**Company collected data from 5000 customers. The objective of this case study is to understand what's driving the total spend of credit card(Primary Card + Secondary card)**

**PROBLEM STATEMENT**

**Prioritize the drivers based on the importance.**

**HOW WE WENT ABOUT SOLVING THE PROBLEM**

* First of all we imported the required libraries in our jupyter notebook in python.
* We then imported our dataset into a data frame df.
* We now did **overall study of the data** in which we did the following
  1. We looked at df.info() to **check the variables** in our dataset and their **data types**
  2. We then checked for the **duplicate rows** in our dataset with df.duplicated().sum()

There were no duplicate rows

* 1. We then checked for the **null values** in our dataset with df.isnull().sum()

The following columns had null values :-

|  |
| --- |
| **lnwireten 3656** |
| **lnwiremon 3656** |
| **lnequipmon 3296** |
| **lnequipten 3296** |
| **lntollten 2622** |
| **lntollmon 2622** |
| **lncardten 1422** |
| **lncardmon 1419** |
| **lnlongten 3** |
| **longten 3** |
| **townsize 2** |
| **commutetime 2** |
| **cardten 2** |
| **lnothdebt 1** |
| **lncreddebt 1** |
|  |

* We now went on to **treat** these **null values** in the following manner.

1. For townsize, lncreddebt and longten, we removed the rows that contained null values due to which lnothdebt, commutetime and lnlongten themselves got treated as the rows that contained null values already got deleted.
2. For all the other variables, as they were in the numeric form, we used KNN Imputer to impute null values.

We will now explain how KNN Imputer works and why was it a good approach to impute values in our case

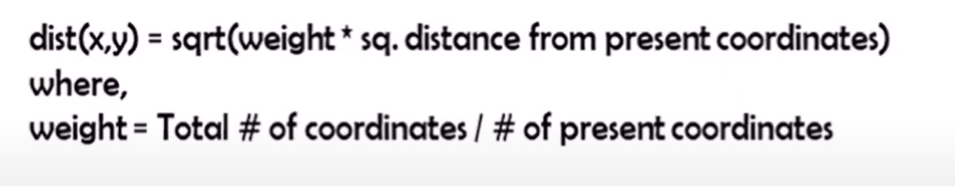
**How KNN Imputer works** :-

In short what KNN Imputer does is that.

* Suppose we have a row containing a single missing value, so the Imputer will look for the rows similar to our row.
* The similar **number of rows** it will look for will depend on the number of neighbours we specify.
* For finding the similar rows, imputer calculates nan\_Euclidian distance of our row from all the other rows not taking into account the column that has null value in our row. Suppose if we have selected 2 neighbors, the Imputer will select 2 rows having the least nan\_Euclidian distance from our row.

While calculating the nan\_Euclidian distance, our imputer also takes into account if other rows have any null values by assigning weights to the sum of squared distances.

The formula for nan\_Euclidian distance is given below :-



* Now the imputer will look for the column in the neighbour rows which has the missing value in the row where the value has to be imputed. From those rows, the imputer will take the values of that column and find their average and will impute that average value in our row that has the missing value.

**For the visual explanation, one can look into the below video :-**

[Step-by-Step procedure of KNN Imputer for imputing missing values | Machine Learning - YouTube](https://www.youtube.com/watch?v=AHBHMQyD75U)

**Why KNN Imputer was a good choice in our case?**

* We took the variables that have more than 1000 missing values in a data frame and applied KNN Imputer on that data frame only so that we fill relevant values as we also had label encoded nominal categorical variables in our original dataset. This made our work easy as all these variables were of numeric datatype.
* It is a very fast working approach and all the variables, each having thousands of null values got imputed very quickly.
* It is a logical approach as rows nearer to one another with respect to all the other columns must also be nearer to one another with respect to the column having the null value.

* After treatment of null values, we went on and did **variable selection**, with **Mutual Information** method of variable selection.

We will now state what is Mutual Information and how it helped us in variable selection.

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**MUTUAL INFORMATION**

A quantity called mutual information measures the amount of information one can obtain from one random variable given another. (In our context, it measures the amount of information one can obtain from an independent variable, given the dependent variable.)

The mutual information between two random variables X and Y can be stated formally as follows:

* **I(X ; Y) = H(X) – H(X | Y)**

Where I(X ; Y) is the mutual information for X and Y, H(X) is the entropy for X and H(X | Y) is the conditional entropy for X given Y. The result has the units of bits.

Mutual information is a measure of dependence or “mutual dependence” between two random variables. As such, the measure is symmetrical, meaning that I(X ; Y) = I(Y ; X).

* We calculated Mutual Information of all the independent variables with respect to our target variable **total\_cardspent.**
* (We added cardspent and card2spent columns to get total\_cardspent as our target variable).
* We then selected variables having mutual information greater than or equal to 0.01.
* We present all the selected features below :-

['carditems', 'card2items', 'income', 'lninc', 'inccat', 'carvalue',

'lncreddebt', 'card', 'carcatvalue', 'creddebt', 'lnothdebt', 'othdebt',

'age', 'agecat', 'addresscat', 'card2', 'employ', 'card2benefit',

'townsize', 'longten', 'retire', 'empcat', 'jobsat', 'ownvcr',

'lncardten', 'card2tenure', 'owncd', 'owndvd', 'multline', 'tenure',

'address', 'longmon', 'carown', 'spousedcat', 'lntollten',

'card2tenurecat', 'pets\_dogs', 'wireless', 'reason', 'ownfax', 'gender',

'internet', 'forward', 'vote', 'bfast', 'cardtenure', 'news',

'cardtenurecat']

* After the variable selection, we did univariate and bivariate analysis simultaneously of the above important variables. We will

now explain one by one what we did with the variables.

* **townsize (Size of hometown)**

For this column, we ran the following command and calculated average of target variable for every category as it

was a categorical variable.

pd.DataFrame(dfi.groupby(['townsize']).agg({"total\_cardspent":['mean']}))

Running the above command, we saw that there was not much difference in the mean of target variable for

different categories , which meant that it was not bringing much change in our target variable so **we dropped it**.

* **gender**

For this column also like townsize we ran the following command and saw that categories of this variable were

bringing considerable change in the mean of target variable so **we kept this variable as it is**. **(We will replace label**

**encoding of nominal categorical variables with their category names later on).**

pd.DataFrame(dfi.groupby(['gender']).agg({"total\_cardspent":['mean']}))

* **age(Age in years) and agecat(Age category)**

**Since we had agecat , we removed age as two variables representing the same factor doesn’t make sense.**

**For agecat**, we ran the following command and saw that categories of this variable were

bringing significant change in the mean of target variable so **we kept this variable as it is**.

pd.DataFrame(dfi.groupby(['agecat']).agg({"total\_cardspent":['mean']}))

* **birthmonth**

We decided that the month in which customer was born wouldn’t make much difference in the overall credit card

spending by that customer, so **we dropped this variable.**

* **employ (Years with current employer) and employcat**

Since we had employcat (employment years in categories) , **we removed employ as two variables representing the**

**same factor doesn’t make sense.**

**For employcat**, we ran the following command and saw that categories of this variable were

bringing significant change in the mean of target variable so **we kept this variable as it is**.

pd.DataFrame(dfi.groupby([‘employcat']).agg({"total\_cardspent":['mean']}))

* **retire (Retired)**

**For retire**, we ran the following command and saw that categories of this variable were

bringing significant change in the mean of target variable so **we kept this variable as it is**.

pd.DataFrame(dfi.groupby([‘retire']).agg({"total\_cardspent":['mean']}))

* **income (Household income in thousands), lninc (Log-income) and inccat (Income category in thousands)**

**Since we had inccat(income category), we removed income and lninc as keeping different features representing**

**the same factor doesn’t make sense.**

**For inccat**, we ran the following command and saw that categories of this variable were

bringing significant change in the mean of target variable so **we kept this variable as it is**.

pd.DataFrame(dfi.groupby([‘inccat']).agg({"total\_cardspent":['mean']}))

* **creddebt (Credit card debt in thousands) and lncredebt (Log-credit card debt)**

For these two variables we made a distribution plot, using the following commands

sns.distplot(dfi["creddebt"])

sns.distplot(dfi["lncreddebt"])

**After plotting them we saw that lncreddebt was more closer to a normal distribution than creddebt and keeping two variables representing a same factor doesn’t make sense, so we deleted creddebt and kept lncreddebt.**

We now did the percentile analysis of lncreddebt in order to treat the outliers as it is a continuous variable.

We used the following command to calculate percentiles.

np.percentile(dfi["lncreddebt"], range(1,101))

We then treated the outliers with suitable flooring and caping.

* **othdebt (Other debt in thousands) and lnothdebt (Log-Other debt)**

For these two variables we made a distribution plot, using the following commands

sns.distplot(dfi["othdebt"])

sns.distplot(dfi["lnothdebt"])

**After plotting them we saw that lnothdebt was more closer to a normal distribution than othdebt and keeping**

**two variables representing a same factor doesn’t make sense, so we deleted othdebt and kept lnothdebt.**

We now did the percentile analysis of lnothdebt in order to treat the outliers as it is a continuous variable.

We used the following command to calculate percentiles.

np.percentile(dfi["lnothdebt"], range(1,101))

We then treated the outliers with suitable flooring and caping.

* **jobsat (Job satisfaction)**

**For jobsat**, we ran the following command and saw that categories of this variable were

bringing considerable change in the mean of target variable so **we kept this variable as it is**.

pd.DataFrame(dfi.groupby([‘jobsat']).agg({"total\_cardspent":['mean']}))

* **spousedcat (Spouse level of education)**

**For spousedcat**, we ran the following command and saw that categories of this variable were

bringing considerable change in the mean of target variable so **we kept this variable as it is**.

pd.DataFrame(dfi.groupby([‘spousedcat']).agg({"total\_cardspent":['mean']}))

* **carditems (Number of items on primary card last month) and card2items (Number of items on secondary card last month)**

For these two variables we made a regression plot with the target variable using the following commands

sns.regplot(dfi["carditems"], dfi["total\_cardspent"])

sns.regplot(dfi["card2items"], dfi["total\_cardspent"])

We now did the percentile analysis of both the variables in order to treat the outliers as they are continuous

variables.

We used the following command to calculate percentiles.

np.percentile(dfi["carditems"], range(1,101))

np.percentile(dfi["card2items"], range(1,101))

We then treated the outliers with suitable flooring and caping.

* **carvalue (Primary vehicle sticker price)**

For carvalue, we made a regression plot with the target variable using the following command

sns.regplot(dfi["carvalue"], dfi["total\_cardspent"])

We now did the percentile analysis of lnothdebt in order to treat the outliers as it is a continuous variable.

We used the following command to calculate percentiles.

np.percentile(dfi["carvalue"], range(1,101))

There were no outliers so no flooring and caping was required.

* **carcatvalue (Primary vehicle price category)**

Since there was significant difference between carvalue and carcatvalue, we are keeping both

**For carcatvalue**, we ran the following command and saw that categories of this variable were

bringing considerable change in the mean of target variable so **we kept this variable as it is**.

pd.DataFrame(dfi.groupby([‘carcatvalue']).agg({"total\_cardspent":['mean']}))

* **card (Primary credit card) and card2 (Secondary credit card)**

**For card and card2**, we ran the following commands and saw that categories of these variables were

bringing considerable change in the mean of target variable so **we kept this variable as it is**.

pd.DataFrame(dfi.groupby([‘card']).agg({"total\_cardspent":['mean']}))

pd.DataFrame(dfi.groupby([‘card2']).agg({"total\_cardspent":['mean']}))

For card variable, there were certain categories that were not bringing considerable change in mean target variable

This we will address while naming the categories later on.

* **address (Years at current address) and addresscat**

**Since we have addresscat, we will remove address.**

**For addresscat**, we ran the following command and saw that categories of this variable were

bringing considerable change in the mean of target variable so **we kept this variable as it is**.

pd.DataFrame(dfi.groupby([‘addresscat']).agg({"total\_cardspent":['mean']}))

* **tenure (Number of months with service)**

For tenure, we made a regression plot with the target variable using the following command

sns.regplot(dfi["tenure"], dfi["total\_cardspent"])

We now did the percentile analysis of tenure in order to treat the outliers as it is a continuous variable.

We used the following command to calculate percentiles.

np.percentile(dfi["tenure"], range(1,101))

There were no outliers so no flooring and caping was required.

* **card2benefit (Benefit program for secondary credit card)**

**For carcatvalue**, we ran the following command and saw that categories of this variable were

bringing considerable change in the mean of target variable so **we kept this variable as it is**.

pd.DataFrame(dfi.groupby([‘carcatvalue']).agg({"total\_cardspent":['mean']}))

There were certain categories that were not bringing a significant change in the mean of target variable, which we will address later on.

* **longten (Long distance over tenure)**

For longten, we made a regression plot with the target variable using the following command

sns.regplot(dfi["longten"], dfi["total\_cardspent"])

We now did the percentile analysis of tenure in order to treat the outliers as it is a continuous variable.

We used the following command to calculate percentiles.

np.percentile(dfi["longten"], range(1,101))

Suitable caping and flooring was done to remove the outliers.

* **card2tenure (Years held secondary credit card) and card2tenurecat**

Since we had card2tenurecat, we removed card2tenure

**For card2tenurecat**, we ran the following command and saw that categories of this variable were

bringing considerable change in the mean of target variable so **we kept this variable as it is**.

pd.DataFrame(dfi.groupby([‘card2tenurecat']).agg({"total\_cardspent":['mean']}))

* **ownvcr, owncd, owndvd, multiline, carown, vote, wireless, reason, ownfax, forward, news, internet, bfast**

|  |  |
| --- | --- |
| **ownvcr** | **Owns VCR** |
| **owncd** | **Owns stereo/CD player** |
| **owndvd** | **Owns DVD player** |
| **multiline** | **Multiple lines** |
| **carown** | **Primary vehicle lease/own** |
| **vote** | **Voted in last election** |
| **wireless** | **Wireless service** |
| **reason** | **Primary reason for being a customer here** |
| **ownfax** | **Owns fax machine** |
| **forward** | **Call forwarding** |
| **news** | **Newspaper subscription** |
| **internet** | **Internet** |
| **bfast** | **Preferred breakfast** |
|  |  |

All the above variables are categorical and thus have been treated in the same way

**For all the mentioned variables**, we ran the following command and saw that categories of these variables were

bringing considerable change in the mean of target variable so **we kept these variable as it is**.

pd.DataFrame(dfi.groupby([‘**variable\_name**']).agg({"total\_cardspent":['mean']}))

* **lncardten (Log-calling card over tenure)**

For lncardten, we made a regression plot with the target variable using the following command

sns.regplot(dfi["lncardten"], dfi["total\_cardspent"])

We now did the percentile analysis of tenure in order to treat the outliers as it is a continuous variable.

We used the following command to calculate percentiles.

np.percentile(dfi["lncardten"], range(1,101))

Suitable caping and flooring was done to remove the outliers.

* **longmon (Long distance last month)**

For longmon, we made a regression plot with the target variable using the following command

sns.regplot(dfi["longmon"], dfi["total\_cardspent"])

We now did the percentile analysis of tenure in order to treat the outliers as it is a continuous variable.

We used the following command to calculate percentiles.

np.percentile(dfi["longmon"], range(1,101))

Suitable caping and flooring was done to remove the outliers.

* **cardtenure (Years held primary credit card) and cardtenurecat**

Since we have cardtenurecat we removed cardtenure

**For cardtenurecat**, we ran the following command and saw that categories of this variable were

bringing considerable change in the mean of target variable so **we kept this variable as it is**.

pd.DataFrame(dfi.groupby([‘cardtenurecat']).agg({"total\_cardspent":['mean']}))

* **lntollten (Log-toll free over tenure)**

For lntollten, we made a regression plot with the target variable using the following command

sns.regplot(dfi["lntollten"], dfi["total\_cardspent"])

We now did the percentile analysis of tenure in order to treat the outliers as it is a continuous variable.

We used the following command to calculate percentiles.

np.percentile(dfi["lntollten"], range(1,101))

Suitable caping and flooring was done to remove the outliers.

* **total\_cardspent (Total amount spent on primary card + secondary card last month)**

**This is our target variable.**

We did the percentile analysis in order to treat the outliers as it is a continuous variable.

We used the following command to calculate the percentiles.

np.percentile(dfi["total\_cardspent"], range(1,101))

Suitable caping and flooring was done to remove the outliers.

* After doing the variable analysis, **we replaced the label encoding of all the nominal categorical variables with there category**

**names.**

The below mentioned variables were replaced by their original categories:-

card2, retire, ownvcr, owncd, owndvd, multiline, carown, vote, wireless, reason, ownfax, gender, forward, news, internet

For the below mentioned variables, since certain categories were not bringing major change in the category wise mean value of

the target variable, we clubbed those categories:-

* card

For card variable, we included “Visa” and “Mastercard” in the “Other” category.

* card2benefit

For card2benefit, we included “Cash back” in the “Other” category

* bfast

For bfast, we took “Energy bar” and “Oatmeal” and made a new category “Others”

* Now we did the **Multivariate Analysis** in the following way :-
* We calculated correlation of every variable with one another with dfi.corr().
* We then made a heatmap with the resulting data frame.
* We transferred the resulting data frame into Microsoft Excel for treating the **Autocorrelation.**
* As a treatment for Autocorrelation, we dropped the following variables:-

"carvalue", "agecat", "longmon", "cardtenurecat", "longten", "lncardten", "lntollten", "tenure"

NOTE: Here we would like to mention that since we calculate Variance Inflation Factor for all the variables later on,

We can also treat the Autocorrelation at that step.

* After doing the multivariate analysis, **we checked the distribution of our target variable “total\_cardspent”**. As we know that our

target variable must be close to normal distribution to run regression effectively.

Since our target variable was not close to normal distribution, we tried square root and log transformations of the variable

and found log transformation to be the closest to normal distribution.

We saved the log transformation of our target variable into a column in our data frame, named **“log\_TotalCardspent” .**

* After transformation of the target variable, we went on to do the **dummy variable creation for the nominal categorical**

**variables**.

* We first of all made a dataframe with all nominal categorical variables.
* We then applied pd.get\_dummies(drop\_first= True) command in order to form dummy variables.
* After creation of dummy variables , we went on to **scale our continuous variables and ordinal categorical variables**, (both of

them were in numeric form).

* We first of all made a dataframe with all the numeric variables
* We then applied MinMax Scaler to scale our variables.
* After scaling the continuous variables we **concatenated the dataset in which we created dummy variables with the dataset in**

**which we scaled our numeric variables.**

* We now went on to **create our regression models.** In which we performed following steps :-
* We first defined our x variable that contained all the independent variables. So we took our concatenated

dataframe after dropping the following columns:-

**"cardspent","card2spent","total\_cardspent", "log\_TotalCardspent"**

* We then defined our y variable that contained our target variable **"log\_TotalCardspent".**
* We then went on to split our dataset into training and testing data.
* We then calculated the **Variance Inflation Factor(VIF)** for all the independent variables

**---------------------------------------------------------------------------------------------------------------------------------------------------**

**Variance Inflation Factor**

**Variance inflation factor** (**VIF**) is a measure of the amount of multicollinearity in a set of multiple regression

variables. Mathematically, the **VIF** for a regression model variable is equal to the ratio of the overall model

**variance** to the **variance** of a model that includes only that single independent variable.

**----------------------------------------------------------------------------------------------------------------------------------------------------**

Basically variance inflation factor is a check of autocorrelation between the variables.

We removed the variables having VIF >10 one by one, removing the variables having lower correlation with our

target variable first.

After the VIF analysis we removed the following variables from our variable x which contained all the independent

variables :-

**"carcatvalue", "lncreddebt", "owncd\_Yes", "ownvcr\_Yes","card2tenurecat", "owndvd\_Yes",**

**"reason\_No response"**

* We then went on to create our regression models applying Grid Search CV and Randomized Search CV where ever

applicable. We achieved following results with train and test data.

|  |  |  |
| --- | --- | --- |
| **MODELS** | **TRAIN DATA**  **(R sq, Adjusted R sq, RMSE)** | **TEST DATA**  **(R sq, Adjusted R sq, RMSE)** |
| **SKLEARN OLS** | **0.643, 0.639, 0.213** | **0.605, 0.594, 0.222** |
| **SGD REGRESSOR WITH GS CV** | **0.643, 0.640, 0.213** | **0.606, 0.595. 0.222** |
| **SGD REGRESSOR WITH RS CV** | **0.621, 0.618, 0.22** | **0.587, 0.575, 0.23** |
| **DECISION TREE REGRESSOR**  **WITH RS CV** | **0.532, 0.527, 0.244** | **0.455, 0.439, 0.261** |
| **DECISION TREE REGRESSOR**  **WITH GS CV** | **0.559, 0.554, 0.237** | **0.477, 0.462, 0.256** |
|  |  |  |
| **RANDOM FOREST REGRESSOR**  **WITH RS CV** | **0.608, 0.604, 0.223** | **0.535, 0.521, 0.241** |
| **K NEAREST REGRESSOR**  **WITH GS CV** | **0.312, 0.305, 0.296** | **0.209, 0.185, 0.315** |
| **SUPPORT VECTOR REGRESSOR**  **WITH RS CV** | **0.643, 0.639, 0.213** | **0.608, 0.596, 0.222** |
| **XG BOOST REGRESSOR**  **WITH RS CV** | **0.666, 0.662, 0.206** | **0.624, 0.613, 0.217** |

**IN THE ABOVE TABLE :-**

* Abbreviations

R sq = R Squared i.e. Coefficient of Determination

RMSE = Root Mean Squared Error

GS CV = Grid Search Cross Validation

RS CV = Randomized Search Cross Validation

SGD = Stochastic Gradient Descent

* With **XG Boost Regressor** we achieved the highest R Squared and lowest RMSE so we take it as our final model