

# CSE 4/587 Data Intensive Computing

## Project Phase #3

### **Vehicle Coupon Recommendation System**

#### **Team Members:**

Sai Kiran Paturi - 50442942

Manoj Paladi - 50496792

Jerry Mathew Oommen - 50560589

## **1. Problem Statement**

Developing an in-vehicle coupon recommendation system, especially for drivers, involves addressing the unique constraints and preferences inherent to driving contexts. The objective is to create a system that delivers relevant coupons to drivers based on real-time factors like destination, time, weather, traffic, and driving behaviour. By leveraging machine learning and data analytics, the system aims to optimize coupon selection and delivery, which enhances driver's coupon acceptance and gives businesses a targeted marketing avenue in the emerging realm of in-vehicle commerce.

1. How can in-vehicle coupon recommendation system accurately predict coupon acceptance while taking into account different situational factors?
2. What techniques improve the precision and flexibility of in-car coupon recommendation systems, offering efficient distribution of customised promotions regardless of varying road conditions?

### **1.1. Background**

These days, everyone commutes from one location to another. In-car coupon recommendation systems will provide drivers with various coupons. These systems help in providing the appropriate coupons to their respective interested customers by employing various strategies that take into consideration the consumer's characteristics. When distributing these coupons, they consider a variety of customer-related and external factors. These systems address the increased demand from customers for convenient and personalised experiences by bringing together data analytics, marketing tactics, and automotive technology. In-car coupon recommendation systems can provide drivers with targeted promotions and advertisements by utilising the huge quantity of data generated within the vehicle ecosystem, such as location data, driving patterns, driver's profiles, and outside variables like traffic and weather temperature, etc.

### **1.2. Contribution**

The dataset was obtained through an Amazon Mechanical Turk poll that included answers to a range of driving scenarios. Each scenario included variables like destination, time, weather, passenger count, and more. In each scenario that was presented, participants were prompted to decide if they would accept a voucher if they were the driver. This dataset provides light on the impact of various situational circumstances on people's propensity to accept incentives while travelling, which makes it a significant contribution to research on decision-making processes in commuting scenarios. The in-vehicle coupon recommendation system can enable companies to launch targeted marketing campaigns that deliver coupons to drivers at the most opportune moments. For example, if the data suggests that drivers are more likely to accept vouchers during certain weather conditions or times of the day, companies can schedule coupon delivery accordingly to maximize effectiveness. A personalized coupon recommendation system can help companies build stronger relationships with customers by demonstrating an understanding of their individual preferences and needs. This can lead to increased customer engagement and loyalty over time, as drivers are more likely to return to companies that provide them with relevant and valuable incentives while traveling.

## **2. Data Sources:**

For this project we used the online dataset from UC Irvine Machine Learning Repository. This dataset contains the features related to the driver and some of the external conditions which all depends in recommendation of the related coupon which makes driver to use.

- a) The dataset contains some personal information about the driver such as gender, age etc and some external information like weather, temperature, location etc.
- b) The dataset also contains information about frequency of visits to restaurants, cafes, and bars of the driver to know his sentiments.

Data Source Link: <https://doi.org/10.24432/C5GS4P>.

## **3. Phase 1 - Summary:**

Phase 1 presents the development of an in-vehicle coupon recommendation system that aims to provide drivers with relevant coupons based on real-time factors such as destination, time, weather conditions, traffic patterns, and driving behavior.

By leveraging machine learning and data analytics techniques, the system seeks to optimize coupon selection and delivery, thereby enhancing coupon acceptance rates and offering businesses a targeted marketing channel within the emerging in-vehicle commerce landscape.

### **3.1. Data cleaning process:**

In the data cleaning stage, various procedures were employed to enhance the quality and integrity of the dataset. Column names were made more descriptive, and data types were converted to appropriate formats to improve computational efficiency and understandability. Irrelevant or highly incomplete columns were identified and dropped to reduce noise and overhead.

Categorical data labels were examined to aid in interpretation, and consistent formatting was applied to enhance clarity. Duplicate rows were removed to prevent skewed results, and data consistency tests were conducted to detect and address anomalies or missing data points. Rows with missing values in critical features were dropped to ensure data completeness, while mode imputation was used to handle missing values in other columns. Finally, categorical variables were encoded to numerical values, facilitating compatibility with machine learning algorithms and improving predictive accuracy.

### **3.2. Exploratory Data Analysis Techniques:**

A comprehensive exploratory data analysis was undertaken to gain insights into the dataset and inform feature selection and modelling processes. Univariate analysis techniques were employed to understand the distribution, characteristics, and behaviors of individual variables, aiding in data cleaning and transformation decisions. Bivariate analysis methods were utilized to analyse the relationship between each feature and the target variable, identifying patterns, trends, and potential correlations. Chi-square tests were conducted to determine the statistical significance of the association between categorical features and the target variable, guiding feature selection.

Feature engineering techniques were applied to create new features or transform existing ones, enhancing the predictive power of the models. Correlation analysis and heatmaps were used to identify relationships between features, detect multicollinearity, and identify potential redundancies, further refining feature selection.

Phase 1 covers data cleaning procedures, exploratory data analysis, feature engineering, correlation analysis, and feature selection processes applied to a dataset.

## **4. Phase 2 – Summary:**

The phase 2 solves the problem of developing an in-vehicle coupon recommendation system that can accurately predict coupon acceptance by drivers based on various situational factors like destination, time, weather, traffic, and driving behaviour. An online dataset from the UC Irvine Machine Learning Repository containing features related to the driver and external conditions was used for this project. The objective was to leverage machine learning and data analytics to optimize coupon selection and delivery, enhancing driver coupon acceptance and providing businesses with a targeted marketing avenue for in-vehicle commerce.

### **4.1. Overview Of Learning Models:**

Several machine learning algorithms were evaluated on this dataset, including logistic regression, random forest, gradient boosting machines, k-nearest neighbours, naive Bayes, support vector machines (with linear and RBF kernels), and XGBoost. Logistic Regression handles binary classification well with categorical features but assumes linear relationships. Random Forests capture nonlinear patterns by combining multiple decision trees but lack interpretability. Gradient Boosting Machines iteratively improve by adding trees to correct prior errors, though computationally expensive. K-Nearest Neighbors is a simple non-parametric method based on similarity but sensitive to outliers. Naive Bayes uses probability for classification tasks like text analysis and spam filtering. Support Vector Machines with linear kernels suit linearly separable data, while RBF kernels model nonlinear relationships. Detailed analysis was carried out on the performance of each model, considering evaluation metrics like accuracy, precision, recall, F1 score, and ROC AUC score. The results showed that tree-based ensemble models like XGBoost, random forests, and gradient boosting machines generally outperformed linear models and other non-parametric models in this classification task.

### **4.2. Best Model Suited For Our Data:**

Among all the models, XGBoost demonstrated the best overall performance, with the highest accuracy, precision, recall, F1 score, and ROC AUC score. However, the choice of model was noted to depend on specific requirements, such as the balance between precision and recall, computational efficiency, and interpretability. Based on feature importance analysis, strategies like improving coupon types, considering age and income criteria, and matching offerings to venue types (bars, coffee houses, etc.) were identified as potential ways to

increase coupon acceptance rates. The report concludes by emphasizing the importance of data quality, feature engineering, and aligning model predictions with actionable business insights

Among the various models evaluated, including Logistic Regression, Random Forest, Gradient Boosting Machines (GBM), K-Nearest Neighbors (KNN), Naive Bayes, and Support Vector Machines (SVM) with linear and Gaussian kernels, the XGBoost model demonstrated the best overall performance across multiple evaluation metrics.

It achieved the highest accuracy of 74.54%, along with the highest precision of 75.35%, recall of 81.02%, F1 score of 78.08%, and ROC AUC score of 73.66%.

### 4.3. The Rationale For Selecting Xgboost:

**Superior Performance Metrics:** XGBoost outperformed all other models in terms of accuracy, precision, recall, F1 score, and ROC AUC score, indicating its ability to effectively classify whether a driver would accept or reject a coupon based on the given situational factors.

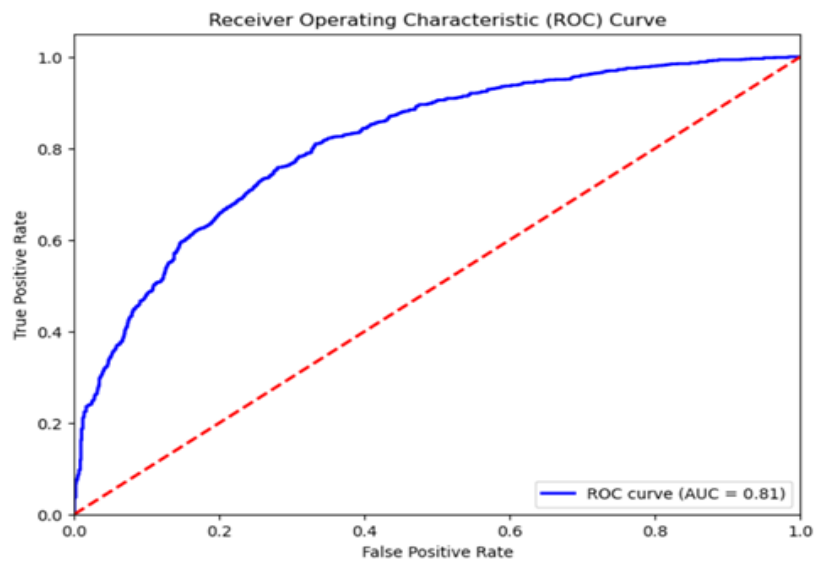
**Regularization and Overfitting Control:** XGBoost incorporates various regularization techniques, such as L1 and L2 regularization, which help mitigate overfitting and improve the model's generalization performance. This is crucial when dealing with a dataset that may contain noise or outliers, ensuring that the model does not memorize the training data but learns meaningful patterns.

**Computational Efficiency:** XGBoost is designed for parallel computing and is highly efficient in processing large datasets. This is advantageous in the context of real-time coupon recommendation systems, where timely predictions are essential for delivering relevant coupons to drivers.

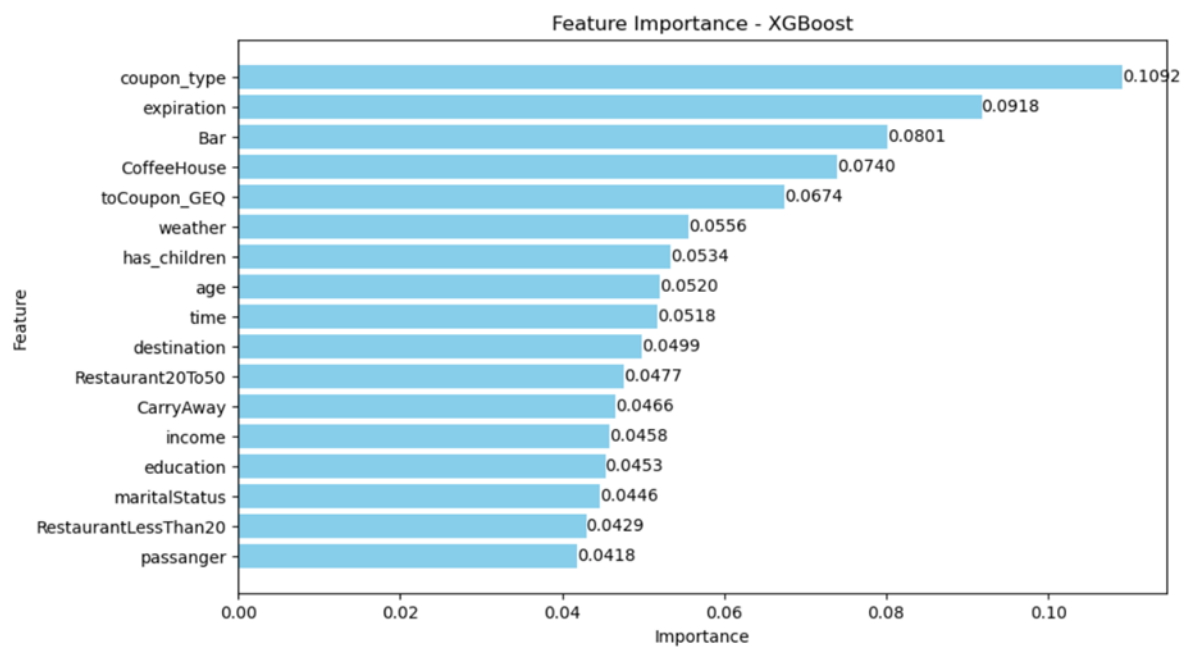
**Feature Importance:** The XGBoost model provides insights into the relative importance of features, which can be valuable for understanding the key factors influencing coupon acceptance. According to the report, features like coupon type, expiration, and bar were identified as the top three most important features for determining the target variable (coupon status).

### 4.4. Results:

Metric	Value
Accuracy	74.54%
Precision	75.35%
Recall	81.02%
F1 Score	78.08%
ROCAUC Score	73.66%



#### 4.5. Feature Importance:



#### 4.6. Model Evaluation Metrics:

S.No	Model	Accuracy	Precision	Recall	F1 Score	ROC AUC Score
1	XGboost	74.54%	75.35%	81.02%	78.08%	73.66%
2	Random Forest	73.35%	73.46%	82.02%	77.50%	72.17%
3	GBM	73.43%	74.52%	79.82%	77.08%	72.56%
4	Logistic	62.89%	62.81%	82.59%	71.35%	60.22%
5	Naïve Bayes	65.47%	67.09%	75.20%	70.91%	64.16%
6	KNN	67.86%	69.34%	76.33%	72.67%	66.71%
7	SVM	66.59%	67.15%	78.89%	72.55%	64.92%

#### 4.7. Hyper Parameter Tuning and Best Parameters

S.No	Model	Parameters
<u>1</u>	Logistic regression	max_iter=1000,C=0.1
<u>2</u>	Random forest	n_estimators=100, random_state=42,max_depth=20
<u>3</u>	Gradient Boosting Machines	n_estimators=300, random_state=42,learning_rate=0.2, max_depth= 5
<u>4</u>	<u>KNN</u>	<u>n_neighbors=7,metric='manhattan', weights='distance'</u>
<u>5</u>	<u>Navie Bayes</u>	<u>categorical NB</u>
<u>6</u>	<u>SVM</u>	<u>(C= 1, gamma= 0.1, kernel='rbf'</u>
<u>7</u>	<u>XGB</u>	<u>learning_rate= 0.1, max_depth= 5, n_estimators= 500</u>

## 5. Phase 3:

During Phase 3, our main goal is to develop a specialized website. This website is intended to be easy to use and engaging, functioning as a display for the findings and forecasts resulting from our detailed analysis in earlier phases.

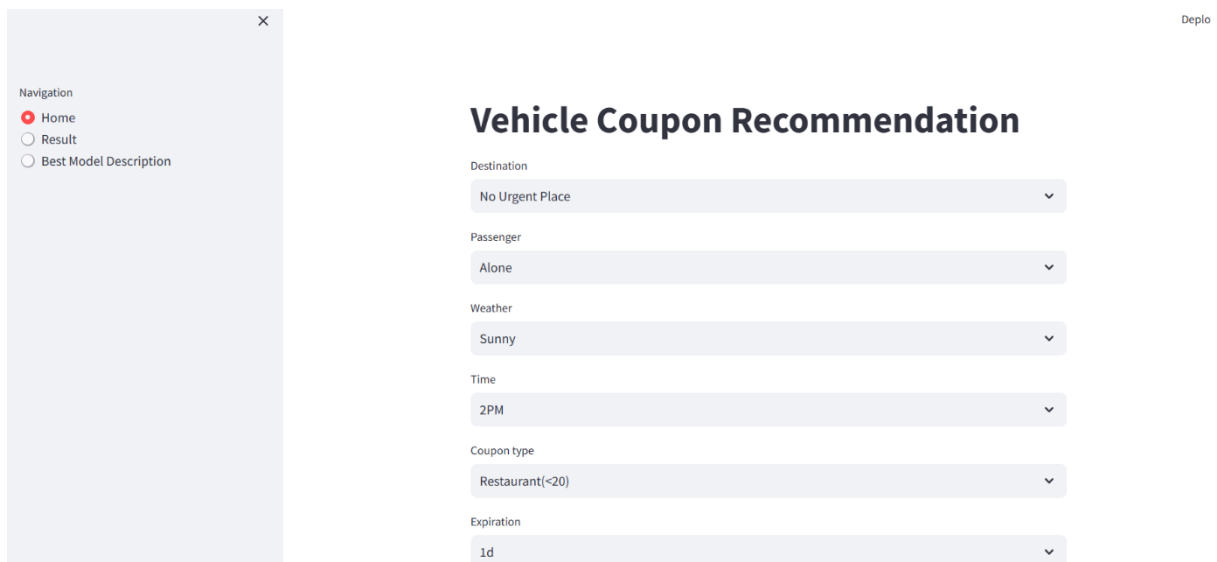
By leveraging the enhanced dataset and utilizing machine learning techniques, the website seeks to provide meaningful and useful insights.

To accurately predict coupon acceptance in a vehicle context while considering different situational factors, our approach involved developing a dedicated website integrated with machine learning models.

By leveraging these models and analysing a refined dataset, we were able to enhance the accuracy of our predictions. The website's integration with machine learning algorithms allowed us to factor in various situational elements, such as road conditions, and user preferences, ensuring more precise coupon recommendations.

### 5.1. Website UI Explanation:

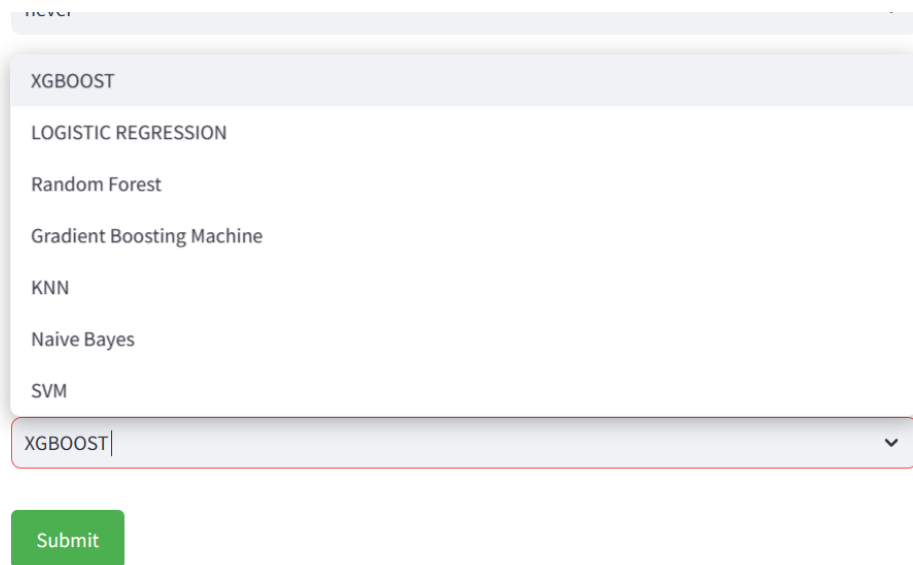
We used Python Streamlit library to develop the website. Python language was used for both front and backend development.



The "Home" page of the website serves as the primary input collection interface, featuring 17 dropdowns and one text input (for Age), enabling users to select or input values for various parameters. These parameters cover a range of factors relevant to the application's domain, allowing users to provide detailed input for analysis and prediction. Additionally, the page offers a selection of seven models, providing users with options to choose the machine learning approach that best suits their needs and preferences. This comprehensive input mechanism ensures flexibility and customization in data input, enhancing the accuracy and relevance of predictions generated by the system.



Different types of models a user can **select** are:



The image shows a web interface for model selection. A dropdown menu is open, displaying a list of machine learning models: XGBOOST, LOGISTIC REGRESSION, Random Forest, Gradient Boosting Machine, KNN, Naive Bayes, and SVM. The 'XGBOOST' option is currently selected and highlighted. Below the dropdown menu is a green rectangular button with the text 'Submit' in white.

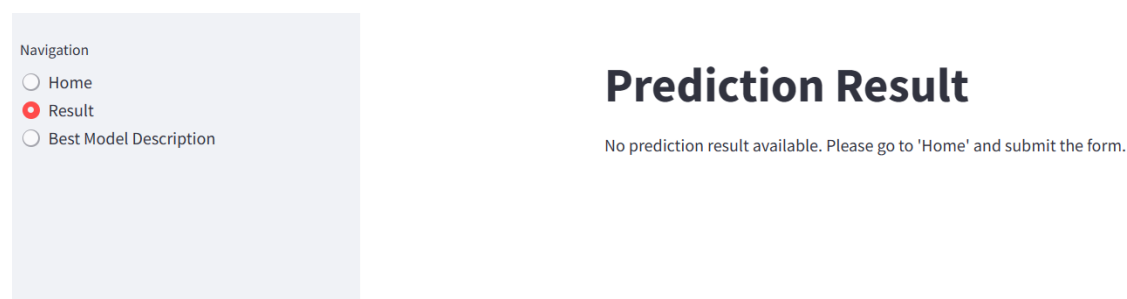
The website provides users with a selection of seven models via a dropdown menu, enabling them to choose their desired model for analysis. Upon selection, the result page displays outcomes specific to the chosen model, offering users immediate feedback tailored to their selection.

In addition, the page showcases results from the best-performing model identified during phase 2, providing additional insights into the most accurate predictions generated by the system. This dual-display approach enhances user engagement and facilitates informed decision-making based on a comprehensive range of model predictions and analyses.

### The Result Page:

#### Result page if no information has been given:

If the user didn't provide any information, the result will be displayed as default, indicating a lack of user input or selection.



The image shows a web page titled 'Prediction Result'. On the left side, there is a navigation sidebar with three radio button options: 'Home', 'Result' (which is selected), and 'Best Model Description'. The main content area on the right has the heading 'Prediction Result' in a large, bold font. Below the heading, a message states: 'No prediction result available. Please go to 'Home' and submit the form.'

## Actual result :

When the user selects a model, provides all required information, and submits the form, the system generates predictions based on the chosen model. Simultaneously, it presents results from the best-performing model, offering users insights into the most accurate predictions obtained by the system.

## Prediction Result

According to the model chosen:

The User will accept the coupon.

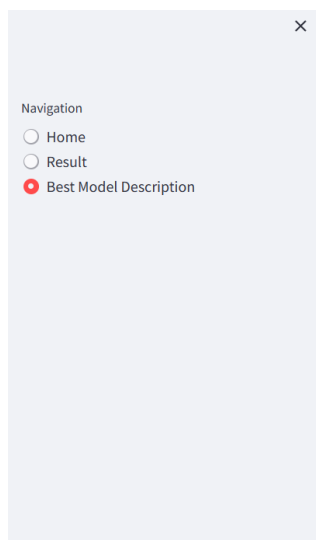
According to the best model (XGBoost):

The User will accept the coupon..

This comprehensive approach ensures that users receive tailored predictions based on their model selection while benefiting from the system's optimized performance through the best model results.

## Best Model Description page:

The website includes a feature where; after viewing the results page, users can choose to access detailed information about the best-performing model. This model description page provides comprehensive insights into why the model is considered the best, including detailed descriptions of its architecture, performance metrics, and feature importance analysis.



## XGboost Model

The XGBClassifier is a component of the XGBoost (Extreme Gradient Boosting) library, known for its efficient implementation of the gradient boosting algorithm. Gradient boosting, an ensemble learning technique, aggregates predictions from numerous decision trees to construct a robust predictive model. XGBoost employs an objective function comprising a loss function and regularization terms, which are optimized during training. Additionally, it integrates L1 (LASSO) and L2 (Ridge) regularization terms into the objective function, serving to mitigate overfitting and enhance the model's ability to generalize to unseen data.

## Key Factors

- Predictive Accuracy:** XGBoost is particularly effective in achieving high predictive accuracy.
- Regularization:** XGBoost incorporates regularization techniques like L1 (LASSO) and L2 (Ridge) regularization to prevent overfitting.
- Speed and Performance:** XGBoost is highly computationally efficient and designed for parallel processing. It can handle large datasets and complex patterns.

## Model Performance:

The model description page highlights key aspects such as the model's accuracy, precision, recall, and F1 score and ROC AUC score showcasing its performance across various evaluation metrics.

### Model performance

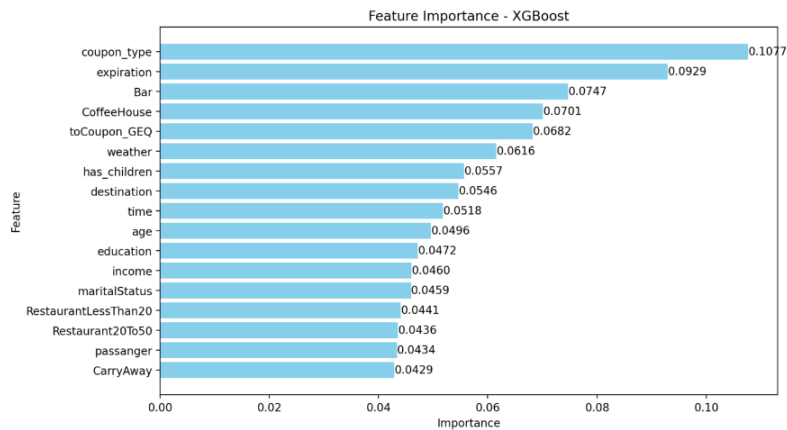
	Metric	Value
1	Accuracy	74.54%
2	Precision	75.35%
3	Recall	81.02%
4	F1 Score	78.08%
5	ROC AUC Score	73.66%

Additionally, users can explore the model's hyperparameters, tuning process, and any specific techniques used to enhance its predictive capabilities. The feature importance section reveals the significance of different features in the model's decision-making process, providing users with valuable insights into the factors driving accurate predictions. Overall, this detailed model description empowers users to understand the inner workings of the best model and gain deeper insights into its performance and predictive capabilities.

## Feature Importance:

The feature importances are dynamically provided based on the user's input, reflecting the specific parameters chosen during data input. This information is presented in a graphical representation, allowing users to visually interpret the significance of different features in the model's decision-making process.

### Feature Impotance



The dynamic nature of these feature importances ensures that users receive tailored insights relevant to their input, enhancing their understanding of the factors driving accurate predictions. This interactive approach enables users to explore the impact of different features on the model's outcomes, facilitating informed decision-making and deeper analysis of the prediction results.

### **Users to upload their own data:**

Users can give path of their datasets and they can run the files named test\_app.py and dic\_project.py to get the results for their dataset.

```
data=pd.read_csv("C:\\Users\\palad\\Desktop\\in+vehicle+coupon+recommendation\\in-vehicle-coupon-recommendation.csv")
work_df=data.copy()
cleaned_df=LabelEncoder(data_cleaning(work_df))
```

## **5.2. Code Documentation:**

There are two files named dic\_project.py and test\_app.py. The python file dic\_project.py will perform data preprocessing of raw data and trains all the 7 models. test\_app.py contains code for developing the website and integrating the models with real time user data to perform classification.

### **5.2.1. dic\_project.py:**

#### **a) import all necessary libraries**

```
1  import streamlit as st
2  import pandas as pd
3  from sklearn.preprocessing import LabelEncoder
4  from sklearn.preprocessing import OneHotEncoder
5  from sklearn.model_selection import train_test_split
6  from sklearn.linear_model import LogisticRegression
7  from sklearn.model_selection import GridSearchCV
8  from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
9  from sklearn.neighbors import KNeighborsClassifier
10 from sklearn.naive_bayes import GaussianNB, MultinomialNB, CategoricalNB, BernoulliNB
11 from sklearn.svm import SVC
12 from xgboost import XGBClassifier
13 from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix
14 import seaborn as sns
15 from sklearn.ensemble import RandomForestClassifier
16 import xgboost as xgb
17 import matplotlib.pyplot as plt
18 import pickle
```

**b) Function name: data\_cleaning(work\_df)**

**Inputs:** Dataframe that contains raw data.

**Process steps:**

1. Changing column names to make it more meaningful.
2. Changing datatypes from object to int.
3. Drop columns that have large number of null values.
4. Changing value names to make it more meaningful.
5. Removing duplicate rows.
6. Encode features with categorical value to numeric values using label encoder and store the reverse transforms to a labels table.

**Outputs:** Cleaned data and labels dictionary for mapping feature values to numeric values.

**c) Function name: model(work\_df)**

**Inputs:** pandas data frame that contains clean data/pre-processed data

**Process Steps:**

1. Training Data is fitted into 7 models: Logistic regression, Random forest, Gradient Boosting Machines, KNN, Naive Bayes, SVM and XGB.
2. Labels and each model are stored into separate pickle files for use by app.

**Outputs:** List of trained classifiers and data frame used for training the model.

**d) Additional steps:**

- Reading the raw data file in csv format
- Calling data\_cleaning and model function
- Storing the classifiers, labels, dataset used for training in sav format using pickle library.

### 5.2.2. test\_app.py

#### 1) import all necessary libraries

```
1 import streamlit as st
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import pandas as pd
5 import pickle
```

#### 2) Loading pre-trained models and label mappings and train dataset:

```
with open('models.sav', 'rb') as file:
    model = pickle.load(file)
with open('labels.sav', 'rb') as file:
    labels = pickle.load(file)
with open('train_df.sav', 'rb') as file:
    x_train = pickle.load(file)
```

The code segment loads pre-trained machine learning models and label encoders from binary files named 'models.sav' and 'labels.sav' using the pickle module in Python. The 'models.sav' file contains the trained machine learning models, while 'labels.sav' stores the label dictionary used for data mapping. By loading these pre-trained components, the application can efficiently utilize learned patterns and transformations, enhancing prediction accuracy without the need for retraining. This approach streamlines the prediction process and ensures consistent performance across different datasets.

#### 3) Function make\_input(values, labels)

**Input:** values: list of values entered by user, dict of labels for mapping

##### Process Steps:

- Take values list from the user and map them.
- The mapped values were transformed to pandas' data frame object

**Output:** A data frame object having a single row.

#### 4) Streamlit Interface and User Input Collection:

```
page = st.sidebar.radio("Navigation", ["Home", "Result", "Best Model Description"])
```

Using sidebar we can navigate to three different pages.

##### In home page:

Using selectbox and text\_input dropdowns and open-ended interface were created.

```
if Button:
    if any(not value or value == "" for value in Values): # Check if both dropdowns are selected
        st.error("Please select options from all dropdowns before submitting.")
    else:
        input_df = make_input(Values, labels)
        predicted_val = model[model_dict[Model]].predict(input_df)
        best_predicted_val = model[6].predict(input_df)
        if best_predicted_val == 1:
            best_predicted_val = "The User will accept the coupon."
        else:
            best_predicted_val = "The User will probably decline the coupon."
        st.session_state.predicted_val = predicted_val
        st.session_state.best_predicted_val = best_predicted_val
```

After submitting it checks whether you selected all options then predicts the value using the trained models.

##### In result Page:

Then you have to navigate to results page manually. You can see the results over there.

```
if page == "Result":
    st.title("Prediction Result")

    # Retrieve and display prediction result
    if "predicted_val" in st.session_state:
        predicted_val = st.session_state.predicted_val
        best_predicted_val = st.session_state.best_predicted_val
        if predicted_val == 1:
            st.write("According to the model chosen:")
            st.markdown(f'<p style="font-size: 20px; background-color: lightgreen; padding: 10px;">The User will accept the coupon.</p>',
                        unsafe_allow_html=True)
        else:
            st.write("According to the model chosen:")
            st.markdown(f'<p style="font-size: 20px; background-color: lightred; padding: 10px;">The User will probably decline the coupon.</p>',
                        unsafe_allow_html=True)
        st.write(f"According to the best model (XGBoost):")
        st.markdown(f'<p style="font-size: 20px; background-color: lightgrey; padding: 10px;">{best_predicted_val}</p>',
                    unsafe_allow_html=True)
    else:
        st.write("No prediction result available. Please go to 'Home' and submit the form.")
```

##### In Best model description:

This contains markdown code model description and real time visualization of feature importance based the data used for the training.

```

importances = xgb_classifier.feature_importances_
feature_names = X_train.columns
sorted_indices = importances.argsort()

# feature importance
plt.figure(figsize=(10, 6))
plt.title("Feature Importance - XGBoost")
plt.barh(range(X_train.shape[1]), importances[sorted_indices], align="center", color='skyblue')
plt.yticks(range(X_train.shape[1]), [feature_names[i] for i in sorted_indices])
plt.xlabel("Importance")
plt.ylabel("Feature")
for i, v in enumerate(importances[sorted_indices]):
    plt.text(v, i, f'{v:.4f}', ha='left', va='center', color='black')
st.pyplot(plt)

```

### 5.3. Instructions:

- 1) Copy the path of raw data csv file. Paste it in read\_scv command in dic\_project.py file
- 2) Run the dic\_project.py file. Now three sav files were created named models, labels, train\_df.
- 3) Before running the test\_app.py file ensure that all there sav files in current working directory.
- 4) Run the test\_app.py using the command  
“**streamlit run** c:.....path/test\_app.py” in the python terminal.
- 5) A local URL is generated you will either directed to html page or paste the URL in the browser.
- 6) Choose all the fields and press submit.
- 7) Navigate to results page using navigation on left side. You can see the results.
- 8) If you wish to know about the beat model for classification ie. XGboost you can navigate to third page where you can see real time Feature importance visualization and evaluation metrics.
- 9) If the user wants to try with his own data. The data should be in csv format and follow the steps from 1 to 8 mentioned above.

## 6. Conclusions:

- All models underwent fine-tuning using grid search, allowing us to identify and implement the optimal parameters from the best-tuned model into the product.
- XGBoost exhibited superior performance compared to the other models assessed, achieving the highest accuracy, precision, recall, F1 score, and ROC AUC score.
- Tree-based models like XGBoost, Random Forest, and GBM generally outperformed linear models (e.g., Logistic Regression) and other non-parametric models (e.g., Naïve Bayes, KNN, SVM) in this classification task.
- When selecting a model, it's important to consider task-specific requirements, such as balancing precision and recall, computational efficiency, and interpretability. XGBoost stands out as a strong choice due to its excellent performance across multiple evaluation metrics.



- The performance of a model is significantly influenced by data quality, effectiveness of feature engineering, and suitability of the modeling approach. Additionally, it's crucial to evaluate whether the model's predictions translate into actionable insights that can improve coupon acceptance rates within practical business constraints.
- Improvising coupon type has major impact on acceptance rate this can be observed from feature importance from XGboost, GBM, and Random forest.
- The models with higher ROC AUC and accuracy scores (XGB, RF, GBM) can help guide the design and execution of these strategies by providing more reliable coupon acceptance prediction.
- Age and income criteria even have major impact on coupon recommendation. This helps design strategies to target audience.
- Offering a proper coupon type based the availability of bar, coffee\_house etc has major impact on acceptance.
- The organization that utilizes this product can leverage the insights from reports 1 and 2, combining domain expertise with data analysis to design effective business strategies.
- They can then validate and refine these strategies using the product, which achieved an accuracy rate of approximately 75%. This allows the company to make informed decisions and optimize their operations based on the insights and capabilities provided by the product.

## **7. Future Scope:**

- Enhancements to this product could include incorporating a user interface feature that allows direct input of data, streamlining the process by automating model selection based on the provided data.
- An additional feature could involve integrating a testing section to assess the success rate of strategies developed by the team.
- Another potential improvement is the implementation of real-time visualization of key metrics influencing coupon acceptance.
- Furthermore, the product could be updated automatically with new data to continuously refine the model's accuracy and relevance.

## **8. References:**

- [1] XGBoost: A Scalable and Accurate Implementation of Gradient Boosting by T. Chen and C. Guestrin .
- [2] Introduction to Random Forest by Will Koehrsen .
- [3] Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron.
- [4] Data Science from Scratch- First Principles with Python by Joel Grus.
- [5] [https://scikit-learn.org/stable/user\\_guide.html](https://scikit-learn.org/stable/user_guide.html)
- [6] Hands-On Gradient Boosting with XGBoost and scikit-learn: A practical guide to building high-performance gradient boosting models for real-world structured and unstructured data" by Corey Wade and Andrii Gakhov