Table of Contents

[SECTION-A 3](#_Toc147528471)

[Project Topic - 3](#_Toc147528472)

[Case Description- 3](#_Toc147528473)

[Data Preprocessing 3](#_Toc147528474)

[Case Formulation and Analysis: 4](#_Toc147528475)

[Variables Chosen: 4](#_Toc147528476)

[SECTION B & C 8](#_Toc147528477)

[EXPLORATORY DATA ANALYSIS 8](#_Toc147528478)

[SECTION D 26](#_Toc147528479)

[DATA CLEANING 26](#_Toc147528480)

[Feature Engineering/ Feature Encoding 31](#_Toc147528481)

[Label Encoding 36](#_Toc147528482)

[SCALING 39](#_Toc147528483)

[Section: E 43](#_Toc147528484)

[Conclusion 43](#_Toc147528485)

[Section-F 44](#_Toc147528486)

[Contribution 44](#_Toc147528487)

[Turnitin Report 45](#_Toc147528488)

# 

# SECTION-A

Project Topic - Understanding Customer Churn in a Telecom Company and building the most accurate model for the data frame.

Case Description- The telecom operator has gathered information from 51,048 checking customers in order to analyse and reduce customer turnover, a significant business concern. Customers quitting their telecom services are referred to as churn, and this may have a considerable effect on income and the viability of an organisation. With the help of this case study's analysis of the dataset and identification of churn-causing variables, the business will be able to put good retention measures into place.

Identification of the best model among the 4 models which are “statsmodel.api”, “naïve\_bayes”, “K-nearest neighbors”, and “Support Vector Machine” model.

Data Preprocessing:

* **Data cleaning:** To guarantee the quality and dependability of the data, the dataset was cleaned to remove problems like missing values, duplicates, and outliers.
* **Label Encoding**: To prepare the data for modelling, categorical variables like "Occupation," "PrizmCode," and "IncomeGroup" were encoded into numerical values.
* **Feature encoding**: To handle categorical variables that were not suited for label encoding, specialised encoding techniques, such as one-hot encoding or ordinal encoding, were used when needed.
* **Engineering of New Features**: To possibly enhance model performance, new features were developed. A "CallCompletionRate" feature, for instance, might be created by dividing "MonthlyMinutes" by "DroppedCalls."
* **Handling Missing Data**: Missing data in the dataset were handled logically and statistically, utilising advanced methods like predictive modelling or imputation with mean/median values to fill in missing values.
* **Feature Scaling**: Scaling of (monthly revenue and income group) and (age and retention offers accepted) was done with the help of min-max methodoly to bring them in the same magnitude for further comparison and understanding the correlation.
* **Model Building**: Model building in the dataset is done statistically, using the Logistic Regression and further more models such as stats.api, naïve bayes, K-nearest neighbors and Support Vector Machine were applied to check which model will give the least error.

## Case Formulation and Analysis:

The primary goal is to identify what causes customer churn and to generate actionable insights to decrease it and use of the best model to create automated systems that can make predictions, recognize patterns, and provide insights based from the dataset. The steps are as follows:

* **EDA (Exploratory Data Analysis):** EDA is used to visualise data distributions, discover patterns, and acquire preliminary insights into turnover rates across various consumer categories.
* **Correlation Analysis**: Look for relationships between data to find possible churn predictions. Examine whether "MonthlyRevenue" or "CustomerCareCalls" correspond with churn rates, for example.
* **Model Building**: Model building in the dataset is done statistically, using the Logistic Regression and further more models such as stats.api, naïve bayes, K-nearest neighbors and Support Vector Machine were applied to check which model will give the least error. The best model for the dataset comes out be Support Vector Machine because it is giving the least misclassified items if we compare it with other models.

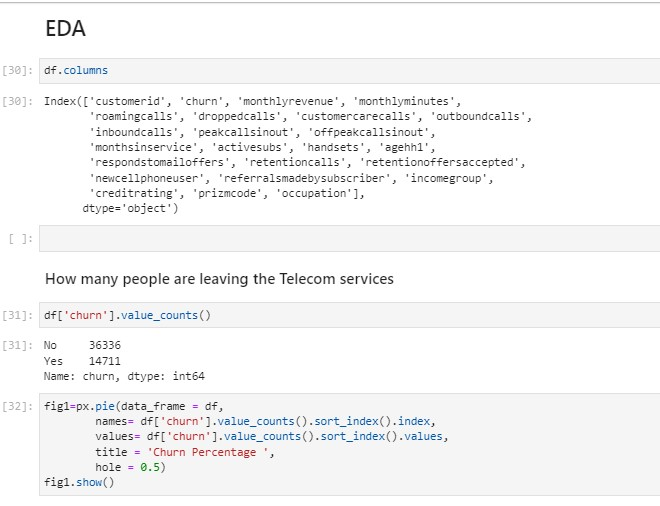
## Variables Chosen:

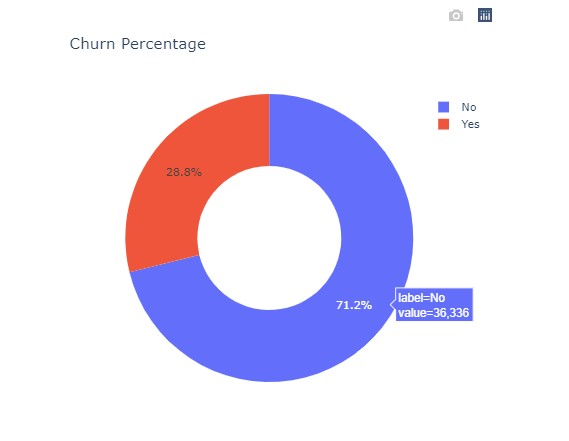
* **CustomerID:** To distinguish each client from other users in the database, a special identification number is given to them.
* **Churn:** The rate at which consumers stop using a telecom company's services is referred to as churn. It shows if a consumer has or has not cancelled their membership. A number of "1" often denotes client turnover, whereas "0" denotes an ongoing customer.
* **MonthlyRevenue:** This variable reflects the monthly revenue from telecom services that a consumer spends. Voice calls, data consumption, and any other services are all included in the price. A higher monthly income might mean that the consumer is using more services or has a premium plan subscription. Because they are more involved in the services, customers with more monthly income may be less inclined to churn.
* **MonthlyMinutes:** This figure indicates all the voice call minutes a client has used in a given month. Customers who make more phone calls per minute may be more engaged with telecom services. A better connection to the service may be shown by more engagement, thereby lowering the risk of churn.
* **RoamingCalls:** This variable reveals how many calls a consumer placed when travelling away from their home network. Customers who often use roaming services can have particular requirements or travel frequently. Understanding their use habits may be used to customise offers or services to their needs and lower turnover.
* **DroppedCalls:** This figure reflects the number of calls that were dropped or terminated before they were finished. Customers may find a large number of lost calls to be inconvenient and may be an indication of poor network performance or service problems. If the problem continues, customers who experience frequent lost calls are more likely to leave the company.
* **CustomerCareCalls:** This variable shows how many times a customer has called the customer service line to ask for help or support. Customers who contact customer service regularly could still have problems or concerns. A large volume of customer service calls may be a sign of unhappiness and raise the possibility of churn.
* **OutboundCalls:** The total number of calls made by the customer to other people. Customers who make more outbound calls may be more engaged with the service. Higher engagement indicates a closer relationship to the telecom operator, which may reduce churn.
* **InboundCalls:** The total number of calls received by the client from others is represented by this variable. Customers with a broader social or professional network may have more incoming calls. Strong social relationships can increase client loyalty and decrease turnover.
* **PeakCallsInOut:** This metric represents the number of calls made or received during peak hours, which are often the busiest hours of the day. The amount of calls made or received during peak hours might show a customer's dependency on the service during peak times. Customers that rely significantly on the service during peak hours may be less inclined to leave.
* **OffPeakCallsInOut**: This variable shows the amount of calls made or received during off-peak hours, which are often the quietest times of the day. The amount of calls made or received during off-peak hours, like peak calls, might reflect the customer's reliance on the service during less busy periods. Customers that routinely utilise the service during off-peak hours may be less prone to churn.
* **MonthsInService:** This is the number of months the consumer has used telecom services. Customer loyalty can be indicated by a longer tenure or a larger number of months in service. Customers who have been with the telecom provider for an extended period of time are less likely to churn.
* **ActiveSubs:** The number of active subscriptions or lines linked with the customer's account is represented by this variable. Customers that have many active subscriptions or lines may be more committed to their telecom provider. Having numerous current subscriptions might imply a better relationship, lowering churn.
* **Handsets:** This is the number of handsets or mobile devices that the consumer has. A customer's number of phones might reflect their level of reliance on telecom services. Customers that have numerous phones are more likely to be invested in the services and are less likely to churn.
* **AgeHH1:** The age of the head of the household to which the consumer belongs is represented by this variable. The age of the household's head might reveal information about the customer's life stage and prospective long-term commitment. Customers who are older or in a stable period of life may be less inclined to churn.
* **RespondsToMailOffers:** This field shows if the consumer has responded to mail offers or promotions from the telecom firm. Customers that respond favourably to postal offers or promotions are likely to be more involved with their telecom provider. Responding to offers can show increased interest and lower the chance of turnover.
* **RetentionCalls:** The number of calls made by the telecom firm to keep the customer's subscription. The frequency of retention calls made by the telecom firm to keep the client might reflect the customer's churn risk. Higher retention call numbers may indicate a higher churn risk.
* **RetentionOffersAccepted:** This metric indicates how many retention offers the consumer has accepted. Customers who accept retention offers are more likely to remain loyal to their telecom supplier. Accepting retention offerings shows a want to keep the connection going, which reduces turnover.
* **NewCellphoneUser:** This variable specifies whether or not the consumer is a first-time mobile user. New mobile customers may have switched carriers or entered the market recently. Understanding their needs and offering a great experience can help decrease churn early in the client journey.
* **ReferralsMadeBySubscriber:** This metric reflects the number of referrals made by the client in order to get new subscribers. Customers who recommend others are likely happy with the telecom services and are more likely to remain loyal. Referrals can reflect a great client experience, which can help reduce turnover.
* **IncomeGroup:** This variable classifies clients according to their income levels. Higher-income customers may have more discretionary income and be less price-sensitive. Customers with higher incomes may be less prone to churn for financial reasons.
* **CreditRating:** It represents the credit rating or score of the customer, indicating their creditworthiness. Customers with higher credit ratings may have a more stable financial situation and be less likely to churn due to financial constraints.
* **PrizmCode:** This variable classifies clients depending on their geographic region or way of life. Customers' preferences and wants might be revealed by categorising them based on their geographic region or lifestyle. Understanding these variables can assist in tailoring offers and services to prevent turnover.
* **Occupation:** It denotes the customer's occupation or profession. The customer's occupation might reveal information about their lifestyle and possible demands. By satisfying unique requirements, tailoring services or offers depending on profession might assist prevent turnover.

# SECTION B & C

## EXPLORATORY DATA ANALYSIS

1. How many people are leaving the telecom services?





**INTERPRETATION**:

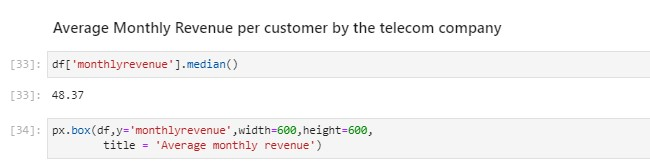
The code df['churn'].value\_counts() is used to count the unique values in the 'churn' column of a DataFrame (df). The output shows the result of this operation, indicating the counts of each unique value in the 'churn' column.Here's the interpretation of the output:

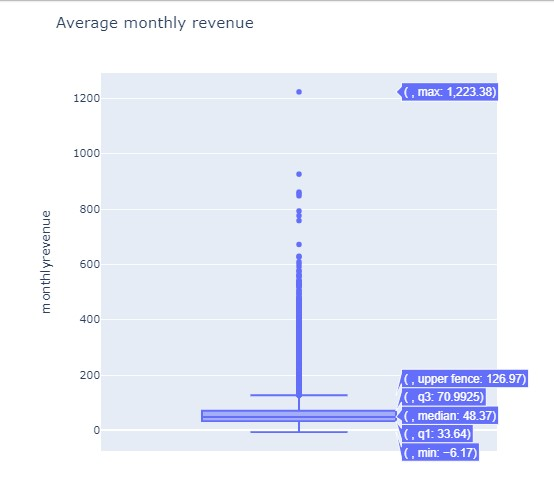
'No' appears 36,336 times in the 'churn' column.

'Yes' appears 14,711 times in the 'churn' column.

The code below generates a pie chart using Plotly Express (px) to visualize the percentage distribution of 'churn' values ('No' and 'Yes') in a DataFrame (df). The chart displays the proportion of 'No' and 'Yes' values, with 'No' being the larger segment. The title is 'Churn Percentage,' and there's a central hole in the pie chart.

1. Average Monthly revenue per customer by the telecom company.



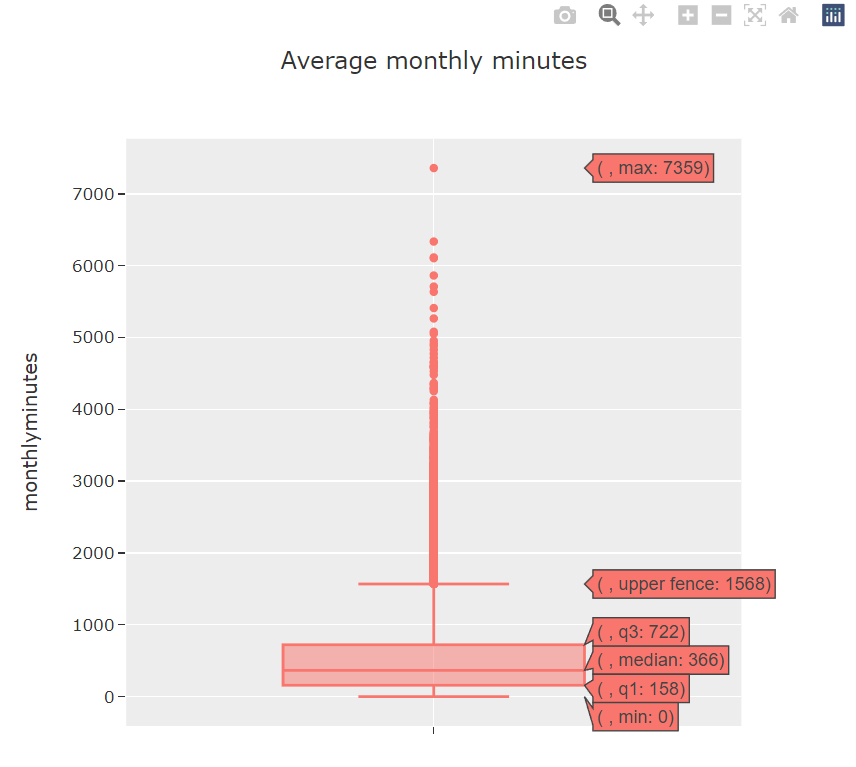


**INTERPRETATION:**

The average monthly revenue per customer is approximately 58.80. This metric indicates the typical monthly spending of customers, which can be valuable for pricing strategies and revenue forecasting in the telecom industry.The "monthlyrevenue" variable has a median (50th percentile) value of 48.37. The data distribution spans from a minimum of - 6.17 (which may indicate errors or outliers) to a maximum of 1223.38. Additionally, the interquartile range (IQR) shows that 25% of customers have a monthly revenue below 33.64, while 25% have revenue above 70.99, indicating variability in customer spending habits.

1. Average monthly minutes per customer

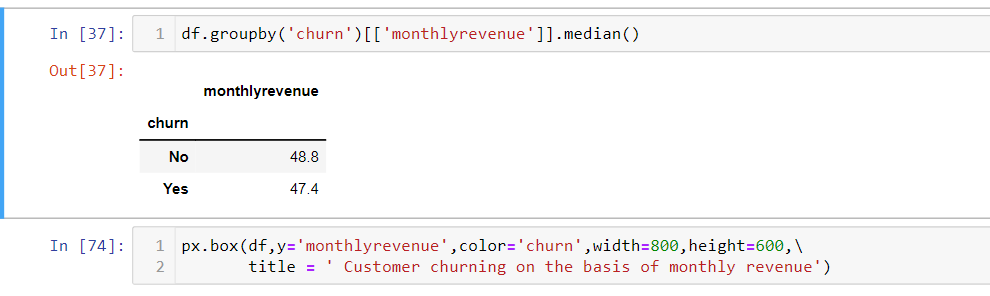


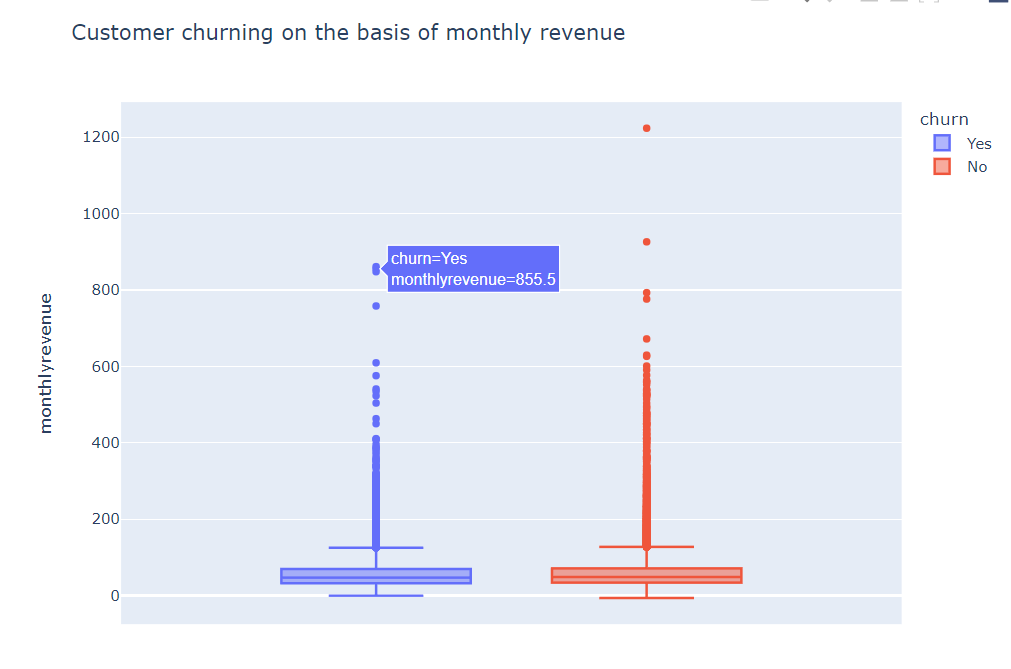


**INTERPRETATION:**

The median monthly minutes used by customers in the dataset is 366 minutes, indicating that half of the customers use fewer than 366 minutes, while the other half use more. The data ranges from 0 minutes (possibly indicating inactive users) to a maximum of 7359 minutes. The interquartile range (IQR) shows that 25% of customers use less than 158 minutes, while 25% use more than 722 minutes, illustrating a wide range of usage patterns.

1. Customer churning on the basis of monthly revenue

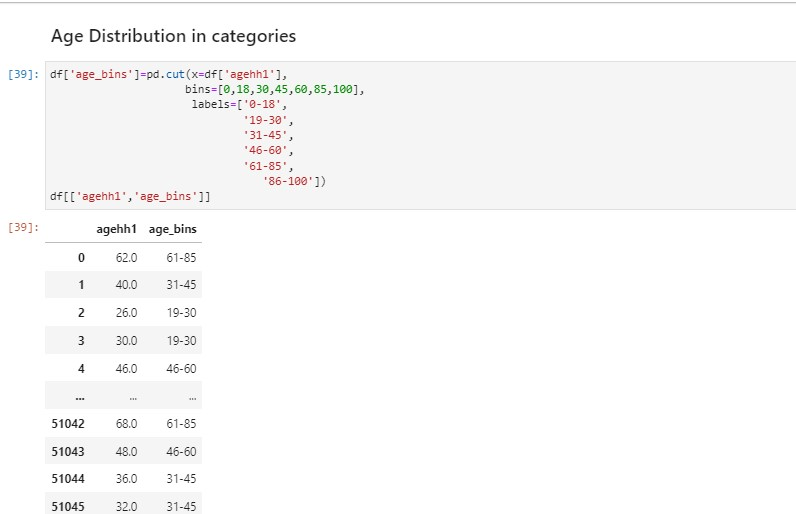




**INTERPRETATION:**

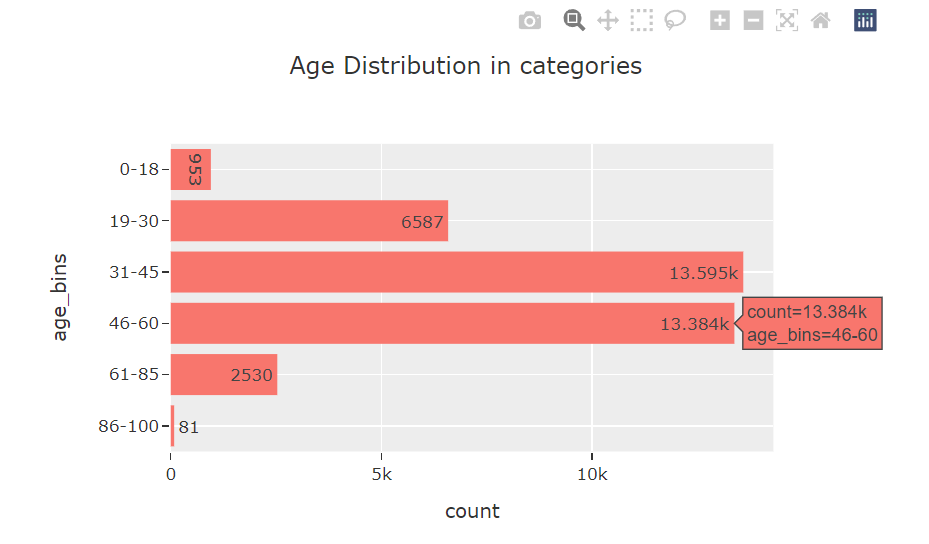
The analysis of customer churn based on monthly revenue reveals that the median monthly revenue for customers who haven't churned ("No") is slightly higher at 48.8, compared to $47.4 for customers who have churned ("Yes"). While the difference is relatively small, it suggests that on average, customers who continue their subscriptions tend to have slightly higher monthly spending compared to those who churn. This insight could be valuable for the telecom company in tailoring retention strategies for different customer segments based on their spending patterns to reduce churn rates.

1. Age Distribution in categories



A screenshot of a computer

Description automatically generated

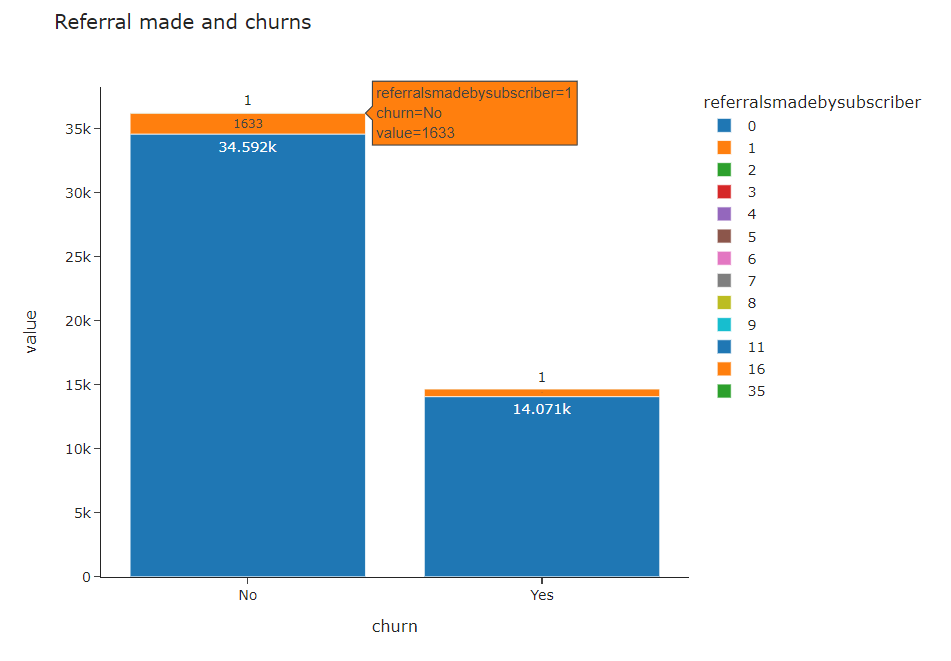


**INTERPRETATION:**

The provided code segments create a new categorical feature "age\_bins" based on the "agehh1" column, which likely represents the age of customers in a dataset. It bins ages into six categories: 0-18, 19-30, 31-45, 46-60, 61-85, and 86-100. The resulting "age\_bins" feature is then used to count the number of customers falling into each age group. The output shows the count of customers in each age group, revealing that the majority fall into the 31-45 and 46-60 age ranges, with fewer customers in the other categories. This helps segment customers by age for further analysis or targeted strategies

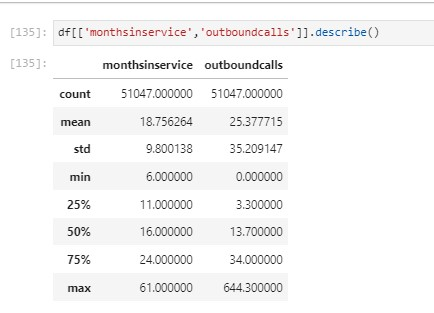
1. Referral made and churns

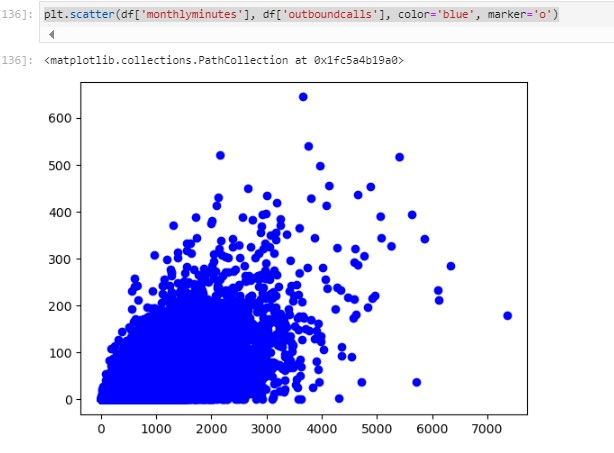


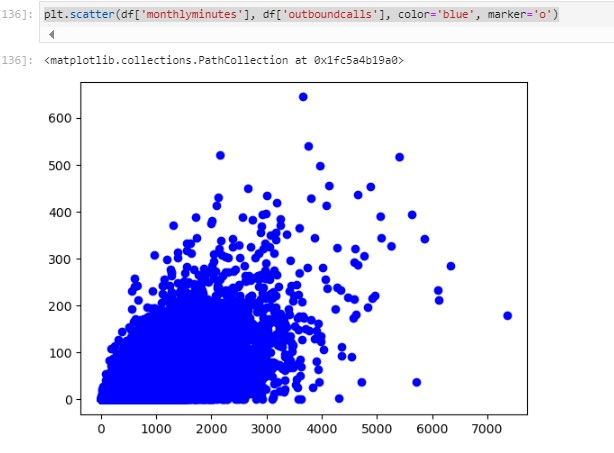


**INTERPRETATION:** The provided code segments analyze the relationship between customer churn and the number of referrals made by subscribers. It groups customers by their churn status (churned or not) and the number of referrals they made. The output, displayed in a bar chart, shows the count of customers falling into each category. For instance, among customers who did not churn ("No"), the majority made 0 referrals, while among churned customers ("Yes"), a significant number also made 0 referrals. This analysis suggests that the majority of customers, both churned and non-churned, have not made referrals, and there's a noticeable drop in the number of customers as the referral count increases. This insight can inform referral programs and customer engagement strategies.

1. Monthly minutes and Outbound calls



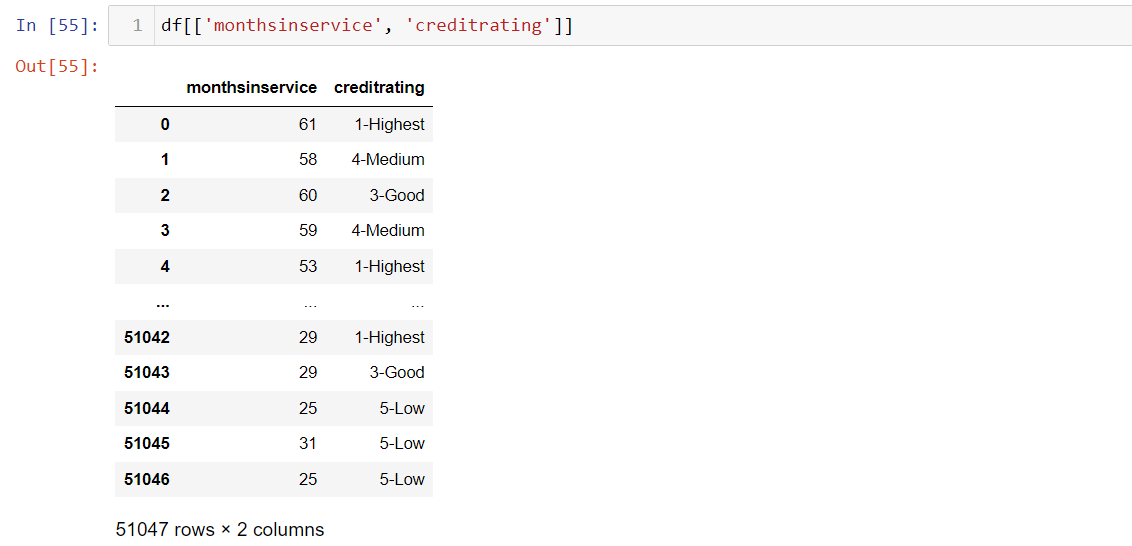


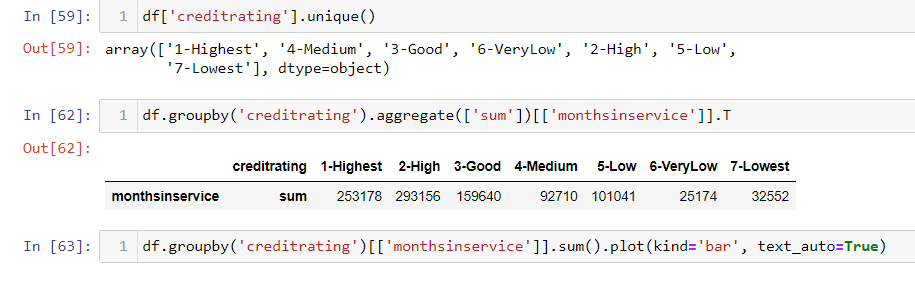


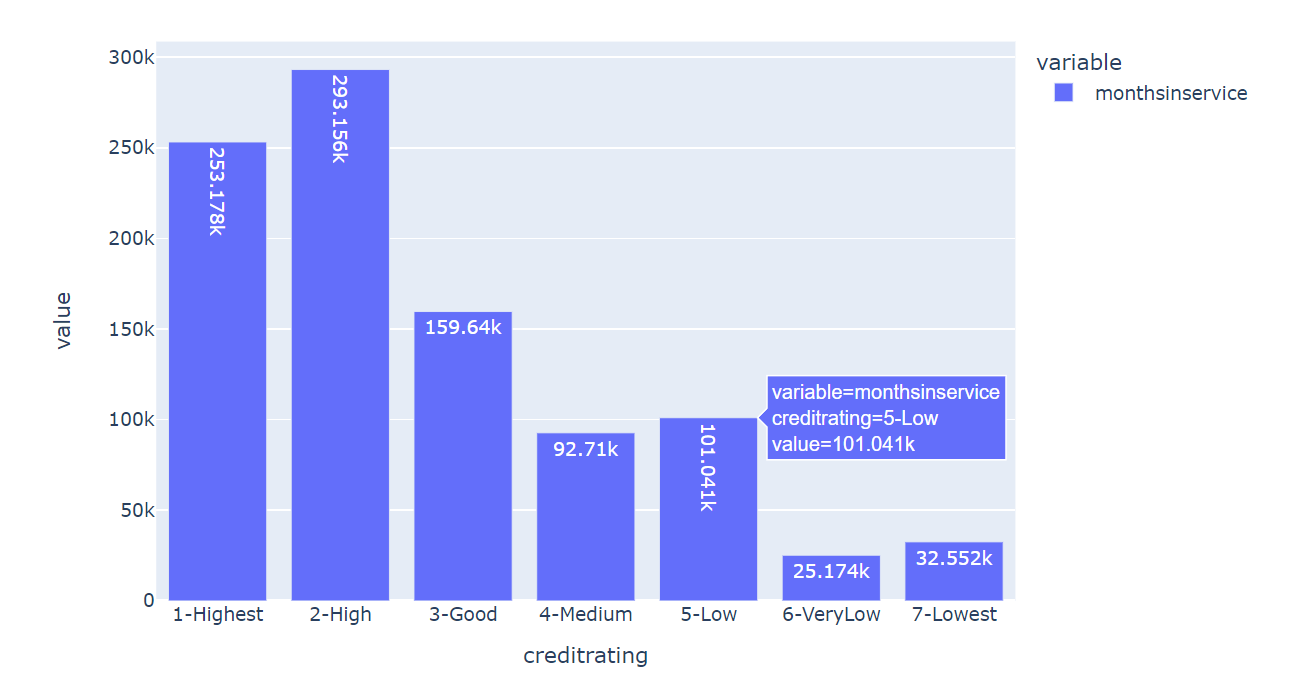
**INTERPRETATION:**

We observe that the data points are clustering in a specific direction (they form a pattern that slopes upwards from left to right), which suggests a positive correlation or relationship between the variables. This suggest that the Customers who make more calls naturally accumulate more call minutes.

1. Months in service & Credit rating







**INTERPRETATION:**

In brief, customers who have used the service for a longer duration are more likely to rate it positively, which can be seen from the above graph.

This is often because longer relationships build trust and familiarity, leading to higher satisfaction and better ratings.

9- Binning of peak calls In & Out

A screenshot of a computer code

Description automatically generated

**INTERPRETATION:**

The code uses the pd.qcut function from the pandas library to divide the continuous numerical column peakcallsinout into 6 equal-sized bins.

It creates a new column called peakcallsinout\_bin to store these bins.

The code then calculates the number of occurrences (counts) of values within each bin using the value\_counts() function.

It sorts the results by the bin index using sort\_index to display them in ascending order.

A screenshot of a computer

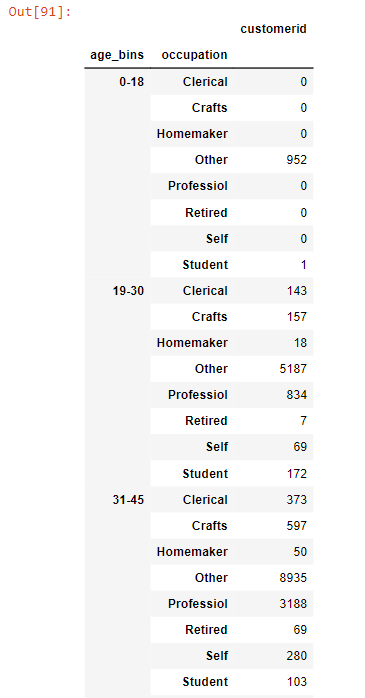
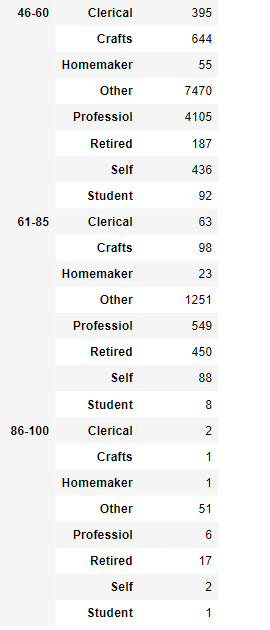
Description automatically generated

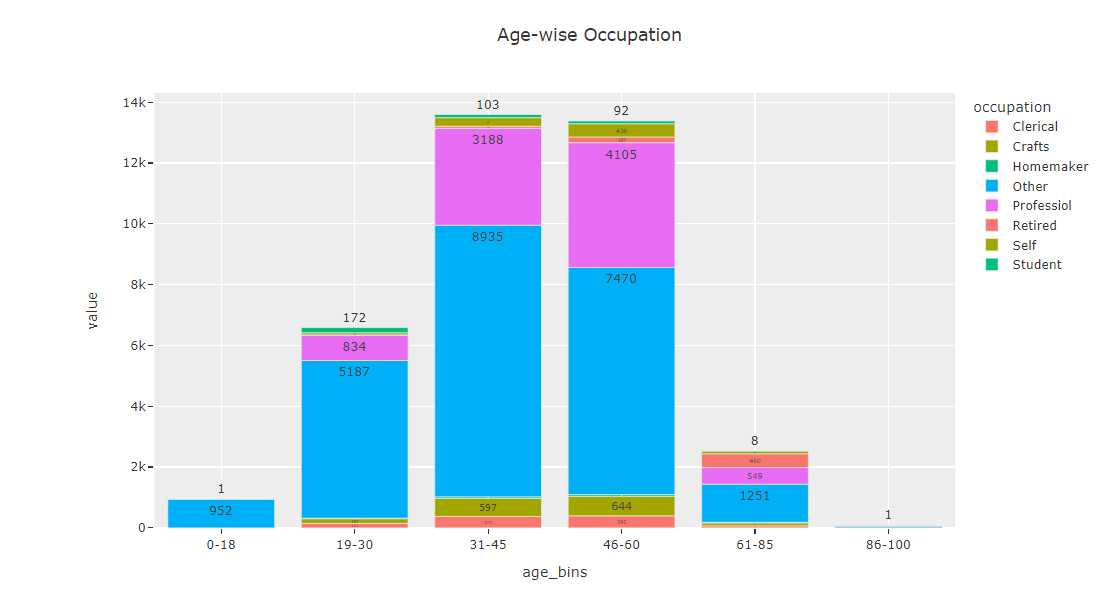
**INTERPRETATION:**

The provided code segments create a new categorical feature " peakcallsinout " based on the "monthlyrevenue" column, which likely represents the age of customers in a dataset. It bins ages into six categories. From the given output we can see that now the missiong values of peakscallinout variable have been filled with corresponding value of monthlyrevenue variable and “inplace= True”, to fix fix it in the dataset so that further missing values are not shown in the peakcallinout feature.

1. Age-wise Occupation



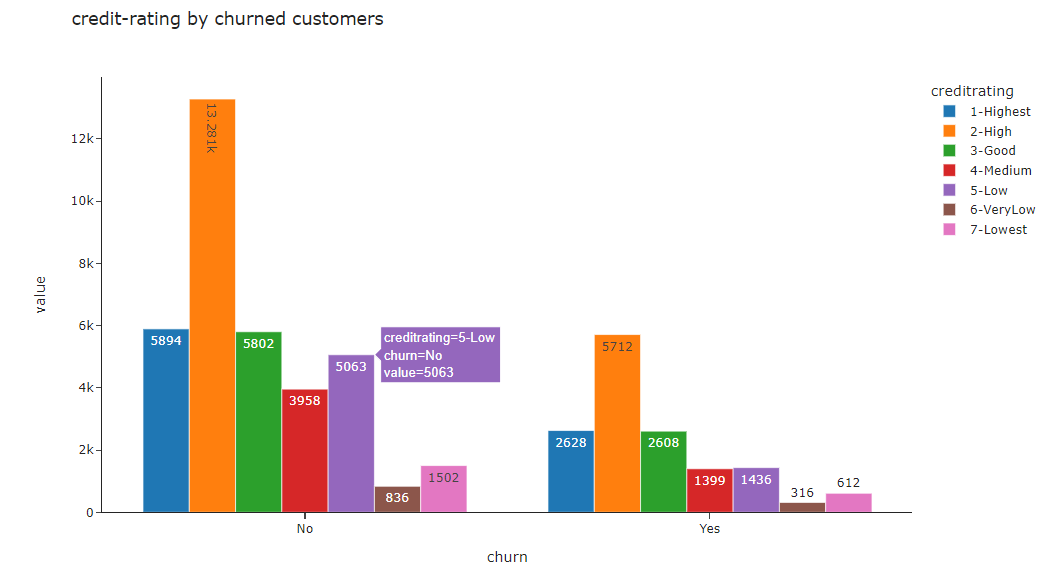


**INTERPRETATION:**

The provided code conducts an analysis of customer demographics (age\_bins) and their respective occupations. It calculates the count of customers in each age group and occupation combination, displaying the results in a bar chart. The output visually represents the distribution of occupations within different age groups. For instance, in the 31-45 age group, "Other" and "Professional" occupations are most common, while the 0-18 age group mostly consists of "Other" occupation. This analysis helps identify occupation preferences among different age segments, which can inform targeted marketing and service strategies based on customer demographics.

1. credit-rating by churned customers





**INTERPRETATION:**

The input code performs a data analysis of customer churn and their respective credit ratings. It calculates the count of customers in each churn status (No churn or Yes churn) and credit rating category, and then it plots the results as a grouped bar chart.

The output visualizes this data, showing how many customers fall into each combination of churn status and credit rating. For instance, it's clear that customers with "High" and "Good" credit ratings have a higher representation among those who have not churned (No), whereas customers with "Medium" and "Low" credit ratings are more prominent among churned customers (Yes). This analysis can guide marketing strategies and retention efforts based on credit ratings.

1. Age distribution of churned customer

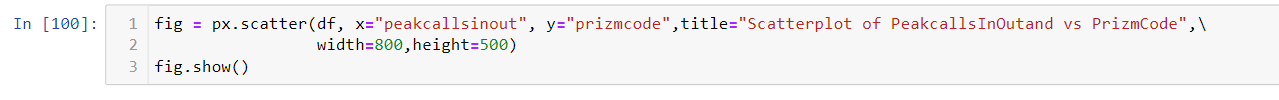


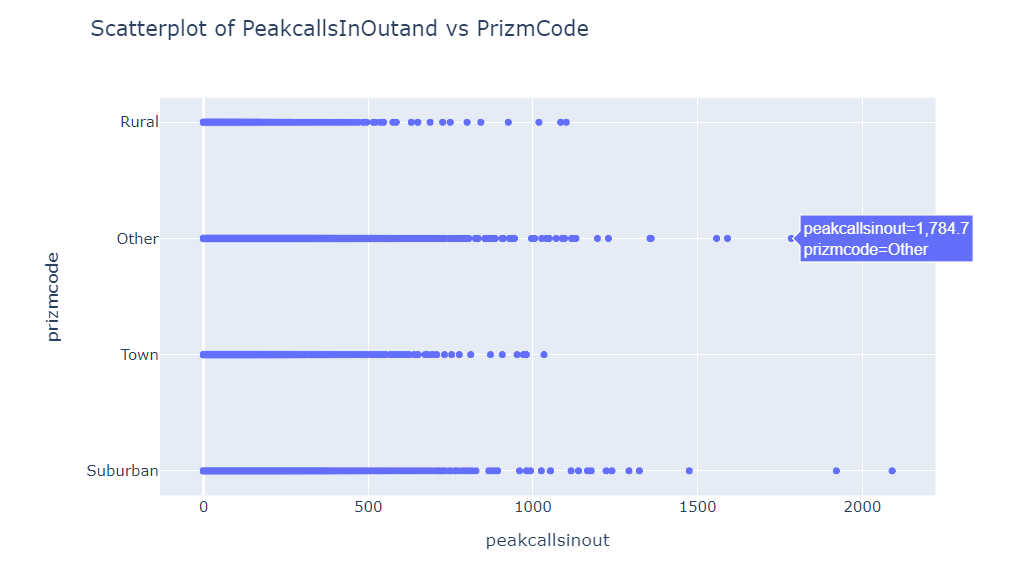


**INTERPRETATION:**

The provided code segments analyze the age distribution of churned customers in a dataset. It groups customers into age categories (age\_bins) and calculates the count of customers who churned ("Yes") and those who did not churn ("No") within each category. The output is visualized in a grouped bar chart. For example, in the 31-45 age group, 9,769 customers did not churn, while 3,826 customers churned. This analysis highlights the age groups with the highest churn rates, helping the telecom company focus its retention efforts on specific age segments, such as 31-45 and 46-60, where churn rates are relatively higher compared to other age groups.

1. Correlation between peak calls in Out & Prizm Code

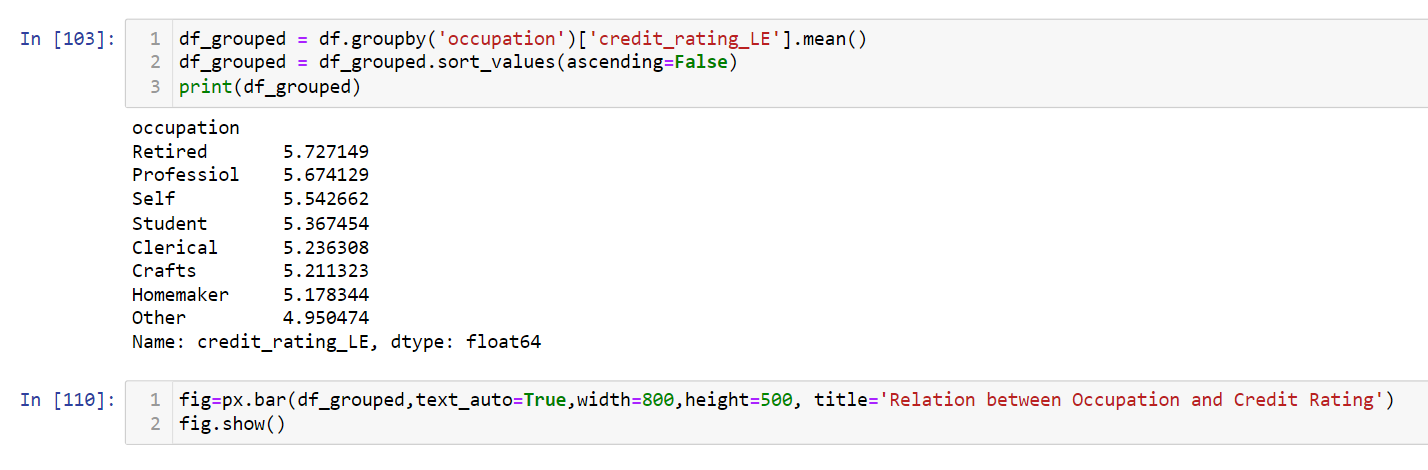


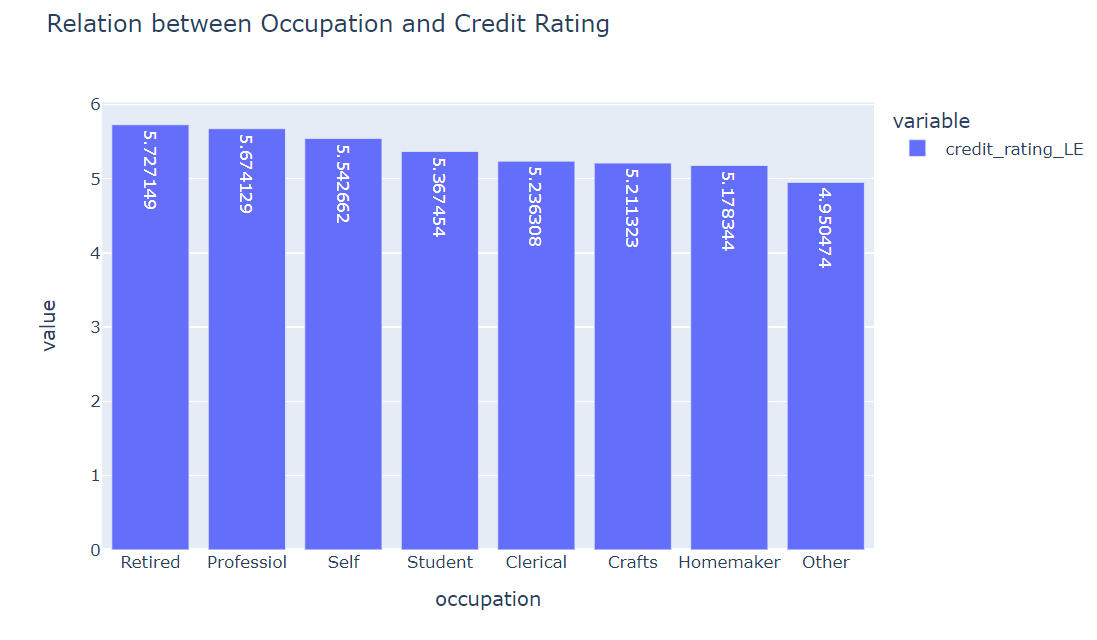


**INTERPRETATION**

The scatterplot demonstrates a positive correlation between the two variables, which means that if one variable's value rises, so too does the value of the other variable. The relationship is not exactly linear, though, as certain locations deviate from the overall pattern. PeakcallsInOutand values are more likely to be higher for residents of locations with higher PrizmCode values. This might be due to the higher likelihood of access to high-speed internet in certain regions, which is necessary for making peak calls. Knowing that correlation does not imply causation is crucial. Even if there is a strong positive correlation between two variables, it does not necessarily follow that one causes the other. It's plausible that a third factor is causing PeakcallsInOutand and PrizmCode to rise simultaneously.

1. Relation between occupation and credit rating

****

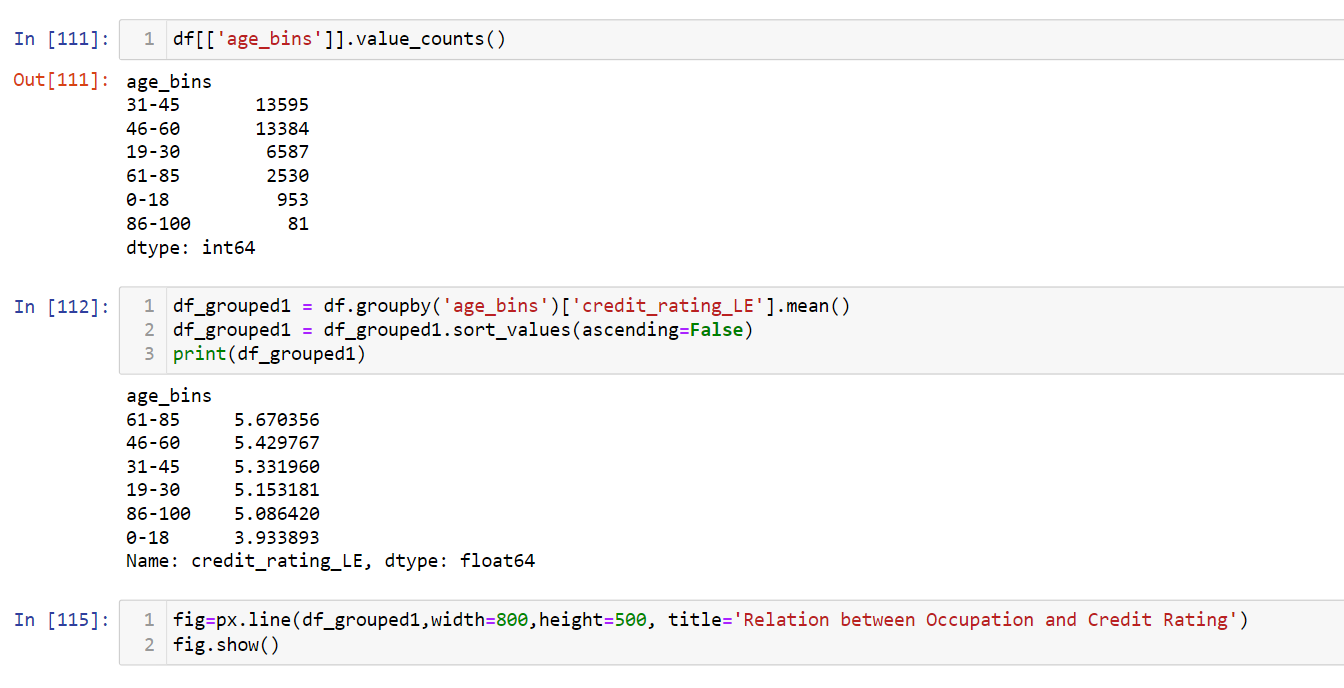
****

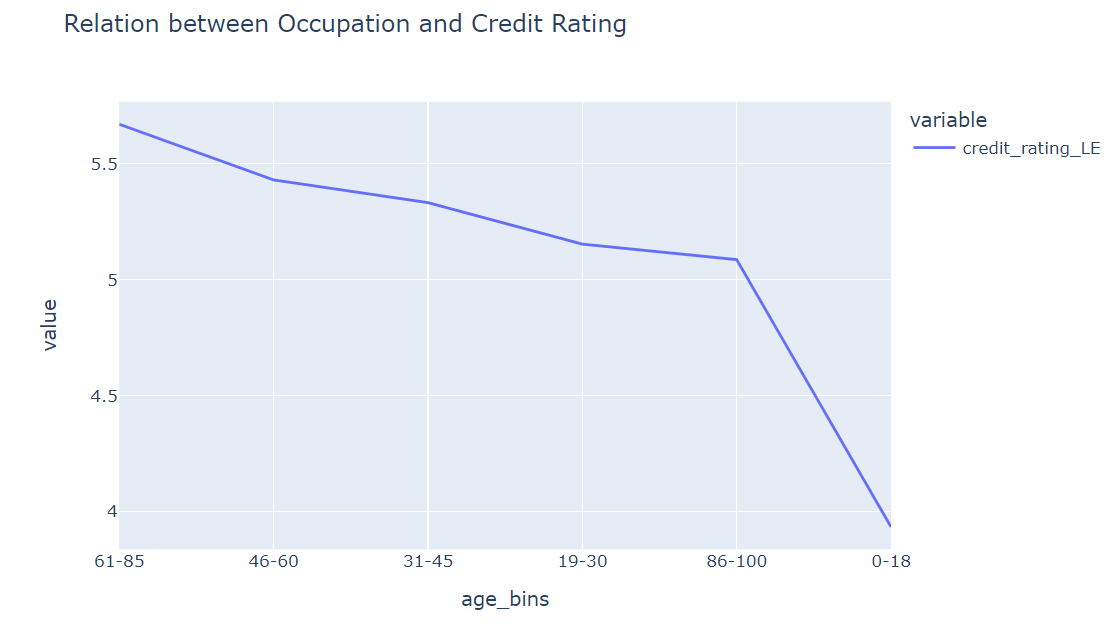
**INTERPRETATION**

The relationship between a occupation and credit rating is favorable. This implies that people tend to have better credit ratings if they have well-paying, steady jobs. This is probably because those with greater wages are more likely to be able to acquire credit and afford to make their payments on time.

It is evident that the average credit ratings for various professions vary significantly from one another. For instance, the average credit score for those working as "Retired" is 5.727, whereas the average credit score for those working as "Other" is 4.950. This shows that some vocations are associated with a higher risk of having bad credit than others.

1. Credit rating based on the age group

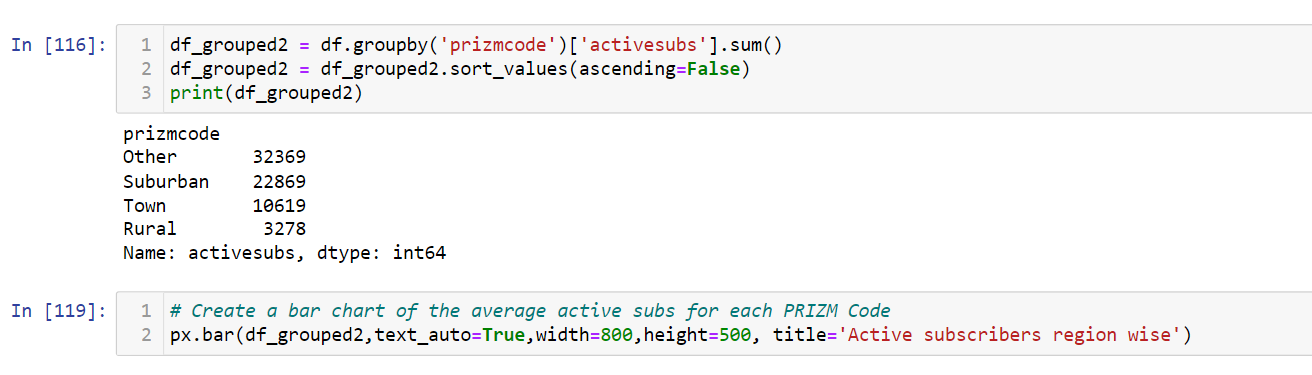
****

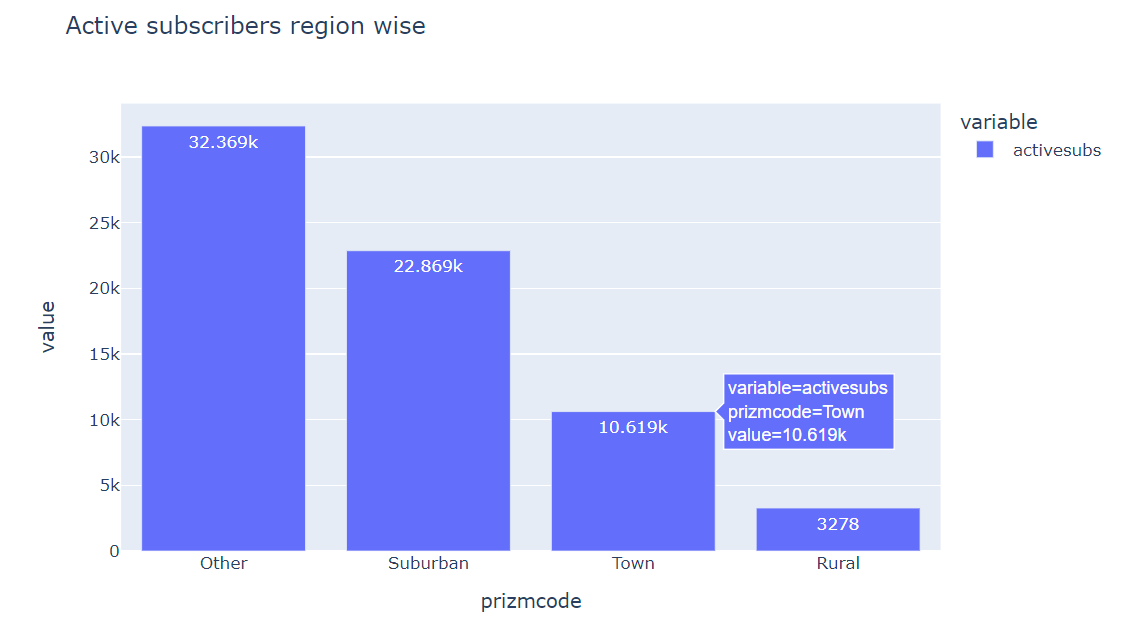
****

**INTERPRETATION:**

The graph demonstrates the significant difference in credit scores between individuals of various ages. For instance, the average credit score for those between the ages of 65 and 74 is 718, while that for those between the ages of 18 and 29 is 653. This shows that some age groups have a higher likelihood of having bad credit than other age groups. Additionally, it demonstrates a positive association between the two factors, indicating that as age rises, so does credit score. The relationship is not exactly linear, though, as certain locations deviate from the overall pattern.

12) Active Subscribers region wise



****

**INTERPRETATION**

The 'df\_grouped2' DataFrame is shown in the output after being grouped, summed, and sorted. It displays the 'prizmcode' as the index and the total of 'activesubs' for each 'prizmcode' category in descending order.

'Other' has the highest sum of 'activesubs' with a value of 32,369.

'Suburban' is the second highest with 22,869 'activesubs.'

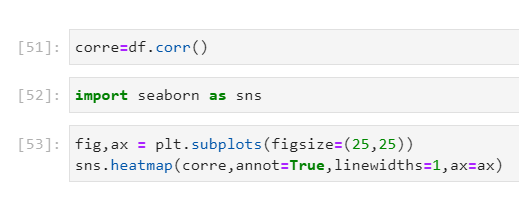
'Town' has 10,619 'activesubs.'

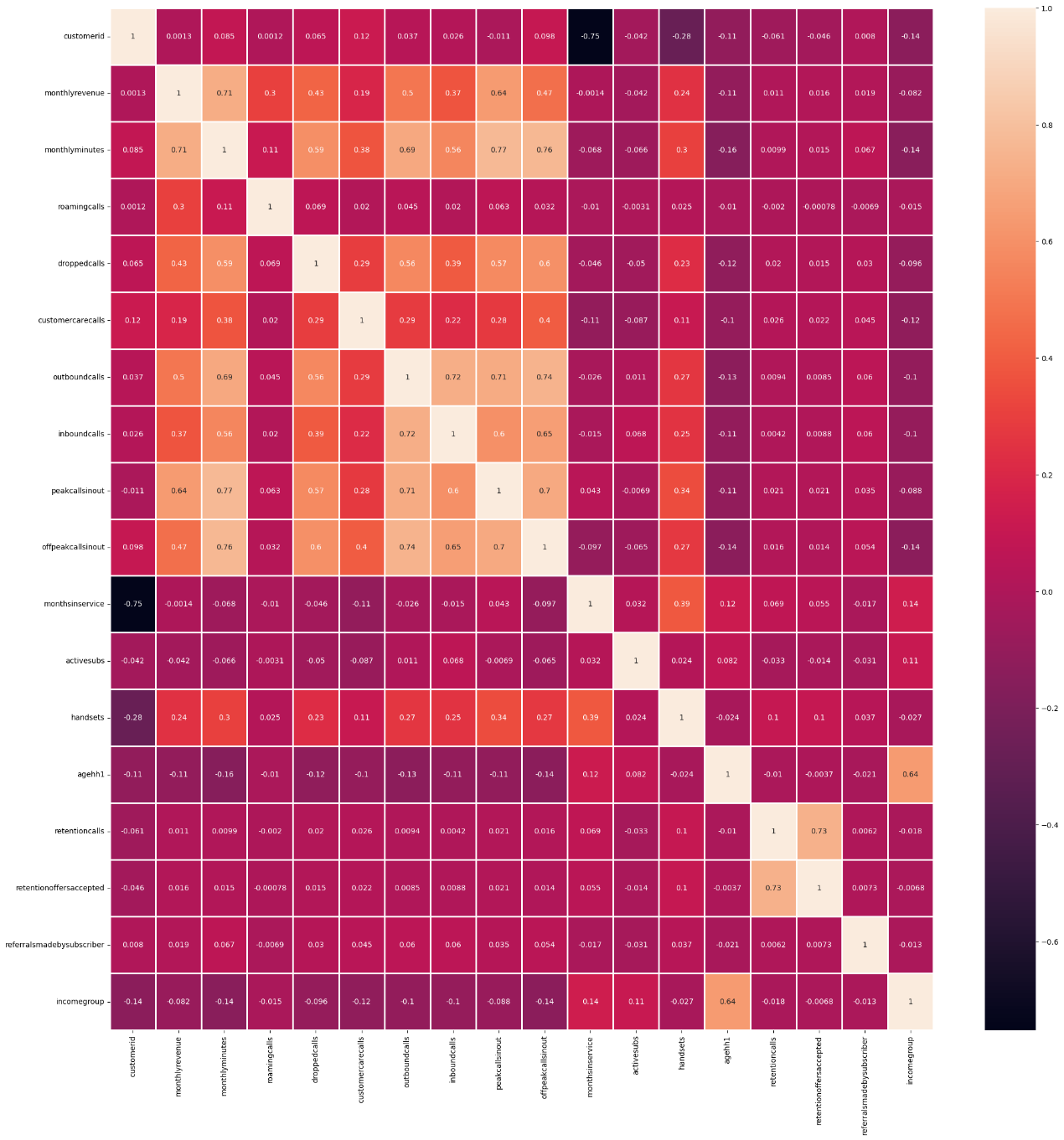
'Rural' has the lowest sum of 'activesubs' with 3,278.

In short, the code sorted the groups in descending order of the sum of the "activesubs" and then grouped the data by "prizmcode," calculating the total number of "activesubs" inside each group. Based on the total "activesubs" values of the output, a ranked list of "prizmcodes" is presented to you.

# SECTION D

## DATA CLEANING







* **Agehh1**

****

**INTERPRETATION**

Since, both the variables are of numeric in nature the missing values in the 'agehh1' is highly correlated with 'incomegroup' with about correlation of 64% as shown in the table. The interpretation of these operations is that the telecom company can use the relationship between 'incomegroup' and 'agehh1' to estimate and fill in missing values in the 'monthlyrevenue' column. By grouping the data based on the number of peak income group of people and age, the company can calculate the average age for each group. This average value is then used as a reference to fill in the missing values in the 'agehh1' column.

* **Monthly Revenue**



**INTERPRETATION**

Since, both the variables are of numeric in nature the missing values in the 'monthlyrevenue' is highly correlated with 'peakcallsinout' with about correlation of 64% as shown in the table. The interpretation of these operations is that the telecom company can use the relationship between 'peakcallsinout' and 'monthlyrevenue' to estimate and fill in missing values in the 'monthlyrevenue' column. By grouping the data based on the number of peak calls made or received, the company can calculate the average monthly revenue for each group. This average value is then used as a reference to fill in the missing values in the 'monthlyrevenue' column.

* **Monthly Minutes**



**INTERPRETATION**

Since, both the variables are of numeric in nature the missing values in the 'monthlyminutes' is highly correlated with 'peakcallsinout' with about correlation of 77% as shown in the table. The interpretation of these operations is that the telecom company can use the relationship between 'peakcallsinout' and 'monthlyminutes' to estimate and fill in missing values in the 'monthlyminutes' column. By grouping the data based on the number of offpeak calls made or received, the company can calculate the average monthly minutes for each group. This average value is then used as a reference to fill in the missing values in the 'monthlyminutes' column.

* **Roming Calls**

****

**INTERPRETATION**

Since, both the variables are of numeric in nature the missing values in the 'roamingcalls' is highly correlated with 'monthlyminutes' with about correlation of 11% as shown in the table. The interpretation of these operations is that the telecom company can use the relationship between 'monthlyminutes' and 'roamingcalls' to estimate and fill in missing values in the 'roamingcalls' column. By grouping the data based on the duration of call, the company can calculate the average romming call minutes for each group. This average value is then used as a reference to fill in the missing values in the 'monthlyrevenue' column.

## Feature Engineering/ Feature Encoding

**One Hot Encoding**

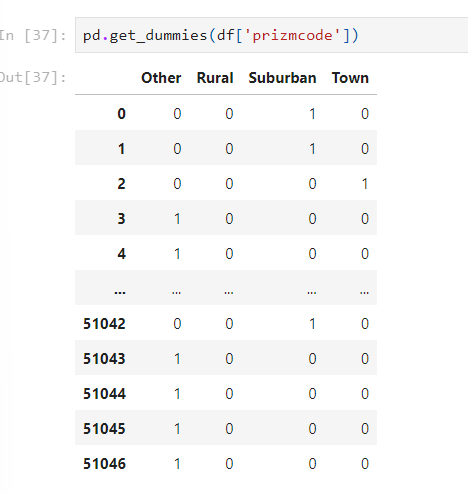
* Churn Rate



The categorical 'churn' column in a DataFrame 'df' is converted into a set of binary columns by the function ‘pd.get\_dummies(df['churn'])’, one for each category in the column, which indicates whether or not each category is present in each row of the DataFrame.

To make the churn column compatible with machine learning algorithms, one-hot encoding is applied to the column. The churn column has two categories, No and Yes, and is a category variable. The churn column is split into two numerical variables, No and Yes, using one-hot encoding. The churn column is now compatible with machine learning techniques because of this.

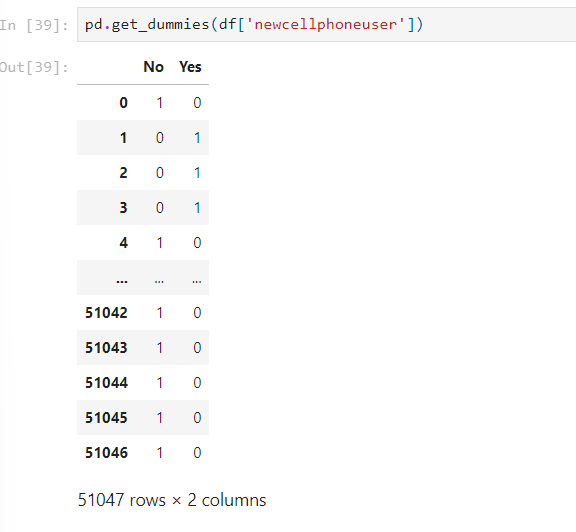
* Prizm Code



The function 'pd.get\_dummies(df['churn'])' creates a set of binary columns, one for each distinct category in the 'churn' column, from a categorical column ('churn') in a DataFrame ('df'). This is a common method for representing category data into numerical values in machine learning.

One-hot encoding is used in the prizmcode column's case to make the column suitable for machine learning methods. A categorical variable with four categories is the prizmcode column. The prizmcode column is one-hot encoded into the four numerical variables Other, Rural, Suburban, and Town.

* New Cell Phone Users



The categorical column 'newcellphoneuser' in DataFrame 'df' is converted into a series of binary columns by the function 'pd.get\_dummies(df['newcellphoneuser'])', where each distinct category in 'newcellphoneuser' becomes a new binary column, signifying the presence or absence of each category for each row in the DataFrame.

To make the 'newcellphoneusers' column compatible with machine learning methods, one-hot encoding is used in this situation. A categorical variable, the newcellphoneusers column has the options No and Yes. The newcellphoneusers column is split into two numerical variables, No and Yes, using one-hot encoding. The newcellphoneusers column can now be used with machine learning methods as a result.

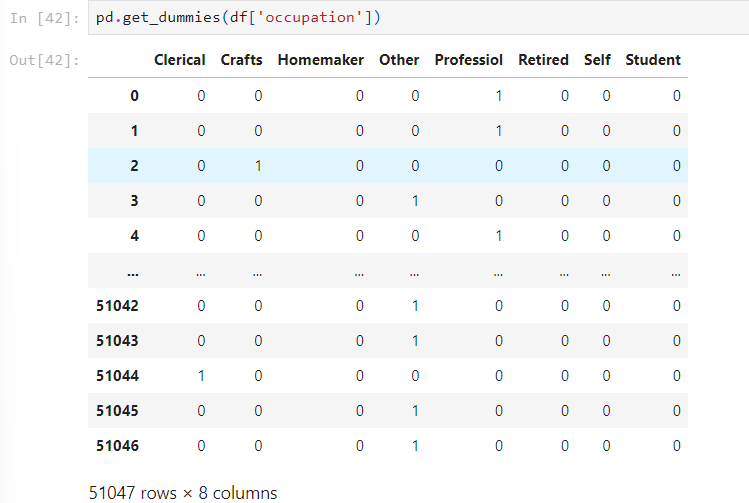
* Responds to Mail Offers



The category column 'respondstomailoffers' in DataFrame 'df' is transformed into numerous binary columns by the function ‘pd.get\_dummies(df['respondstomailoffers'])’. Each distinct category in "respondstomailoffers" is converted into a separate binary column, where 1 denotes the presence of that category for each row in the DataFrame and 0 denotes its absence.

One-hot encoding is used in the respondstomailoffer column's case to make the column suitable for machine learning techniques. A categorical variable, the respondstomailoffer column has two categories: No and Yes. The respondstomailoffer column is split into two number variables, No and Yes, using one-hot encoding. This enables machine learning methods to use the respondstomailoffer column.

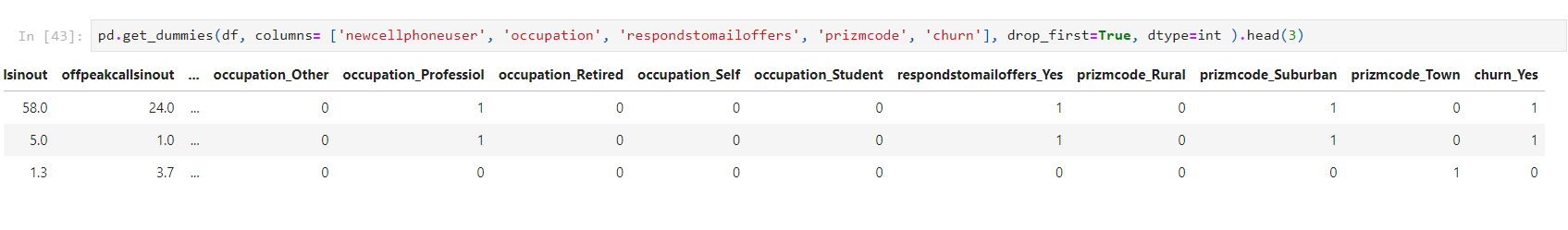
* Occupation



The categorical column 'occupation' in DataFrame 'df' is taken and transformed into a set of binary columns by the function pd.get\_dummies(df['occupation']). Each distinct occupational category in "occupation" is converted into a separate binary column, where 1 denotes the presence of that occupation for each row in the DataFrame and 0 denotes its absence.

To make the occupation column compatible with machine learning techniques, one-hot encoding is used in this scenario. There are eight categories in the occupation column, which is a categorical variable. The occupation column is one-hot encoded into eight number variables, one for each classification. This enables machine learning algorithms to use the occupation column.

Dropping the first columns of each encoded variable to avoid the curse of dimensionality and multi collinearity



With the use of one-hot encoding, this method transforms the category columns 'newcellphoneuser', 'occupation','respondstomailoffers', 'prizmcode', and 'churn' from a DataFrame called 'df' into binary columns. To avoid multicollinearity, it eliminates one of the binary columns for each categorical variable and specifies that the remaining columns should have an integer data type. The last three rows of the modified DataFrame are shown.

## Label Encoding

* Credit Rating



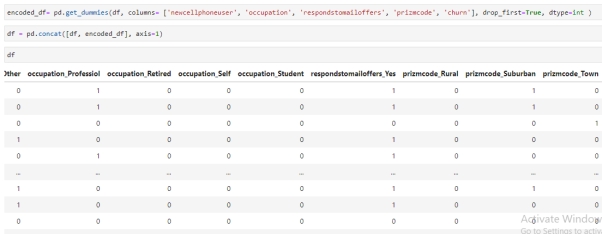
In the DataFrame 'df', this code adds a new column called 'credit\_rating\_LE' by mapping the values from the 'creditrating' column to numbers using a prepared mapping dictionary called 'credit\_rating\_mapping'. In order to make analysis and modeling easier, this enables the translation of category credit rating labels into numerical scores. Both the original "creditrating" column and the newly added "credit\_rating\_LE" column, which contains numerical representations of the credit ratings, will be present in the resultant DataFrame.

After the occupation column has been label encoded, machine learning algorithms can use it as input. The label encoded occupation column could be one of the input factors used to train a machine learning algorithm to predict whether or not a client is likely to churn.

The result is a list of integers, one for each customer, as we can see. The integer denotes the customer's occupational category, where 0 is the first category, 1 denotes the second category, and so forth.

The links between the many categories of the categorical variable are not preserved by label encoding, it is vital to note. For example, the integer value 4 does not indicate that the Professional category is more important than the Clerical category.

Adding columns of the variables encoded to the intial data frame.



The DataFrame's category variables is encoded using the Pandas get\_dummies() function. A DataFrame is passed to the get\_dummies() method, which outputs a DataFrame with new binary variables for each category of each categorical variable.

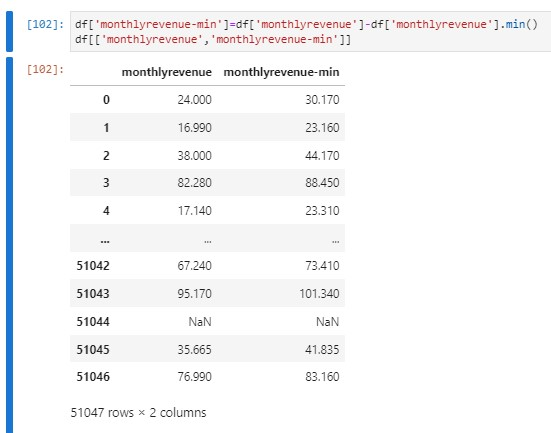
The we used Pandas' concat() function to add the encoded columns to the original DataFrame. A new DataFrame with the columns of the two input DataFrames is returned by the concat() method, which accepts two DataFrames as input.

A good method for preparing data for machine learning is to add the encoded columns to the initial DataFrame. Feature engineering is made possible and the categorical variables are made compatible with machine learning techniques.

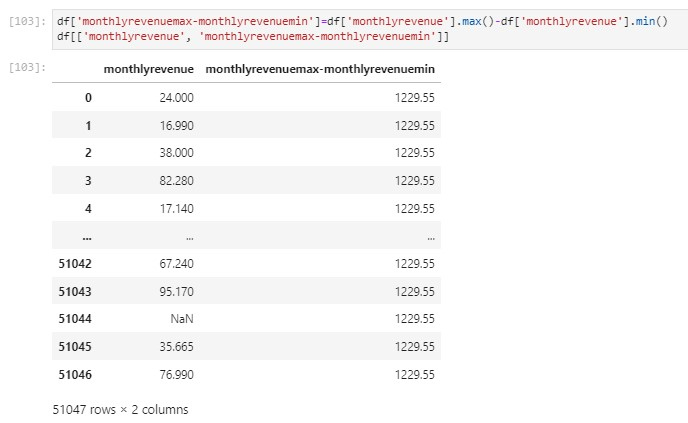
## SCALING

Scaling in the context of data pre-processing involves the transformation of numerical features within a dataset to adhere to a specific scale or distribution. Scaling is advantageous in the context of distance-based methods and regularization techniques, where disparities in scales can adversely affect model performance.

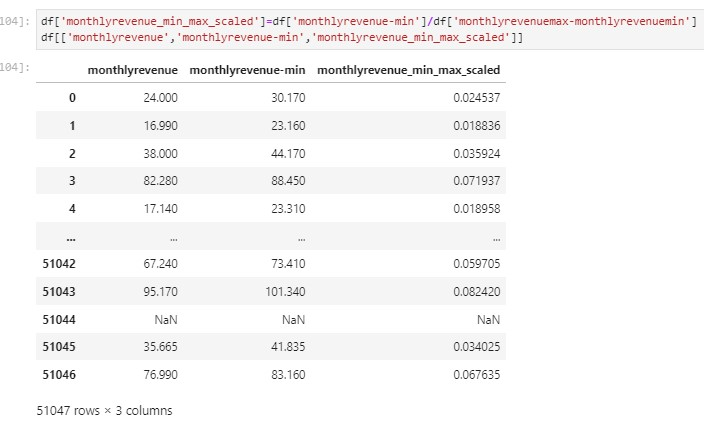
Scaling has been performed between the variables Monthlyrevenue and Income group as they both had values which were drastically apart from each other.



df[‘monthlyrevenue-min'] calculates the difference between each value in the 'monthlyrevenue' column and the minimum value in that same column, effectively centering the data around zero.

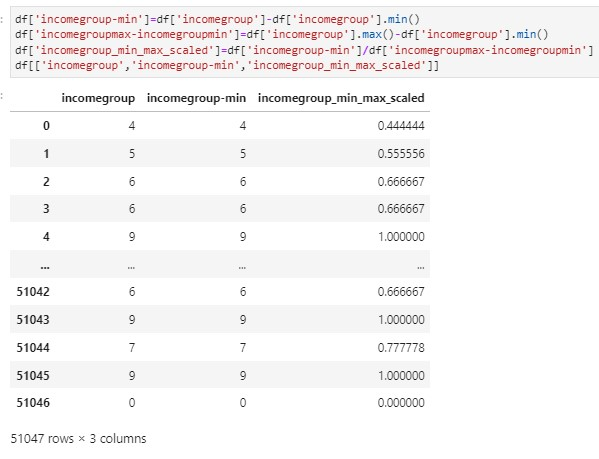


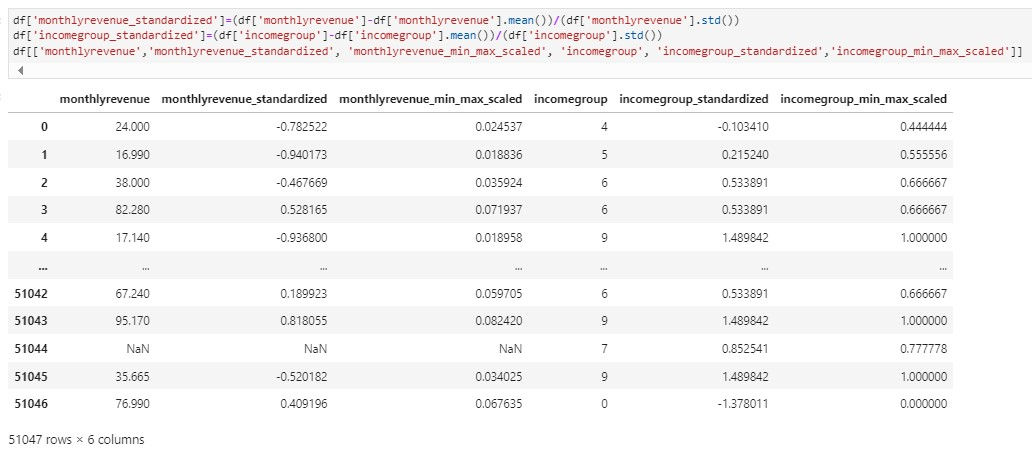
df['monthlyrevenuemax-monthlyrevenuemin'] computes the difference between the maximum value and the minimum value in the 'monthlyrevenue' column.



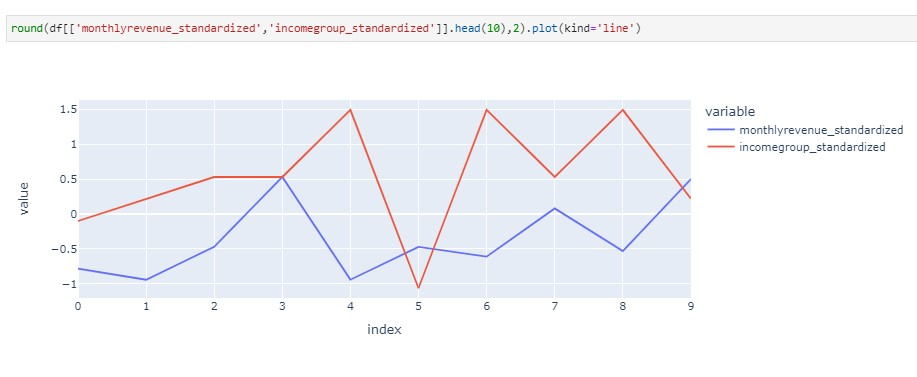
This scaling process transforms the 'monthlyrevenue' column's values to a new range between 0 and 1, making them comparable and interpretable on the same scale.

Similar steps are taken for the variable income group as well and the following results are obtained.





The result is that both the 'monthlyrevenue\_standardized' and 'incomegroup\_standardized' columns now contain standardized values. Standardization makes it easier to compare and analyze the two columns because they are now on the same scale and have a mean of 0 and a standard deviation of 1, which simplifies their interpretation and use



## DEFINING X AND Y

A screenshot of a computer program

Description automatically generated

X contains the features (independent variables) of the dataset, excluding the 'churn' column, which is the target variable.

y contains the values of the 'churn' column, which is the target variable.

The shapes show the dimensions of the data:

"Shape of X": It prints the number of rows and columns in X.

"Shape of y": It prints the number of elements in y

## LOGISTIC REGRESSION

A screenshot of a computer program

Description automatically generated

In this code snippet, a machine learning task involving logistic regression is executed. The dataset is split into training and testing subsets using the `train\_test\_split` function from `sklearn.model\_selection`.

This separation is crucial to train the model on one portion of the data (`X\_train` and `y\_train`) and assess its performance on unseen data (`X\_test` and `y\_test`). A logistic regression model is initialized and fitted (`logit\_sklearn.fit()`) on the training data to learn the patterns between the independent variables (`X`) and the target variable (`y`). Subsequently, predictions are made on the test set using the trained model (`results.predict(X\_test)`), resulting in `y\_pred` containing the predicted values.

Finally, a DataFrame is generated to display a comparison (`pd.DataFrame({'y\_actual': y\_test, 'y\_pred': y\_pred}).head()`) between the actual (`y\_test`) and predicted (`y\_pred`) values for the initial instances in the test set, allowing an initial assessment of the model's predictive performance on unseen data.

CONFUSION MATRIX

A screenshot of a computer

Description automatically generated

* logit\_sklearn\_cf = metrics.confusion\_matrix(y\_test, y\_pred)-> This code calculates a confusion matrix using metrics.confusion\_matrix from Scikit-Learn's metrics module. The confusion matrix provides a table that summarizes the model's predictions versus the actual values. It shows counts of true negatives (TN), false positives (FP), false negatives (FN), and true positives (TP). These values help assess the model's .
* tn, fp, fn, tp = metrics.confusion\_matrix(y\_test, y\_pred).ravel(): This line retrieves the individual values of true negatives (tn), false positives (fp), false negatives (fn), and true positives (tp) from the confusion matrix using ravel() method. These values represent the counts of correctly and incorrectly classified instances based on the model's predictions compared to the actual values in the test set.

Classification Report:

* target\_names = ['No (1)', 'Yes (0)']: This defines the names for the classes (labels) in the classification report.
* The report includes metrics such as precision, recall, F1-score, and support for each class label ('No' and 'Yes'). It provides a summary of the model's performance, indicating how well it predicts each class.

The classification report unveils critical insights into the model's performance. There is a glaring imbalance between the classes, with class 'Yes' (0) significantly outnumbering class 'No' (1). Remarkably, the model excels in predicting instances labeled as 'Yes' (0), showcasing high precision (72%), recall (100%), and an impressive F1-score of 83%. However, the model's inability to correctly identify instances of class 'No' (1) raises concern, as it displays a low precision (38%) and an alarmingly zero recall, resulting in an F1-score of 0.00. The overall accuracy of 71% might seem satisfactory, yet it is largely influenced by the model's strong performance in predicting the majority class. The weighted average metrics (e.g., weighted F1-score of 0.60) emphasize the need for improved predictive ability across both classes and addressing the imbalance issue.

A screenshot of a computer

Description automatically generated

* logit\_sklearn\_cf: This variable represents a trained logistic regression classifier from scikit-learn.
* ConfusionMatrixDisplay(logit\_sklearn\_cf, display\_labels={0: 'False', 1: 'True'}): This line creates a ConfusionMatrixDisplay object using the trained classifier logit\_sklearn\_cf. It also specifies the display labels for the classes; in this case, 'False' is mapped to class 0 and 'True' is mapped to class 1.
* .plot(): This method is used to visualize or plot the confusion matrix represented by the ConfusionMatrixDisplay object created in the previous step. The confusion matrix is a table that shows the counts of true positive, true negative, false positive, and false negative predictions made by a classification model.
* It displays a visual representation of the confusion matrix for the specified logistic regression classifier (logit\_sklearn\_cf), allowing us to analyze its performance in predicting the classes labeled 'False' and 'True'.

The accuracy\_score function computes the accuracy, which is the ratio of correctly predicted data points (the number of true positives and true negatives) to the total number of data points. The accuracy\_score is 0.71 (or 71%), it means that the model correctly predicted the class of 71% of the data points in the test set. Higher accuracy values generally indicate better performance.

## DECISION TREE CLASSIFIER

A screenshot of a computer

Description automatically generated

The provided code initializes a Decision Tree Classifier using the Gini impurity criterion and a specified random seed of 50. Subsequently, the model is trained using the training dataset, consisting of features (`X\_train`) and corresponding target labels (`y\_train`). After training, predictions are made on the test dataset (`X\_test`) using the trained model. These predictions are stored in a DataFrame (`y\_pred\_df`), which juxtaposes the actual target labels (`y\_test`) with the predicted labels (`y\_pred\_dtc`). By displaying this DataFrame, the code allows for a direct comparison between the model's predictions and the true values in the test set, offering a clear view of where the model's predictions align or diverge from the actual values, facilitating an assessment of the model's performance on this specific dataset.

A screenshot of a computer

Description automatically generated

X\_test.shape[0] retrieves the number of samples in the test set (X\_test) and ((y\_test) != (y\_pred)) creates a boolean array that represents where the predicted values differ from the actual values. It evaluates to True where the predictions are incorrect and False where they match.

The output indicates that out of a total of 10,210 samples in the test set, the model made incorrect predictions for 2,910 samples. These are instances where the predicted values do not match the actual values in the test set. This count represents the number of misclassified outcomes by the model on the given test dataset.

The detailed classification report showcases metrics such as precision, recall, F1-score, and support for each class (0 and 1) in the target variable. For class 0, the precision, recall, and F1-score are 0.33, 0.36, and 0.34, respectively, indicating relatively lower performance. Conversely, for class 1, these metrics are higher at 0.74 (precision), 0.71 (recall), and 0.72 (F1-score), suggesting better predictive capability for this class. The support values indicate the number of instances for each class in the test set (2909 for class 0 and 7301 for class 1). The overall accuracy of the model on the entire test set is 61%, while the macro and weighted averages for precision, recall, and F1-score stand around 0.53 and 0.62, respectively.

## NAÏVE BAYES MODEL

A screenshot of a computer

Description automatically generated

The Gaussian Naive Bayes model assumes feature independence and Gaussian (normal) distribution of continuous features to classify data based on Bayesian probability.

Gaussian Naive Bayes Model Creation: The code initializes a Gaussian Naive Bayes (GNB) classifier (gnb).The gnb.fit(X\_train, y\_train) line trains the GNB model using the training dataset (X\_train features and y\_train target labels). After training, the GNB model (gnb\_results) is used to predict labels for the test dataset (X\_test) using gnb\_results.predict(X\_test). The code creates a DataFrame (y\_pred\_df) to compare the actual target labels (y\_test) from the test set with the predicted labels (y\_pred\_gnb) generated by the Gaussian Naive Bayes model.

A screenshot of a computer

Description automatically generated

The results reveal that among the 10,210 samples present in the test set, the model inaccurately predicted 3,074 samples, signifying instances where the model's predictions did not align with the actual values in the test dataset.

Furthermore, examining the detailed classification report, for class 0, the precision, recall, and F1-score are recorded as 0.33, 0.36, and 0.34, respectively, indicating relatively modest performance for this class. Conversely, for class 1, the model showcases higher metrics at 0.74 (precision), 0.71 (recall), and 0.72 (F1-score), suggesting a more robust predictive capability for this particular class. The support values associated with each class indicate the number of instances present in the test set, with 2,909 instances for class 0 and 7,301 instances for class 1.

## KNN CLASSIFIERS

A screenshot of a computer

Description automatically generated

K-nearest neighbors (KNN) is a simple algorithm for classification and regression. It classifies new data points based on the majority class of their k nearest neighbors. It's non-parametric, lazy, and relies on distance metrics.

The code initializes a Knn classifier (knn).The knn.fit(X\_train, y\_train) line trains the KNN model using the training dataset (X\_train features and y\_train target labels). After training, the KNN model (Knn\_results) is used to predict labels for the test dataset (X\_test) using knn\_results.predict(X\_test). The code creates a DataFrame (y\_pred\_df) to compare the actual target labels (y\_test) from the test set with the predicted labels (y\_pred\_gnb) generated by the Knn classifier model.

A screenshot of a computer

Description automatically generated

The results reveal that among the 10,210 samples present in the test set, the model inaccurately predicted 3,717 samples, signifying instances where the model's predictions did not align with the actual values in the test dataset.

Furthermore, examining the detailed classification report, for class 0, the precision, recall, and F1-score are recorded as 0.31, 0.22, and 0.26, respectively, indicating relatively modest performance for this class. Conversely, for class 1, the model showcases higher metrics at 0.72 (precision), 0.80 (recall), and 0.76 (F1-score), suggesting a more robust predictive capability for this particular class. The support values associated with each class indicate the number of instances present in the test set, with 2,909 instances for class 0 and 7,301 instances for class 1.

## SVM CLASSIFIER

A screenshot of a computer

Description automatically generated

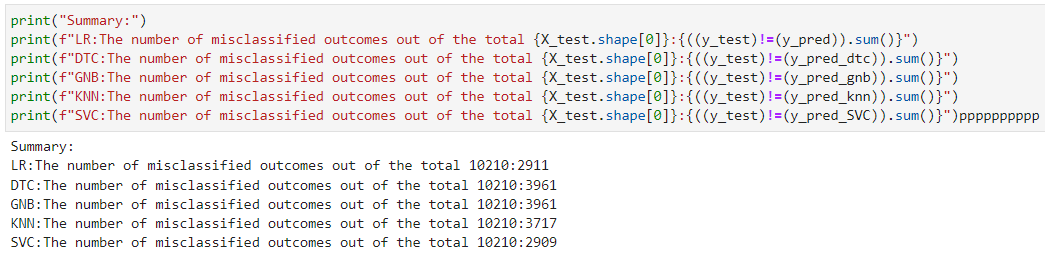
A screenshot of a computer

Description automatically generated

The results reveal that among the 10,210 samples present in the test set, the model inaccurately predicted 2909 samples, signifying instances where the model's predictions did not align with the actual values in the test dataset.

Furthermore, examining the detailed classification report, for class No (1), the precision, recall, and F1-score are recorded as 0.00, 0.00, and 0.00, respectively, indicating relatively modest performance for this class. Conversely, for class Yes (0), the model showcases higher metrics at 0.72 (precision), 1.00(recall), and 0.83 (F1-score), suggesting a more robust predictive capability for this class. The support values associated with each class indicate the number of instances present in the test set, with 2,909 instances for class No (1) and 7,301 instances for class Yes (0).

## FINAL SUMMARY REPORT OF ALL THE MODELS USED



Total Test Cases: The number 10210 in each output line indicates that there were 10,210 test cases used for evaluation.

Misclassified Counts: The numbers following the colon represent the count of misclassified outcomes for each model, as follows:

* LR: 2911
* DTC: 3961
* GNB: 3961
* KNN: 3717
* SVC: 2909

The code evaluates the performance of five machine learning models by comparing their predictions on a test set to the actual outcomes. It prints the number of misclassified outcomes for each model, providing a way to assess their relative accuracy.

Finding:

* Best Performing Models: Based on the misclassification counts, the best-performing models in this evaluation are LR and SVC, with 2911 and 2909 misclassifications, respectively.
* Worst Performing Models: The worst-performing models are DTC and GNB, with 3961 misclassifications each.

# Section: E

## Conclusion

Based on the observations from the case description and exploratory data analysis (EDA) of the telecom dataset, the following findings can be summarized. These findings provide insights into customer behavior, demographics, and factors influencing churn. They can inform retention strategies and targeted marketing efforts to reduce customer turnover and improve business performance.

* Customer Spending Habits - The average monthly revenue per customer is approximately $58.80, with a wide range of spending habits. Monthly revenue ranges from as low as -$6.17 to as high as $1,223.38, with considerable variability. This data is important for pricing strategies and revenue forecasting.
* Usage Patterns-The median monthly minutes used by customers is 366 minutes, indicating variability in usage patterns. Usage ranges from 0 minutes (possibly inactive users) to a maximum of 7,359 minutes.
* Churn and Monthly Revenue - Customers who haven't churned ("No") tend to have slightly higher median monthly revenue ($48.8) compared to those who have churned ("Yes") with $47.4. Customers who continue their subscriptions tend to have slightly higher monthly spending.
* Demographic Analysis - Age and occupation demographics reveal patterns in customer preferences. Occupation preferences differ among various age groups, which can inform targeted marketing strategies.
* Customers who did not leave the service have made more referrals than those who have churned.
* Customers who make more calls tend to accumulate more call minutes.
* Customers who have used the service for a longer duration tend to rate it more positively.
* Credit Rating and Churn - Credit rating affects churn, with customers having "High" and "Good" credit ratings less likely to churn ("No"), while those with "Medium" and "Low" credit ratings are more likely to churn ("Yes").
* Occupation and Credit Rating - Occupation can affect credit rating, with certain professions associated with better credit scores.For example, "Retired" individuals tend to have higher credit scores compared to those in the "Other" category.
* Age and Credit Rating - Older age groups tend to have higher average credit scores. There is a positive correlation between age and credit score.
* Prizm Code and Active Subscribers- The "Prizm Code" category "Other" has the highest number of active subscribers, followed by "Suburban," "Town," and "Rural."
* After using all the model which are stats.api model, naïve bayes, K-nearest neighbors, Decision Tree and Support Vector Machine, it can be seen that SVM is the most preferred model for our dataset as it giving the least misclassified item that is 2909, the results can be seen as follows:

LR:The number of misclassified outcomes out of the total 10210:2911.

DTC:The number of misclassified outcomes out of the total 10210:3961

GNB:The number of misclassified outcomes out of the total 10210:3961

KNN:The number of misclassified outcomes out of the total 10210:3717

SVC:The number of misclassified outcomes out of the total 10210:2909

* Best Performing Models: Based on the misclassification counts, the best-performing models in this evaluation are LR and SVC, with 2911 and 2909 misclassifications, respectively.
* Worst Performing Models: The worst-performing models are DTC and GNB, with 3961 misclassifications each.

# Section-F

## Contribution

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **ROLL NO.** | **RATING (10)** | **CONTRIBUTION** |
| Mrinal Kumar Chaubey | JL22PG107 | 10 | Problem Statement, Feature engineering & Interpretation |
| Prashant Pal | JL22PG135 | 10 | Missing Values & Binning, Feature Scaling, Visualization & EDA, Interpretation |
| Ritukona Chakraborty | JL22PG162 | 10 | Feature Scaling, Visualization, EDA, Conclusion, Interpretation |
| Udita Saini | JL22PG224 | 10 | Problem Identification, Feature Engineering, EDA, Visualization, Interpretation,  Model Building |

# Turnitin Report

