|  |
| --- |
| **CECS 550: Pattern Recognition**        **Group-9**    **Project Report**    **REPEAT BUYER’S PREDICTION FOR E-COMMERCE**    **Submitted by**          Rishikesh Chava 029404386  Rajeev Sai Nitturu 030823908  Aashrith Racherla 030839755          Under the guidance of Prof. Mahshid Fardadi |

# ABSTRACT

Promotions are commonly used by merchants to attract new customers, but many of these customers are only interested in one-time deals. This means that the effect of promotions on long-term sales is limited. To optimize their ROI and lower the cost of promotions, merchants must distinguish between one-time buyers and potential loyal customers and focus their efforts on converting the latter.

The project provides a dataset with information on promotional shopping events from an e-commerce platform. Your task is to develop a system that will enhance the ROI by predicting the likelihood that new buyers will make another purchase from the same merchant within 6 months, while also reducing promotional costs and identifying one-time buyers.

# DATA PRE-PROCESSING

* Reading csv files into data frames using pandas library.

Text

Description automatically generated

* Renaming the seller\_id column in user behavior logs to merchant\_id • Merge user behavior logs with user profile and training data
* Filter merged\_dataset\_training based on the item\_id range.
* Drop the data without labels.
* Replace null values in the age range column with 0.
* Replace null values in the age range column with 2 (unknown) • Next, we repeated the same process for merged\_dataset\_testing.

Text

Description automatically generated

# 1. DATA VISUALIZATION

## 1.1 Graph for Action Type Column

Chart, bar chart

Description automatically generated

We may infer from the information provided that a tracking mechanism is in place to keep tabs on user activity on a website or application. Depending on how engaged a user is with the website/application, the tracking system categorizes their actions.

Four action types:

Action Type 0 (CLICK): When a person clicks on a link or button but then does nothing further.

Action Type 1 (ADD TO CART): When a person adds an item to their shopping cart.

Action Type 2 (PURCHASE): When a customer makes a purchase.

Action Type 3 (ADD TO FAVORITE): when a person adds a product to their list of favorites.

According to the bar graph, a considerable portion of users fall into category 0 (click), which means they are simply browsing or using the website without making any big purchases or adding items to their shopping carts. However, a relatively small percentage of users are actively interacting with the website/application and adding items to their cart, as shown by the minimum percentage of users that fall into category 1 (add-to-cart). Around 40% of the users fall into category 2 (Purchase), which means these users are actively using website for making purchases and 10% of users fall into category 3 (add-to-favorite), adding items to favorites depending on their interests while browsing the website.

**1.2 Bar Graph showing the Repeat Buyers by age range for Women.**

Chart, bar chart

Description automatically generated

The bar graph illustrates the association between women and repeat customers. According to the graph, women between the ages of 20 and 40 are more likely to go shopping than women of other ages. Despite having fewer inputs overall, the data indicates that these women tend to make fewer repeat purchases than women between the ages of 60 and 65. These findings could have several implications for businesses targeting female consumers. First off, focusing on women between the ages of 20 and 40 could be a profitable move because they have a larger customer base. However, businesses need to identify and address the reasons why these women are not making as many repeat purchases. However, despite having a smaller market share, women between the ages of 60 and 65 appear to be more devoted buyers. Businesses may need to think about how to entice and keep these customers, possibly by providing targeted promotions or loyalty programs.

Overall, this graph demonstrates the significance of comprehending consumer behavior and preferences when creating marketing strategy. By analyzing data on repeat customers across different age ranges, businesses can identify trends and patterns that can inform their marketing and sales strategies, and ultimately lead to increased customer satisfaction and loyalty.

**1.3 Bar Graph showing the Repeat Buyers by age range for Men.**

Chart, bar chart

Description automatically generated

The bar graph illustrates the association between men and repeat. The plot indicates that men between the ages of 30 and 40 are more likely to shop than other age groups and tend to make repeat purchases compared to those who do not make any repeat purchases. These findings have significant implication for companies that market to men. First off, focusing on men between the ages of 30 and 40 may be a profitable tactic because this demographic represents a sizeable customer base that is likely to make repeat purchases. Understanding the causes of this behavior is crucial. The data also suggests that there could be a chance to improve repeat sales from males in other age groups who don't currently make any repeat purchases. To do this, we might have to figure out why they behave the way they do and remove any obstacles that might prevent them from making more purchases in the future.

Overall, this graph demonstrates the significance of comprehending consumer behavior and preferences when creating marketing strategy. By analyzing data on repeat customers across different age ranges, businesses can identify trends and patterns that can inform their marketing and sales strategies, and ultimately lead to increased customer satisfaction and loyalty.

**1.4 Stack Plot showing the Repeat Buyers by Age Range and Gender:**

Chart, bar chart

Description automatically generated

**1.5 A Pie Chart showing portion of Repeat and Non-Repeat Buyers.**

Chart, pie chart

Description automatically generated

The above pie chart indicates the proportion of repeat buyers and non-repeat buyers. It indicates that 93.6% of the users are the non-repeat buyers whereas the remaining 6.4% are the repeat buyers.

This shows that the majority of the buyers or the customers are making the purchases only once. And only a few percent of people are re-visiting and making purchases from the ECommerce website more than once.

This visualization is very important for the companies so that they may take some actions or implement some kind of new advancements in their business so that they may increase the number of repeat buyers. This also helps them in identifying the areas of improvement.

## 1.6 Relationship between action type and age range

Chart, scatter chart

Description automatically generated

The graph represents the relationship between the action type and age range of the customers who interacted with a product on an e-commerce website. As per the above data we can observe a tight correlation between the clicks and purchases done by customers and a weak correlation between clicks and add-to-cart. The data suggests that customers are making a decision to purchase a product after viewing its description.

## 1.7 Age range distribution between repeat and non-repeat buyers

Chart, line chart

Description automatically generated

The x-axis represents the age range of customers, and the y-axis represents the density of customers in that age range. The plot is split by yellow and blue color to distinguish between repeat and non-repeat buyers. Based on the above data we can observe between age 20 – 40 there are more repeated buyers and conversely from age 30 – 70 the non-repeated users’ density is high.

## 1.8 Relationship between action type, age range and frequency

Chart

Description automatically generated

The plot generated by the code is a scatter plot that visualizes the relationship between action type, age range, and frequency. The x-axis represents the action types, where 0 corresponds to click, 1 corresponds to add-to-cart, 2 corresponds to purchase, and 3 corresponds to add-tofavorite. The y-axis represents the age ranges. Each point in the plot represents a combination of action type and age range, and the size of the point represents the frequency of that combination. According to the above data most no of purchases are made by users between age 20 – 40.

## 1.9 Distribution of action type by gender

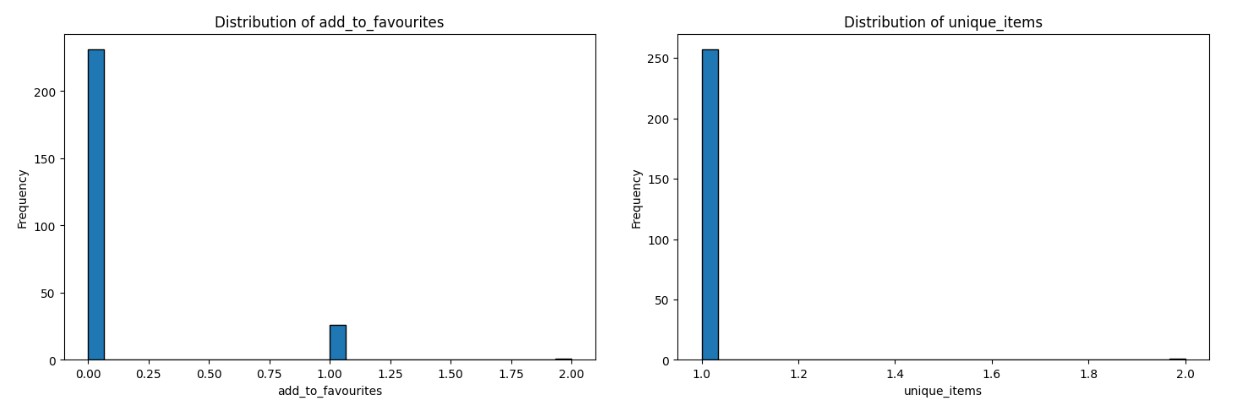
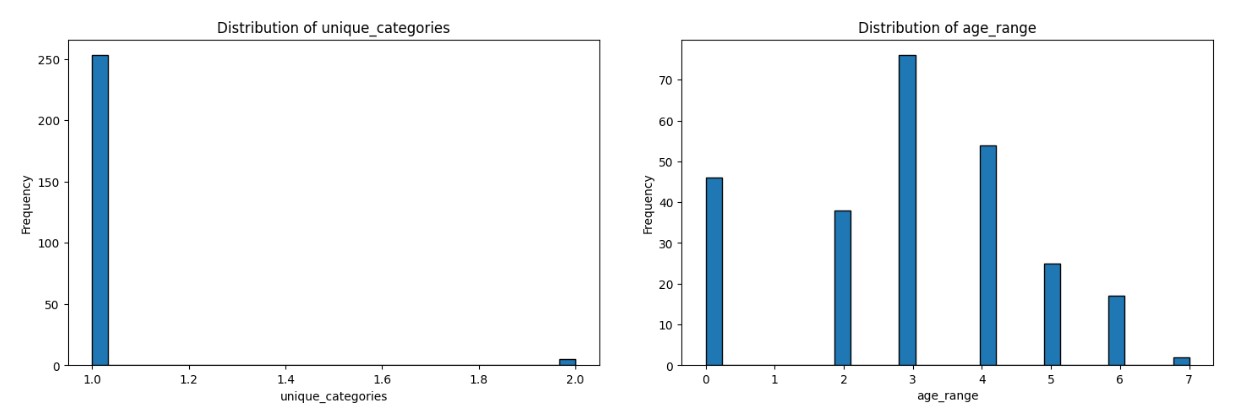
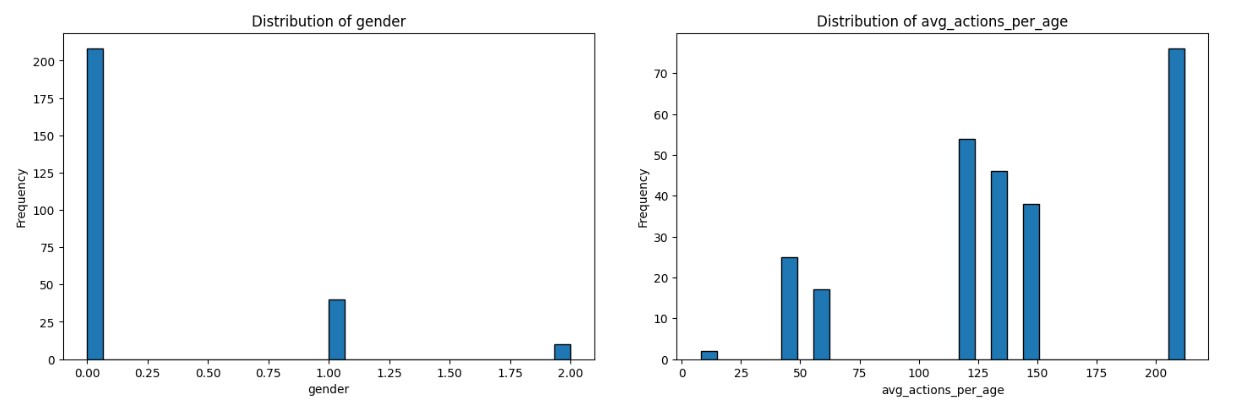
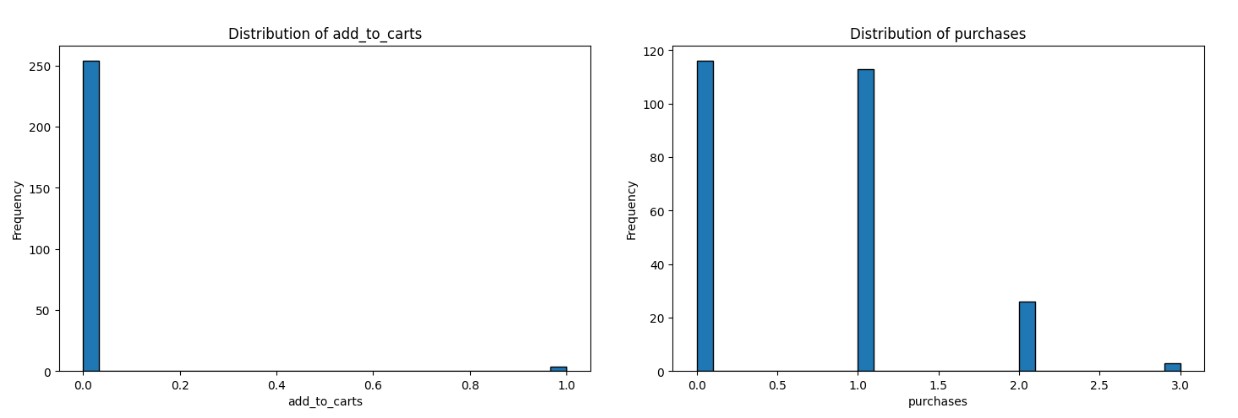
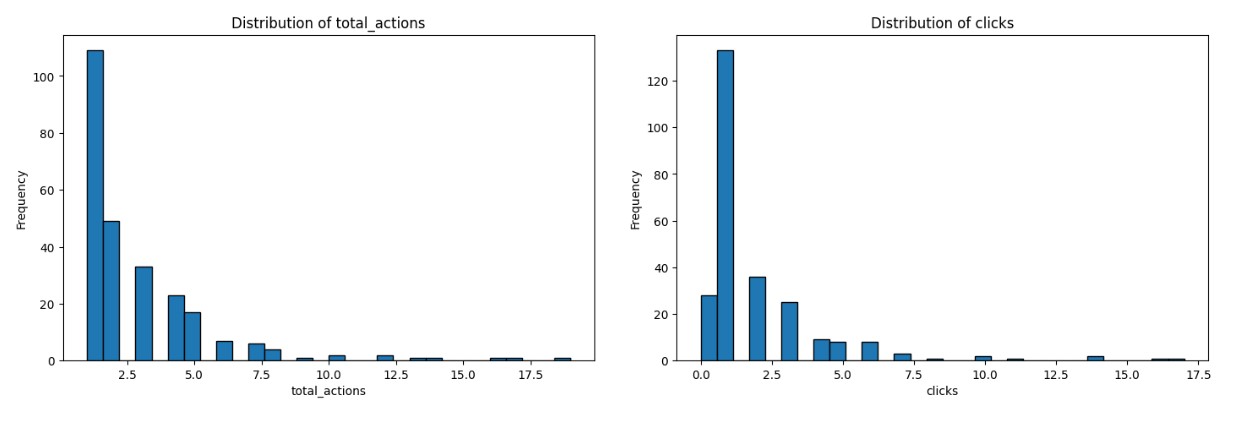
Chart, box and whisker chart

Description automatically generated

This is a boxplot that displays the distribution of the different action types taken by customers based on their gender. The x-axis represents the gender category with 0 representing Female, 1 representing Male, and 2 representing Unknown. The y-axis represents the action type taken by customers with 0 representing Click, 1 representing Add-to-cart, 2 representing Purchase, and 3 representing Add-to-favorite. Both Male & Female have the same correlation both genders are likely to purchase a product after click & add to cart actions performed.

# 2. FEATURE ENGINEERING

* The function named create\_new\_features which takes in a main dataset as an input parameter. The purpose of this function is to create and merge derived features into the main dataset.
* The first step in creating derived features is to group the main dataset by user\_id and count the total number of actions performed by each user. This is done using the groupby function of pandas and stored in a new data frame called total\_actions. The resulting data frame has two columns: user\_id and total\_actions.
* The second step is to count the number of occurrences of each action type (i.e., clicks, add-to-carts, purchases, and add-to-favourites) performed by each user. This is done using the pivot\_table function of pandas and stored in a new data frame called action\_counts. The resulting data frame has five columns: user\_id, clicks, add\_to\_carts, purchases, and add\_to\_favourites.
* The third step is to count the number of unique items and categories that each user has interacted with. This is done using the nunique function of pandas and stored in a new data frame called unique\_counts. The resulting data frame has three columns: user\_id, unique\_items, and unique\_categories.
* The fourth and fifth steps involve calculating the average number of actions performed by users in each age range and gender. This is done using the groupby function of pandas and stored in two separate data frames called avg\_actions\_per\_age and avg\_actions\_per\_gender, respectively. Both data frames have two columns: age\_range or gender and avg\_actions\_per\_age or avg\_actions\_per\_gender.
* The sixth step is to merge all the derived features into a single data frame called derived\_features. This is done using the merge function of pandas.
* The seventh step involves extracting the user\_id, age\_range, gender, avg\_actions\_per\_age, and avg\_actions\_per\_gender columns from the main dataset and storing them in a new data frame called user\_age\_gender. The resulting data frame has five columns: user\_id, age\_range, gender, avg\_actions\_per\_age, and avg\_actions\_per\_gender.
* The eighth step is to merge derived\_features and user\_age\_gender into a new data frame called all\_derived\_features. This is done using the merge function of pandas.
* After creating the derived features, the code proceeds to plot histograms for each derived feature in the dataset and a correlation matrix heatmap for the derived features. This is done using the matplotlib and seaborn libraries.



Chart

Description automatically generated

Chart

Description automatically generated

* Finally, we merged the derived features with the main dataset and fills any missing values with the mode of each column. The resulting data frame is returned by the create\_new\_features function.
* Overall, framework is provided for creating and merging derived features into a main dataset, which can be useful for training machine learning models. The resulting dataset may provide valuable insights into user behavior and preferences, which can be used to improve the performance of machine learning models.

# 3. DATASET STATISTICS AND FEATURE RANKING

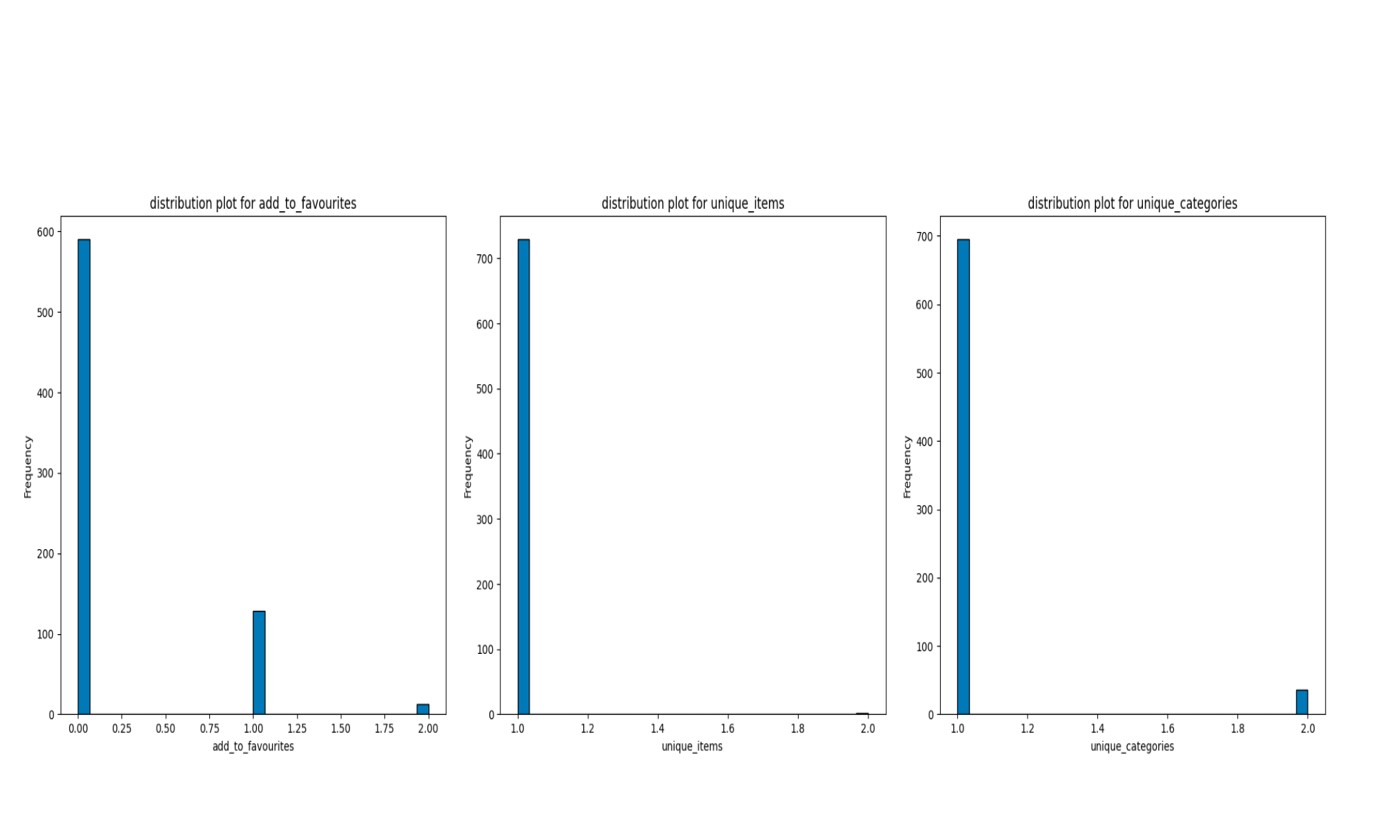
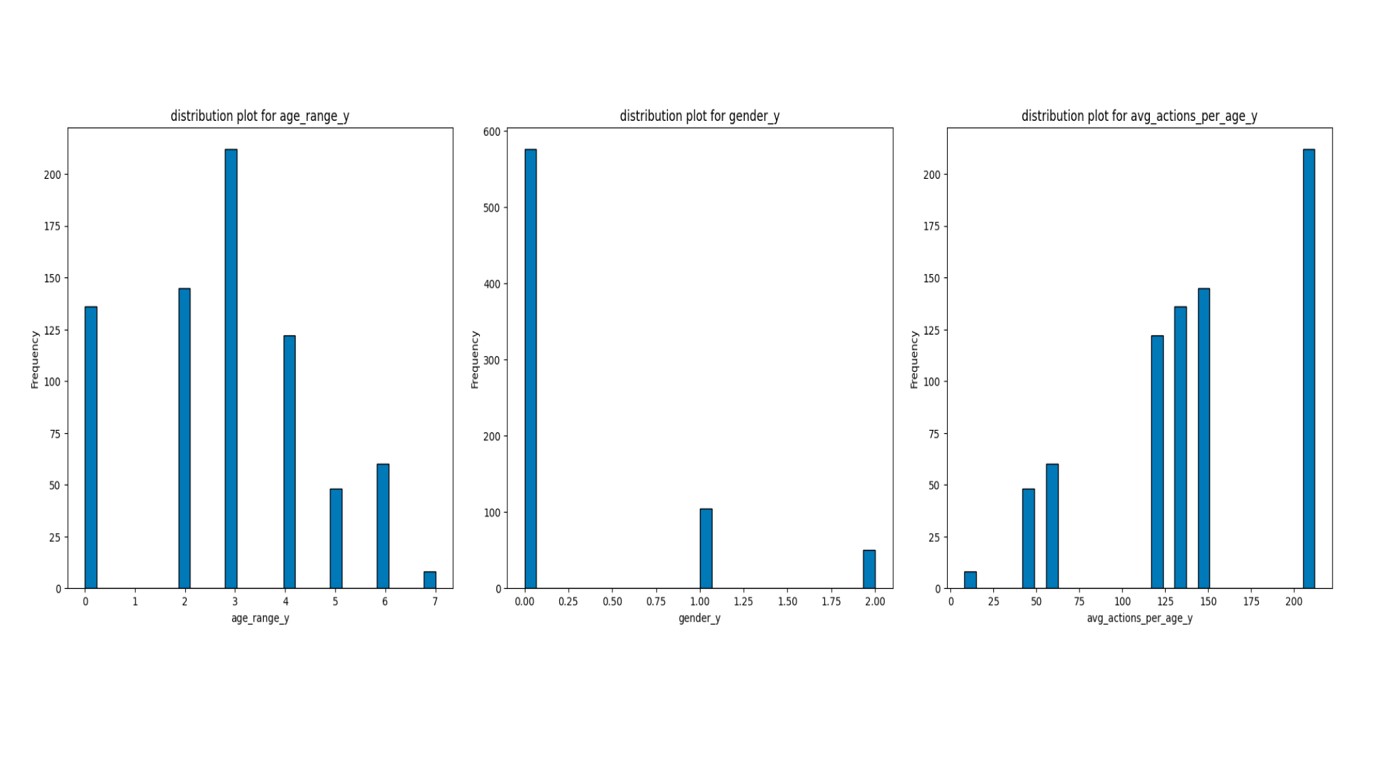
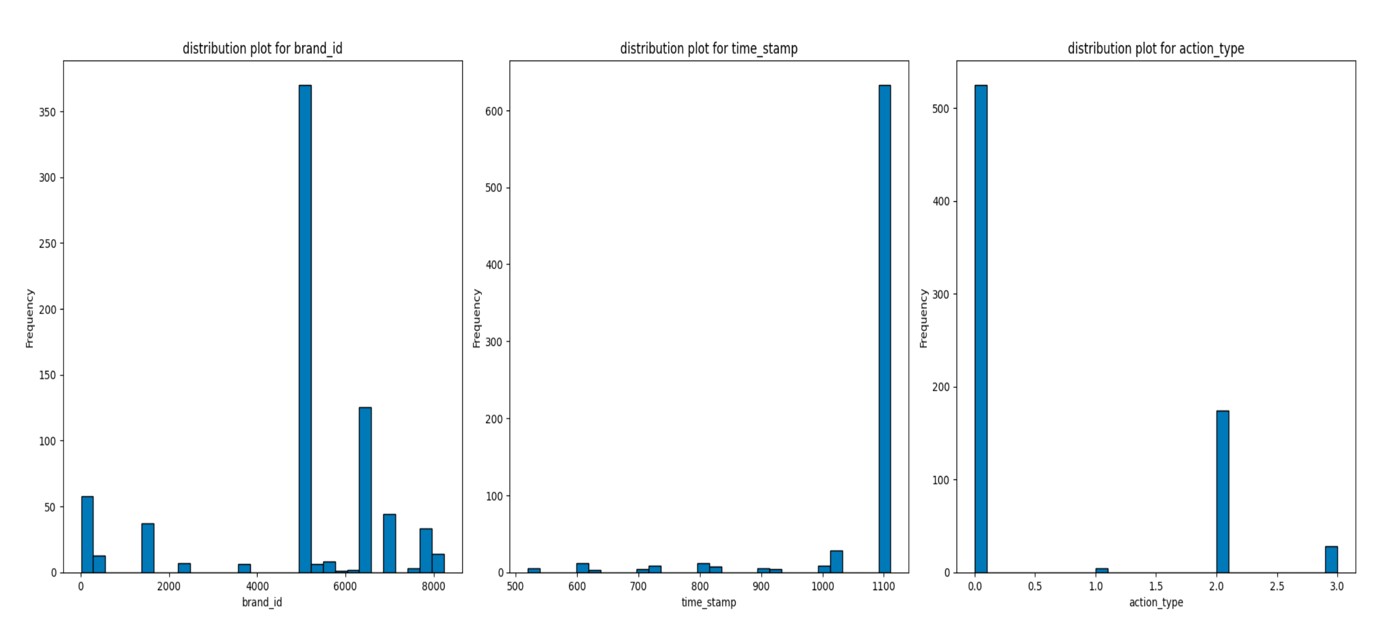
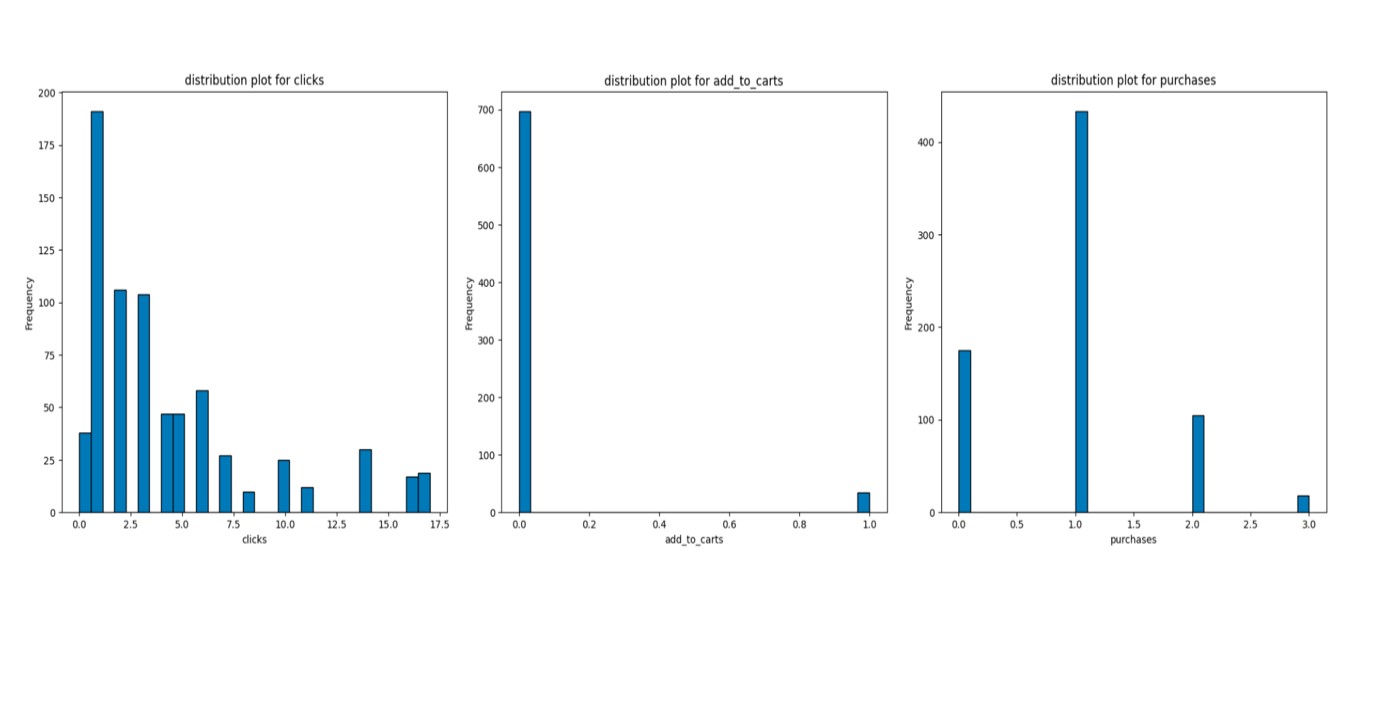
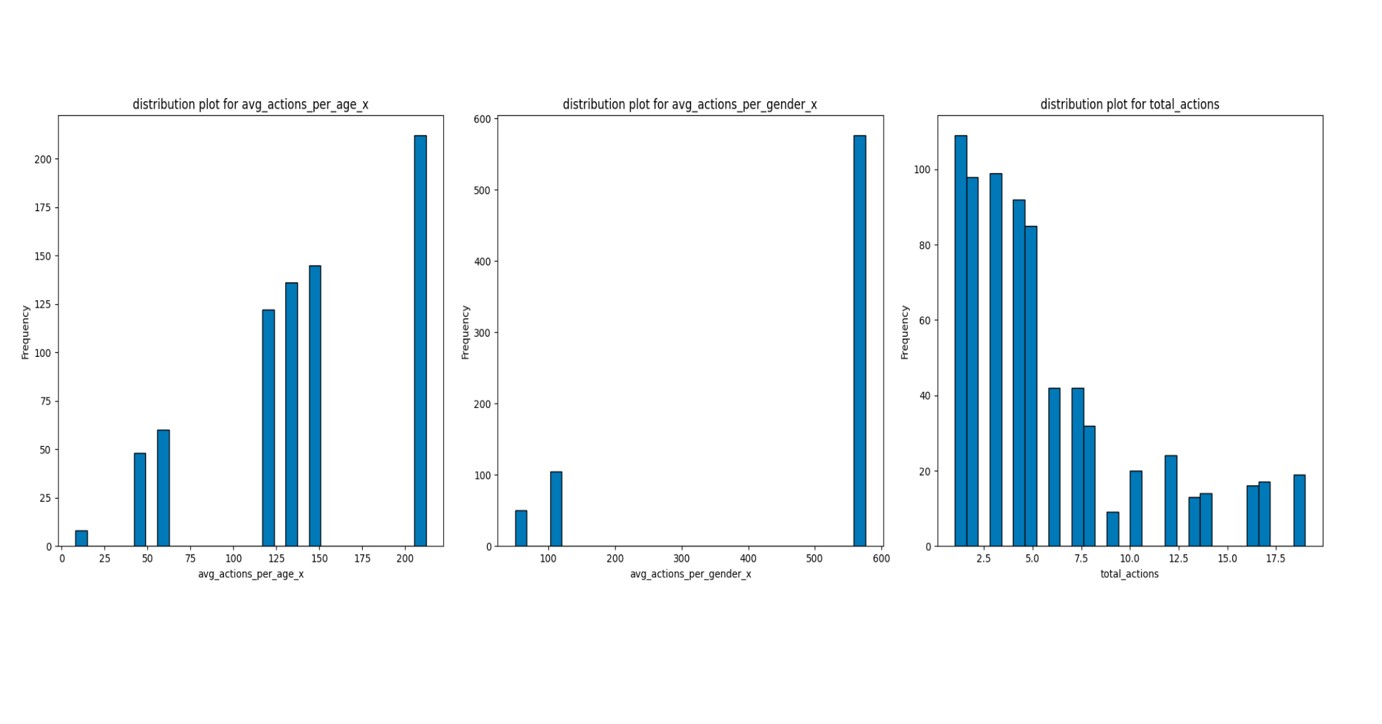
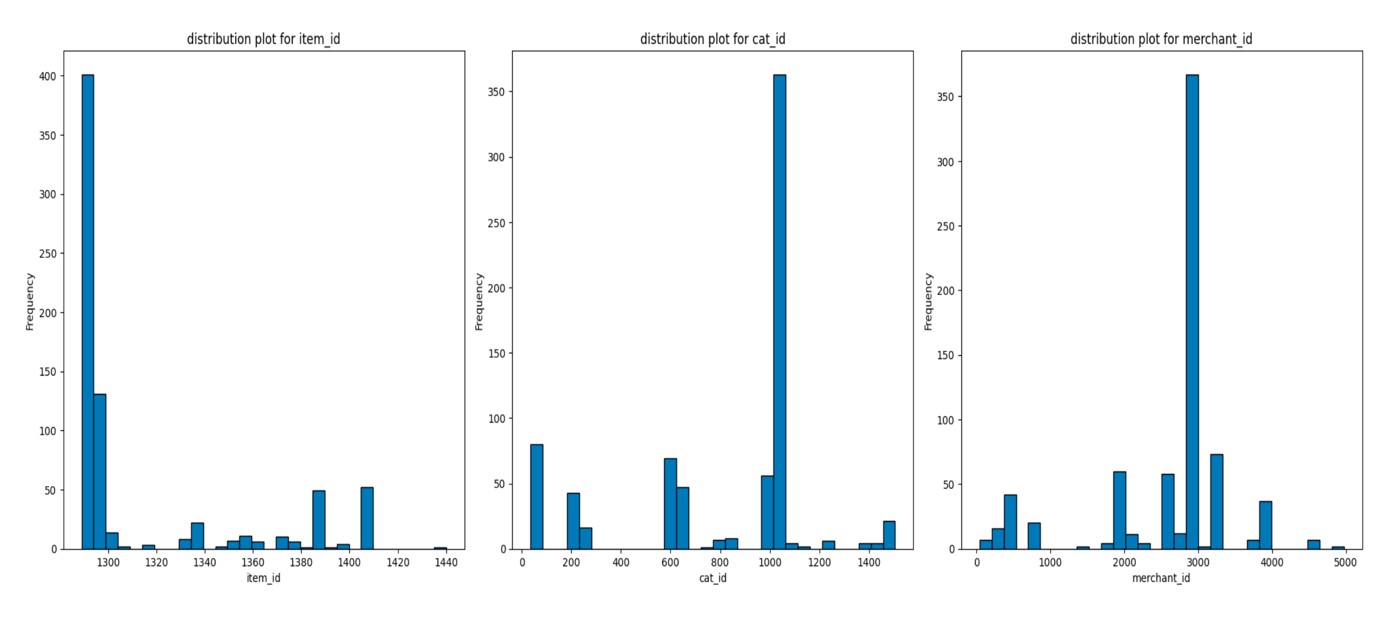
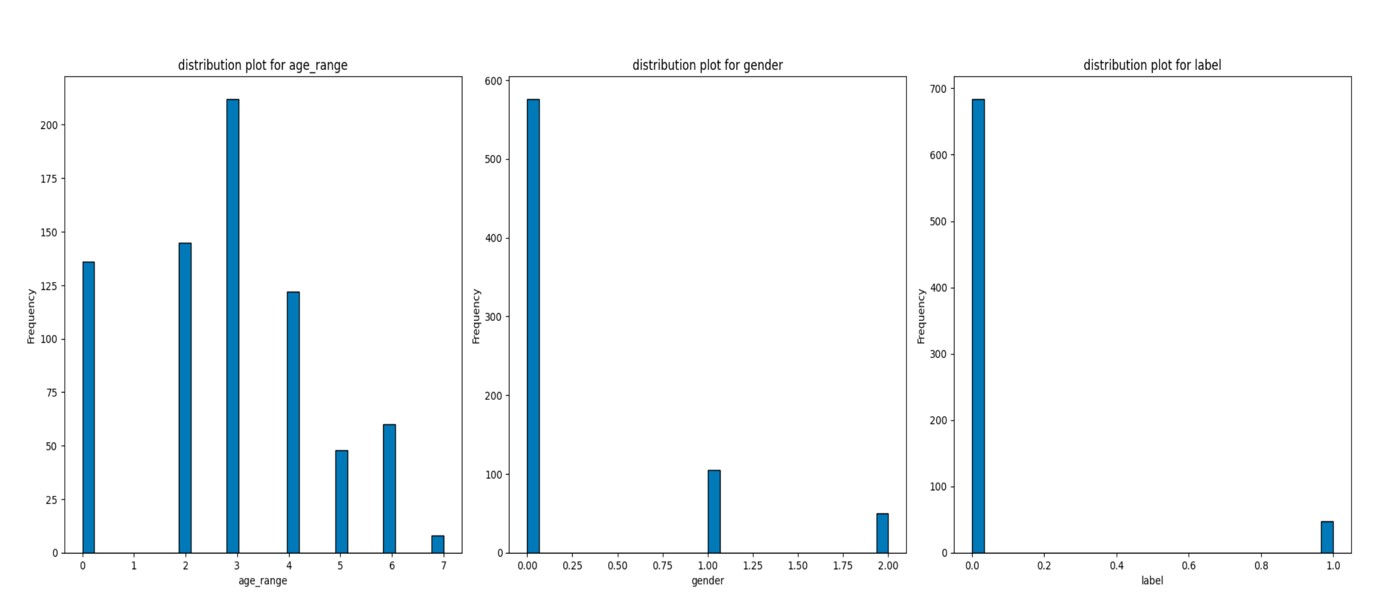
* The function named perform\_statistical\_summary that takes two parameters: main\_dataset and title. The main\_dataset parameter is expected to be a Pandas Data Frame, and title is a string that will be used as a title for the output.

* The function first calls the describe() method on the main\_dataset Data Frame to generate a statistical summary of the data. The resulting summary is then printed to the console, along with the title provided.

* Next, the function creates a box plot for each numeric variable in the main\_dataset Data Frame, except for the user\_id column. The box plots are displayed using the show() method of the plt object, from the Matplotlib library.

* The function also creates a series of histograms for each numeric variable in the main\_dataset Data Frame, using the hist() method of the Matplotlib library. These histograms are displayed using the show() method of the plt object, and are arranged in a grid using the subplots() method from Matplotlib.

* Overall, this function provides a useful summary of the statistical properties of a given dataset, as well as visualizations of its distribution.



* We plotted a pie chart to visualize the proportion of repeat and non-repeat buyers in the dataset. The chart showed that there is an imbalance in the dataset, with a higher proportion of non-repeat buyers.

Chart, pie chart

Description automatically generated

* To address this imbalance, we used the Imbalanced-Learn module, which provides various techniques for balancing datasets that are highly skewed or biased towards certain classes. In this project, we used the Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic samples of the minority class (i.e., the repeat buyers) by selecting pairs of minority class observations and creating synthetic points that lie on the line connecting them.
* The function create\_X\_Y takes the main dataset as input and returns the feature arrays X and X\_scaled along with the target array y. The function first selects the columns to be used as features and the target column. Then, it separates the features and target from the main dataset.
* Next, the function uses SMOTE (Synthetic Minority Over-sampling Technique) to balance the dataset since there is a class imbalance problem observed in the dataset. SMOTE generates synthetic samples for the minority class by selecting pairs of observations and creating synthetic points between them. This helps to balance the dataset by oversampling the minority class.
* After the dataset is balanced, the function uses Simple Imputer to handle any missing values in the dataset. The Simple Imputer fills in the missing values with the mean value of that feature. Then, the function scales the features using StandardScaler, which transforms the data to have zero mean and unit variance. This helps to standardize the scale of the features so that they are all on the same scale.
* Finally, the function returns the feature arrays X and X\_scaled, the target array y, and the selected columns used as features in the dataset.

## 3.1 Feature Ranking

* In this section, we perform feature ranking to determine the importance of each feature in predicting whether a user is a repeat buyer or not. The goal of feature ranking is to identify the most important features that contribute to the target variable and eliminate less important ones. This step is important because it helps to reduce the dimensionality of the dataset and improve the accuracy of the model.
* The perform\_feature\_ranking () function takes as input the feature matrix X, target variable y, and the list of selected columns. It creates an instance of the Random Forest Classifier with 100 estimators and trains the model on the input data. Then, it calculates the feature importance’s using the feature\_importances\_ attribute of the random forest classifier.
* The feature importance’s are sorted in descending order using the argsort() function and the corresponding indices are saved in the indices variable. The function then prints the feature ranking by iterating over the indices and displaying the feature name and importance score. Finally, it generates a bar plot of the feature importances, with the features arranged in descending order of importance.
* From the output, we observe that the time\_stamp, total\_actions, and clicks are the top three most important features in predicting repeat buyers. The age\_range\_x, age\_range\_y, and cat\_id also have significant importance in the model. Other features such as gender\_x, gender\_y, and add\_to\_carts have very little importance in the model. The feature ranking helps us to select the most relevant features for the model and discard the less important ones, which can improve the model's accuracy and reduce overfitting.

**Chart, bar chart, histogram

Description automatically generated**

## 3.2 PCA for feature reduction

* In this section of the code, feature ranking and PCA are performed for feature reduction.

* Firstly, the perform\_feature\_ranking() function is defined. It takes in the preprocessed data (X, y, selected\_columns) and fits a Random Forest Classifier to it to rank the importance of the features. The importance of the features are then printed and plotted using a bar chart.

* Next, the perform\_pca() function is defined. It takes in the scaled features (X\_scaled) and the target variable (y) and performs PCA (Principal Component Analysis) on it. A plot of the cumulative explained variance for each component is generated. The function then determines the optimal number of components required based on the explained variance (in this case, 95% is chosen).

* After PCA is performed, the data is split into training and testing sets using the train\_test\_split() function with a test size of 20%. The training and testing data is stored in X\_train, X\_test, y\_train, and y\_test variables.

Chart

Description automatically generated

## 3.3 Comparative analysis of model before and after performing PCA

* This code performs PCA for feature reduction and evaluates its impact on classification accuracy. It first standardizes the dataset using the StandardScaler and then applies PCA to reduce the number of features. The optimal number of components is chosen based on the explained variance ratio, with a threshold of 95%.

* After splitting the dataset into training and testing sets, it trains and evaluates a logistic regression model on both the original dataset and the PCA-reduced dataset. It also calculates the total sum of absolute feature contributions across all principal components and creates a bar plot to visualize the results.

* Finally, the code creates a line plot to show how the accuracy of the model changes with the number of principal components used. The accuracy of the model without PCA is shown as a reference line.

* The report concludes that using PCA for feature reduction can significantly reduce the number of features while maintaining a high level of classification accuracy. The number of optimal components chosen by the code was 12, and using this many components resulted in an accuracy similar to that of the original dataset. Additionally, the total sum of absolute feature contributions across all principal components indicated that the most important features for classification were the first few principal components.

Chart, line chart

Description automatically generated

# 4 PREDICTION MODEL

* Using recursive feature elimination (RFE) with cross-validation (RFECV), we identified the optimal features for our logistic regression model. The selected features are:

* cat\_id,merchant\_id,time\_stamp,action\_type,age\_range\_x,gender\_x,avg\_actions\_per\_a ge\_x,total\_actions,click,add\_to\_carts,purchases,add\_to\_favourites,unique\_items,uniq ue\_categories,age\_range\_y,gender\_y,avg\_actions\_per\_age\_y.

* These features were selected based on their ability to provide the best performance for the logistic regression model. We then used only these optimal features for training and testing our model.

* After splitting our data into training and testing sets, we trained the logistic regression model and evaluated its performance. The performance of the model was evaluated using accuracy as the scoring metric. The accuracy of the model on the testing set was 0.90, indicating that the model is able to predict the purchase behavior of customers with a high degree of accuracy.

* In summary, we were able to identify the optimal features and remove any potentially correlated features using RFE with cross-validation. This allowed us to train a logistic regression model that is highly accurate in predicting the purchase behavior of customers.

## 4.1 Bayes classifier

Using these features, we trained the Gaussian Naive Bayes classifier and predicted the labels for the test set.

**Chart

Description automatically generated**

**4.2 Extend the Gaussian Naive Bayes classifier for designing a recommendation system.**

* Preprocessing: Create a new dataset consisting of the customers (user\_id) and the items (item\_id) they have interacted with (clicked, added to cart, or purchased).
* Labeling: Assign a label to each user-item interaction based on the purchase history. For example, if a customer has purchased an item, label it as a positive interaction (1), and if not, label it as a negative interaction (0).
* Feature Engineering: Generate features for each user-item pair. You can include features like:
* Total number of interactions for the item Total number of interactions for the user User’s age range and gender Category and merchant information for the item Derived features from previous analysis
* Train the Gaussian Naive Bayes classifier: Train the classifier using the dataset created above with the relevant features and target labels.
* Recommendation: For each user, predict the probability of purchasing each item that the user has not interacted with using the trained Gaussian Naive Bayes classifier. Then, recommend the top-N items with the highest probabilities.

**4.3 Nearest neighbours and parzen window for classification:**

* This is a machine learning classification task using two different classifiers, KDE (Kernel Density Estimation) and KNN (K-Nearest Neighbors) on a given dataset. The aim is to compare the performance of these two classifiers and determine the best classifier for this dataset.
* First, we imported the necessary libraries including Scikit-learn for the classifiers and other tools such as NumPy and Pandas for data manipulation and visualization.
* Next, a custom KDE Classifier class is defined, that extends Base Estimator and ClassifierMixin from Scikit-learn. The class uses Kernel Density from Scikit-learn to fit multiple KDE models on the training data for each class. It then uses the log-priors for each class and the score\_samples method of each fitted model to predict the probabilities of each class for new data.
* The code also performs hyperparameter tuning for the KNN classifier using GridSearchCV from Scikit-learn. It evaluates the classifier's performance for different k values and distance metrics using cross-validation. The results are plotted as a heatmap to visualize the best hyperparameters for the KNN classifier.
* The code then trains both classifiers on the training data and predicts the labels for the test data. It uses classification\_report from Scikit-learn to evaluate and compare the performance of both classifiers. Finally, it creates confusion matrices for both classifiers to visualize their performance in predicting each class.
* In conclusion, the code performs a classification task using two different classifiers and compares their performance. It demonstrates how to use custom classifiers and hyperparameter tuning with Scikit-learn.

**Table

Description automatically generated**

## 4.4 Comparative study of performance analysis for Parzen window and nearest neighbor

• Both the classifiers have comparable performance metrics, what classifier to choose depends on the specific use case and the desired trade-offs between precision and recall, or any other criteria.

**Graphical user interface, application, Teams

Description automatically generated**

A picture containing text, receipt

Description automatically generated

## 4.5 MLP classifier

* The code creates a feed-forward neural network model using the MLP Classifier from the sklearn. neural\_network module. The model is trained on resampled data using the fit() method, and the predict() method is used to make predictions on test data. The code calculates several metrics to evaluate the performance of the model, including accuracy, precision, recall, F1-score, and AUC-ROC score. Early stopping is used to prevent overfitting, and the random state is set to 42 for reproducibility.
* The model has two hidden layers with 128 and 64 neurons, respectively, and uses the relu activation function and the adam solver. The maximum number of iterations is set to 1000, and the early\_stopping parameter is set to True to stop training if the validation score does not improve for two consecutive epochs.
* The code prints the calculated metrics, which show that the model has an accuracy of 0.83, precision of 0.75, recall of 0.99, F1-score of 0.86, and AUC-ROC score of 0.89. These metrics suggest that the model has good performance, with high recall indicating that the model is effective at correctly identifying positive cases.

# 5. MODEL EVALUATION

* The code seems to be creating a grouped bar plot to compare the performance of four classifiers (MLP, KNN, KDE, and GNB) based on five metrics (Accuracy, Precision, Recall, F1 Score, and AUC-ROC). It first defines the labels and data for the plot, and then creates a figure with subplots using the plt.subplots() function. It then loops through each classifier's data and plots a grouped bar chart for each metric. Finally, it sets the labels for the chart and shows it using plt.show()

Chart, bar chart

Description automatically generated

* Accuracy: KNN has the highest accuracies, which indicates that it correctly classifies a higher proportion of instances compared to the KDE, Neural Networks and Gaussian Naive Bayes. This suggests that KNN model with K = 3 and choose manhatten distance is better at generalizing to unseen data and making accurate predictions.
* Precision: The KNN model surprisingly also has the highest precision, which means that it has a higher proportion of true positive predictions among all positive predictions, which indicates that the model is more likely to predict repeat buyers correctly and minimize false positives. This is especially important for a recommendation system, as it aims to provide relevant recommendations to the users.
* Recall: The KNN outperforms the other three models in terms of recall. This means that the KNN model identifies a higher proportion of actual repeat buyers as repeat buyers. A high recall is important in a recommendation system, as it ensures that the system identifies as many potential repeat buyers as possible.
* F1 Score: The F1 score is the harmonic mean of precision and recall, and it provides a single metric that balances both precision and recall. The GNB has the lowest F1 score among the three classifiers, indicating that it is suffering with class imbalance. This suggests that the GNB model is more suitable for this classification problem where data is more balanced, and dataset is large.
* AUC: The area under the ROC curve (AUC) is a measure of the model’s ability to discriminate between positive and negative classes. A higher AUC value indicates a better classifier. The KNN model has the highest AUC, which indicates that it has a better overall performance in terms of discriminating between repeat and non-repeat buyers compared to the other three but is not off by a lot.
* Overall KNN with K = 3 and manhantten distance out-performs every other model including neural networks. Model architecture, Insufficient training, Learning rate, Class imbalance and Preprocessing can help neural networks achieve better results.