CAPSTONE PROJECT
BUSINESS REPORT

Vivek Sagar

1 October 2023

Mentor: Abhay Poddar

CONTENTS:

1) Introduction to the business problem

In the realm of financial risk assessment, the probability of credit card default holds paramount importance. This project delves into the predictive power of data analysis, employing python as a tool to analyse historical record data. By crafting a robust model, we aim to accurately forecast the likelihood of credit card holders defaulting on payments.

A) Problem Statement

This business problem is a supervised learning example for a credit card company. The objective is to predict the probability of default (whether the customer will pay the credit card bill or not) based on the variables provided. There are multiple variables on the credit card account, purchase and delinquency information which can be used in the modelling.

PD modelling problems are meant for understanding the riskiness of the customers and how much credit is at stake in case the customer defaults. This is an extremely critical part in any organisation that lends money [both secured and unsecured loans].

B) Need of the project

Credit card companies operate in a landscape where minimising risk and maximising profitability is crucial. Then analysis of credit card default probability play is a pivotal role in achieving this balance. By accurately assessing the likelihood of default, companies can proactively manage potential losses, optimise credit limits and tailor interest rates for individual customers. This analysis empowers companies to make informed decisions, enhance customer relationships and maintain a healthier bottom line. As a data analyst, delving into this realm provides an opportunity to contribute to the industry's stability and success.

2) EDA and Business Implication

A) Univariate analysis

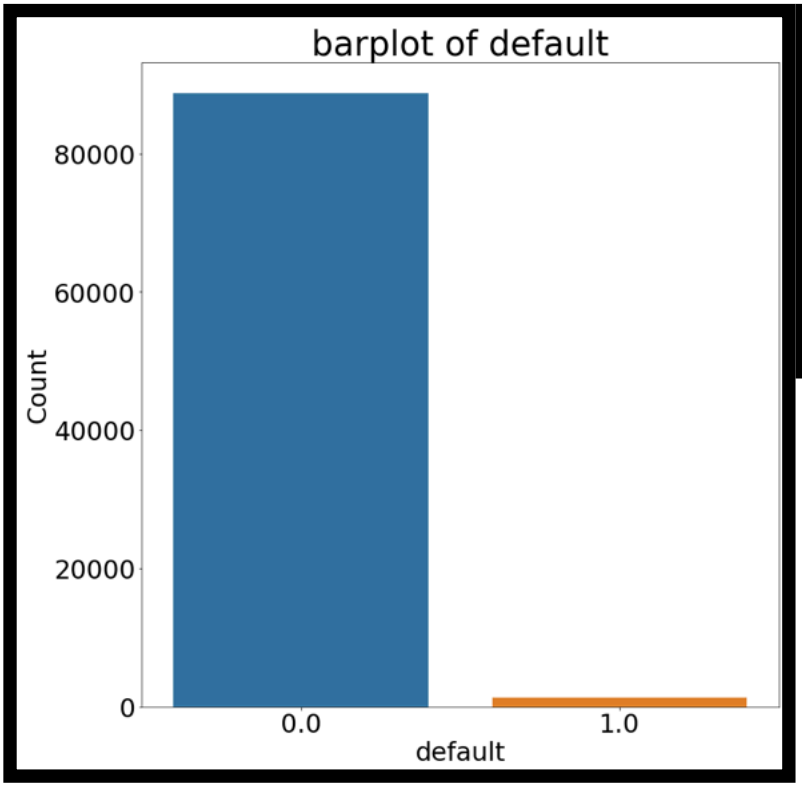
Since we have a lot of variables in the dataset, only the most important ones are shown here, rest can be found in the attached python Jupyter notebook (.ipynb file).

Distribution of data for categorical variables:

We have used a combination of **bar-plot** and **value counts** to represent the categorical variables.

Default:

- 'Default' is the target variable. As we can see it is highly imbalanced with only 1288 defaulters (class 1) out of total 99979 credit card users, composing only 1% of the dataset.

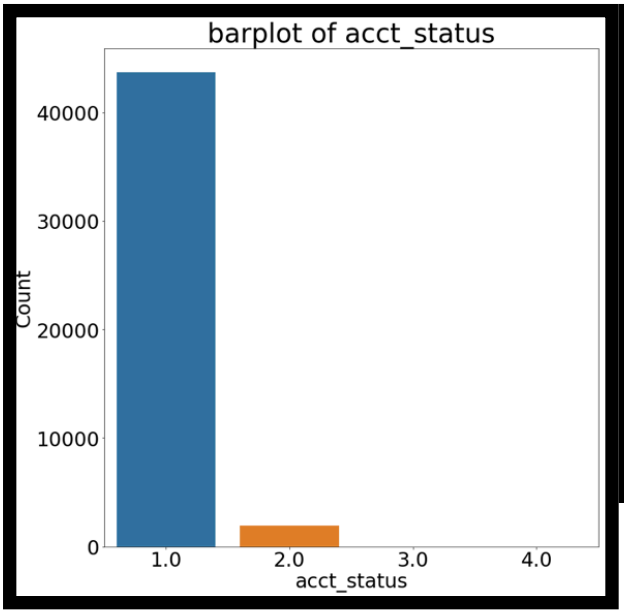



```
Count of default
--------------------------------
default
0.00    88688
1.00     1288
Name: count, dtype: int64

Proportion of default
--------------------------------
default
0.00   0.99
1.00   0.01
Name: proportion, dtype: float64
```

acct_status:

- It represents account status of the credit card user. According to the data dictionary it is supposed to have binary values 0 and 1, with 0 representing inactive account status and 1 representing the account as active.
- But as we can see in the dataset, it is showing 4 categories namely 1,2,3 and 4.
- Category 1 constitutes the majority with 96% of the total values, category 2 making 4% of the data. Category 3 and 4 have negligible values.
- We will need to delve deeper or refer to the company to confirm about the nature of these unknown categories. Most likely these are data feeding mistakes.

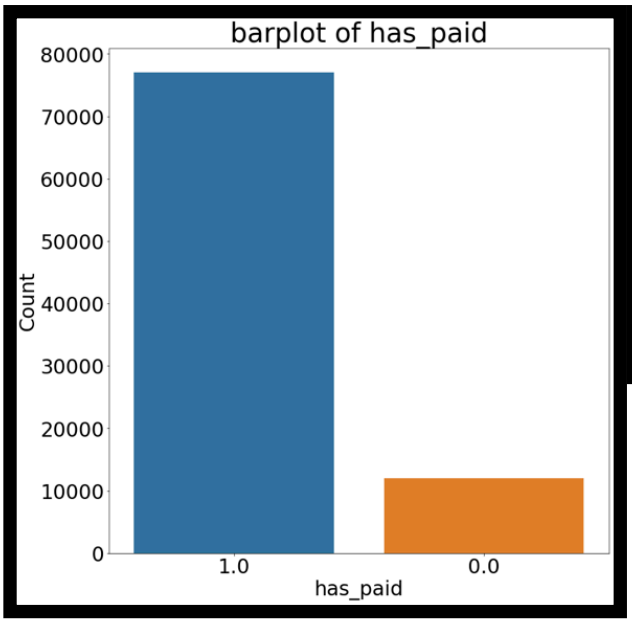



```
Count of acct_status
--------------------------------
acct_status
1.00    43693
2.00     1900
3.00        7
4.00        3
Name: count, dtype: int64

Proportion of acct_status
--------------------------------
acct_status
1.00   0.96
2.00   0.04
3.00   0.00
4.00   0.00
Name: proportion, dtype: float64
```

has_paid:

- It shows whether the customer has paid the current credit card bill or not.
- 87% of the users have paid the current credit card bill while 13% haven't.

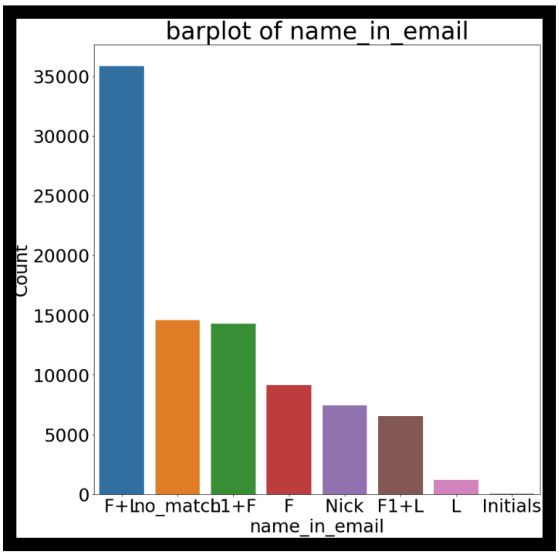



```
Count of has_paid
--------------------------------
has_paid
1.00    77002
0.00    11940
Name: count, dtype: int64

Proportion of has_paid
--------------------------------
has_paid
1.00   0.87
0.00   0.13
Name: proportion, dtype: float64
```

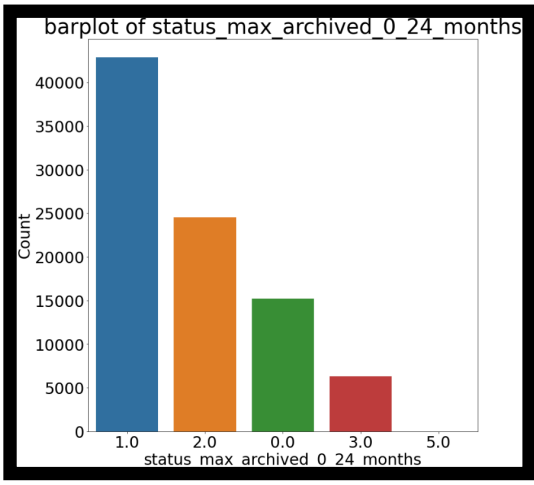
name_in_email:

- Here looking at the bar plot, we have an anomaly, as there seems to be a large scale data feeding error.
- We will have to refer to the company to know the logic behind the naming as they are getting shown in terms of F and L.
- Overall this variable is not making any sense as it is not showing name of the credit card users, hence we will drop this variable before moving on to model building.



status_max_archived_0_24_months:

- It shows maximum number of times the account was in archived status in last 24 months i.e. last 2 years.
- Here looking at the bar plot, 48% of users have gone into archived status at least once while 17% of the users have never gone into archive status in last 2 years.
- 28% of the users have gone twice into archived status and only 7% of the users have gone thrice into the archive the status in last 24 months.
- There is also a category named 5, denoting that the number of users that have gone into account the status 5 times, in last 24 months is 16 which is negligible.

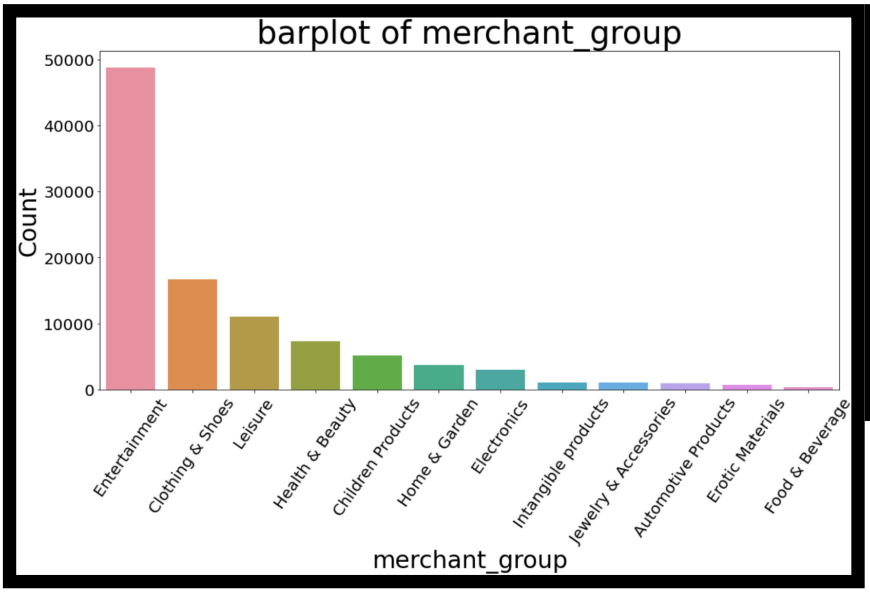



```
Count of status_max_archived_0_24_months
-----------------------------------------
status_max_archived_0_24_months
1.00    42847
2.00    24534
0.00    15204
3.00     6341
5.00       16
Name: count, dtype: int64

Proportion of status_max_archived_0_24_months
-----------------------------------------
status_max_archived_0_24_months
1.00   0.48
2.00   0.28
0.00   0.17
3.00   0.07
5.00   0.00
Name: proportion, dtype: float64
```

merchant_category and merchant group:

- There is not much given about these two variables in the data dictionary, so we will assume that these represent the category and group where users have transacted the most using their credit card.
- According to the assumption made above we can see that, "**Entertainment**" overall is the most transacted merchant group with "**clothing & shoes**" coming at the second place.
- "Diversified Entertainment" is the most transacted merchant category"Youthful shoes and clothing" coming at the second place.




```
Proportion of merchant_group
--------------------------------
merchant_group
Entertainment           0.49
Clothing & Shoes        0.17
Leisure                 0.11
Health & Beauty         0.07
Children Products       0.05
Home & Garden           0.04
Electronics             0.03
Intangible products     0.01
Jewelry & Accessories   0.01
Automotive Products     0.01
Erotic Materials        0.01
Food & Beverage         0.00
Name: proportion, dtype: float64
```

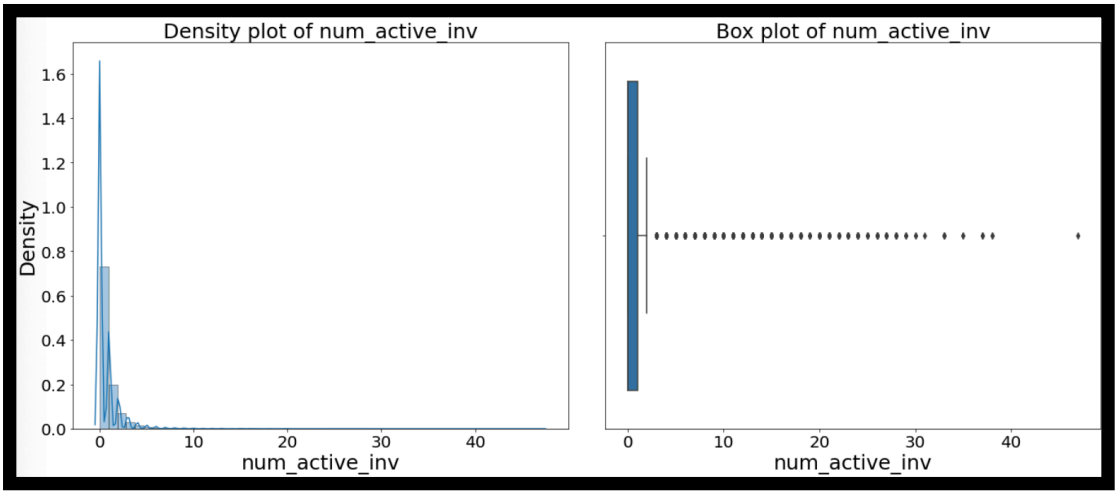
Distribution of data for continuous variables:

- We have used a combination of density plot and box plot to represent the continuous numeric variables.
- Density plot was used instead of histogram because histogram takes longer execution time and density plot overall is a better and improved version of histogram.
- Since we have a lot of continuous variables in the dataset, only the most important ones are shown here, rest can be found in the attached python Jupyter notebook (.ipynb file).

The plot combinations and inferences are shown below:

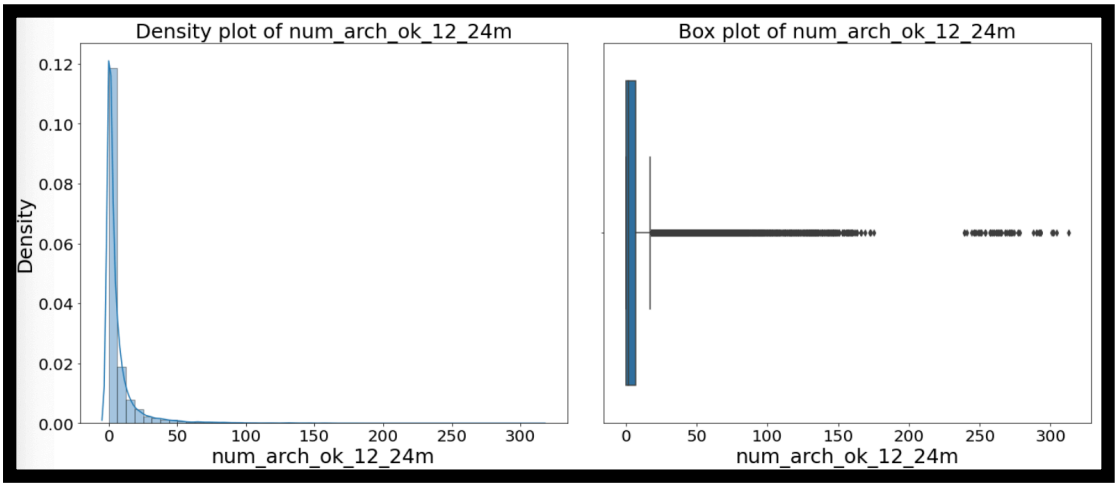
num_active_inv:

- This represents Total number of active invoices per user, I.e. Unpaid bills.
- Looking at the proportion of this variable, 69% users have no active invoices which is a good thing.
- Here, the distribution is highly right skewed. Majority of active invoices count fall in the range of 1 to 5 but there are a lot of outliers with few users having more than 30 active invoices.



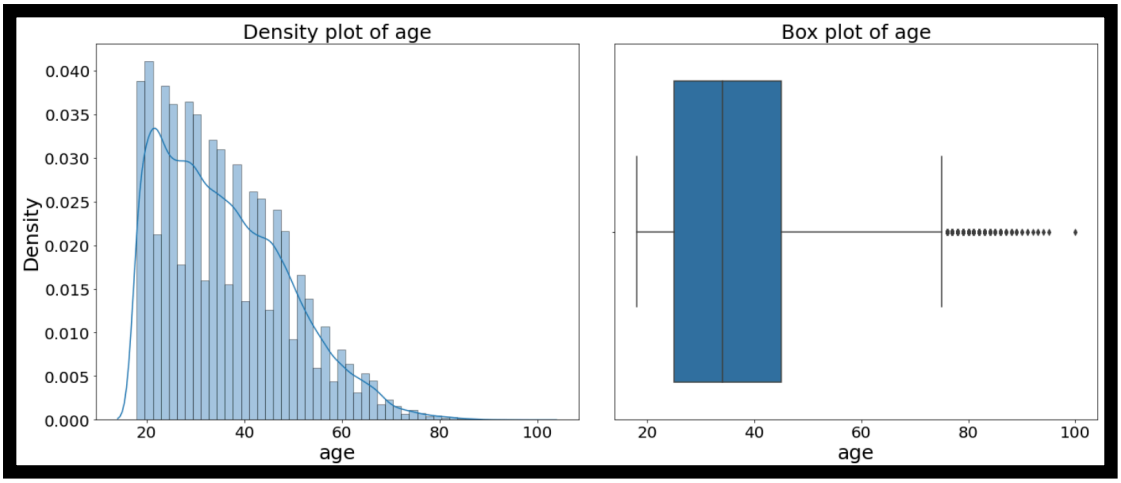
num_arch_ok_12_24m:

- This represents number of archived purchases that were paid between 24 months in the past to the present date and 12 months in the past to the present date.
- Here also, the distribution is highly right skewed with the right tail almost reaching negligible values around 50. But some users have, as high as 300 archived purchases paid during this period I.e. there are a lot of outliers,



Age:

- Almost 95% of the users fall in the range of 20 years old to around 80 years old.
- Though there are outliers present in this too, it is relatively less compared to other continuous variables.



Inferences from Univariate analysis

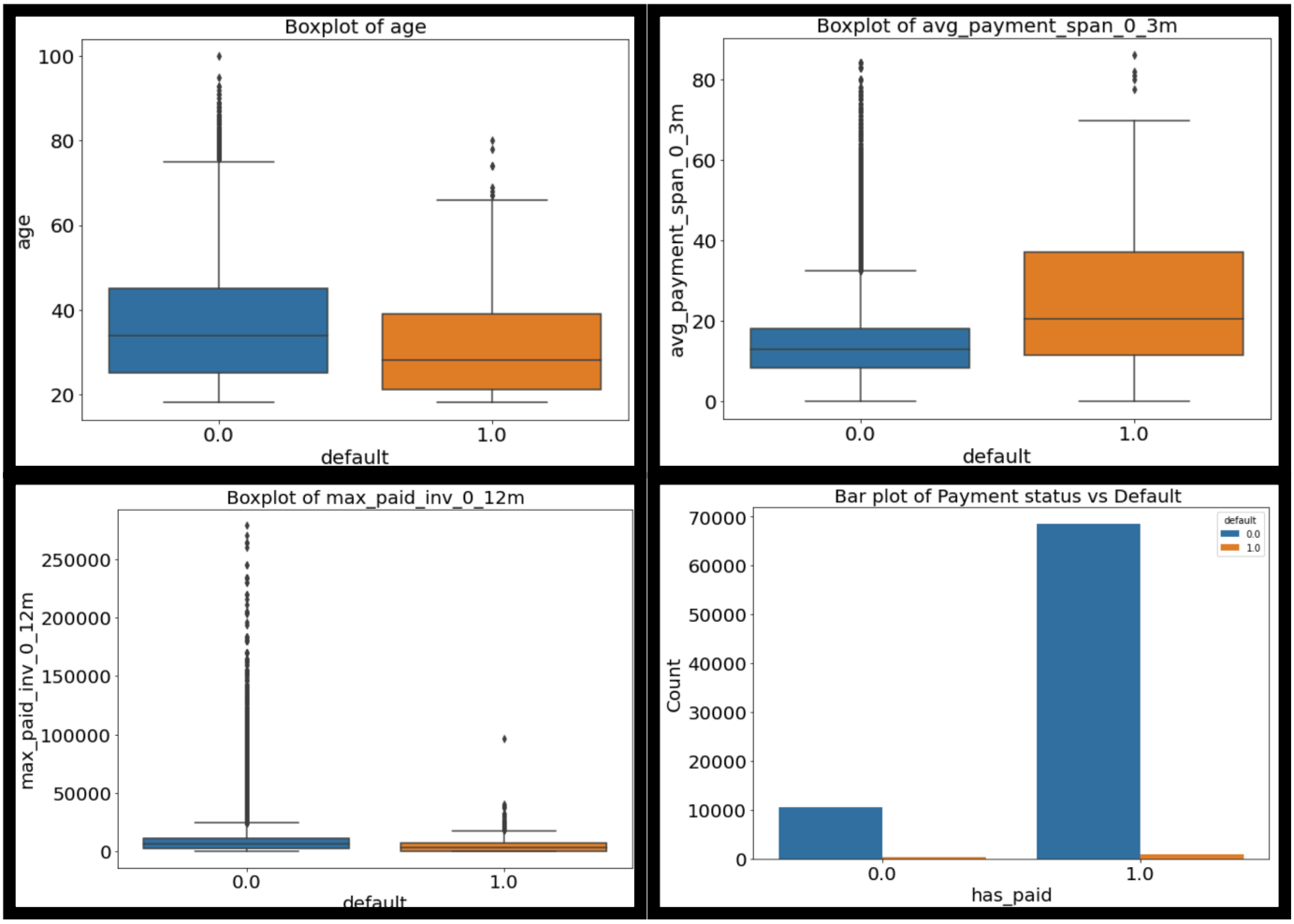
There seems to be some abnormalities in the dataset:

- It's worth noting that the second last row has all the values missing, hence we will look for rows with a lot of missing values and drop such rows since imputing large amounts of missing values row-wise will distort our dataset.
- Variable "acct_status" should only have 0 and 1 as values as per the data dictionary, but in the actual dataset it has 1,2,3 and 4 as values. We need to find the reasoning behind this nomenclature or else drop this variable too if the logic cannot be found.
- we will also drop such variables which are not required for our model building namely 'userid', 'name_in_email'.
- Almost all the continuous numeric variables have highly skewed boxplots, with only a few exceptions.
- There are a lot of Outliers in almost all the variables. These will need to be treated.
- Another thing to note is that variables are on different scales, so some form of scaling will be needed to be done on the dataset.
- For this we will be using Standard-Scaler feature from scikitlearn library of Python.

B) Bivariate and Multivariate analysis against the target variable

- Since defaulters make only 1% of the dataset, bar plot comparison of categorical variables with the target variable is unlikely to produce any meaningful insights.
- For checking the correlation between independent variables we will be using heatmap instead of scatter-plots, reason being heat-map are better at quantifying the correlation.
- Due to the dataset having large number of numerical variables, the pair-plot becomes illegible as it produces all the possible combination of correlated variables in one single visualisation.
- Due to this, we use heat-maps as our main tool to check correlation between the independent continuous variables as here we can identify the degree of correlation just by looking at colour of the individual boxes.

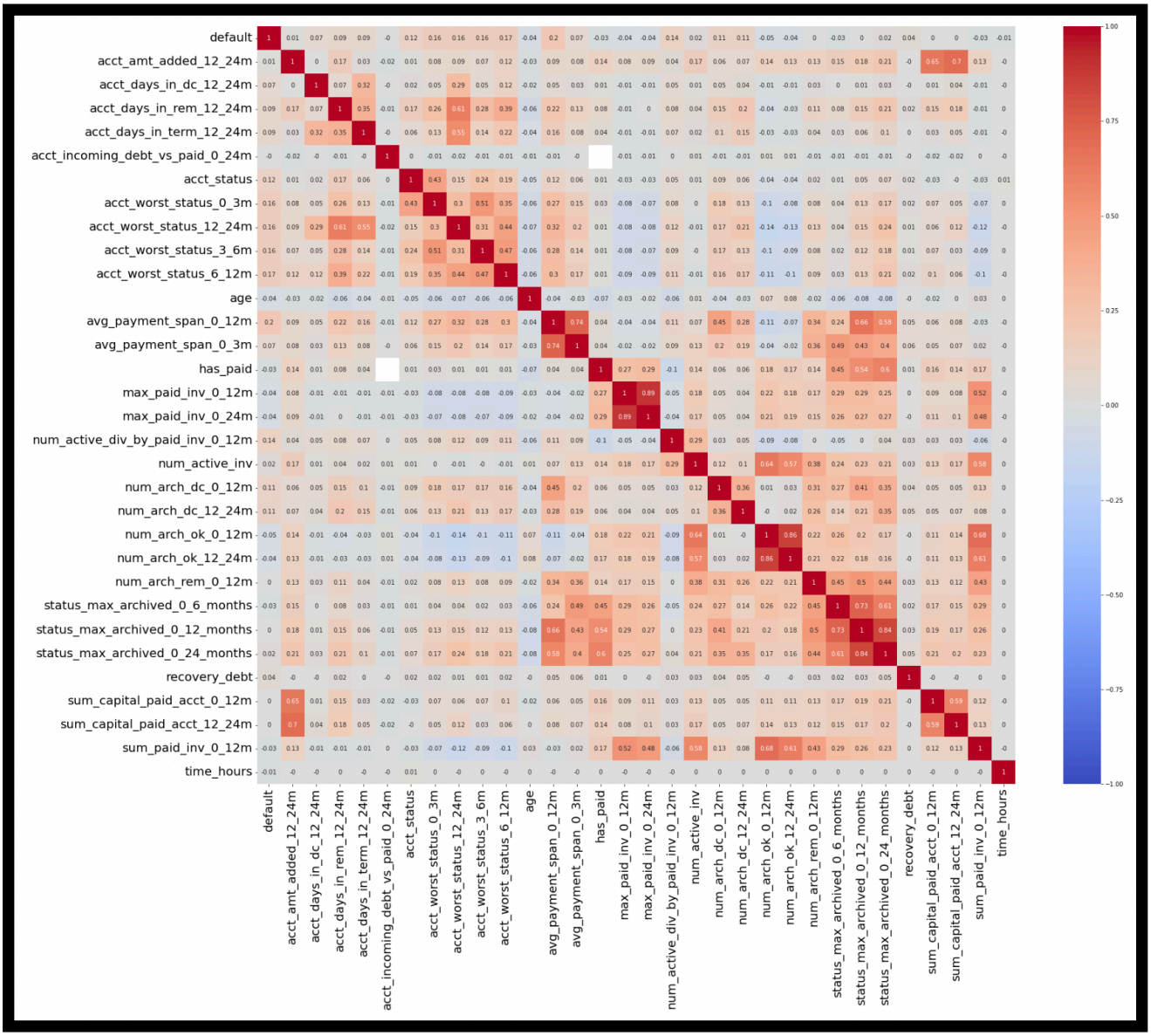
The plot combinations and inferences are shown below:



Inferences from Bivariate analysis

- **Age vs default:** As we can see, mean age of defaulters is around 30 yrs whereas that of non defaulters is close to 40 yrs meaning younger users in the age group of 20 to 30 yrs are more likely to default.
- **Average payment span in last 3 months vs default:** Here, on an average, defaulters have taken around 20 days to make the payment of their credit card bill while non defaulters on an average have taken around 15 days.
- Also there are a lot of outliers in this variable specially in case of non-defaulters, some taking as long as 80 days to pay the bill.
- This means that in general users who take more than 15 days to make payment for their credit card bills are more likely to default and should be prioritised over, with respect to enquiry calls and payment intimation mails.
- But still this cannot be taken too seriously due to the presence of outliers in non-defaulters.
- **Maximum bill amount payment in last 12 months vs default:** Here, we see that, there is not much difference between defaulters and non-defaulters. Except for the fact that there are very few outliers in case of defaulters and a lot of outliers in case of non-defaulters.
- One reason for this can be the fact that non defaulters may be more well to do and therefore are not afraid of spending more in contrast to defaulters.
- **Current payment status vs default:** Here, with respect to default, there is not much difference, as there is similar proportion of defaulters among both the users who have payed the current credit card bill and the users who haven't.

Heat map:



Inferences from Multivariate analysis

- As we can see in the heat map, there are a few variables that are highly correlated.
- For example, 'avg_payment_span_0_12m' with 'avg_payment_span_0_3m' I.e. average payment span of last 12 months to average payment span of last 3 months.
- Another highly correlated pair is max paid invoice in last 12 months to max paid invoice in last 24 months.
- This shows that there is multicollinearity present in the data and thus we will have to reduce it during model building.
- For this, we will use **VIF(Variance Inflation Factor)** technique.
- VIF is a statistical measure used in regression analysis to assess multicollinearity among independent variables. It measures the degree to which the accuracy of a regression model is affected when independent variables are correlated with each other.

C) Summary of Business insights

- The dataset was highly imbalanced I.e. out of almost one hundred thousand users present in the dataset, only 1% were defaulters.
- Such high data imbalance is not suitable for model building even though the domain is such that data imbalance is a given.
- To solve this issue we will have to use a resampling technique like SMOTE to balance the target class I.e. “Defaults”
- The dataset contained a lot of uncleaned data, there were 7 variables present which had more than 50% of data as missing while one variable had 29% data as missing.
- Remaining variables had around 11% of the data as missing, barring few exceptions.
- Most of the variables were highly right skewed with a lot of Outliers in the data.
- Due to this high degree of noise and unclean data, lot of variables were lost while cleaning the data and we were finally left with just 14 variables out of the original 36.
- Further detailed to the point business insights will be available after model building in the later phases.

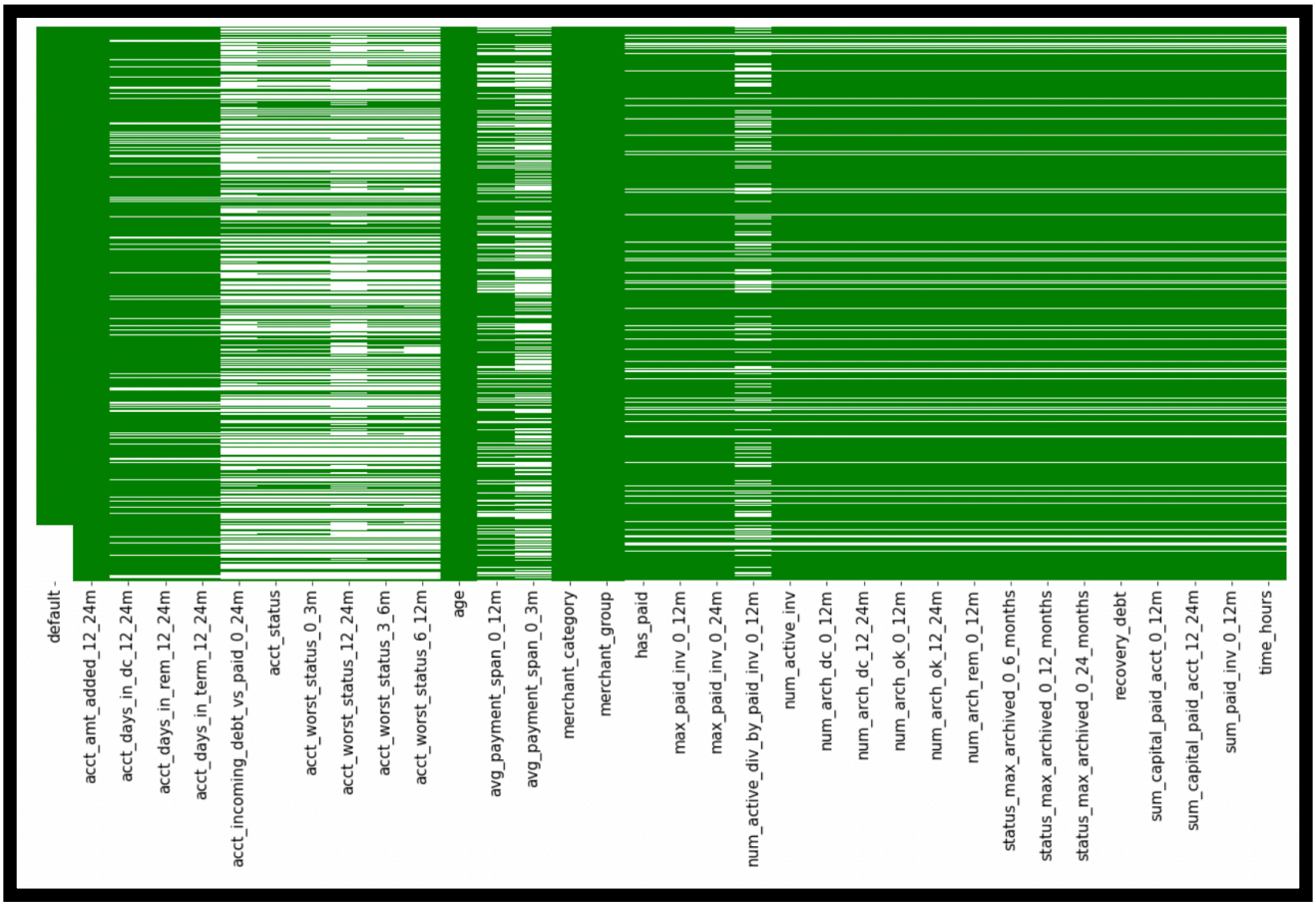
D) Business implications

- As we can see, the dataset is highly imbalanced with only 1288 defaulters (class 1) out of total 99979 credit card users, composing only 1% of the dataset.
- The credit card company should try to add more defaulters in coming future to this existing dataset so as to make it more balanced, as balanced datasets lead to more robust model building.
- There is mis-representation of data in the variable ‘Account status’, ‘name of users in email’ and all the variables related to ‘number of days the account was in worst status for different periods of time’.
- Hence, the company should look into this and take corrective measures.
- Based on the ‘merchant group’ variable, ‘Entertainment’ is the most transacted group, almost 50% users have some transaction related to entertainment.
- Hence the credit card company should target users who have more number of transactions or high value transactions in the entertainment group.
- With respect to ‘age’, younger users in the age group of 20 to 30 yrs are more likely to default and thus should be prioritised by the credit card company.
- ‘Maximum bill amount payed in last 12 months' as well as ‘Current payment status’ i.e whether the user has paid the current credit card bill, has almost no effect on whether the user will default or not.

3) Data Cleaning and Pre-processing

A) Missing Value treatment

Visually inspecting the missing values in the dataset using heatmap (white bars show missing values)



- **There are a lot of missing values in the dataset.**
- Total missing values is around 20% of the whole dataset. Therefore we should be quite careful as to how we are going to treat these missing values.
- Among them we should drop the variables which have at least 30% of the column as missing values, as imputing such large number of missing values will lead to what we call as an Analyst bias.
- It refers to an approach related bias, since missing values imputation differ from analyst to analyst and there is no sure shot way to do it.
- **After dropping the variables with more than 30% missing values, we are left with 24 independent variables.**

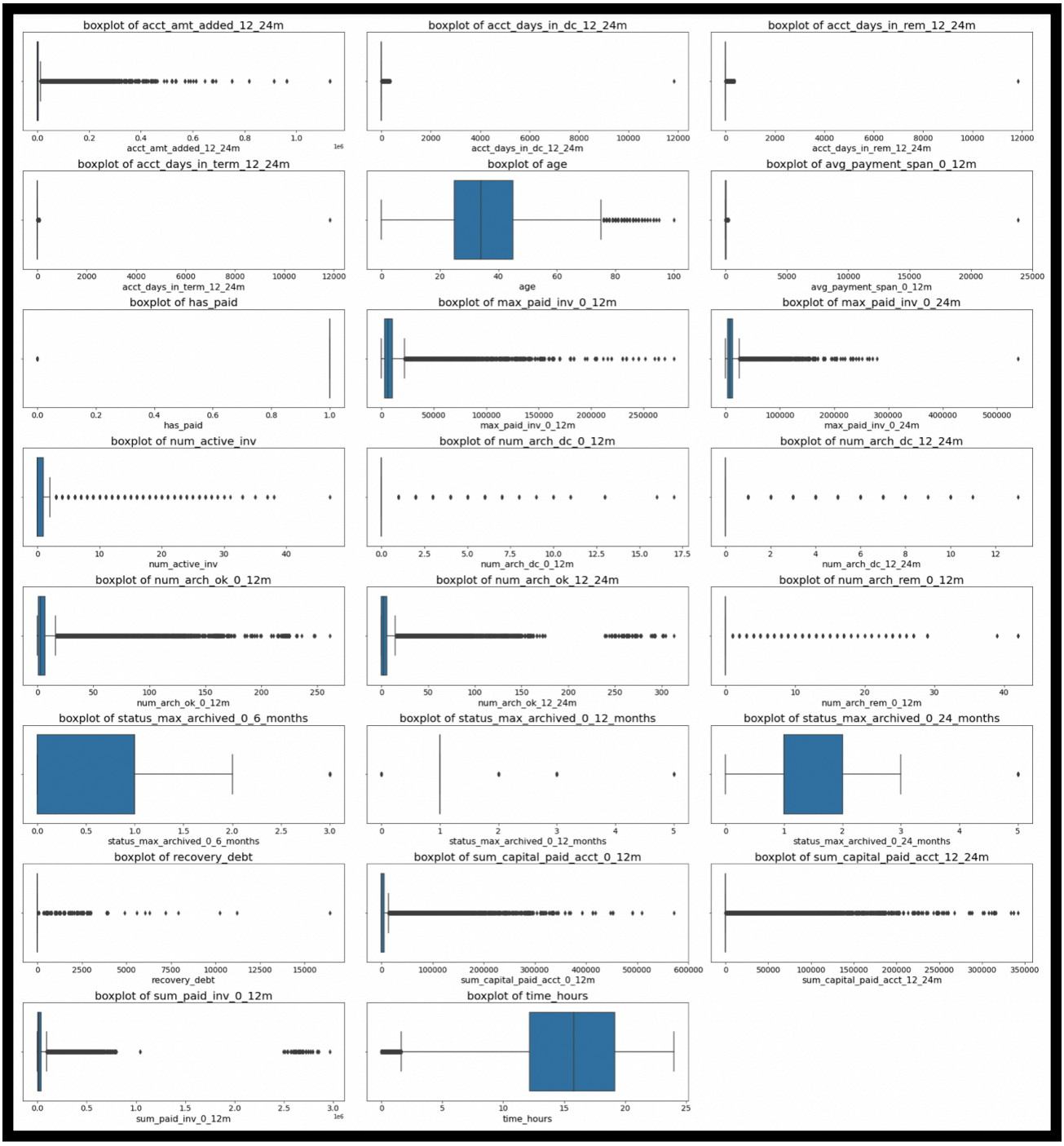
Imputing the remaining missing values:

- Since the number of missing values in the 'merchant_category' and 'merchant_group', we will simply drop those missing values.
- The target variable 'default' has around 10% values as missing, so while imputing the remaining missing values, we will impute the target variable using mode value i.e. 0 in this case and resample the data using “SMOTE” later on.
- For other continuous variables we will be imputing the missing values with median value as there are outliers present in the data.
- **As we can see, missing values from all the variables have been treated:**

```
default                                0
acct_amt_added_12_24m                  0
acct_days_in_dc_12_24m                 0
acct_days_in_rem_12_24m                0
acct_days_in_term_12_24m               0
age                                    0
avg_payment_span_0_12m                 0
merchant_category                      0
merchant_group                         0
has_paid                               0
max_paid_inv_0_12m                     0
max_paid_inv_0_24m                     0
num_active_inv                         0
num_arch_dc_0_12m                      0
num_arch_dc_12_24m                     0
num_arch_ok_0_12m                      0
num_arch_ok_12_24m                     0
num_arch_rem_0_12m                     0
status_max_archived_0_6_months         0
status_max_archived_0_12_months        0
status_max_archived_0_24_months        0
recovery_debt                          0
sum_capital_paid_acct_0_12m            0
sum_capital_paid_acct_12_24m           0
sum_paid_inv_0_12m                     0
time_hours                             0
dtype: int64
```

B) Outlier treatment

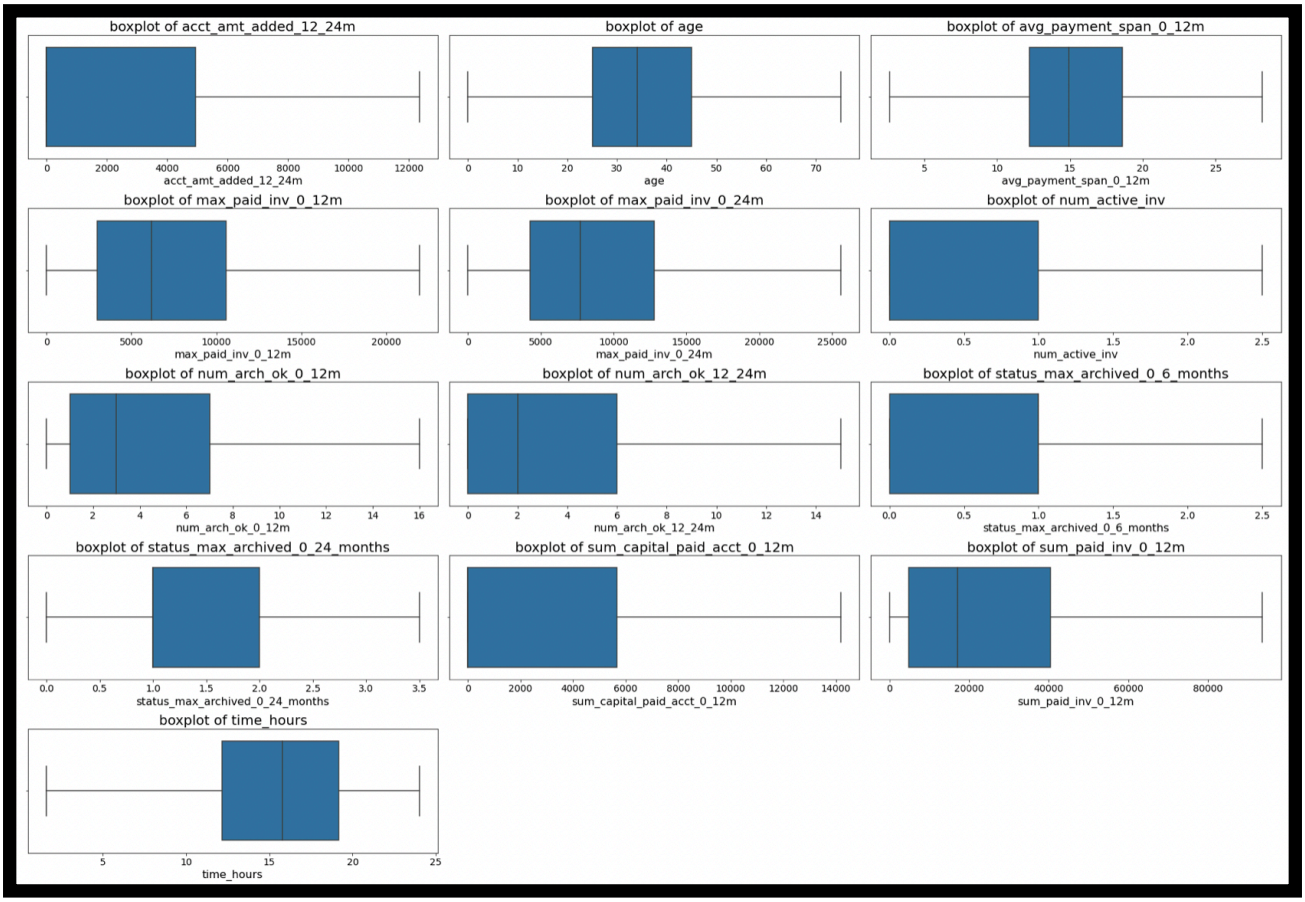
Checking the outliers per column using boxplot:



- All the variables have outliers except 'time_hours'.
- Some variables have one or two outliers while some have a large number of outliers.
- Thus, we are going to use Inter-quartile-range (IQR) method for treating the outliers.

All the outliers got treated using the IQR method, but some of the variables were found to have only zeroes left while some had only ones left after the outlier treatment. Therefore after dropping such variables, finally we were left with 14 variables.

Checking the final boxplots:



C) Variable transformation

Since the variables present in the dataset are on different scales, we needed to bring them to the same scale as we had performed K-means Clustering as part of our project, which is distance based algorithm. For this we used Standard-scaler from scikitlearn library of Python.

Standard-scaler basically does z-score scaling on the dataset, bringing the mean to 0 and standard deviation to 1.

After scaling here is a look of top 5 rows of the scaled dataset:

	acct_amt_added_12_24m	age	avg_payment_span_0_12m	max_paid_inv_0_12m	max_paid_inv_0_24m	num_active_inv	num_arch_ok_0_12m	num_arch_ok_1
0	-0.59	-1.24	-0.49	2.25	2.20	2.12	1.61	
1	-0.59	1.08	1.58	0.97	0.60	-0.56	0.84	
2	-0.59	-1.08	0.66	2.25	2.20	0.78	1.22	
3	-0.59	0.00	-1.76	2.25	2.20	0.78	2.19	
4	-0.59	-0.85	-0.44	-0.07	-0.30	-0.56	-0.71	

After scaling here is a look at the descriptive summary of the scaled dataset:

	count	mean	std	min	25%	50%	75%	max
acct_amt_added_12_24m	99968.00	-0.00	1.00	-0.59	-0.59	-0.59	0.39	1.85
age	99968.00	-0.00	1.00	-2.78	-0.85	-0.15	0.70	3.02
avg_payment_span_0_12m	99968.00	0.00	1.00	-2.09	-0.57	-0.14	0.44	1.96
max_paid_inv_0_12m	99968.00	0.00	1.00	-1.18	-0.71	-0.21	0.47	2.25
max_paid_inv_0_24m	99968.00	0.00	1.00	-1.26	-0.68	-0.21	0.47	2.20
num_active_inv	99968.00	-0.00	1.00	-0.56	-0.56	-0.56	0.78	2.79
num_arch_ok_0_12m	99968.00	0.00	1.00	-0.90	-0.71	-0.32	0.45	2.19
num_arch_ok_12_24m	99968.00	0.00	1.00	-0.80	-0.80	-0.40	0.41	2.24
status_max_archived_0_6_months	99968.00	-0.00	1.00	-1.27	-1.27	0.25	0.25	2.54
status_max_archived_0_24_months	99968.00	0.00	1.00	-1.57	-0.28	-0.28	1.00	2.93
sum_capital_paid_acct_0_12m	99968.00	-0.00	1.00	-0.59	-0.59	-0.59	0.40	1.88
sum_paid_inv_0_12m	99968.00	0.00	1.00	-0.93	-0.77	-0.36	0.42	2.21
time_hours	99968.00	0.00	1.00	-2.93	-0.69	0.09	0.80	1.83
default	89968.00	0.13	33.34	0.00	0.00	0.00	0.00	10000.00

D) Removal of unwanted variables

- Here we will be dropping the variables that are redundant to the model building exercise namely **'userid'** and **'name_in_email'**.
- We will also be dropping the categorical variables with "object" datatype namely '**merchant_category**' and '**merchant_group**' as there won't be much to get out of them even after encoding them as numeric variables using one-hot encoding or label encoding.
- This is due to high cardinality in them, i.e. they have high number of unique categories. 'merchant_category' has 58 unique categories while 'merchant_group' has 13.

4) Model Building

A) Model Selection

Our target variable "default" is a binary variable with 0 denoting non-defaulting credit card users and 1 denoting defaulting credit card users.

So this is essentially a classification problem where our main goal is to predict the users that will default.

To solve this problem we will be using the following techniques or models:

1. **Logistic Regression:**
   - Logistic Regression is a statistical model used for binary classification.
   - It models the relationship between a binary dependent variable and one or more independent variables.
   - It estimates the probability of an instance belonging to a specific class.

2. **Linear Discriminant Analysis (LDA):**
   - Linear Discriminant Analysis is a dimensionality reduction and classification technique.
   - It finds linear combinations of features that best separate multiple classes.
   - LDA aims to maximise the between-class variance and minimise the within-class variance.

3. **Bagging (Bootstrap Aggregating) with Random Forest as base estimator:**
   - Bagging is an ensemble machine learning technique.
   - It builds multiple independent models (typically decision trees) using random subsets of the training data.
   - The final prediction is often an average or majority vote of predictions from these models, reducing variance and improving accuracy.
   - For our problem we will be using **Random Forest Classifier** as the Base Estimator for the Bagging model instead of Decision Trees.
   - **Random Forest Classifier:** It itself is an ensemble learning method that combines multiple decision trees to improve classification accuracy and reduce overfitting. But at the same time can be used as a base estimator for Bagging model, this improves the chances of it being a successful model rather than when being used as a standalone.

4. AdaBoost (Adaptive Boosting):

- **Boosting**: Boosting is an ensemble technique that combines weak learners to create a strong learner.
- It iteratively adjusts the weights of misclassified instances to focus on difficult-to-classify examples.
- Algorithms like AdaBoost and Gradient Boosting are commonly used for boosting.
- AdaBoost is an ensemble learning technique used for classification.
- It combines multiple weak classifiers to create a strong classifier.
- It assigns higher weights to misclassified instances, focusing on difficult-to-classify samples in each iteration.

5. Gradient Boosting:

- Gradient Boosting is an ensemble learning method used for both classification and regression.
- It builds an ensemble of decision trees sequentially.
- Each tree corrects the errors made by the previous ones by fitting to the residuals, optimising a loss function.

B) Model Training

Here we utilise historical data to train and fine-tune the selected models. This was done by splitting the data into training and testing sets to evaluate model performance. Train to Test data ratio was kept at 70:30.

SMOTE

SMOTE (Synthetic Minority Over-sampling Technique) is an algorithm used in machine learning to address class imbalance by generating synthetic examples for the minority class, thereby increasing its representation in the dataset.

Splitting the data into train and test sets:

- The original data I.e. the predictors and response are split into train and test data before applying SMOTE because we apply SMOTE only on the training data and not on the testing data.

- This is because we want to preserve the original data for testing our model and as mentioned above SMOTE introduces synthetic examples in our data which is okay to be trained on but not to test the data.

Checking the class proportion of target variable and applying SMOTE on train data:

```
default
0.00   0.99
1.00   0.01
Name: proportion, dtype: float64
```

- As we can see the minority class i.e. the condition where the credit card user is a defaulter, has a representation of only 1% in the target variable. This suggests a very high data imbalance.
- SMOTE as a technique is generally applied if minority class is below 5%, thus here we will be applying SMOTE to balance the target variable.
- For this we will be taking a sampling strategy of 0.67 i.e. a distribution ratio of 60:40 among the target class 0 and 1 respectively.
- A ratio of 50:50 would be oversampling since our original data was highly imbalanced, so even after resampling some imbalance should be there to keep the analysis authentic.
- As we can see below, after resampling the data, we got our new resampled predictors and response namely "X_res" and "y_res" and these will be our new training data which will be used for our model building later on.

```
Resampled class proportion:
---------------------------
default
0.00   0.60
1.00   0.40
Name: proportion, dtype: float64


Original class proportion:
--------------------------
default
0.00   0.99
1.00   0.01
Name: proportion, dtype: float64
```

C) Model Tuning

- For tuning the models thus built, we have used two techniques:
  - Predicting classes based on their **optimum threshold value** than the default one I.e. 0.5.
  - Using **GridSearchCV** to optimise the hyper-parameters of the built models.

NOTE:

All the models were built and their corresponding performance metrics were identified which are explained below in the model validation part.

Their corresponding classification reports, confusion matrices and ROC curves along with all the relevant charts can be found in the attached python Jupyter notebook (.ipynb file).

5) Model Validation

- To evaluate the performance of the above mentioned modelling techniques we generally use "Classification report" based performance metrics namely 'Accuracy', 'Precision' and 'Recall' along with ROC(Region of convergence) and AUC(Area under the curve) scores.
- Since our goal is to predict the credit card users that will default, the performance metric '**Recall**' becomes the most important metric for us.
- Reason being, we are more concerned with correctly predicting the actual defaulters rather than the checking likelihood of these positive predictions being correct, which is measured by 'Precision'.
- In business terms this means that we are okay even if some non-defaulters are predicted as defaulters as long as we are able to predict most of the actual defaulters.
- Another way to evaluate the performance of the above mentioned modelling techniques is by using Confusion Matrix and the two kinds of errors associated with it namely, Type I and Type II Errors.
  - Type I Error (aka False Positive) - Occurs when the model incorrectly predicts a user as a defaulter, leading to the rejection of a potentially creditworthy customer.
  - Type II Error (aka False Negative) - Occurs when the model fails to predict that a user will default, resulting in the approval of a high-risk customer who may default on payments.
  - Based on our business problem, out of the two, '**Type II Error'** is the one we must mitigate in order for our model to perform better.

- Based on the steps mentioned earlier, all the respective models were built and tuned. The final selected version of each model along with their performance metrics are shown below:

Performance metrics Table (comparison of all selected models)

Sl. No.	Model Name	Accuracy		Precision		Recall		AUC	
		Train	Test	Train	Test	Train	Test	Train	Test
1	Logistic Regression	0.73	0.69	0.63	0.03	0.81	0.81	0.816	0.829
2	LDA	0.72	0.67	0.62	0.03	0.82	0.82	0.811	0.827
3	Bagging (Random Forest as estimator)	0.99	0.95	0.98	0.04	1.0	0.13	0.999	0.790
4	ADA Boosting	0.84	0.80	0.75	0.04	0.90	0.65	0.941	0.813
5	Gradient Boosting	0.84	0.85	0.79	0.04	0.82	0.53	0.964	0.810

Comparison of all the selected models wrt Type I and Type II Errors

Sl. No.	Model Name	Train		Test	
		False Positives (Type I Error)	False Negatives (Type II Error)	False Positives (Type I Error)	False Negatives (Type II Error)
1	Logistic Regression	22016	8997	9234	72
2	LDA	23496	8556	9947	70
3	Bagging (Random Forest as estimator)	1145	230	1044	337
4	ADA Boosting	13641	4710	5870	137
5	Gradient Boosting	5685	7664	2462	259

Performance metrics detailed analysis

LDA and Logistic Regression are the best performing models with respect to Recall as well as type II error, our target performance metrics. Also, LDA model has a slight edge over Logistic regression in terms of recall value.

Hence, we would choose LDA to be our final model if we have to choose one model out of the two.

Performance Metrics with respect to test data for the Best-Fit LDA Model:

Classification report:

```
Updated classification_report for train data:
-----------------------------------------------
              precision    recall  f1-score   support

         0.0       0.84      0.66      0.74     69081
         1.0       0.62      0.82      0.70     46284

    accuracy                           0.72    115365
   macro avg       0.73      0.74      0.72    115365
weighted avg       0.75      0.72      0.72    115365

=====================================================

Updated classification_report for test data:
----------------------------------------------
              precision    recall  f1-score   support

         0.0       1.00      0.66      0.80     29608
         1.0       0.03      0.82      0.06       386

    accuracy                           0.67     29994
   macro avg       0.51      0.74      0.43     29994
weighted avg       0.98      0.67      0.79     29994
```

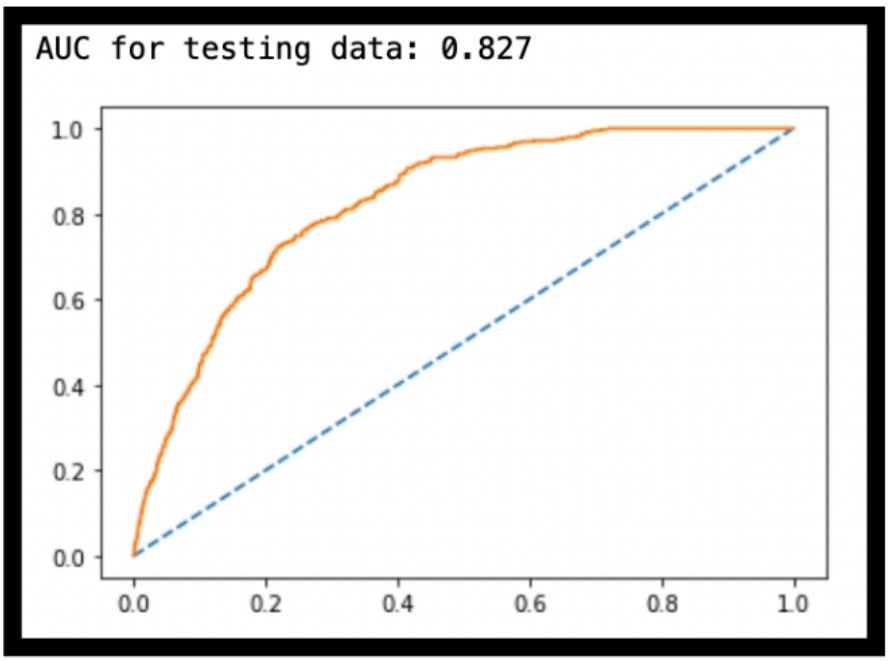
- For predicting Credit card Defaulters (Label 1):
  - **Recall (82%)** – Out of all the credit card users who actually defaulted, 82% of them have been predicted correctly .
  - **Precision (3%)** – Out of every 100 predictions of defaulters, only 3 were predicted correctly I.e. a lot of false positives got generated due to the highly imbalanced nature of original data.
  - **Accuracy (67%)** – 67% of total predictions are correct.
- The selected model shows no overfitting or under-fitting with respect to recall values.

ROC Curve and AUC Scores:

A good model will have an ROC curve that approaches the upper-left corner of the plot. The closer the curve is to the upper-left corner, the better the model's discriminatory power. The AUC value ranges from 0 to 1, where a higher AUC indicates better model performance.

- AUC = 1: Perfect model, the ROC curve covers the entire area, and the model makes perfect predictions.
- AUC = 0.5: Random model, the ROC curve is a diagonal line from the bottom-left to the top-right, indicating no discriminatory power.

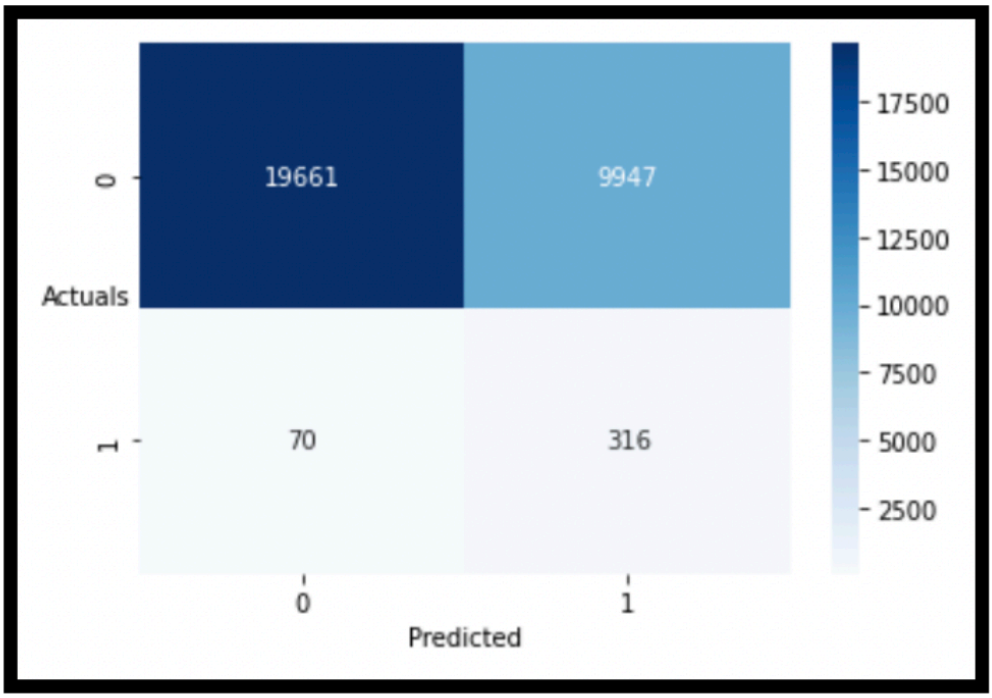
AUC and ROC for the testing data:



- In our case, **for LDA model, AUC Scores = 0.8**, which indicates that the model is performing relatively better, it's not perfect but not too bad also as the value is nearer to 1 than to 0.5. The model is average when it comes to ROC curve.

Confusion Matrix:

Updated Confusion matrix for the test data based on optimum cut-off of 0.395:



- **Type I and Type II Errors:** With respect to type I (False Positives) and type II (False Negatives) errors, as mentioned earlier, our main focus is to minimise type II Errors.
  - Looking at the type II Errors, the **model with the least type II error is LDA (70).**
  - This means that out of a sample of 386 actual defaulters in the test data , our best-fit model failed to identify 70 defaulters, which comes around 18%. This is the best we could do across all the models built.

6) Final interpretation / recommendation

A) Insights from the final (most optimum) model i.e. LDA Model

- Looking at our most optimum model i.e. tuned LDA model, the top 3 predictors based on the Linear discriminant function that play a major role in predicting classes are as follows (Linear discriminant function is a linear combination of the input features that maximises the separation between classes):
  - **'num_active_inv'(0.65) :** This represents Total number of active invoices per user, I.e. Unpaid bills. It is positively correlated to predicting classes meaning, based on analysis we can say that **users with large number of active invoices are 60% more likely to belong to class 1(defaulters).** Hence, this becomes our most important predictor.
  - **'status_max_archived_0_6_months'(0.27) :** This represents maximum number of times the account was in archived status in the last six months. It is also positively correlated to predicting classes meaning, based on analysis we can say that **users with higher number of archived status counts in last 6 months are around 25% more likely to belong to class 1(defaulters).**
  - **'num_arch_ok_12_24m'(-0.25) :** This represents number of archived purchases that were paid between 24 months in the past to the present date and 12 months in the past to the present date. But this variable is negatively correlated to predicting classes meaning, based on analysis we can say that **users with higher number of archived purchases in the given period are 25% less likely to belong to class 1 (defaulters). And it makes sense also, since these users are clearing their pending bills thus expressing their willingness to continue using their credit cards.**
- Also looking at the performance evaluation of the LDA model, we can see that there is a high variation between the train and test 'Precision' values. This doesnot necessarily mean that there is overfitting.
- This variation is due to the fact that we have used SMOTE to resample the training data in order to balance it. This was done in order to properly train the machine learning models, as original dataset had too few positive classes to properly train the models.
- There is also a high imbalance between the precision and recall values in the test data with precision values being quite low. Ideally we want a good balance between recall and precision, but that is not the case here.

- The main reason for such a low precision value is the highly imbalanced nature of the dataset (only 1% of the total users i.e. 1000 users amongst almost 1 lakh users were defaulters while the rest were non-defaulters.
- The extremely low precision value suggests that the model is producing a large number of false positives for each true positive it identifies. While this may seem rather unusual, it is not uncommon in certain situations such as highly imbalanced datasets or when the priority is maximising recall.
- Moreover another reason for this high imbalanced nature of dataset is the domain to which it belongs i.e. Credit card default prediction or more broadly financial risk assessment. Since finance is the base for almost all economic activities in the world, financial risk assessments are taken very seriously due to which the ratio of defaulters to non-defaulters is usually quite low leading to such high imbalances.

B) Model Implication on the business and recommendations

- Since 'Total number of active invoices per user' is our most important predictor, the credit card company should device specific plans around this variable.
  - ★ **One recommendation here would be to clusters of users based on number of active invoices. It can offer lucrative discounts and offers to users who have a lot of active invoices encouraging them to settle their bills.**
  - ★ **Company can also cross match such customers to the merchant groups they transacted the most and promise lucrative offers for related products on the condition that they settle their active invoices in a limited time period.**
- Another important predictor is 'maximum number of times the account was in archived status in the last six months'.
  - ★ **Here the recommendation would be to allocate dedicated human resource and time to setup direct communication line with customers who are still not written off yet whose accounts went in archived status most number of times in last 6 months.**
- Our third most important predictor is 'number of archived purchases that were paid between 24 months in the past to the present date and 12 months in the past to the present date'.
  - ★ **Here the company can provide discount coupons related to user's most transacted merchant groups for users with higher number of archived purchase payments aesthete users settled their archived purchases on their own accord, meaning they are willing to stay with the credit card company. Providing lucrative offers to such users will encourage them to engage more in transactions through the company's credit card.**

- Another main highlight of our model is the high variation between Precision and Recall value and what it means for the business.
- In the context of a credit card company, the implications of low precision and high recall in a credit card default prediction model can be particularly significant due to the unique nature of the business and its focus on risk management and customer satisfaction.

Operational Costs:

- High False Positives: Low precision means a high rate of false positives, where the model incorrectly predicts credit card defaults. This is also clearly visible in our final model.
- Investigating these false positives can be costly in terms of resources, time, and manpower.

★ **One major recommendation for the business here would be, conducting manual reviews of flagged accounts.** This is absolutely essential in order to keep track and make sense out of the business.

Customer Experience:

- Customer Inconvenience: Customers who are subject to false positives may experience inconvenience, such as temporary card suspensions, transaction declines, or account freezes. This can lead to customer dissatisfaction and potentially damage the company's reputation.

★ **Hence, here the recommendation would be to allocate dedicated human resource and time to mitigate such issues as soon as they occur.**

Risk Management:

- Risk Mitigation: High recall indicates that the model captures most of the actual defaults, this can help the company mitigate credit risk. Identifying a large portion of true defaults is essential for minimising financial losses.

Regulatory Compliance:

- Credit card companies are subject to various regulatory requirements and standards. High recall can be essential to ensure compliance with regulatory standards related to risk assessment and fraud prevention.

Financial Impact:

- Loss Mitigation: While high recall helps mitigate financial losses from defaults, low precision can lead to financial losses due to the approval of high-risk customers who eventually default.
- Here the recommendation would be to try to strike a balance between precision and recall based on its risk tolerance and business objectives. Adjusting the prediction threshold can help achieve a practical trade-off between the two metrics.
- This can not be achieved to a greater extent using this dataset, as one way to achieve this balance is by optimising threshold values which has already been done.

★ **So here the recommendation would be that the company use this model to operate for now and try to add to this dataset more defaulters, so that overtime the iterative models can strike a better balance between precision and recall values.**

Continuous Monitoring and Improvement:

- Feedback and Adaptation: Continuous monitoring of model performance, feedback collection, and adaptation are crucial for optimising the model's performance over time as mentioned above.

★ **Hence, here the recommendation would be that the company should be prepared to make adjustments based on evolving conditions and business priorities.**