****

Machine Learning Algorithm Using Python

**Predicting Direct Debit for Allianz**

**Group Project**

Submitted by:

**Group Number – 7**

|  |  |
| --- | --- |
| **Name** | **ID** |
| Kajal Bhanot | 23A3HP603 |
| C. Sai Bharadwaja | 23A3HP640 |
| Sk Asif | 23A3HP666 |

Submitted to Prof. (Dr.) Surajit Ghosh Dastidar

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Item** | **Page No.** |
| 1 | Importing the File | 1 |
| 2 | Data Cleaning and Exploratory Data Analysis | 2 |
| 3 | Partitioning the Dataset | 9 |
| 4 | Model Building and Evaluation | 11 |
| 5 | Conclusion | 17 |

**Chapter 1. Importing the File**

Prior to opening and importing the file, the following libraries were imported into the python environment:

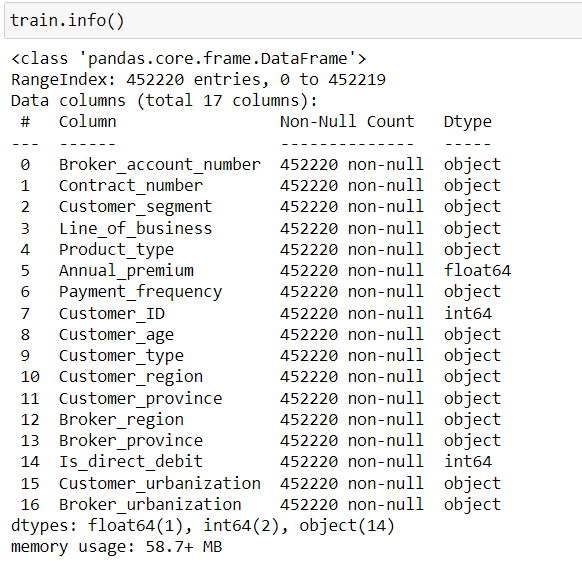
* **Pandas:** Used for data manipulation and analysis, it provides data structures like DataFrames, which allow for easy handling of tabular data.
* **Numpy:** It provides support for several scientific numerical computing methods in Python. It introduces powerful data structures, such as arrays, and functions for mathematical operations.
* **Matplotlib:** It is a plotting library that allows users to create static, interactive, and animated visualizations in Python. The pyplot module helps create bar charts, line charts, histograms, etc.
* **Seaborn:** Built on top of Matplotlib, provides a high-level interface for creating attractive and informative statistical graphics. It simplifies the process of creating complex visualizations, such as heatmaps and categorical plots, with less code.

The **%matplotlib inline** command displays Matplotlib plots directly within the Jupyter Notebook cells rather than a separate window.

**Importing the Dataset:** The data is in an Excel file with multiple sheets. The **pd.read\_excel()** function is used to read data from an Excel file (.xls or .xlsx) into a Pandas DataFrame. The data required for analysis in this case is present in a sheet name "AllianzDirectDebitData." The file can be stored either in system path (C drive) or a different folder. This data frame was named as “train.”



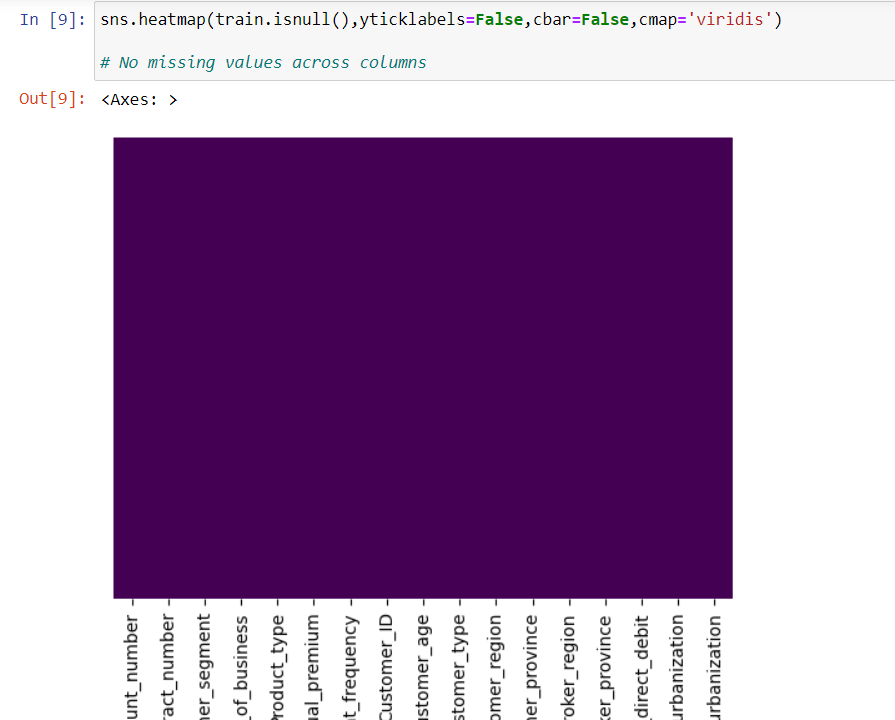
**Understanding the Imported Data:** Using the **.info()** function, the number of datapoints for each column and the type of data present in each column were identified. There were a total of 17 columns, 14 of which were of object data type, 2 were of integer data type, and 1 was of float data type.



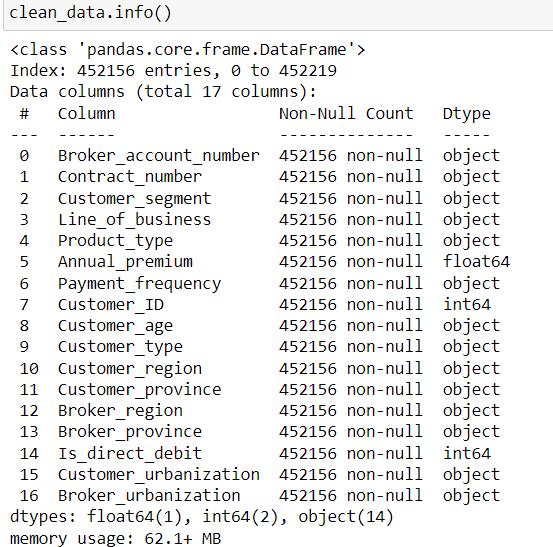
Using the **.head()** and **.tail()** functions, the first five and last five rows of data were observed in the data frame.

**Chapter 2. Data Cleaning and Exploratory Data Analysis**

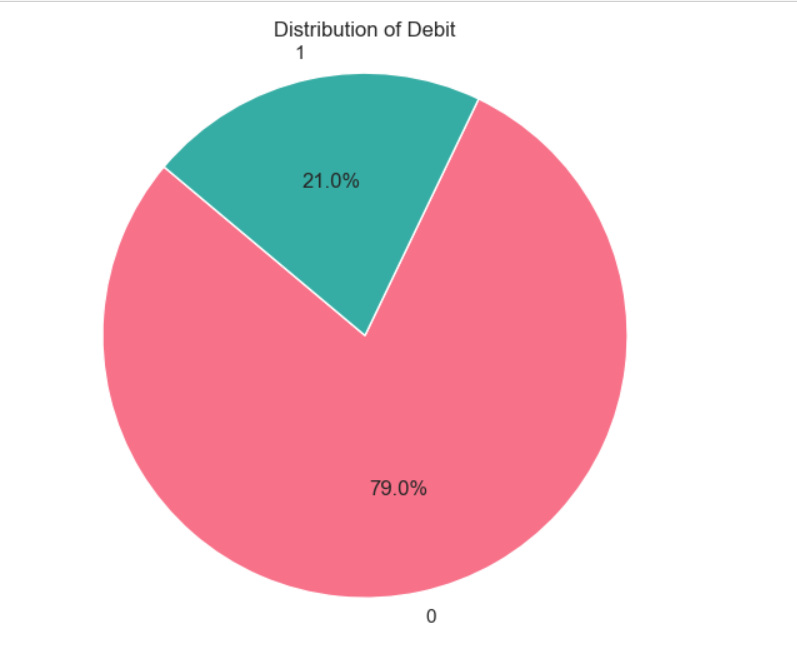
Before proceeding to exploratory data analysis, the dataset was checked for missing values and duplicate values. A quick heatmap analysis of datapoints for each column did not reveal any missing values.



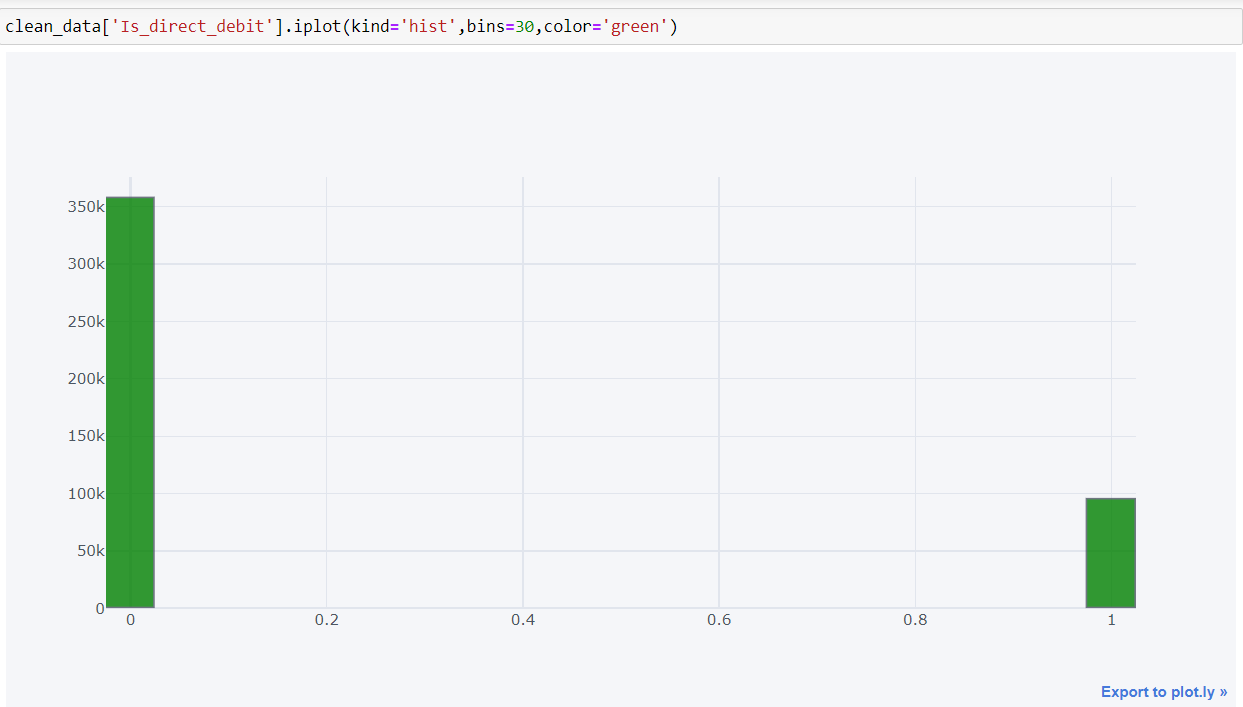
Using the **.drop\_duplicates()** function, rows with the same data for all columns were removed. Using **.dropna()** function, cells with NaN values were removed. Finally, the clean dataset had 4,52,156 datapoints per each column, as against the original dataset that had 4,52,220 datapoints per column.



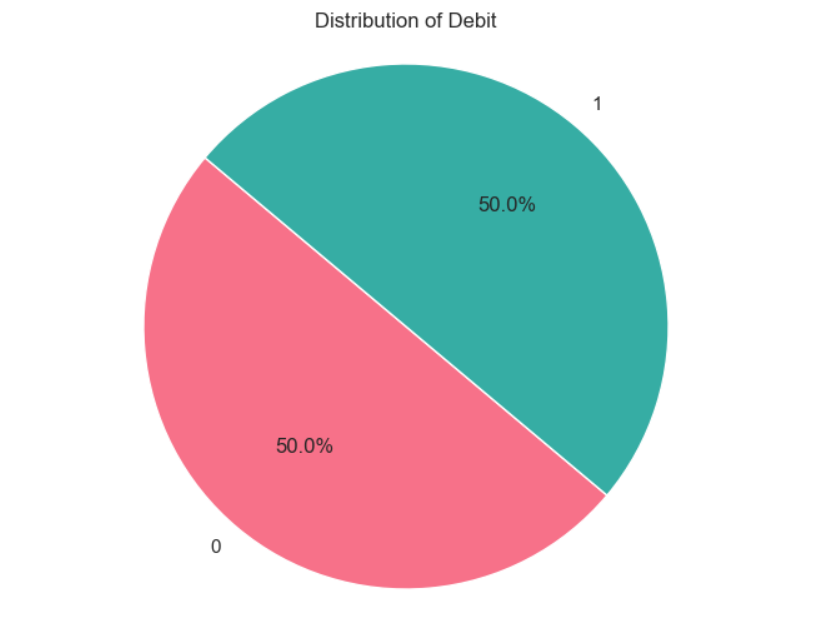
**Exploratory Data Analysis:** The variable of interest in this data is “**Is\_direct\_debit**”, which is a binomial categorical variable. This variable has an imbalanced distribution, with 79% of datapoints being 0 (no direct debit) and 21% being 1 (direct debit).



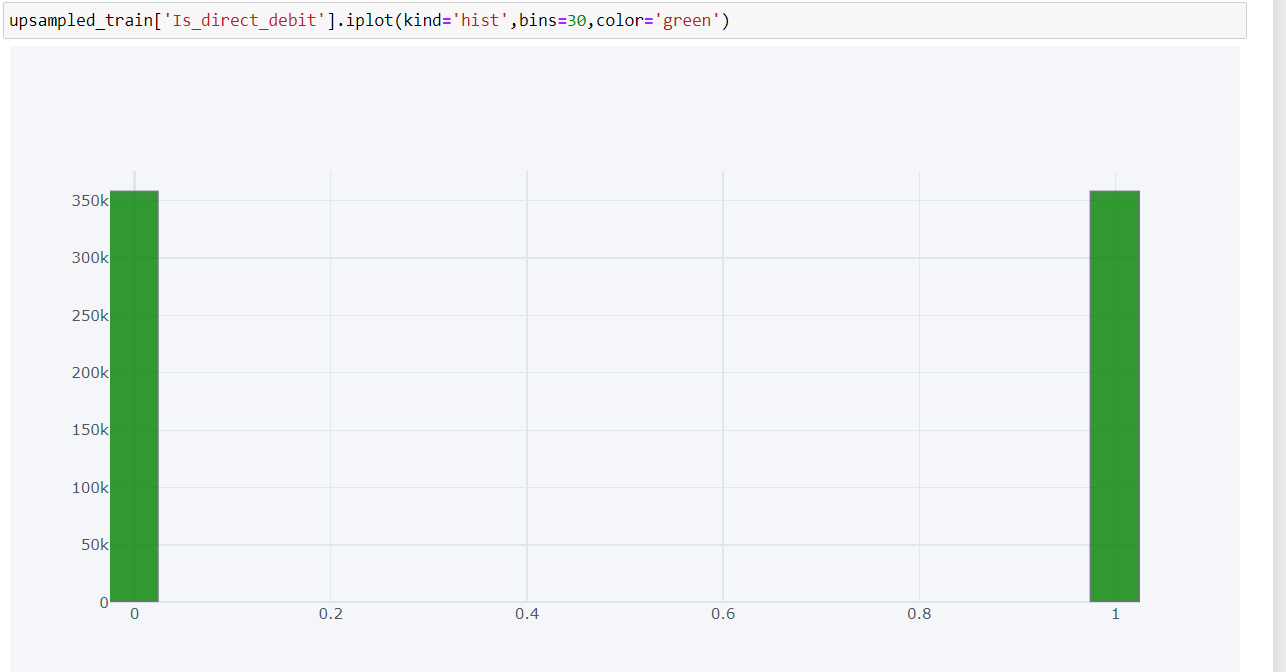
Through the **cufflinks** library, the number of datapoints for no direct debit and direct debit were identified to be 3,57,120 and 95,035, respectively.



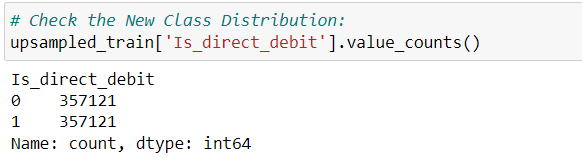
**Upsampling the Data:** Since the data is imbalanced, we decided to upsample the minority class data. Upsampling is a better alternative as compared to downsampling the majority class as it might lead to loss of key information from the data. After upsampling, the datpoints in both categories were balanced.



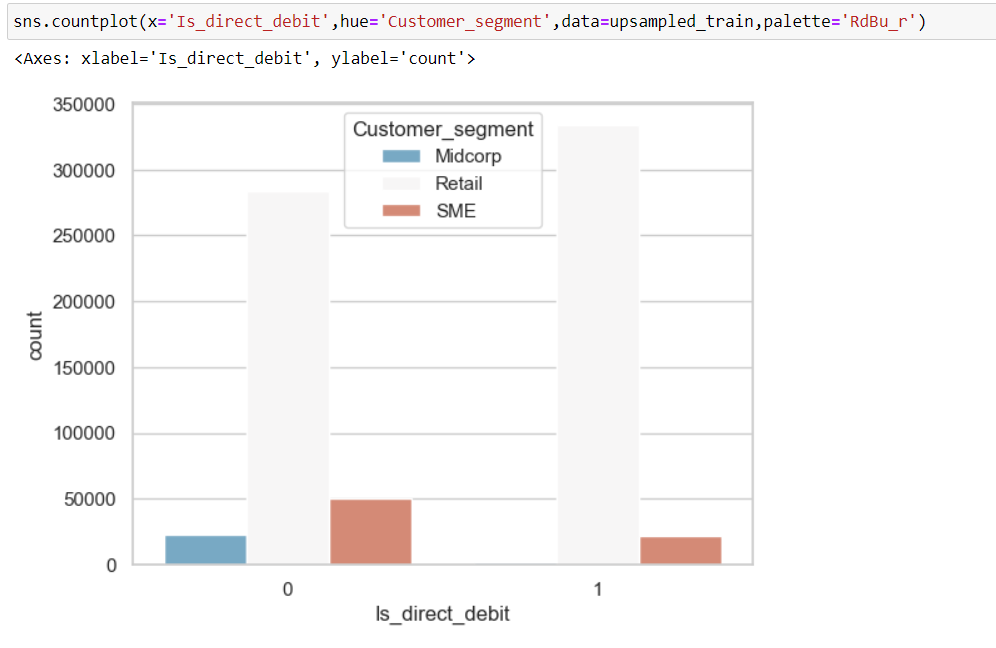
The same was verified through a histogram plot using the cufflinks library.



The final number of datapoints in both categories were as follows:



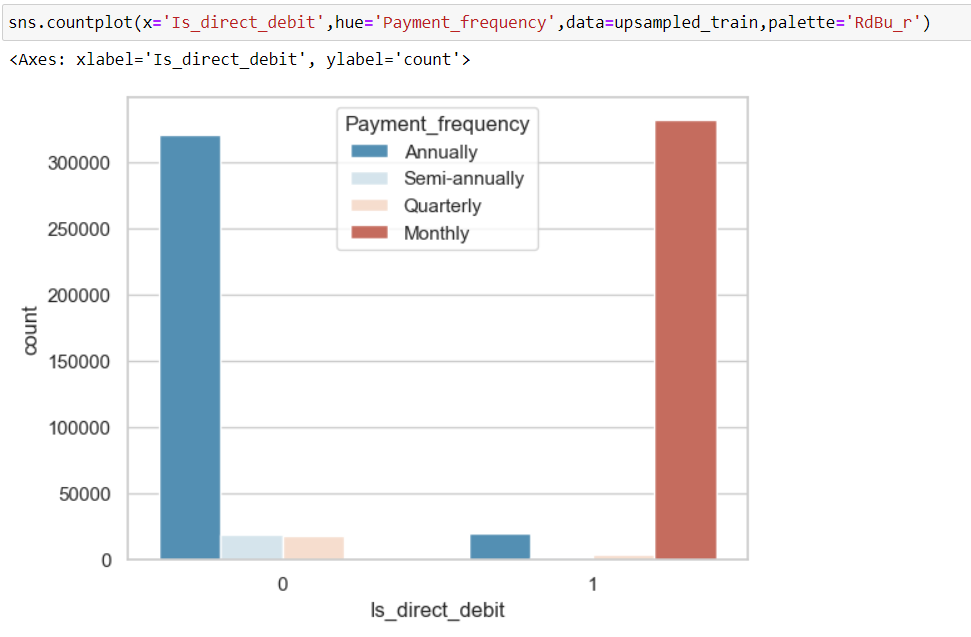
**Debit Distribution Based on Customer Segment:**



The retail customer segment leads the no direct debit section with 2,80,000 datapoints, followed by SME (5000 datapoints) and Midcorp (2500 datapoints).

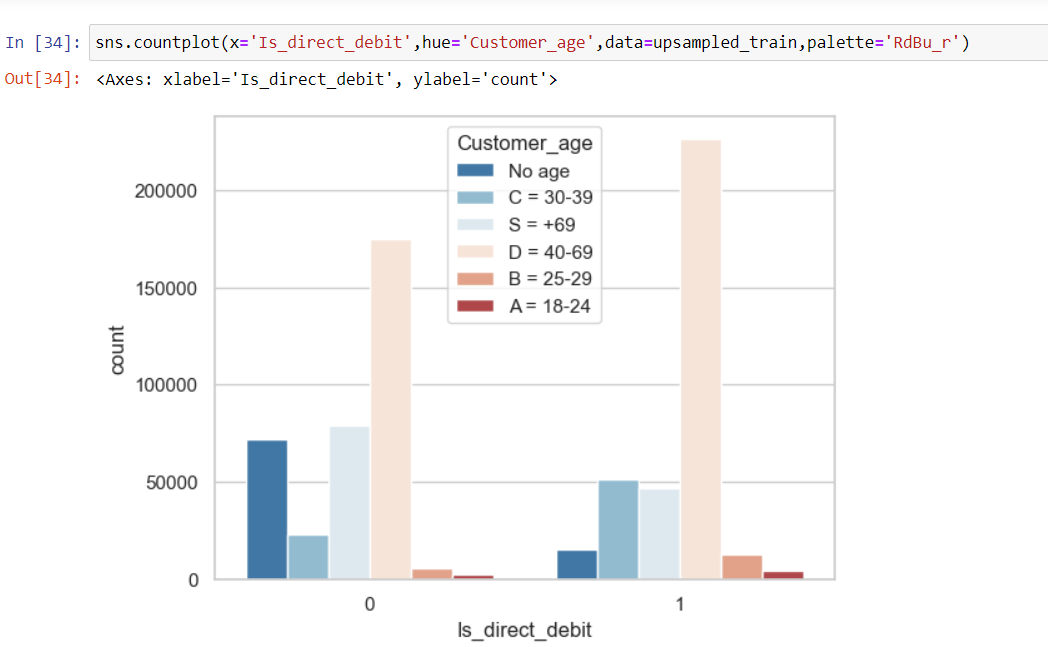
**Debit Distribution Based on Payment Frequency:**

Annual payments had the greatest number of customers with no direct debit (over 3,00,000).

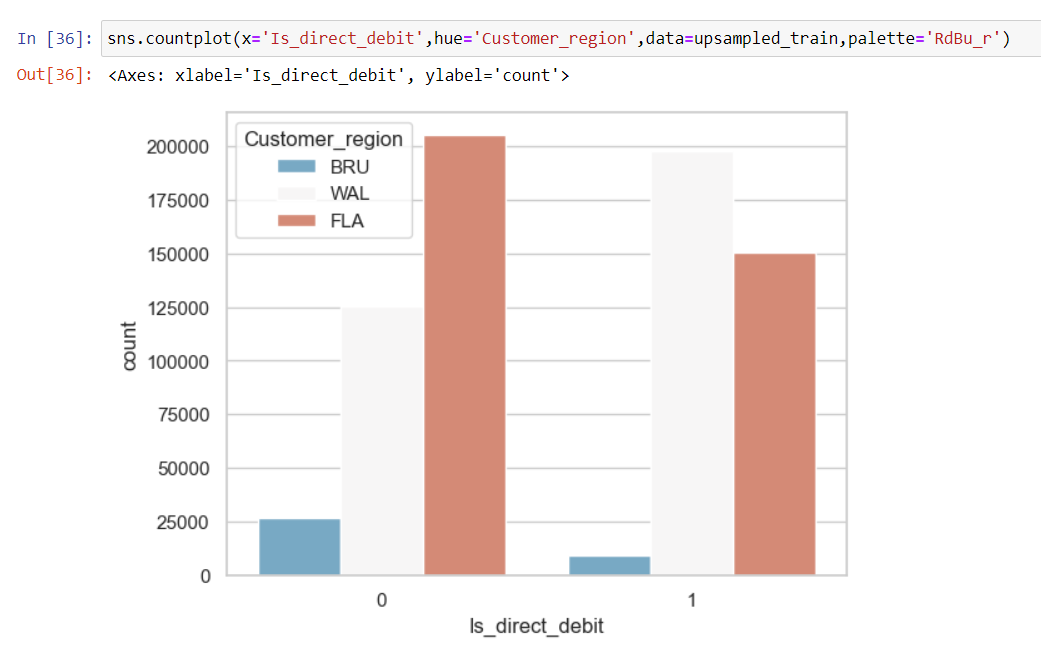


**Debit Distribution Based on Customer Age:**

Physical customers aged between 40-69 years had the highest instances of no direct debit, followed by those over 69 years and enterprises (No Age).

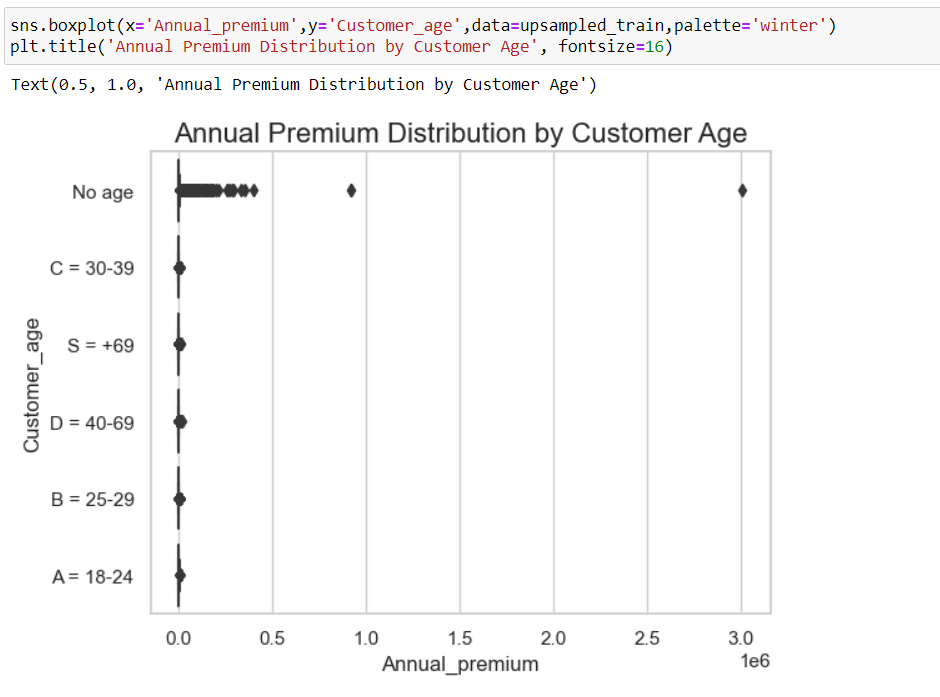


**Debit Distribution Based on Customer Region:**



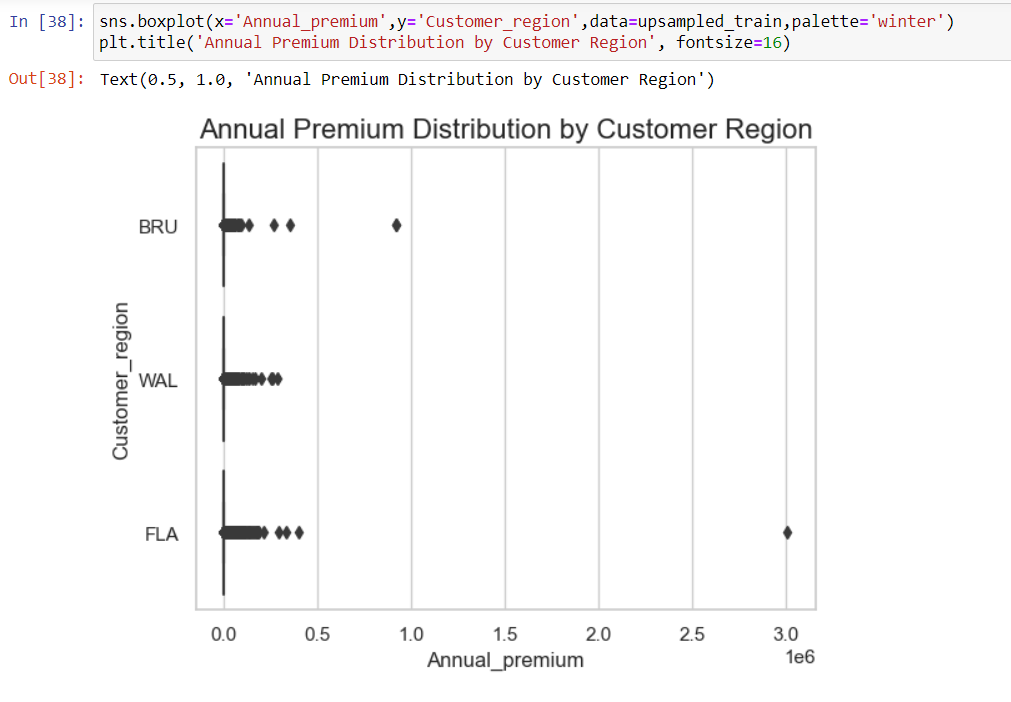
Most number of customers with no direct debit were from Flanders, followed by Wallonia, and Brussels.

**Annual Premium Distribution by Customer Age:**

****

The annual premium data is right-skewed for enterprises (No Age), indicating that outliers are on the positive side of the box plot.

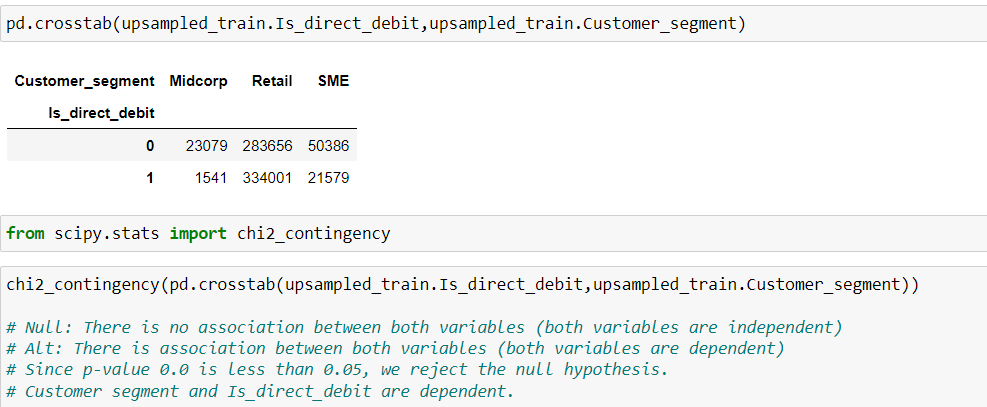
**Annual Premium Distribution by Customer Region:**



More outliers are present in annual premiums from Flanders and Brussels. Moreover, data from all 3 regions is right-skewed since all outliers are on the positive side of the box plot.

**Choosing the Independent Variables:**

At first, identifier variables like customer ID and Broker Account Number were dropped. Since a majority of the variables were nonnumerical categorical in nature, we opted for chi square test to test if there is association between “Is\_direct\_debit” and rest of the categorical variables.



Based on the results of chi square test, the following variables were shortlisted:

* Customer\_segment
* Line\_of\_business
* Payment\_frequency
* Customer\_age
* Customer\_type
* Customer\_region
* Broker\_region

Product\_type was dropped out of variables as it had over 61 categories all distributed unevenly across customer segments, making the model complex. Also, customer province, along with broker region and province, added more complexity to the data and were of less importance for model building. Customer and Broker urbanization were dropped as they were insignificant (p-value > 0.05) in association with “Is\_direct\_debit”. The final variables considered for modelling were as follows:

* Annual Premium
* Customer Segment
* Payment Frequency
* Customer Age
* Customer Type
* Customer Region

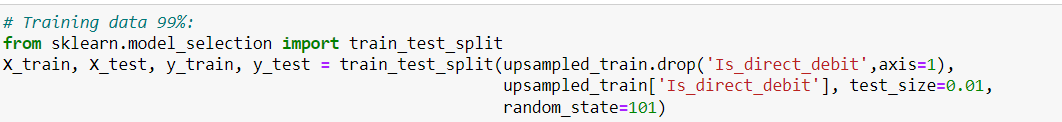
**Chapter 3. Partitioning the Dataset**

For the modelling activity, data was split in the following two ways:

* 99:1 train-test split: Control model used to compare if the trained model was overfitting
* 70:30 train-test split: Actual model used for predicting debit distribution

If the training and test partition results had a wider difference between recall, precision, and F1 scores, then the model is said to be overfitting. Otherwise, the model is a good fit for predicting the debit distribution

**Train-test split:**



**Importing the Function:** The code begins by importing the train\_test\_split function from the sklearn.model\_selection module. This function is a utility from the Scikit-learn library, which is widely used for machine learning in Python. It allows for easy splitting of datasets into training and testing subsets.

**Preparing the Data:**

* **upsampled\_train.drop('Is\_direct\_debit', axis=1)**:
  + This part of the code is used to create the feature set (independent variables) for training.
  + The drop method removes the column named 'Is\_direct\_debit' from the upsampled\_train DataFrame.
  + The axis=1 parameter indicates that a column is being dropped (as opposed to a row).
  + The resulting DataFrame contains all columns except for 'Is\_direct\_debit', which is assumed to be the target variable.
* **upsampled\_train['Is\_direct\_debit']**:
  + This part selects the target variable (dependent variable) for training, which is the column 'Is\_direct\_debit' from the upsampled\_train DataFrame. This column contains the labels or outcomes that the model will learn to predict.

**Splitting the Data**

* **train\_test\_split(...)**:
  + This function takes the feature set and the target variable as inputs and splits them into training and testing sets.
* **Parameters**:
  + **test\_size=0.01 (for comparison) and 0.30 (for prediction)**: This parameter specifies the proportion of the dataset to include in the test split. In this case, 1% of the data will be used for testing, while 99% will be used for training.
  + **random\_state=101**: This parameter is used to seed the random number generator for reproducibility. By setting a specific random state, you ensure that the results of the split will be the same each time you run the code, which is important for consistency in experiments.

**Output Variables**

* The function returns four variables:
  + **X\_train**: The training feature set (99% (70%) of the original features).
  + **X\_test**: The testing feature set (1% (30%) of the original features).
  + **y\_train**: The training target variable (99% (70%) of the original labels).
  + **y\_test**: The testing target variable (1% (30%) of the original labels).

**Chapter 4. Model Building and Evaluation**

**1. LOGISTIC REGRESSION**

A screenshot of a computer

Description automatically generated

**Understanding the Class Labels:**

Given the class labels:

* **1:** Direct Debit
* **0:** No Direct Debit

We can interpret the model's performance more specifically in terms of its ability to predict whether a customer will use Direct Debit.

**Interpreting the Results:**

* **High Precision for Class 1 (1.00):** The model is very accurate in identifying customers who will use Direct Debit. When it predicts a customer will use Direct Debit, it's highly likely that they will.
* **High Recall for Class 1 (0.93):** The model captures a significant portion of the actual Direct Debit customers. However, there might be some Direct Debit customers that the model misses.
* **F1-score for Class 0 (0.97):** This indicates that the model is effectively balancing precision and recall for the "No Direct Debit" class. It suggests that the model is accurate in identifying instances that are not Direct Debit customers and capturing a significant portion of the actual "No Direct Debit" cases.
* **F1-score for Class 1 (0.96):** This indicates that the model is also performing well for the "Direct Debit" class. It suggests that the model is accurate in identifying Direct Debit customers and capturing a significant portion of the actual "Direct Debit" cases.

**Overall Assessment:**

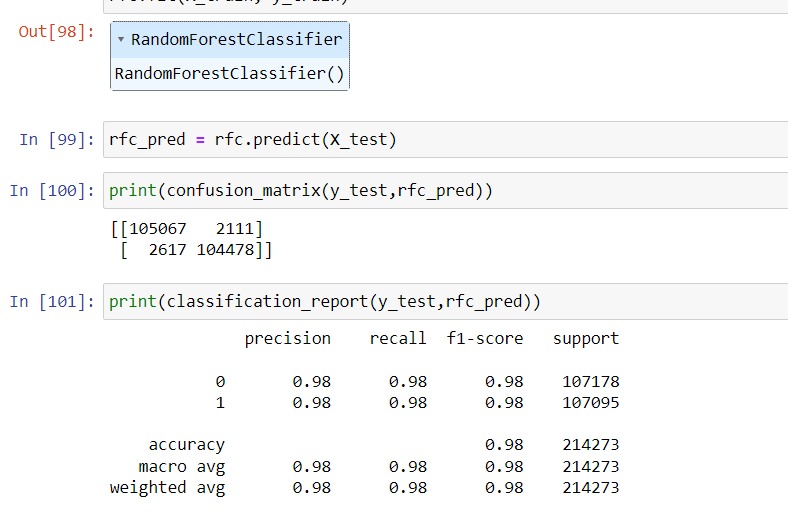
The model performs well in predicting whether a customer will use Direct Debit. It's particularly strong in identifying potential Direct Debit customers.

**Further Considerations:**

* **Cost-Benefit Analysis:** If the cost of missing a Direct Debit customer is high, it might be worth exploring ways to improve recall.
* **False Positive Rate:** While the model is good at identifying Direct Debit customers, it's also important to consider the rate of false positives (customers predicted to use Direct Debit but who don't). This could impact operational costs or customer satisfaction.

The logistic regression model is a suitable choice for predicting Direct Debit usage. Its high precision indicates its accuracy in identifying potential Direct Debit customers, and the relatively high recall suggests its effectiveness in capturing a significant portion of the actual Direct Debit customers.

**2. RANDOM FOREST**



**Analyzing the Model with Class Labels**

Given the class labels:

* **1:** Direct Debit
* **0:** No Direct Debit

We can interpret the model's performance more specifically in terms of its ability to predict whether a customer will use Direct Debit.

**Key Performance Indicators:**

* **Precision for Class 1:** This indicates the proportion of customers predicted to use Direct Debit who do use Direct Debit. A high precision value suggests that the model is accurate in identifying Direct Debit customers.
* **Recall for Class 1:** This indicates the proportion of actual Direct Debit customers whom the model correctly predicts. A high recall value suggests that the model is effective in capturing most of the Direct Debit customers.

**Interpreting the Results:**

* **High Precision for Class 1 (0.98):** The model is very accurate in identifying customers who will use Direct Debit. When it predicts a customer will use Direct Debit, it's highly likely that they will.
* **High Recall for Class 1 (0.98):** The model captures a significant portion of the actual Direct Debit customers. This suggests that the model is effective in identifying most of the customers who are likely to use Direct Debit.
* **F1-score for Class 1 (0.98):** This indicates that the model effectively balances precision and recall for predicting Direct Debit customers. It suggests that the model is both accurate in identifying Direct Debit customers and capturing a significant portion of the actual Direct Debit customers.

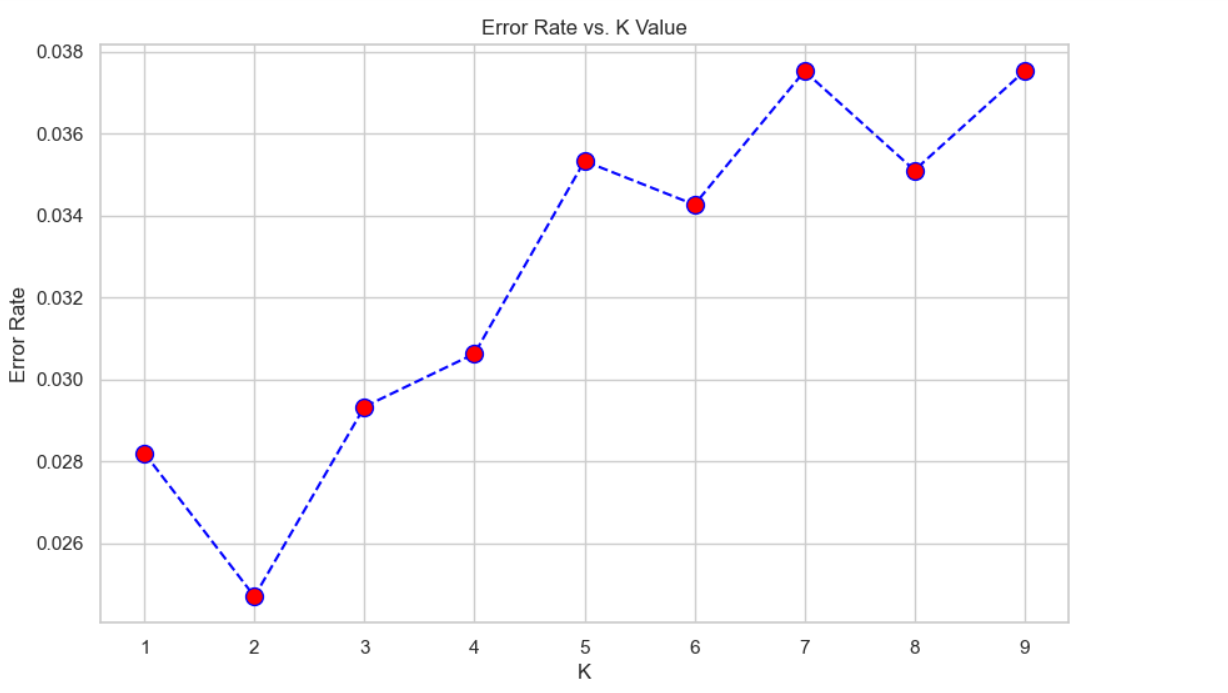
**Interpretations:**

* **Balanced Classes:** The support for both classes is relatively similar, indicating a balanced distribution of Direct Debit and No Direct Debit customers in the dataset. This makes it easier to interpret the model's performance without being influenced by class imbalance.
* **F1-Score and Balanced Classes:** The high F1-score for Class 1, combined with balanced classes, suggests that the model is performing well in accurately predicting Direct Debit customers, even considering the distribution of the classes in the dataset.

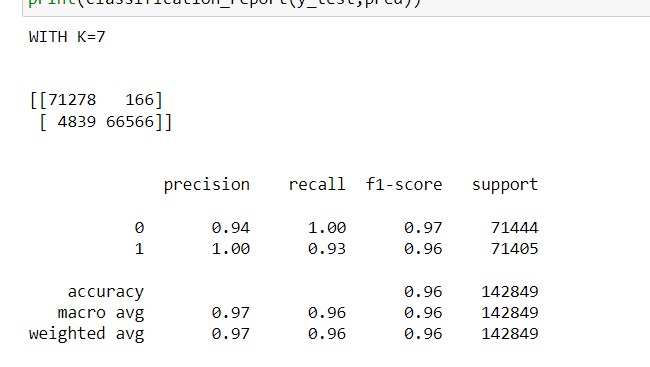
**Overall Assessment:**

The Random Forest Classifier performs **exceptionally well** in predicting whether a customer will use Direct Debit. Its high precision and recall values indicate that it is both accurate in identifying Direct Debit customers and effective in capturing most of the actual Direct Debit customers.The Random Forest Classifier is a suitable choice for predicting Direct Debit usage. Its strong performance across both precision and recall demonstrates its ability to effectively identify and predict Direct Debit customers.

**3. K-NEAREST NEIGHBOUR:**

****

Error rate has started to stabilize between 0.035 and 0.038 after K=7.



**Analyzing the Machine Learning Model**

**Interpretation:**

* **Precision:**
  + **Class 0:** The model is very accurate in identifying instances that are not Direct Debit customers. 94% of the instances predicted as "No Direct Debit" are indeed "No Direct Debit."
  + **Class 1:** The model is perfectly accurate in identifying Direct Debit customers. 100% of the instances predicted as "Direct Debit" are actually "Direct Debit."
* **Recall:**
  + **Class 0:** The model is capturing all instances of "No Direct Debit." 100% of the actual "No Direct Debit" cases are correctly predicted.
  + **Class 1:** The model is capturing a significant portion of "Direct Debit" cases, but there might be some "Direct Debit" instances that it's missing. 93% of the actual "Direct Debit" cases are correctly predicted.
* **F1-score:**
  + **Class 0 (0.97):** The model is effectively balancing precision and recall for the "No Direct Debit" class.
  + **Class 1 (0.96):** The model is also performing well for the "Direct Debit" class, but it might be missing some instances.

**Overall Assesment:**

The KNN model with K=7 is a suitable choice for this classification task, demonstrating strong performance across both classes. The high F1-scores for both classes and the balanced nature of the dataset indicate that the model is effective in capturing both Direct Debit and No Direct Debit customers accurately, providing a reliable prediction tool for this classification task.

**Chapter 5. Conclusion**

K-Nearest Neighbors (KNN) is a simple, yet powerful, supervised machine learning algorithm used for classification and regression tasks. The core idea behind KNN is to classify a data point based on the classes of its nearest neighbors in the feature space.

* **K**: The number of nearest neighbours to consider when making a classification decision. In our case, K is set to 7.
* **Distance Metric**: KNN typically uses distance metrics (like Euclidean distance) to determine the "closeness" of data points in the feature space.

**Feature Space:** Each customer is represented as a point in a multi-dimensional space, where each dimension corresponds to one of the independent variables (features).

**Finding Neighbors:** For a given customer (the customer of interest), KNN identifies the K nearest customers (neighbors) based on the specified distance metric.

**Voting Mechanism:** The algorithm then classifies the customer of interest based on the majority class among the K neighbors. If the majority of the nearest neighbors belong to the "direct debit" class, the customer is classified as likely to be in direct debit, and vice versa.

Choosing K=7 means that the algorithm will consider the 7 nearest neighbors when making a classification decision. The choice of K can significantly impact the model's performance. With the independent variables being Annual Premium, Customer Segment, Payment Frequency, Customer Age, Customer Type, and Customer Region, KNN at K=7 is a better model at classifying whether a customer will be in direct debit or no direct debit based on the similar features of the nearest 7 customers around the customer of interest.