictions. However, XGBoost doesn't stop there. It creates additional trees, with each new tree focused on correcting the errors made by the previous ones. For instance, if the first tree incorrectly predicts that a customer will make a stop due to their age, subsequent trees may prioritize other features like income or store type to refine the prediction.

What truly distinguishes XGBoost is its utilization of a gradient descent optimization algorithm during this process. It seeks to minimize a loss function, effectively reducing the difference between its predictions and the actual target values. By continuously adjusting its model based on the gradients of this loss function, XGBoost can discover the optimal model that best fits the data. This mechanism allows XGBoost to capture intricate relationships and patterns within your dataset, even in cases of high dimensionality or complex data structures.

In binary classification project, XGBoost's ability to iteratively refine its predictions, driven by the gradient descent optimization, can lead to highly accurate results. It excels in capturing the nuanced factors that influence customer purchase decisions, ensuring the model's performance improves with each iteration.

In summary, Random Forest, SVM, and XGBoost has been choosen for binary classification project due to their proven effectiveness, versatility, and suitability for our dataset. Each of these models has unique strengths and characteristics that can be beneficial in different scenarios, giving you a well-rounded approach to tackle your classification task.

**Model tuning and testing:**

**Random Forest:**

Grid search has been used by use for the model tuning to find the most suitable hyperparameter in our model.

Grid search is a systematic and effective approach to fine-tune the hyperparameters of a Random Forest model for our binary classification project. The `param\_grid` we've defined provides a range of values for key hyperparameters, allowing us to explore various configurations.

Starting with 'n\_estimators,' we consider values like 100, 150, and 200, which represent the number of trees in our ensemble. This exploration helps us strike a balance between model accuracy and computational efficiency, as more trees can capture complex patterns but may come at the cost of longer training times.

Next, 'max\_depth' is crucial for controlling the depth of individual decision trees. We've included options like `None` (unlimited depth), 10, 20, and 30. This range allows us to investigate the impact of tree depth on model complexity and overfitting, helping us find the optimal level of depth for our data.

Min\_samples\_split' and 'min\_samples\_leaf' are vital for regulating the granularity of tree splits and leaf nodes. By testing values such as 2, 3, and 4 for 'min\_samples\_split' and 1, 2, and 4 for 'min\_samples\_leaf,' we can determine how finely the tree divides the data and how detailed the leaf nodes become.

In the results, the best hyperparameters for the Random Forest model are as follows: {'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 3, 'n\_estimators': 100}

When comparing Random Forest with hyperparameter tuning (Model Tuning) to Random Forest without tuning (default parameters), the results show that the tuned model achieved an accuracy of 0.74464, while the untuned model had an accuracy of 0.7313. The difference between the two accuracies is 0.0151.

This comparison demonstrates that hyperparameter tuning has led to an improvement in model performance. The tuned model, with carefully selected hyperparameters, is more accurate than the default Random Forest model, resulting in a performance gain of 0.0151 in terms of accuracy.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Figure 12** Random Forest Classifier result with using the best hyperparameter

**SVM:**

The `param\_grid` we selected, containing 'C' and 'gamma' values of [0.1, 1, 10, 100] and [0.001, 0.01, 0.1, 1] respectively, holds significant importance in the realm of hyperparameter tuning, especially within the domain of Support Vector Machines (SVMs).

'C', a crucial hyperparameter, controls the balance between maximizing the margin (separation between data points) and minimizing classification errors. Smaller 'C' values, like 0.1, encourage a broader margin, permitting some misclassification in exchange. Conversely, larger 'C' values, such as 100, push for a narrower margin, reducing misclassifications on the training data.

Meanwhile, 'gamma', represented by values like 0.001 or 1, shapes the decision boundary's flexibility. Smaller 'gamma' values result in a smoother, gradual decision boundary, favoring better generalization but possibly underfitting the training data. In contrast, larger 'gamma' values create a more intricate and tightly fitting boundary, a potential risk for overfitting. Employing these 'C' and 'gamma' values within hyperparameter tuning techniques like grid search systematically evaluates the SVM model's performance with various configurations. The goal is to identify the optimal combination of 'C' and 'gamma' that maximizes the model's accuracy on a validation dataset, thus enhancing its predictive power on new, unseen data.

In the results, the best hyperparameters for the SVM model are as follows: {'C': 10, 'gamma': 0.01}

When comparing SVM with hyperparameter tuning (Model Tuning) to SVM without tuning (default parameters), the results show that the tuned model achieved an accuracy of 0.7369, while the untuned model had an accuracy of 0.6937. The difference between the two accuracies is 0.0151.

This comparison demonstrates that hyperparameter tuning has led to an improvement in model performance. The tuned model, with carefully selected hyperparameters, is more accurate than the default SVM model, resulting in a performance gain of 0.0432 in terms of accuracy.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Figure 13** SVM result with using the best hyperparameter

**XGBoost (Extreme Gradient Boosting):**

This grid defines a set of candidate values for three key hyperparameters: 'n\_estimators', 'max\_depth', and 'learning\_rate'. 'n\_estimators' represents the number of weak learners, typically decision trees, in the ensemble. By exploring values such as 50, 100, and 150, we investigate different ensemble sizes to determine the ideal number of weak learners for our specific task. 'max\_depth' governs the maximum depth of each individual tree, offering control over model complexity. Testing values like 3, 4, and 5 allows us to assess the impact of varying tree complexity on your model's performance. Lastly, 'learning\_rate' regulates the learning speed of the boosting algorithm, with lower values favoring better generalization. By experimenting with rates such as 0.01, 0.1, and 0.2, we are able to strike a balance between training speed and model accuracy.

In the results, the best hyperparameters for the XGBoost model are as follows: {'learning\_rate': 0.2, 'max\_depth': 5, 'n\_estimators': 150}

When comparing XGBoost with hyperparameter tuning (Model Tuning) to XGBoost without tuning (default parameters), the results show that the tuned model achieved an accuracy of 0.7515, while the untuned model had an accuracy of 0.7183. The difference between the two accuracies is 0.0332.

This comparison demonstrates that hyperparameter tuning has led to an improvement in model performance. The tuned model, with carefully selected hyperparameters, is more accurate than the default XGBoost model, resulting in a performance gain of 0.0332 in terms of accuracy.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Figure 14** XGBoost result with using the best hyperparameter

**3. Results and Discussion**

**Experimental results:**

**Random Forest Classifier**

As shown in Figure 15, the model performed exceptionally well on the training dataset, achieving high accuracy, precision, recall, and F1-scores for both classes. These metrics suggest that the model is effective in distinguishing between 'False' and 'True' instances, and it has a strong overall performance.

A screenshot of a computer

Description automatically generated

Figure 15

As shown in Figure 16, the model shows reasonably good performance in terms of precision, recall, and F1-score on testing dataset. It performs better on the "True" class, which indicates that it is relatively better at identifying positive cases. However, there is room for improvement in correctly identifying negative cases (False).

A screenshot of a computer

Description automatically generated

Figure 16

**Support Vector Machine**

As shown in Figure 17, the model exhibits reasonably good overall accuracy on training data. It shows a balanced trade-off between precision and recall for both "True" and "False" classes, as indicated by F1-scores around 0.76 and 0.83, respectively. The model's macro and weighted averages for precision, recall, and F1-score are all around 80%, suggesting consistent performance across classes.

A screenshot of a computer

Description automatically generated

Figure 17

From Figure 18, The SVM model exhibits reasonable performance on the testing dataset, with balanced precision and recall scores for both classes. The model maintains its effectiveness in distinguishing "True" instances while also demonstrating a fair level of precision for "False" instances.

A screenshot of a computer screen

Description automatically generated

Figure 18

**XGBoost Classifier**

According to Figure 19, the XGBoost Classifier performed well on the training dataset with an overall accuracy of 85.54%. Even though there is a slight signal of imbalance between "False" and "True" in recall, it exhibited good precision and F1-scores for both classes, indicating a balanced performance in predicting both "False" and "True" classes.

A screenshot of a computer

Description automatically generated

Figure 19

According to Figure 20, the model performed well on the testing dataset, but with some differences compared to the training data. While accuracy, precision, and F1-scores remained consistent, recall for the "False" class decreased on the testing data, which may indicate some challenges in correctly identifying "False" samples in the testing set.

A screenshot of a computer

Description automatically generated

Figure 20

**Discussion:**

Random Forest Classifier demonstrates best results when it comes to predicting with training dataset. Even though the results of predicting with testing datasets pales in comparison, the model still gives out one of the best results when compared to others. The Area Under Curve for ROC Curve for this model is 0.7362, very few digits away from the largest one, is a very good indication that it is a very good model. However, it takes up a lot of time for training the model.

SVM has the most consistent result of all when comparing the prediction of training dataset and testing dataset, which poses the least threat towards overfitting. However, it cannot perform well enough as there are overlapping feature distributions. Both training and testing result came in last when compared to other models. It is easily noticed when comparing its Precision-Recall Graph and ROC Curve with others.

XGBoost Classifier shows the best result among the models when predicting with testing dataset. Regardless of not having the best training accuracy, it has less overfitting issues when taking the classification reports into account, especially when compared to the random forest classifier model in this case. Despite having very good results, it is a complex model that has a lot of hyperparameters to be set.

In accordance to the computed p-value, storeType, visitCoffeeHouse and offerEndSoon indicates stronger evidence that their correlation with the output are statistically significant. Therefore, they are the most important feature when compared to others and are more likely to influence the output.

**4. Conclusion**

In this study, we conducted a comprehensive analysis of in-vehicle radio advertising effectiveness prediction using various machine learning models including Random Forest Classifier, Support Vector Machine (SVM), and XGBoost Classifier. The primary goal of this project is to determine whether a driver would make a stop at a local store after encountering a promotional advertisement on the car radio.

During the data exploration and preprocessing phase, the dataset explored which consisted of diverse features related to driver demographics, driving conditions, preferences, and store promotion details. The importance of data like store type, visit frequency to coffee houses, and offer expiration dates were visualized. The handling missing values process, encoding categorical variables, and transforming data are involved in data preprocessing session that ensure data quality and model compatibility.

During the model selection and training phase, there are three powerful machine learning models chosen, they are Random Forest Classifier, SVM, and XGBoost Classifier. These models were chosen for their versatility and effectiveness in handling binary classification tasks, and the data transformed into a suitable format for each of them. Each model was trained and validated using an 80:20 train-test split to assess its performance.

Lastly, during the model tuning and testing phase, each model's hyperparameters defined to optimize performance. Grid search was employed to identify the best hyperparameter combinations for each model. The tuned models consistently outperformed their default counterparts, showcasing the importance of hyperparameter optimization.

For the results and discussion, Random Forest Classifier achieved high accuracy and exhibited strong precision, recall, and F1-scores on the training dataset. While it performed well on the testing dataset, there was room for improvement in identifying negative cases (False). The model's ROC curve indicated good performance.

Besides, Support Vector Machine (SVM) demonstrated balanced performance on both training and testing datasets with consistent precision, recall, and F1-scores. While it had the least overfitting issues, it struggled with overlapping feature distributions. Its ROC curve and Precision-Recall graph showed room for improvement.

XGBoost Classifier excelled in predicting the testing dataset with balanced precision and recall. It exhibited good overall accuracy and balanced performance on both training and testing datasets. The ROC curve and Precision-Recall graph showed strong performance.

In conclusion, among the models tested, XGBoost Classifier emerged as the top performer in terms of predictive accuracy and balanced performance on unseen data.