Double-click (or enter) to edit

Refresh

A.Introduction

The purpose of this notebook is to conduct a comprehensive Exploratory Data Analysis (EDA) and draw insights from the Home Credit dataset. Home Credit is a non-banking financial institution, focused on providing loans to people with little or no credit history. Our aim is to understand the factors influencing the credit default rate.

The dataset contains demographic, loan, and other financial information about each client, making it a rich source to analyze and predict their credit behavior. However, an initial glance at our target variable - credit default, shows a significant class imbalance which might pose a challenge in later stages of predictive modeling. This intriguing aspect motivates us to hypothesize and investigate certain key factors which could potentially influence the likelihood of credit default.

Our main hypotheses revolve around the following factors:

1. Demographic Factors: These include age, gender, and occupation, and we hypothesize that these can play a role in the likelihood of a client defaulting.
2. Economic Factors: The economic status of the client, such as their total income, might influence their ability to repay loans.
3. Loan Characteristics: Certain characteristics of the loan itself, like loan amount and term, could potentially affect the client's repayment behavior.
4. External Ratings: Ratings from external sources can provide a broader perspective on the client's financial behavior and might be linked to their default probability.

In the following sections, we'll dive deeper into these hypotheses, conducting a detailed EDA to uncover the patterns hidden in the data.

B.Technical Setup

> This notebook leverages the strengths of both Python and R. We will use Python for efficient data processing and preliminary visualizations, and R for in-depth visualizations and statistical analyses. This interplay between Python and R is made possible by the `rpy2` library, which allows us to run R code inside a Python environment.

B.1 Python Packages

```
import yaml
from google.colab import drive

import os
import gc

import pandas as pd
import numpy as np

from scipy.stats import zscore
from scipy.stats import mstats
import scipy.stats as stats
from scipy.stats import skew
import statsmodels.api as sm
import statsmodels.formula.api as smf


import matplotlib.pyplot as plt
import matplotlib.ticker as plticker
import seaborn as sns
import plotly.graph_objs as go
import plotly
from plotly.subplots import make_subplots

from sklearn.decomposition import PCA

import warnings
warnings.filterwarnings("ignore")

drive.mount("/content/drive")
%matplotlib inline
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

B.2 R Packages

To use R within our Python notebook, we first need to install rpy2 and other necessary components, then import and install our R packages. Finally, we will activate R magic commands to allow us to use R in our notebook.

Firstly, install rpy2 and required libraries.

```
!pip install rpy2==3.5.1
!apt-get install libmpc-dev
!apt-get install libgmp3-dev
!apt-get install libmpfr-dev
```

Secondly, we import and install the R packages we'll be using for our analysis.

```
#Installing R packages via rpy2
# import rpy2's package module
import rpy2.robjects.packages as rpackages
from rpy2.robjects.vectors import StrVector # R vector of strings


# import R's utility package
utils = rpackages.importr('utils')
base = rpackages.importr('base')
```

```
# select a mirror for R packages
utils.chooseCRANmirror(ind=1) # select the first mirror in the list

# R package names
packnames = ('ggplot2', 'tidyverse', 'ggstatsplot', 'devtools', 'PMCMRplus', 'rstantools', 'ggside', 'ggridges', 'ggblanket', 'see', 'gg

# Selectively install what needs to be install.
names_to_install = [x for x in packnames if not rpackages.isinstalled(x)]
if len(names_to_install) > 0:
    utils.install_packages(StrVector(names_to_install))
```

```
import rpy2.robjects as robjects

r_code = """
if (!requireNamespace("devtools", quietly = TRUE))
    install.packages("devtools")
devtools::install_github("wilkelab/ggtext")
"""
robjects.r(r_code)
```

After installing necessary packages, activate R magic commands.

```
#@title
%load_ext rpy2.ipython

import rpy2
import rpy2.robjects as robjects
from rpy2.robjects.packages import importr, data
from rpy2.robjects import pandas2ri
from functools import partial
from rpy2.ipython import html
html.html_rdataframe=partial(html.html_rdataframe, table_class="docutils")
import rpy2.ipython.html
rpy2.ipython.html.init_printing()


pandas2ri.activate()
rpy2.__path__
```

```
['/usr/local/lib/python3.10/dist-packages/rpy2']
```

Lastly, load the necessary R libraries.

```
#@title
#Loading R Libraries
%%R

library(ggstatsplot)
library(dplyr)
library(ggplot2)
library(PMCMRplus)
library(rstantools)
library(ggside)
library(statsExpressions)
library(wesanderson)
library(ggblanket)
library(ggridges)
library(see)
library(ggforce)
library(scales)
library(RColorBrewer)
library(ggtext)
```

```
%%R

#Theme settings for Data Viz
theme_text <- theme(
    # This is the new default font in the plot
```

```
    text = element_text(family = "Roboto", size = 8, color = "black"),
    plot.title = element_text(
      family = "Lobster Two",
      size = 20,
      face = "bold",
      color = "#2a475e"
    ),
    # Statistical annotations below the main title
    plot.subtitle = element_text(
      family = "Roboto",
      size = 15,
      face = "bold",
      color="#1b2838"
    ),
    plot.title.position = "plot", # slightly different from default
    axis.text = element_text(size = 10, color = "black"),
    axis.title = element_text(size = 12)
  )

theme_bg <- theme(
    axis.ticks = element_blank(),
    axis.line = element_line(colour = "grey50"),
    panel.grid = element_line(color = "#b4aea9"),
    panel.grid.minor = element_blank(),
    panel.grid.major.x = element_blank(),
    panel.grid.major.y = element_line(linetype = "dashed"),
    panel.background = element_rect(fill = "#fbf9f4", color = "#fbf9f4"),
    plot.background = element_rect(fill = "#fbf9f4", color = "#fbf9f4")
  )

# Define the theme
my_dark_theme <-
  theme(
    plot.background = element_rect(fill = "#121212"),
    panel.background = element_rect(fill = "#121212"),
    panel.border = element_blank(),
    panel.grid.major = element_line(color = "#2a2a2a", linetype = "solid"),
    panel.grid.minor = element_line(color = "#2a2a2a", linetype = "dotted"),
    axis.line = element_line(color = "#bbbbbb"),
    axis.text = element_text(color = "#e0e0e0", size = 12, family = "Arial"),
    axis.title = element_text(color = "#ffffff", size = 14, family = "Arial"),
    axis.ticks = element_line(color = "#e0e0e0"),
    plot.title = element_text(color = "#ffffff", size = 18, hjust = 0.5, family = "Arial"),
    plot.subtitle = element_text(color = "#e0e0e0", size = 14, hjust = 0.5, family = "Arial"),
    plot.caption = element_text(color = "#e0e0e0", size = 12, hjust = 1, family = "Arial"),
    legend.title = element_text(color = "#ffffff", size = 12, family = "Arial"),
    legend.text = element_text(color = "#e0e0e0", size = 12, family = "Arial"),
    legend.key = element_rect(fill = "#222222", color = "#bbbbbb")
  )
```

B.3 Utility Functions

2 cells hidden

C.Data Collection

Data Loading and Preliminary Analysis

Given the size of our datasets, we've divided our analysis into two notebooks. The first notebook focused on preprocessing, and in this second notebook, we'll focus on exploratory data analysis and testing our hypotheses.

We'll be using three preprocessed datasets in this notebook:

1. Application Data: Contains information about each loan application at Home Credit. This is a combination of the `application_train` and `application_test` datasets from the original Home Credit datasets.
2. Bureau Data: Data about client's previous credits from other financial institutions. Each row represents one credit.
3. Previous Application Data: Contains information about the client's previous applications at Home Credit.

Let's start by loading these datasets.

```
1 app_df_cln = pd.read_parquet('/content/application_train_test_cln.parquet')
```

```
1 app_df_cln.head()
```

	SK_ID_CURR	TARGET	CONTRACT_TYPE	GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	..
0	100002	1.0	Cash loans	M	N	Y	0	202500.0	406597.5	24700.5	.
1	100003	0.0	Cash loans	F	N	N	0	270000.0	1293502.5	35698.5	.
2	100004	0.0	Revolving loans	M	Y	Y	0	67500.0	135000.0	6750.0	.
3	100006	0.0	Cash loans	F	N	Y	0	135000.0	312682.5	29686.5	.
4	100007	0.0	Cash loans	M	N	Y	0	121500.0	513000.0	21865.5	.

5 rows × 67 columns

```
1 analyze_dataframe(app_df_cln)
```

	Column	Data Type	Unique Count	Unique Sample	Missing Values	Missing Percentage
0	SK_ID_CURR	int64	343602	[100002, 100003, 100004, 100006, 100007]	0	0.0000
1	TARGET	float64	2	[1.0, 0.0, nan]	47037	13.6894
2	CONTRACT_TYPE	object	2	[Cash loans, Revolving loans]	0	0.0000
3	GENDER	object	2	[M, F]	0	0.0000
4	FLAG_OWN_CAR	object	2	[N, Y]	0	0.0000
...	...	...	...	...	...	...
61	FLAG_DOCUMENT_18	int64	2	[0, 1]	0	0.0000
62	FLAG_DOCUMENT_19	int64	2	[0, 1]	0	0.0000
63	FLAG_DOCUMENT_20	int64	2	[0, 1]	0	0.0000
64	FLAG_DOCUMENT_21	int64	2	[0, 1]	0	0.0000
65	AMT_REQ_CREDIT_BUREAU_YEAR	float64	24	[1.0, 0.0, nan, 2.0, 4.0]	45632	13.2805

66 rows × 6 columns

D.Exploratory Data Analysis

In this section, I'll conduct an exploratory data analysis on two selected datasets: the application data and the previous application data. Given the large number of variables in the datasets (66 in the application data and 36 in the previous application data), I will not be exploring each variable individually. Instead, I will focus on variables that I believe, based on my understanding of the data and the domain, could significantly influence the loan application outcome. This section will be organized as follows:

Exploring the Application Data

In this subsection, I'll delve into the application data. Starting with the target variable, I'll examine its distribution and the balance of its classes. Then, I'll investigate key variables such as gender, number of children, family status, occupation, total income, and age. For each of these, I'll analyze their relationship with the target variable, using a combination of descriptive statistics, visualizations, and hypothesis tests to validate observed trends and relationships.

Exploring the Previous Application Data

Next, I'll turn to the previous application data. Although the target variable isn't present in this dataset, I'll conduct a similar analysis to understand the distributions of key variables and their relationships. This could provide insights into past application trends, informing future loan application outcomes.

Merging the Datasets and Joint Analysis

After independently exploring the datasets, I'll merge the application data with the previous application data. With the merged dataset, I'll examine interactions between variables from both original datasets. Again, as part of this analysis, I'll run hypothesis tests whenever I observe interesting patterns or relationships. This joint analysis will inform my understanding of the variables' joint impact on the loan application outcome.

By embedding hypothesis testing within the EDA, this approach allows for immediate testing of insights, ensuring a dynamic and integrated data analysis process that leads into the next step: feature engineering.

```
app_df_cln['LOG_AMT_INCOME_TOTAL'] = np.log(app_df_cln['AMT_INCOME_TOTAL'])

app_df_viz = app_df_cln.copy(deep=True)

app_df_viz = app_df_viz.filter(regex='^(?!.*FLAG_DOCUMENT)')

# Replace 0 with 'No Default' and 1 with 'Default' in the 'TARGET' column
# app_df_viz['TARGET'] = app_df_viz['TARGET'].replace({0: 'No Default', 1: 'Default'})
```

```
%%R -i app_df_viz

rdf <- app_df_viz
```

```
app_df_viz.to_csv('app_df_viz.csv', index=False)
```

D.1. Application Data Analysis

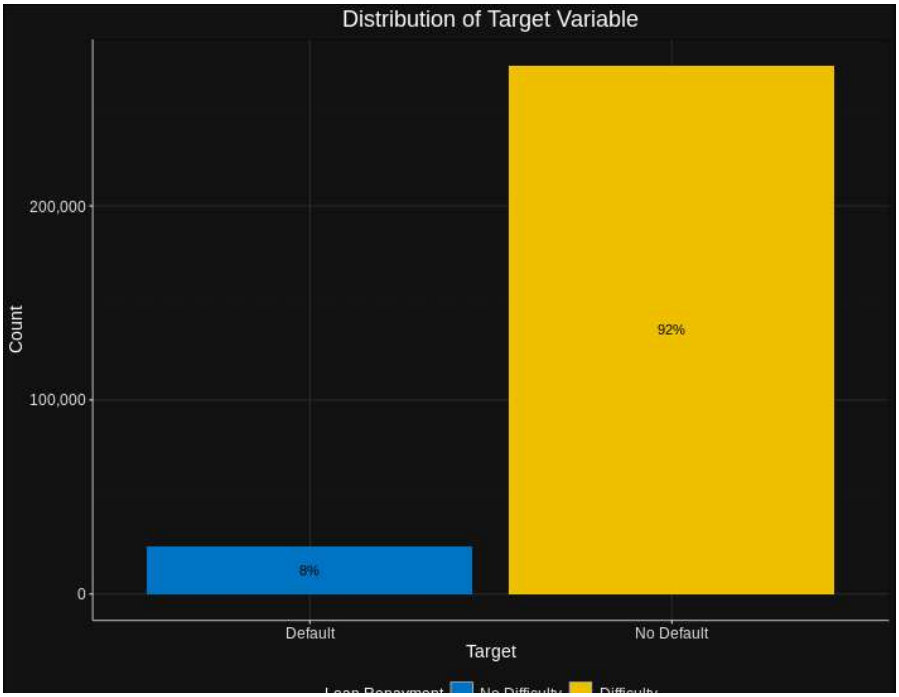
D.1.a. Analysis of Target Variable

First, let's examine the target variable, which indicates whether the loan applicant had trouble repaying their loan. A brief statistical summary and visualization of its distribution can provide insights into the class balance of the dataset.

```
# Imbalance Check
print("\nImbalance Check\n")
target_count = app_df_cln['TARGET'].value_counts()
print(f'Class 0: {target_count[0]}')
print(f'Class 1: {target_count[1]}')
print(f'Proportion: {round(target_count[0] / target_count[1], 2)} : 1')
print(f'Percentage of Majority Class: {round(target_count[0] / sum(target_count) * 100, 2)}%')
```

```
Imbalance Check

Class 0: 272248
Class 1: 24317
Proportion: 11.2 : 1
Percentage of Majority Class: 91.8%
```

```
%%R -w 25 -h 20 -u cm

#bar plot of TARGET
rdf %>%
  select(TARGET) %>%
  mutate(TARGET=as.factor(TARGET)) %>%
  na.omit() %>%
  # Create bar plot
  ggplot(aes(x=TARGET, fill=TARGET)) +
    theme_minimal() +
    geom_bar() +
    labs(title = 'Distribution of Target Variable',
        x = 'Target',
        y = 'Count') +
    scale_fill_manual(values = c("#0073C2FF", "#EFC000FF"),
                      labels = c("No Difficulty", "Difficulty"),
                      name = "Loan Repayment") +
    scale_y_continuous(labels=comma) +
    my_dark_theme +
    geom_text(aes(label = scales::percent(..count../sum(..count..))),
              stat = 'count',
              position=position_stack(vjust=0.5),
              color = 'black') +
    theme(legend.position = 'bottom')
```



> The bar plot visualizes the distribution of the TARGET variable, representing default and non-default instances. The x-axis shows two classes: "No Default" (Target=0) and "Default" (Target=1), while the y-axis represents the count of observations in each class.
>
> In the plot, "No Default" has a count of 272,248, and "Default" has a count of 24,317. This indicates a class imbalance, with an approximate ratio of 11.2:1 between the majority and minority classes.
>
> The plot visually illustrates the distribution, with the "No Default" class dominating, accounting for around 91.8% of the total observations. Conversely, the "Default" class represents a smaller portion of the dataset.

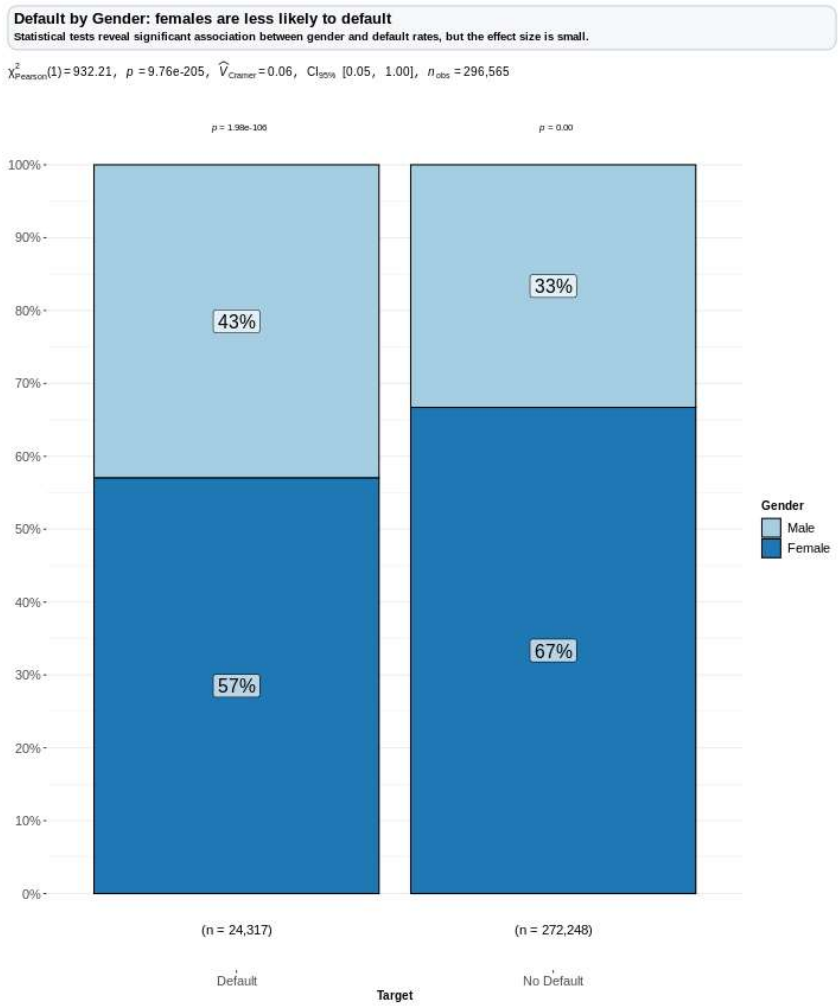
D.1.b. Analysis of Demographic Variable

4.2.1. Impact of Gender on Default

In this section, we'll examine the relationship between gender and the default rate using a stacked bar chart and a Chi-Squared test. Our null hypothesis (H0) is that there is no relationship between gender and default rate, while our alternative hypothesis (H1) is that there is a relationship between the two.

```
%%R -w 25 -h 30 -u cm

rdf %>%
  select(TARGET, GENDER) %>%
  na.omit() %>%
  mutate(GENDER = ifelse(GENDER=='F', 'Female', ifelse(GENDER=='M', 'Male', GENDER))
  ) %>%
  ggbarstats(
      x = GENDER,
      y = TARGET,
      title = "",
      xlab = "Target",
      legend.title  = "Gender",
      ggsignif.args = list(textsize = 20, tip_length = 0.1),
      label.args = list(alpha = 0.7, size=5.5, fill = "white"),
      ggplot.component = list(ggplot2::scale_x_discrete(guide = ggplot2::guide_axis(n.dodge = 1)),
                              theme(plot.title.position = "plot",
                                    axis.text.x = element_text(size = 11),
                                    axis.text.y = element_text(size = 11),
                                    legend.text = element_text(size=11),
                                    plot.title = ggtext::element_textbox_simple(
                                        size = 14,
                                        lineheight = 1,
                                        linetype = 1,
                                        box.color = "#C6CED5",
```

```
                                    fill = "#F7F8F9",
                                    r = grid::unit(8, "pt"),
                                    padding = margin(5, 5, 5, 5),
                                    margin = margin(0, 0, 10, 0))
                                )
                            ),
    palette = "Paired",
    results.subtitle = TRUE,
    bf.message = FALSE
  ) +
  labs(title="<span style='font-size:13pt; font-weight:normal;'>Default by Gender: females are less likely to default</span><br>
  <span style='font-size:9.5pt; font-weight:normal;'>Statistical tests reveal significant association between gender and default rates,

```

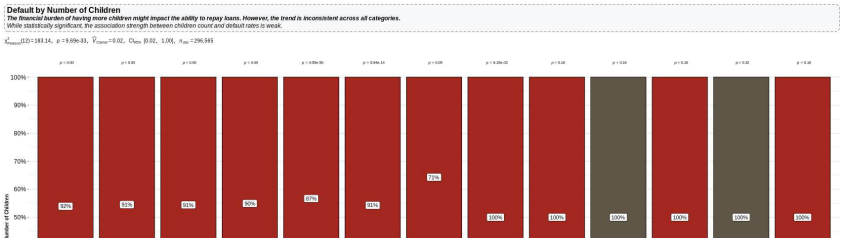


> From the Chi-Squared test, we obtained a Chi-square statistic of 932.21 and a p-value of 9.76, which is less than 0.05. This indicates that we can reject the null hypothesis and conclude that there is a statistically significant relationship between gender and default rate. However, the Cramer's V value is 0.06, indicating a weak association.
>
> Our stacked bar chart provides further insight into this relationship. Among the defaults (n = 243,127), 57% were female and 43% were male. Among those who did not default (n = 272,248), the proportion of females was even higher at 67%, while males made up 33%. This clearly shows that females are less likely to default compared to males, further supporting our test results.
>
> The p-values indicated on the bars (p = 1.983e-6 for default and p = 0.00 for no default) refer to the results of a hypothesis test comparing the observed proportion of males and females in that category (default vs. non-default) to the expected proportion if there were no difference in default rates between genders. These very small p-values provide strong evidence against the null hypothesis, suggesting a significant difference in default (and non-default) rates between genders.

> Although we have statistical evidence that gender and default rates are associated, the effect size is relatively small, meaning the practical impact of this association may be limited. The confidence interval [0.05, 1.00] and our large sample size (296,565) further support these findings. It represents the range within which we can be 95% certain that the true value of Cramer's V (the measure of association between gender and default rate) lies. As this interval is fairly wide and includes very low values (close to 0), it further supports our earlier statement that although the association is statistically significant, the strength of the association is weak.

4.2.3. Number of Children

```
%%R -w 60 -h 30 -u cm

theme_ggtext <- theme(plot.title.position = "plot",
                      axis.text.x = element_text(size = 12, colour = "black"),   # More readable size and color
                      axis.text.y = element_text(size = 12, colour = "black"),   # More readable size and color
                      legend.text = element_text(size=11, colour = "black"),     # More readable size and color
                      plot.title = ggtext::element_textbox_simple(
                          size = 16, lineheight = 0.8,linetype = 2,
                          box.color = "#808080", fill = "#F7F8F9",  # Neutral color scheme
                          r = grid::unit(8, "pt"),
                          padding = margin(5, 5, 5, 5),
                          margin = margin(0, 0, 10, 0)),
                      legend.position = "bottom",   # Keep legend at right for professional look
                      plot.background = element_rect(fill = "#FFFFFF"),  # White background for plot
                      panel.background = element_rect(fill = "#FFFFFF"),  # White background for panel
                      panel.grid = element_line(colour = "#DDDDDD"))  # Subtle gridlines

rdf %>%
  select(TARGET, GENDER, CNT_CHILDREN) %>%
  na.omit() %>%
  mutate(GENDER = ifelse(GENDER=='F', 'Female', ifelse(GENDER=='M', 'Male', GENDER))) %>%
  ggbarstats(
      y = CNT_CHILDREN,
      x = TARGET,
      title = "Default by Number of Children<br>
      <i><span style='font-size:12pt; font-weight:normal;'>The financial burden of having more children might impact the ability to repay
      <i><span style='font-size:12pt; font-weight:normal;'> While statistically significant, the association strength between children cou
      xlab = "Default Status",
      ylab = "Number of Children",
      legend.title  = "Target",
      package = "wesanderson",
      palette = "BottleRocket1",   # Back to a known palette
      bf.message=FALSE
    ) + theme_ggtext

```



> The chi-squared test performed on the number of children (CNT_CHILDREN) reveals a statistically significant association with the default rates (X^2 = 183.14, p < .0001), despite the low effect size (V_cramer = 0.02) confirmed by the confidence interval [0.02, 1.00]. Out of 296,565 observations after handling missing values, we can see an interesting trend: the default rates seem to increase as the number of children increases.
>
> For applicants with no children (CNT_CHILDREN = 0), the default rate is the lowest at 8%. Applicants with one to five children show a slightly higher default rate at around 9%, with the exception of those with four children where the default rate jumps to 13%. Beyond five children, the default rate shows a sharp increase, peaking at CNT_CHILDREN = 9 and 10, where all applicants defaulted on their loans. However, it is worth noting that applicants with seven, eight, eleven, or twelve children had no defaults. Given the small sample size and 0 variance in these categories, this observation might not generalize to a larger population.
>
> Because of this, these categories may be considered as rare, and their analysis may require special attention or alternative treatment, such as grouping them into a single category or excluding them from the model to avoid skewed results.
>
> These results suggest that the financial burden of having more children might impact the ability to repay loans. However, the trend is not consistent across all categories, warranting further investigation.

Conclusion:

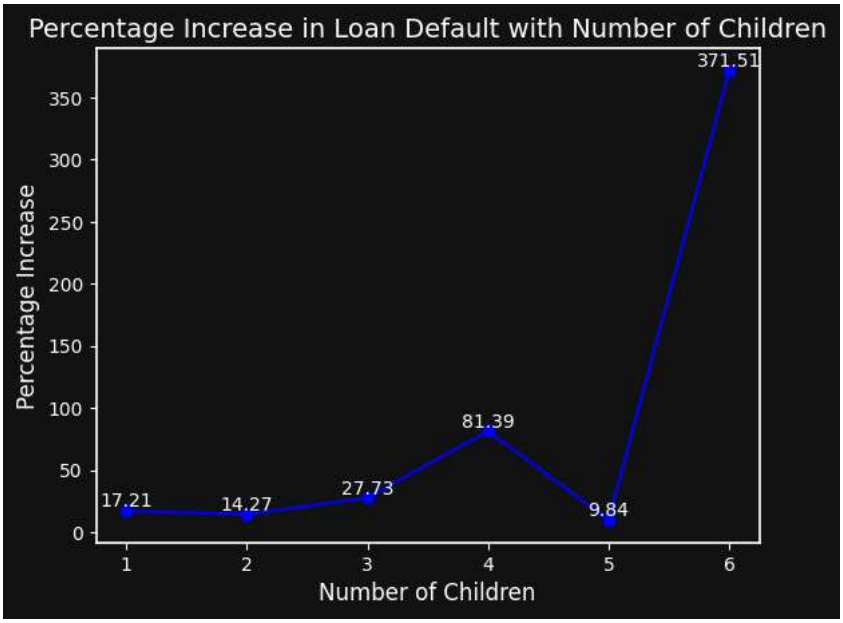
> - While the number of children appears to influence default rates, inconsistencies in certain categories suggest further analysis is required.
> - In other words, the financial burden of having more children might impact the ability to repay loans. However, the trend is not consistent across all categories, warranting further investigation.
> - CNT_CHILDREN above 7 need to be excluded to avoid skewed modeling results

```
%%R

rdf %>%
  select(TARGET,CNT_CHILDREN) %>%
  na.omit() %>%
  mutate(CNT_CHILDREN = as.factor(CNT_CHILDREN),
         TARGET = ifelse(TARGET=='No Default', 0, ifelse(TARGET=='Default', 1, TARGET)),
         TARGET = as.numeric(TARGET)) %>%
  glm(TARGET ~ ., data = ., family = binomial(link = "logit")) %>%
  summary()

```

```
Call:
glm(formula = TARGET ~ ., family = binomial(link = "logit"),
    data = .)

Coefficients:
                Estimate Std. Error  z value Pr(>|z|)
(Intercept)     -2.465446   0.008173 -301.643  < 2e-16 ***
CNT_CHILDREN1    0.159167   0.016503    9.645  < 2e-16 ***
CNT_CHILDREN2    0.133707   0.023311    5.736 9.70e-09 ***
CNT_CHILDREN3    0.244581   0.056631    4.319 1.57e-05 ***
CNT_CHILDREN4    0.595044   0.145085    4.101 4.11e-05 ***
CNT_CHILDREN5    0.093868   0.395294    0.237  0.81230
CNT_CHILDREN6    1.549156   0.483115    3.207  0.00134 **
CNT_CHILDREN7   -8.100581  45.154676   -0.179  0.85763
CNT_CHILDREN8   -8.100581  84.476663   -0.096  0.92361
CNT_CHILDREN9   13.031474  84.476663    0.154  0.87740
CNT_CHILDREN10  -8.100581  84.476663   -0.096  0.92361
CNT_CHILDREN11  13.031474 119.468043    0.109  0.91314
CNT_CHILDREN12  -8.100581  84.476663   -0.096  0.92361
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 168221  on 296564  degrees of freedom
Residual deviance: 168064  on 296552  degrees of freedom
AIC: 168090

Number of Fisher Scoring iterations: 9
```

```
import matplotlib.pyplot as plt

# Data
num_children = [1, 2, 3, 4, 5, 6]
percentage_increase = [17.21, 14.27, 27.73, 81.39, 9.84, 371.51]

# Plot
plt.plot(num_children, percentage_increase, marker='o', linestyle='-', color='blue')
plt.xlabel('Number of Children')
plt.ylabel('Percentage Increase')
plt.title('Percentage Increase in Loan Default with Number of Children')

# Add data labels
for i, j in zip(num_children, percentage_increase):
    plt.text(i, j, f'{j:.2f}', ha='center', va='bottom')

# Display the plot
plt.show()
```

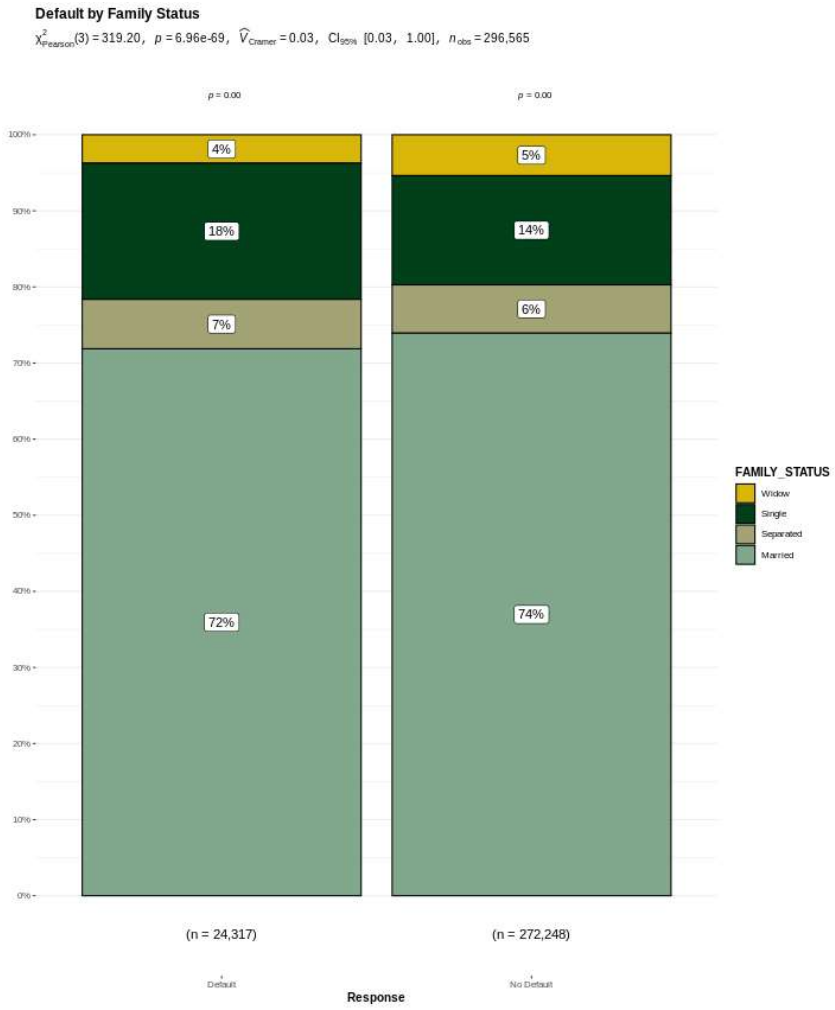


4.2.4. Family Status

Family Status & Target Variable

```
%%R -w 25 -h 30 -u cm

p_fam_target <- rdf %>%
                  select(TARGET, FAMILY_STATUS) %>%
                  na.omit() %>%
                  ggbarstats(
                      x = FAMILY_STATUS,
                      y = TARGET,
                      title = "Default by Family Status",
                      xlab = "Response",
                      legend.title  = "FAMILY_STATUS",
                      package = "wesanderson",
                      palette = "Cavalcanti1",
                      bf.message=FALSE
                    )

extract_stats(p_fam_target)
```

```
%%R -w 25 -h 30 -u cm
p_fam_target
```



The statistical analysis performed on Family Status with respect to default rates reveals a significant association, as indicated by the Pearson's Chi-squared test (X^2 = 319, df = 3, p < .0001). Despite the significant p-value, the effect size is small (Cramer's V = 0.0327), falling within the confidence interval [0.0295, 1.00]. The observations from 296,565 customers are taken into account after handling missing values.

When we look at the descriptive data, we see that among the defaulters, the percentage distribution according to Family Status is as follows: Widow 3.74%, Single 17.9%, Separated 6.5%, and Married 71.9%. On the other hand, among those who did not default, the distribution is: Widow 5.38%, Single 14.4%, Separated 6.35%, and Married 73.9%.

The pattern suggests a slightly higher propensity to default among Married and Single customers. However, the impact of Family Status on default is rather minimal, given the small effect size. Furthermore, the data also implies that while there may be a significant association between Family Status and default rates, the rate of default is still predominantly low (8.2%) compared to non-default (91.8%).

To summarize, while Family Status has a statistically significant association with default rates, the practical impact may not be as substantial, hence necessitating further exploration into other variables or combinations of variables to better understand their influence on loan default.

```
%%R

rdf %>%
  select(TARGET,CNT_FAM_MEMBERS) %>%
  na.omit() %>%
  mutate(CNT_FAM_MEMBERS = as.factor(CNT_FAM_MEMBERS),
         TARGET = ifelse(TARGET=='No Default', 0, ifelse(TARGET=='Default', 1, TARGET)),
         TARGET = as.numeric(TARGET)) %>%
  glm(TARGET ~ ., data = ., family = binomial(link = "logit")) %>%
```

```
summary()
```

```
Call:
glm(formula = TARGET ~ ., family = binomial(link = "logit"), 
    data = .)

Coefficients:
                   Estimate Std. Error  z value Pr(>|z|)    
(Intercept)        -2.37454    0.01406 -168.866  < 2e-16 ***
CNT_FAM_MEMBERS2   -0.10975    0.01702   -6.446 1.15e-10 ***
CNT_FAM_MEMBERS3    0.04835    0.02098    2.305 0.021184 *  
CNT_FAM_MEMBERS4    0.03159    0.02680    1.179 0.238588    
CNT_FAM_MEMBERS5    0.12619    0.06026    2.094 0.036250 *  
CNT_FAM_MEMBERS6    0.55292    0.14602    3.787 0.000153 ***
CNT_FAM_MEMBERS7   -0.12416    0.42493   -0.292 0.770132    
CNT_FAM_MEMBERS8    1.52724    0.48815    3.129 0.001756 ** 
CNT_FAM_MEMBERS9   -8.19149   48.77263   -0.168 0.866621    
CNT_FAM_MEMBERS10   1.68139    1.22483    1.373 0.169828    
CNT_FAM_MEMBERS11  12.94056  119.46804    0.108 0.913743    
CNT_FAM_MEMBERS12  -8.19149   84.47666   -0.097 0.922752    
CNT_FAM_MEMBERS13  12.94056  119.46804    0.108 0.913743    
CNT_FAM_MEMBERS14  -8.19149   84.47666   -0.097 0.922752    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 168221  on 296564  degrees of freedom
Residual deviance: 168075  on 296551  degrees of freedom
AIC: 168103

Number of Fisher Scoring iterations: 9
```

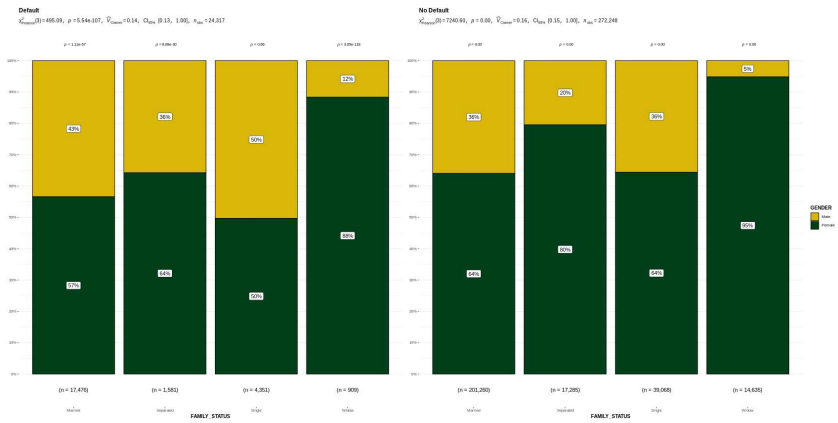
Family Status and Gender and Target Variable

```
%%R -w 60 -h 30 -u cm

p_fam2 <- rdf %>%
            select(TARGET, GENDER, FAMILY_STATUS) %>%
            na.omit() %>%
            mutate(GENDER = ifelse(GENDER=='F', 'Female', ifelse(GENDER=='M', 'Male', GENDER)),
                   TARGET = ifelse(TARGET==0, 'No Default', ifelse(TARGET==1, 'Default', TARGET))) %>%
            grouped_ggbarstats(
              x = GENDER,
              y = FAMILY_STATUS,
              grouping.var = TARGET,
              package = "wesanderson",
              palette = "Cavalcanti1",
              bf.message = FALSE
              )

extract_stats(p_fam2)
```

```
$subtitle_data
# A tibble: 1 × 13
  statistic    df p.value method                    effectsize        estimate
      <dbl> <int>   <dbl> <chr>                     <chr>                <dbl>
1     7241.     3       0 Pearson's Chi-squared test Cramer's V (adj.)    0.163
  conf.level conf.low conf.high conf.method conf.distribution  n.obs expression
       <dbl>    <dbl>     <dbl> <chr>       <chr>              <int> <list>    
1       0.95    0.154         1 ncp         chisq             272248 <language>

$caption_data
NULL

$pairwise_comparisons_data
NULL

$descriptive_data
# A tibble: 8 × 5
  FAMILY_STATUS GENDER counts  perc .label
  <fct>         <fct>   <int> <dbl> <chr>  
1 Married       Male    72431 36.0  36%    
2 Separated     Male     3539 20.5  20%    
3 Single        Male    13899 35.6  36%    
4 Widow         Male      757  5.17 5%     
5 Married       Female 128829 64.0  64%    
```

```
6 Separated     Female  13746 79.5  80%
7 Single        Female  25169 64.4  64%
8 Widow         Female  13878 94.8  95%

$one_sample_data
# A tibble: 4 × 10
  FAMILY_STATUS counts  perc N             statistic    df p.value method .label
  <fct>          <int> <dbl> <chr>             <dbl> <dbl>   <dbl> <chr>  <chr>
1 Widow          14635  5.38 (n = 14,635)     11764.     1       0 Chi-s… list(…
2 Single         39068 14.4  (n = 39,068)      3251.     1       0 Chi-s… list(…
3 Separated      17285  6.35 (n = 17,285)      6027.     1       0 Chi-s… list(…
4 Married       201260 73.9  (n = 201,260)    15804.     1       0 Chi-s… list(…
# ℹ 1 more variable: .p.label <chr>

$tidy_data
NULL

$glance_data
NULL
```

```
1 %%R -w 60 -h 30 -u cm
2 p_fam2
```



Interpretation:

This statistical analysis investigates the association between Family Status, Gender and the Target (loan default) variable. The Pearson's Chi-squared test reports a highly significant result (X^2 = 7241, df = 3, p < .0001), indicating that there is indeed an association among these three variables.

However, although statistically significant, the effect size, as given by the adjusted Cramer's V, is relatively small (0.163), which suggests that the practical impact of Family Status and Gender on default rates might not be very substantial. The observations from 272,248 customers were taken into account for this analysis.

The data suggest that women, regardless of their family status, have a higher representation in each category, except for separated individuals where men slightly outweigh women. Notably, widowed women comprise a significant 94.8% of the total, while widowed men only represent 5.17%.

In summary, while there is a statistically significant association between Family Status, Gender, and default rates, the actual impact on default rates is rather small according to the effect size. This implies that although Family Status and Gender can influence default rates to some extent, other factors or combinations of factors may play a larger role in determining default rates. Further research into these additional variables is recommended for a more comprehensive understanding of loan default.

4.2.5. Occupation

```
%%R

occupation_summary <- rdf %>%
  group_by(GENDER, OCCUPATION_TYPE) %>%
  summarise(EXT_SOURCE_3_mean = mean(EXT_SOURCE_3, na.rm = TRUE),
            Freq = n())


occupation_summary
```

```
`summarise()` has grouped output by 'GENDER'. You can override using the
`.groups` argument.
# A tibble: 38 × 4
# Groups:   GENDER [2]
   GENDER OCCUPATION_TYPE       EXT_SOURCE_3_mean  Freq
   <chr>  <chr>                             <dbl> <int>
 1 F      Accountants                       0.510 10572
 2 F      Cleaning staff                    0.504  4778
 3 F      Cooking staff                     0.497  6044
 4 F      Core staff                        0.494 24336
 5 F      Drivers                           0.491  1029
 6 F      HR staff                          0.483   587
 7 F      High skill tech staff             0.506  8254
 8 F      IT staff                          0.512   179
 9 F      Laborers                          0.502 26165
10 F      Low-skill Laborers                0.509   355
# ℹ 28 more rows
# ℹ Use `print(n = ...)` to see more rows
```

```
%%R

occupation_summary2 <- rdf %>%
  mutate(TARGET=)
  group_by(OCCUPATION_TYPE) %>%
  summarise(default_prob = mean(TARGET, na.rm = TRUE)*100,
            Freq = n())


occupation_summary2
```

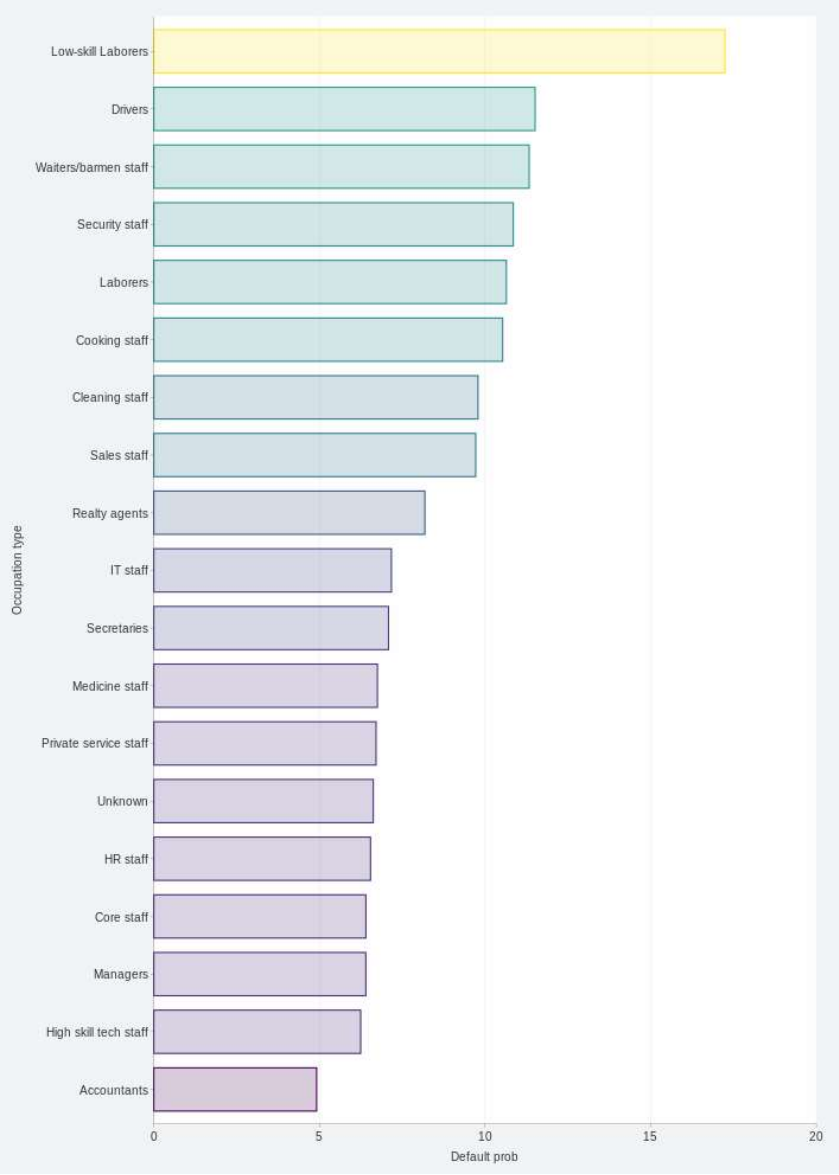
```
# A tibble: 19 × 3
   OCCUPATION_TYPE       default_prob   Freq
   <chr>                        <dbl>  <int>
 1 Accountants                     NA  10876
 2 Cleaning staff                  NA   5162
 3 Cooking staff                   NA   6672
 4 Core staff                      NA  30666
 5 Drivers                         NA  20704
 6 HR staff                        NA    630
 7 High skill tech staff           NA  12650
 8 IT staff                        NA    551
 9 Laborers                        NA  62408
10 Low-skill Laborers              NA   2334
11 Managers                        NA  23580
12 Medicine staff                  NA   9624
13 Private service staff           NA   2949
14 Realty agents                   NA    854
15 Sales staff                     NA  36255
16 Secretaries                     NA   1433
17 Security staff                  NA   7474
18 Unknown                         NA 107294
19 Waiters/barmen staff            NA   1486
```

```
%%R

mean(occupation_summary3$default_prob)
```

```
[1] 8.665014
```

```
%%R -w 25 -h 35 -u cm

occupation_summary3 %>%
  mutate(OCCUPATION_TYPE=reorder(OCCUPATION_TYPE, desc(default_prob))) %>%
  gg_errorbar(
    x = default_prob,
    y = OCCUPATION_TYPE,
    col=default_prob,
    stat = "summary",
    fun.data = \(x) mean_se(x, 1.96),
    width = 0.25,
    x_include = 0
  ) +
  geom_bar(
    stat = "summary",
    fun = \(x) mean(x),
    alpha = 0.2,
    width = 0.75)
```

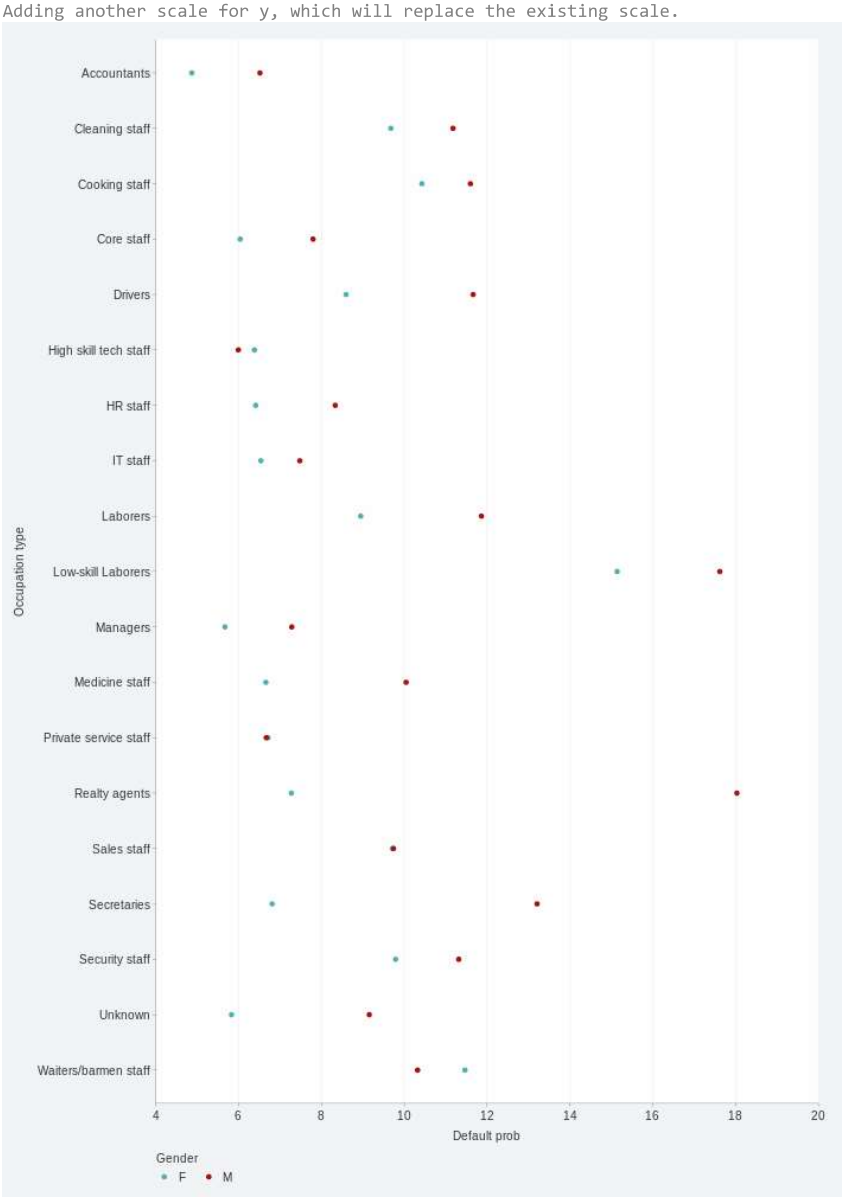
```
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
Scale for y is already present.
Adding another scale for y, which will replace the existing scale.
```



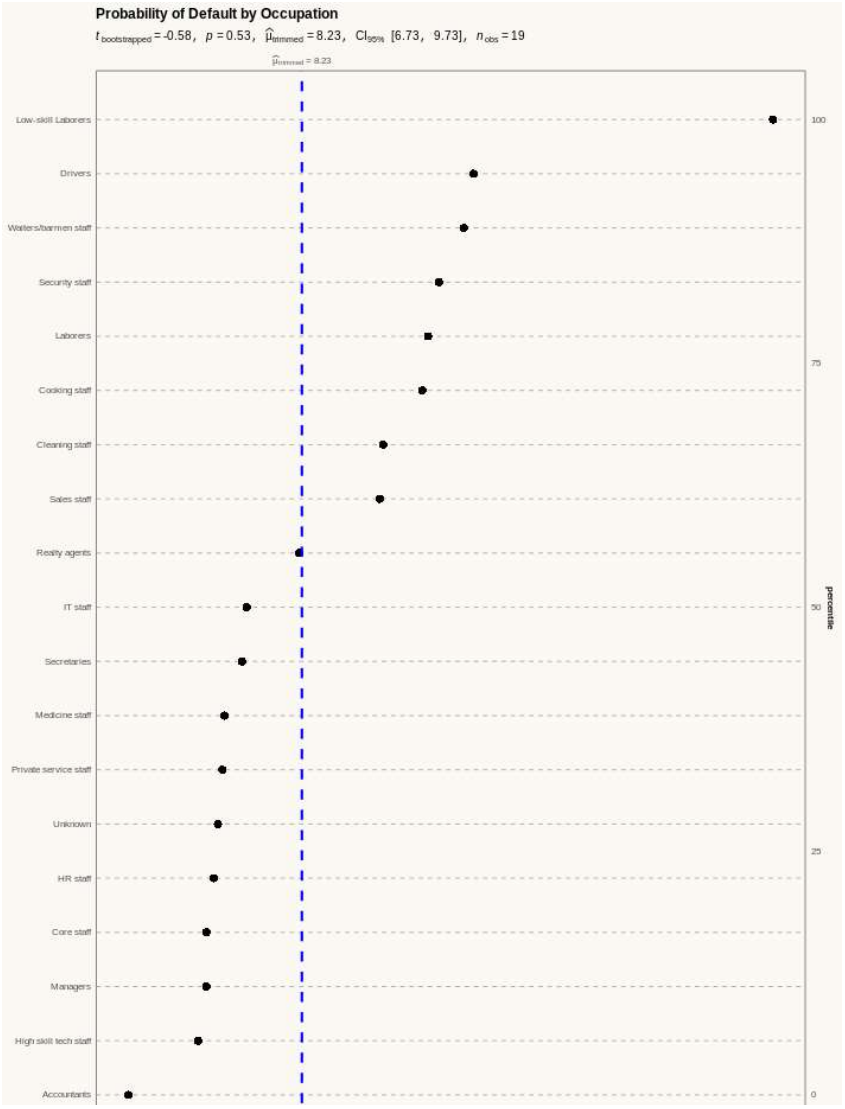

```
%%R -w 25 -h 35 -u cm

occupation_summary2 %>%
  gg_point(x=default_prob,
```

```
5           y=OCCUPATION_TYPE,
6           col=GENDER)
```

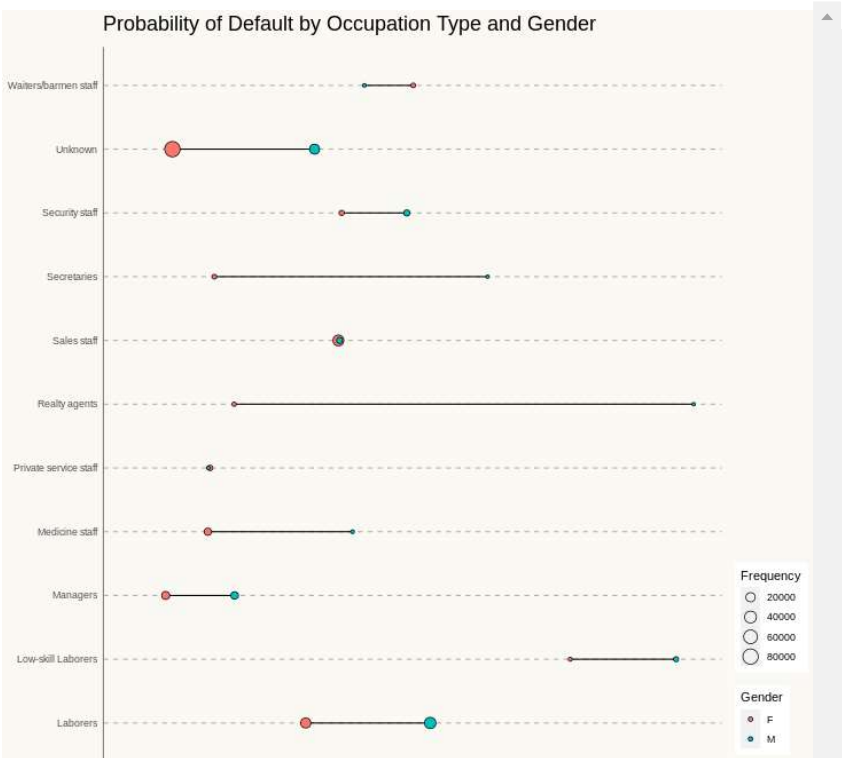
```
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
Scale for y is already present.
Adding another scale for y, which will replace the existing scale.
```




```
%%R -w 25 -h 35 -u cm

ggdotplotstats(
  data       = occupation_summary3,
  y          = OCCUPATION_TYPE,
  x          = default_prob  ,
  test.value = 8.665,
  type       = "robust",
  title      = "Probability of Default by Occupation",
  xlab       = "Probability of Default"
) + theme_bg
```



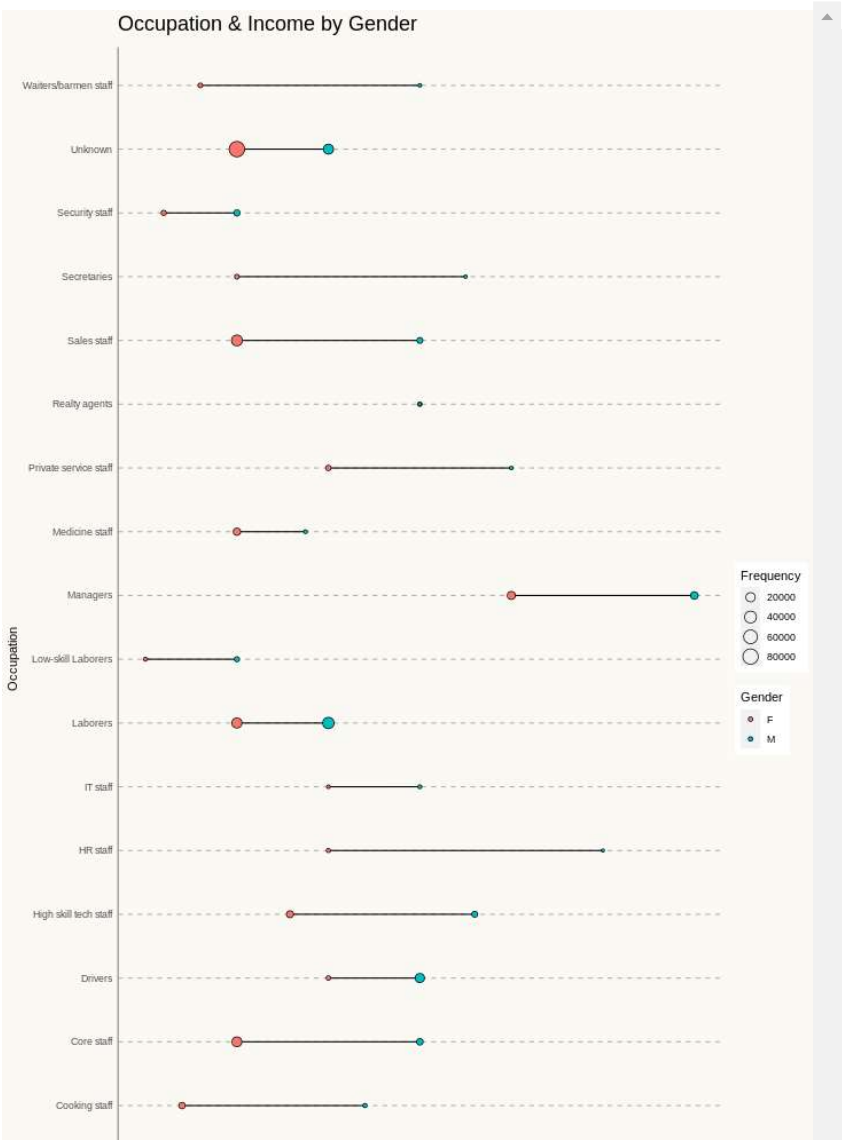

```
%%R -w 25 -h 40 -u cm

ggplot(data = occupation_summary2, mapping = aes(x = OCCUPATION_TYPE, y = default_prob, fill = GENDER)) +
    geom_line(aes(group = OCCUPATION_TYPE)) + geom_point(aes(size = Freq), shape = 21) +
    labs(x = NULL, y = "Probability of Default",
    fill = "Gender", size = "Frequency", title = "Probability of Default by Occupation Type and Gender") +
    coord_flip() +
    theme(plot.title = element_text(size = 18, margin = ggplot2::margin(b = 10)),
          plot.subtitle = element_text(size = 10, color = "darkslategrey")) + theme_bg
```




```
%%R

occupation_income <- rdf %>%
  select(OCCUPATION_TYPE, AMT_INCOME_TOTAL, GENDER) %>%
  group_by(OCCUPATION_TYPE, GENDER) %>%
  summarise(AMT_INCOME_TOTAL_medi = median(AMT_INCOME_TOTAL, na.rm = TRUE)*100,
            Freq = n())


occupation_income
```

```
`summarise()` has grouped output by 'OCCUPATION_TYPE'. You can override using
the `.groups` argument.
# A tibble: 38 × 4
# Groups:   OCCUPATION_TYPE [19]
   OCCUPATION_TYPE GENDER AMT_INCOME_TOTAL_medi  Freq
   <chr>           <chr>                  <dbl> <int>
 1 Accountants     F                   17100000 10572
 2 Accountants     M                   20250000   304
 3 Cleaning staff  F                   11250000  4778
 4 Cleaning staff  M                   13500000   384
 5 Cooking staff   F                   12150000  6044
 6 Cooking staff   M                   16650000   628
 7 Core staff      F                   13500000 24336
 8 Core staff      M                   18000000  6330
 9 Drivers         F                   15750000  1029
10 Drivers         M                   18000000 19675
# ℹ 28 more rows
# ℹ Use `print(n = ...)` to see more rows
```

```
%%R -w 25 -h 40 -u cm

ggplot(data = occupation_income, mapping = aes(x = OCCUPATION_TYPE, y = AMT_INCOME_TOTAL_medi, fill = GENDER)) +
    geom_line(aes(group = OCCUPATION_TYPE)) + geom_point(aes(size = Freq), shape = 21) +
    labs(y = 'Income', x = "Occupation",
    fill = "Gender", size = "Frequency", title = "Occupation & Income by Gender") +
    coord_flip() +
    theme(plot.title = element_text(size = 18, margin = ggplot2::margin(b = 10)),
          plot.subtitle = element_text(size = 10, color = "darkslategrey")) + theme_bg
```




```
%%R -w 25 -h 40 -u cm

ggplot(data = rdf, mapping = aes(x = OCCUPATION_TYPE, y = AMT_INCOME_TOTAL, fill = GENDER)) +
    geom_line(aes(group = OCCUPATION_TYPE)) + geom_point(aes(size = Freq), shape = 21) +
    labs(x = NULL, y = "Probability of Default",
    fill = "Gender", size = "Frequency", title = "Probability of Default by Occupation Type and Gender") +
    coord_flip() +
    theme(plot.title = element_text(size = 18, margin = ggplot2::margin(b = 10)),
          plot.subtitle = element_text(size = 10, color = "darkslategrey")) +
    theme_bg
```

```
app_df_cln['OCCUPATION_TYPE'].value_counts()
```

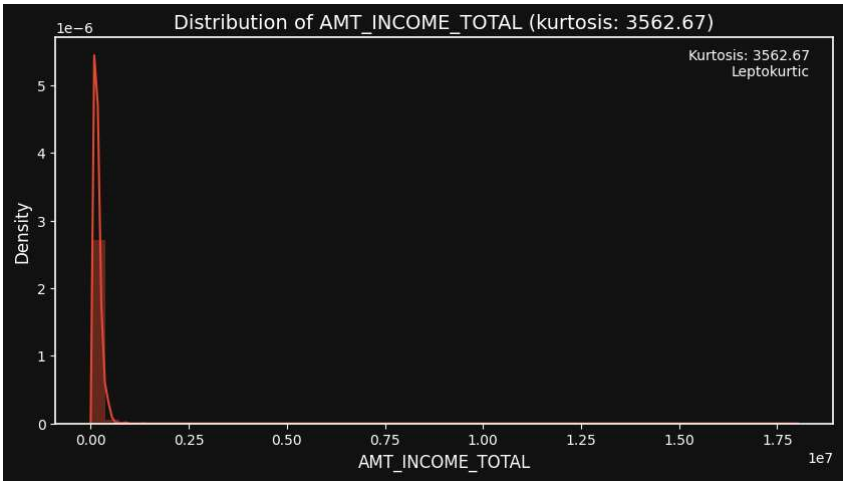
```
Unknown                 107294
Laborers                 62408
Sales staff              36255
Core staff               30666
Managers                 23580
Drivers                  20704
High skill tech staff    12650
Accountants              10876
Medicine staff            9624
Security staff            7474
Cooking staff             6672
Cleaning staff            5162
Private service staff     2949
Low-skill Laborers        2334
Waiters/barmen staff      1486
Secretaries               1433
Realty agents              854
HR staff                   630
IT staff                   551
Name: OCCUPATION_TYPE, dtype: int64
```

- IT staff

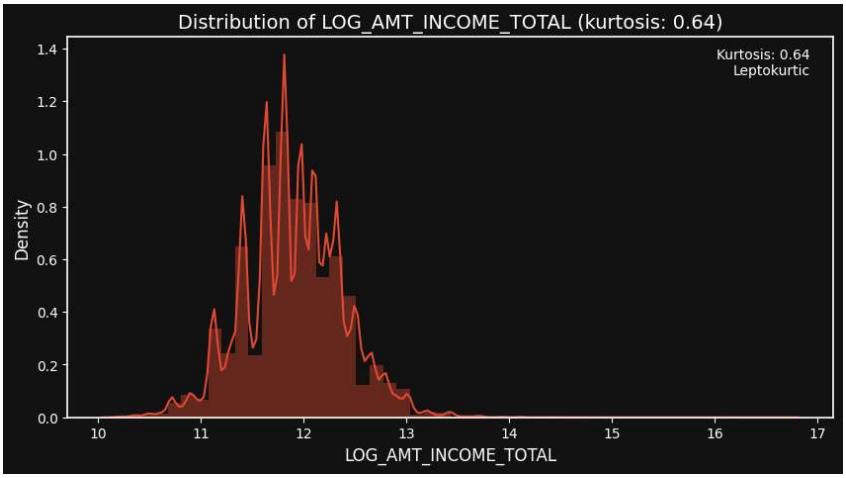
```
%%R -w 25 -h 40 -u cm

library(ggplot2)

ggplot(data = occupation_summary, mapping = aes(x = OCCUPATION_TYPE, y = EXT_SOURCE_3_mean, fill = GENDER)) +
  geom_line(aes(group = OCCUPATION_TYPE)) +
  geom_point(aes(size = Freq), shape = 21) +
  labs(x = NULL, y = "EXT_SOURCE_3 Scores", fill = "Gender", size = "Frequency", title = "External Credit Score by Occupation and Gender") +
  coord_flip() +
  scale_fill_manual(values = c("M" = "darkcyan", "F" = "brown3")) +  # Set custom colors for each gender
  theme(
    plot.title = element_text(size = 18, margin = ggplot2::margin(b = 10)),
    plot.subtitle = element_text(size = 10, color = "darkslategrey")
  ) +
  theme_bg
```

4.2.6. Total Income

```
plot_dist_kurtosis(app_df_cln, ['AMT_INCOME_TOTAL'])
```




```
plot_dist_kurtosis(app_df_cln, ['LOG_AMT_INCOME_TOTAL'])
```



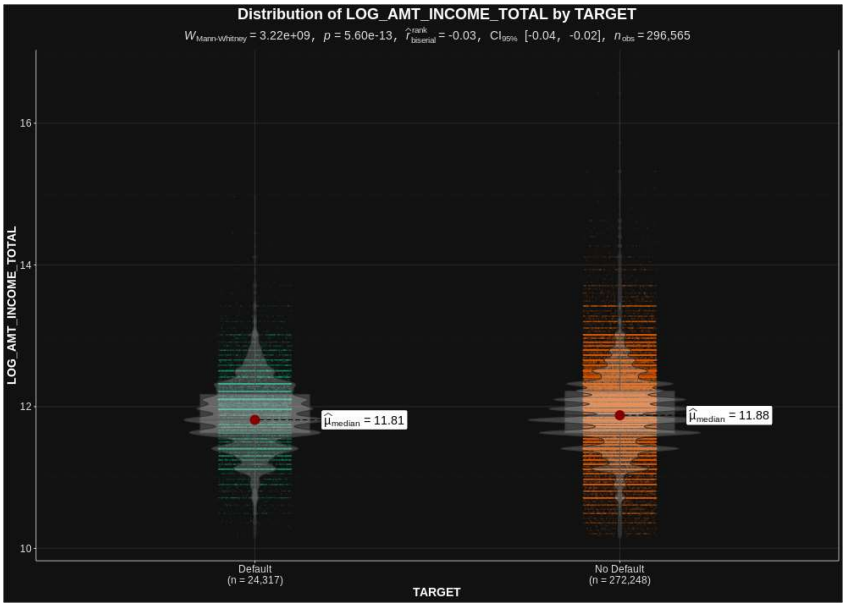
4.2.7. Impact of Total Income on Target Variable

```
temp_df = app_df_cln[['TARGET', 'AMT_INCOME_TOTAL']].copy(deep=True)
temp_df = temp_df.dropna()
temp_df['LOG_AMT_INCOME_TOTAL'] = np.log(temp_df['AMT_INCOME_TOTAL'])
# Add constant for statsmodels
temp_df['intercept'] = 1.0

# Define the logistic regression model
logit_model = sm.Logit(temp_df['TARGET'], temp_df[['intercept', 'LOG_AMT_INCOME_TOTAL']])

# Fit the model using robust standard errors
result = logit_model.fit(cov_type='HC3')

# Print the summary
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.283525
```

```
          Iterations 7
                          Logit Regression Results
==============================================================================
Dep. Variable:                 TARGET   No. Observations:               296565
Model:                          Logit   Df Residuals:                   296563
Method:                           MLE   Df Model:                            1
Date:                Mon, 26 Jun 2023   Pseudo R-squ.:              0.0003217
Time:                        06:31:26   Log-Likelihood:                -84084.
converged:                       True   LL-Null:                       -84111.
Covariance Type:                 HC3   LLR p-value:                 1.894e-13
============================================================================================
                               coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------------
intercept               -1.1962      0.160     -7.499      0.000      -1.509      -0.884
LOG_AMT_INCOME_TOTAL    -0.1026      0.013     -7.645      0.000      -0.129      -0.076
============================================================================================
```

```
%%R -w 35 -h 25 -u cm

rdf %>%
  select(TARGET, LOG_AMT_INCOME_TOTAL) %>%
  na.omit() %>%
  mutate(TARGET = ifelse(TARGET==0, 'No Default', ifelse(TARGET==1, 'Default', TARGET))) %>%
  ggbetweenstats(
    x     = TARGET,
    y     = LOG_AMT_INCOME_TOTAL,
    type = 'non-parametric',
    title = "Distribution of LOG_AMT_INCOME_TOTAL by TARGET",
    point.args = list(position = ggplot2::position_jitterdodge(dodge.width = 0.2),
                      alpha=0.1, size = 1, stroke = 0, na.rm = TRUE),
    centrality.point.args = list(size = 5, color = "darkred"),
    centrality.label.args = list(size = 5, nudge_x = 0.3, segment.linetype = 2,
    min.segment.length = 0),
    bf.messAMT_INCOME_TOTAL=FALSE,
  ) + my_dark_theme
```

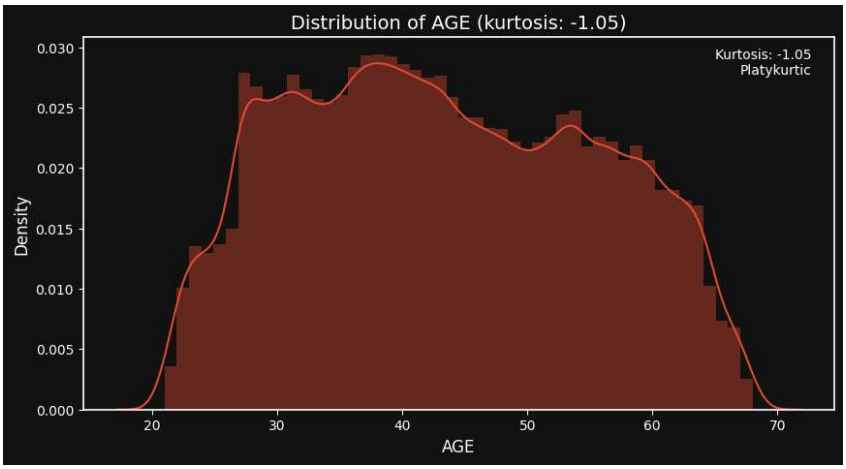


4.2.8. Relationship between Total Income and External Credit Score

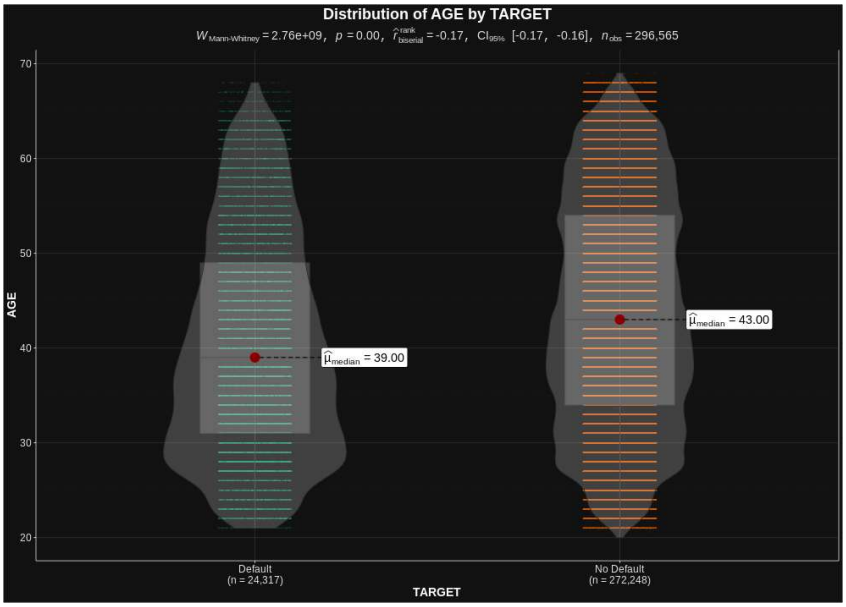
```
iris |>
  mutate(Species = str_to_sentence(Species)) |>
  gg_blank(
    x = Sepal.Width,
    y = Sepal.Length,
    col = Species,
    facet = Species,
    col_legend_place = "r", #two legends look better on right
    col_title = "\nSpecies") + #hack to add space between legends
  ggdensity::geom_hdr(col = NA) +
  labs(alpha = "Probs")
```

4.2.9. AGE

```
plot_dist_kurtosis(app_df_cln, ['AGE'])
```

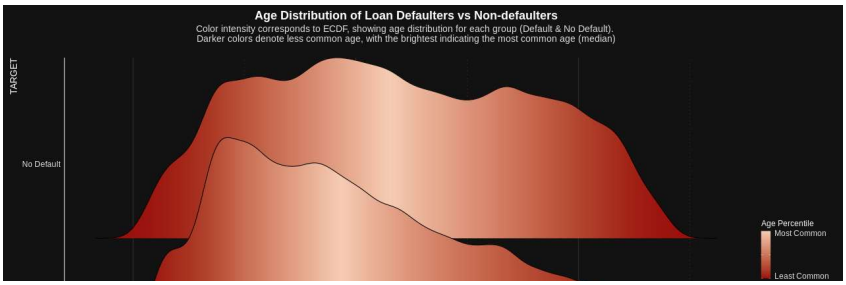



```
%%R -w 35 -h 25 -u cm

rdf %>%
  select(TARGET, AGE) %>%
  na.omit() %>%
  mutate(TARGET = ifelse(TARGET==0, 'No Default', ifelse(TARGET==1, 'Default', TARGET))) %>%
  ggbetweenstats(
    x     = TARGET,
    y     = AGE,
    type = 'non-parametric',
    title = "Distribution of AGE by TARGET",
    point.args = list(position = ggplot2::position_jitterdodge(dodge.width = 0.2),
                      alpha=0.1, size = 1, stroke = 0, na.rm = TRUE),
    centrality.point.args = list(size = 5, color = "darkred"),
    centrality.label.args = list(size = 5, nudge_x = 0.3, segment.linetype = 2,
    min.segment.length = 0),
    bf.message=FALSE,
  ) + my_dark_theme
```




```
%%R -w 45 -h 25 -u cm

palette1 <- wes_palette("BottleRocket1", 7)
palette2 <- wes_palette("BottleRocket2", 5)
palette3 <- wes_palette("Royal2", 5)

rdf %>%
  select(TARGET, FAMILY_STATUS, AGE, CNT_CHILDREN) %>%
  na.omit() %>%
  mutate(TARGET = ifelse(TARGET==0, 'No Default', ifelse(TARGET==1, 'Default', TARGET))) %>%
  filter(CNT_CHILDREN < 6) %>%
  ggplot(aes(x = AGE, y = TARGET, fill=0.5 - abs(0.5-stat(ecdf)))) +
    stat_density_ridges(geom='density_ridges_gradient', calc_ecdf=TRUE) +
    theme_ridges(grid = FALSE, center_axis_labels = FALSE) +
    scale_fill_gradient(low = palette1[3], high = palette3[2],
                        labels = c("0.0" = "", "0.25" = "Most Common \n \n \n \nLeast Common", "0.4" = ""),
                        breaks = c(0, 0.25, 0.51)) +
    my_dark_theme +
    labs(title = "Age Distribution of Loan Defaulters vs Non-defaulters",
          subtitle = "Color intensity corresponds to ECDF, showing age distribution for each group (Default & No Default). \n Darker color
         fill = "Age Percentile") +
    theme(axis.text.y = element_text(vjust = -12),
          panel.grid.major.y = element_blank(),
          panel.grid.minor.y = element_blank())
```

```
WARNING:rpy2.rinterface_lib.callbacks:R[write to console]: Picking joint bandwidth
```



4.2.10. Probability of Default by Age

```
%%R

model <- glm(TARGET ~ AGE, data = rdf, family = binomial)

# View the model summary
summary(model)
```

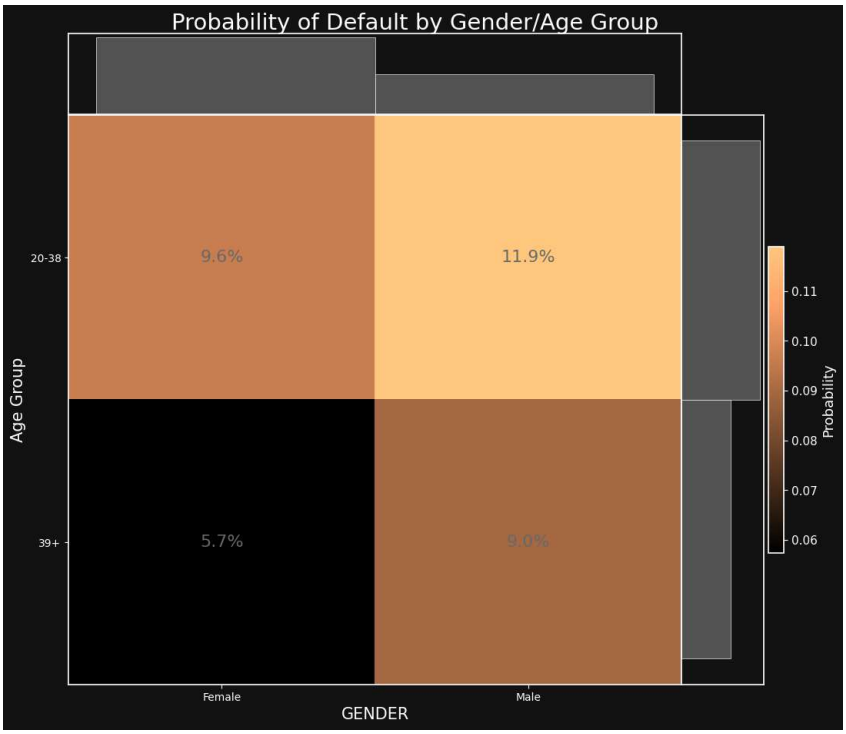
```

Call:
glm(formula = TARGET ~ AGE, family = binomial, data = rdf)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.3723426  0.0244769  -56.07   <2e-16 ***
AGE         -0.0248681  0.0005809  -42.81   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 168221  on 296564  degrees of freedom
Residual deviance: 166329  on 296563  degrees of freedom
  (47037 observations deleted due to missingness)
AIC: 166333

Number of Fisher Scoring iterations: 5
```

```
plot_prob_progression(app_df_cln['AGE'].dropna().reset_index(drop=True),
                      app_df_cln['TARGET'].dropna().reset_index(drop=True),
                      x_intervals=10, use_quartiles=True,  mean_line=True,
                      title='Probability of Default by Age',
                      text="For each 1 year increase in Age, the odds of loan default\ndecrease by approximately 2.5%")
```

> The above plot depicts how the youngest age group (20-38) is the most likely group to default on their loans as the probability of default is shown to be above average. This group belongs to a 'vulnerable' group or 'underpriviledged' group while the older group is 'priviledged' group.

```
app_df_fe['Age_Group'] = np.where((app_df_fe['AGE'] >= 20) & (app_df_fe['AGE'] <= 38), 0, 1)
```

```
temp_df = app_df_fe.dropna(subset='TARGET')
plot_prob_contour_map(
    temp_df.GENDER.replace({'M':'Male', 'F':'Female'}),
    temp_df.Age_Group.replace({0:'20-38', 1:'39+'}),
    temp_df.TARGET,
    xlabel='GENDER', ylabel='Age Group',
    title = 'Probability of Default by Gender/Age Group',
    annotate=True,
    plot_type='grid'
)
```



Despite having higher income on avergae across occupation types, male still have higher probability of default on both priviledged and unpriviledge group.

3. Feature Engineering

```
analyze_dataframe(app_df_cln).sort_values('Missing Percentage', ascending=False)
```

	Column	Data Type	Unique Count	Unique Sample	Missing Values
39	FLOORSMAX_MEDI	float64	48	[0.0833, 0.2917, nan, 0.1667, 0.3333]	150698
38	YEARS_BEGINEXPLUATATION_MEDI	float64	202	[0.9722, 0.9851, nan, 0.9811, 0.9806]	148098
40	TOTALAREA_MODE	float64	4858	[0.0149, 0.0714, nan, 0.0612, 0.1417]	146660
37	EXT_SOURCE_3	float64	814	[0.1393757800997895, nan, 0.7295666907060153, ...	58388
15	DAYS_EMPLOYED	float64	50	[1.0, 3.0, 0.0, 8.0, 4.0]	53294
...	...	...	...	...	...
28	HOUR_APPR_PROCESS_START	int64	24	[10, 11, 9, 17, 16]	0
29	REG_REGION_NOT_LIVE_REGION	int64	2	[0, 1]	0

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder

# Drop rows where TARGET is missing
app_df_cln = app_df_cln.dropna(subset=['TARGET'])
app_df_cln = app_df_cln[app_df_cln['CNT_CHILDREN'] < 6]
app_df_cln['Occupation_Gender'] = app_df_cln['OCCUPATION_TYPE'].astype(str) + '(' + app_df_cln['GENDER'].astype(str) + ')' #combine categc
# Define the threshold for the number of instances to collapse
threshold = 1000
# Calculate category counts
category_counts = app_df_cln['ORGANIZATION_TYPE'].value_counts()
# Identify categories to collapse
categories_to_collapse = category_counts[category_counts <= threshold].index
# Replace those categories with 'Other'
app_df_cln['ORGANIZATION_TYPE'] = app_df_cln['ORGANIZATION_TYPE'].replace(categories_to_collapse, 'Other')

# Drop SK_ID_CURR
app_df_cln = app_df_cln.drop(columns=['SK_ID_CURR'])

# Define features and target
X = app_df_cln.drop(columns=['TARGET'])
y = app_df_cln['TARGET']

# Stratify sampling to split the data into train, validation, and test sets
X_temp, X_test, y_temp, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp, test_size=0.2, stratify=y_temp, random_state=42)  # 0.25 x 0.8 = 0.2
```

```
!pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.6.1-py2.py3-none-any.whl (81 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.9/81.9 kB 3.0 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.25.0)
```

```
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.2.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.10.1)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.13.5)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.5.3)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (2022.7.
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_encoders)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_enc
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.9.0->category_encoders)
Installing collected packages: category_encoders
Successfully installed category_encoders-2.6.1
```

```
from category_encoders import TargetEncoder, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

# Define columns to be encoded
target_enc_cols = ['OCCUPATION_TYPE', 'ORGANIZATION_TYPE', 'Occupation_Gender']
onehot_enc_cols = ['CONTRACT_TYPE', 'GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'INCOME_TYPE', 'EDUCATION_TYPE', 'FAMILY_STATUS', 'WEEKDA

# Create encoder objects
target_enc = TargetEncoder(cols=target_enc_cols)
onehot_enc = OneHotEncoder(cols=onehot_enc_cols, use_cat_names=True)

# Apply target encoder to training data
target_enc.fit(X_train, y_train)
X_train_encoded = target_enc.transform(X_train)
X_val_encoded = target_enc.transform(X_val)

# Apply one hot encoder to training data
onehot_enc.fit(X_train_encoded, y_train)
X_train_encoded = onehot_enc.transform(X_train_encoded)
X_val_encoded = onehot_enc.transform(X_val_encoded)
```

```
X_test_encoded = target_enc.transform(X_test)
X_test_encoded = onehot_enc.transform(X_test_encoded)
```

Missing Value Imputation

```
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBRegressor
from sklearn.metrics import accuracy_score

# Specify your model
model = RandomForestClassifier(random_state=42)

# Create the Simple Imputer and the XGBoost Imputer
simple_imputer = SimpleImputer(strategy='mean')
xgboost_imputer = IterativeImputer(estimator=XGBRegressor(random_state=42), random_state=42)

# Create pipelines
simple_pipeline = make_pipeline(simple_imputer, model)
xgboost_pipeline = make_pipeline(xgboost_imputer, model)

# Fit the pipelines and evaluate
for pipeline in [simple_pipeline, xgboost_pipeline]:
    pipeline.fit(X_train_encoded, y_train)
    y_pred = pipeline.predict(X_val_encoded)
    print(f"Model accuracy: {accuracy_score(y_val, y_pred)}")

```

```
Model accuracy: 0.9181578670038992
```

```
!pip install gcimpute
```

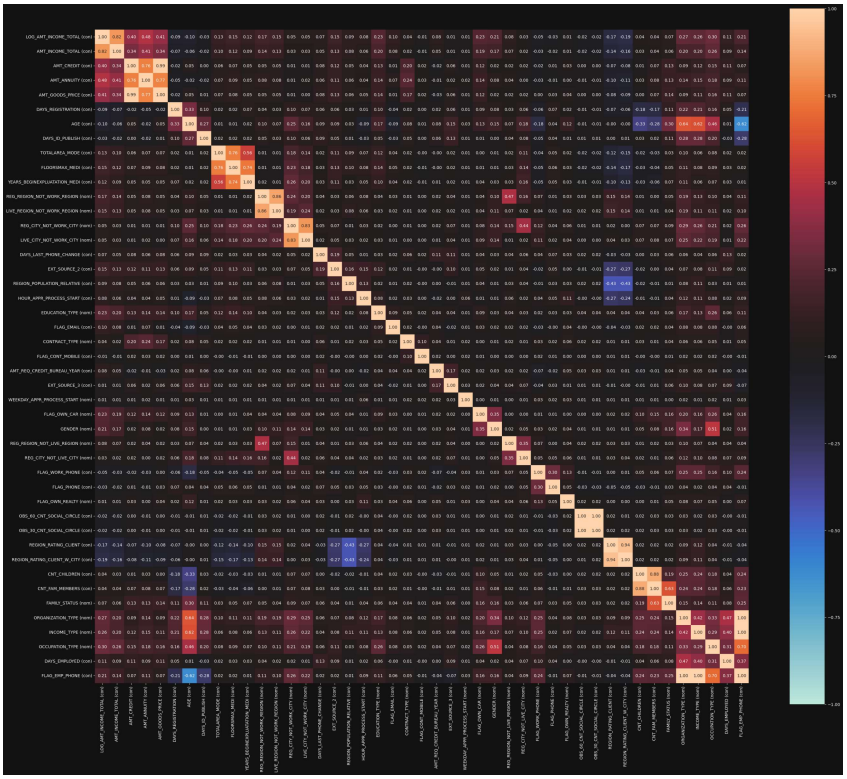
```
Collecting gcimpute
  Downloading gcimpute-0.0.4.tar.gz (749 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 749.3/749.3 kB 13.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: numpy>=1.16.4 in /usr/local/lib/python3.10/dist-packages (from gcimpute) (1.25.0)
Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.10/dist-packages (from gcimpute) (1.5.3)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from gcimpute) (1.10.1)
Requirement already satisfied: statsmodels>=0.11.0 in /usr/local/lib/python3.10/dist-packages (from gcimpute) (0.13.5)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-packages (from gcimpute) (4.65.0)
Requirement already satisfied: importlib_resources in /usr/local/lib/python3.10/dist-packages (from gcimpute) (5.12.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->gcimpute) (2.8.2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->gcimpute) (2022.7.1)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.11.0->gcimpute) (0.5.3)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.11.0->gcimpute) (23.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.2->statsmodels>=0.11.0->gcimpute) (1.16.
Building wheels for collected packages: gcimpute
  Building wheel for gcimpute (setup.py) ... done
  Created wheel for gcimpute: filename=gcimpute-0.0.4-py3-none-any.whl size=808886 sha256=0d21d1a96781b7e7e441fcf5d2859fb24780ef216c500
  Stored in directory: /root/.cache/pip/wheels/27/af/4e/4152194797bfad3e0c102e75dde40fe6641bb2720a2cfe380a
Successfully built gcimpute
Installing collected packages: gcimpute
Successfully installed gcimpute-0.0.4
```

```
from gcimpute.gaussian_copula import GaussianCopula
from gcimpute.helper_data import generate_mixed_from_gc
from gcimpute.helper_evaluation import get_smae
from gcimpute.helper_mask import mask_MCAR
import numpy as np

# generate and mask 15-dim mixed data (5 continuous variables, 5 ordinal variables (1-5) and 5 boolean variables)
X = generate_mixed_from_gc(n=2000)
X_mask = mask_MCAR(X, mask_fraction=0.4)

# model fitting
model = GaussianCopula(verbose=1)
X_imp = model.fit_transform(X=X_mask)

# Evaluation: compute the scaled-MAE (SMAE) for each data type (scaled by MAE of median imputation)
smae = get_smae(X_imp, X, X_mask)
print(f'The SMAE across continous variables: mean {smae[:5].mean():.3f} and std {smae[:5].std():.3f}')
print(f'The SMAE across oridnal variables: mean {smae[5:10].mean():.3f} and std {smae[5:10].std():.3f}')
print(f'The SMAE across boolean variables: mean {smae[10:].mean():.3f} and std {smae[10:].std():.3f}')
```

Feature Selection

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder

# Drop rows where TARGET is missing
app_df_cln = app_df_cln.dropna(subset=['TARGET'])

# Drop SK_ID_CURR
app_df_cln = app_df_cln.drop(columns=['SK_ID_CURR'])

# Define your features and target
X = app_df_cln.drop(columns=['TARGET'])
y = app_df_cln['TARGET']

# Stratify sampling to split the data into train, validation, and test sets
X_temp, X_test, y_temp, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp, test_size=0.2, stratify=y_temp, random_state=42)  # 0.25 x 0.8 = 0.2
```

```
temp_df = app_df_viz.drop(['SK_ID_CURR', 'TARGET'], axis=1)
nominal_columns = ['ORGANIZATION_TYPE', 'GENDER', 'CONTRACT_TYPE', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'WEEKDAY_APPR_PROCESS_START',
                   'INCOME_TYPE', 'EDUCATION_TYPE', 'FAMILY_STATUS', 'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
                    'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'OCCUPAT
# temp_df = temp_df[nominal_columns + ['EXT_SOURCE_3', 'TOTALAREA_MODE', 'REGION_RATING_CLIENT_W_CITY', 'LOG_AMT_INCOME_TOTAL', 'EXT_SOURC
#                                      'DAYS_LAST_PHONE_CHANGE']]


corr_result = nominal.associations(temp_df, nominal_columns=nominal_columns, nom_num_assoc='correlation_ratio', clustering=True, mark_colu
```



```
from dython import nominal

def find_top_correlated_variables(data, target_variable, nominal_columns=[], top_n=10):

    # Calculate the correlation matrix using dython's nominal associations
    corr_matrix = nominal.associations(data, nominal_columns=nominal_columns, nom_num_assoc='correlation_ratio', mark_columns=False, clust

    # Get the absolute correlation values with the target variable
    target_corr = corr_matrix[target_variable].abs()

    # Sort the correlation values in descending order
    top_correlated = target_corr.sort_values(ascending=False)

    # Exclude the target variable itself
    top_correlated = top_correlated.drop(target_variable)

    # Select the top N correlated variables
    top_n_variables = top_correlated[:top_n]

    return top_n_variables

temp_df = app_df_viz.drop(['SK_ID_CURR', 'TARGET'], axis=1)
nominal_columns = ['ORGANIZATION_TYPE', 'GENDER', 'CONTRACT_TYPE', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'WEEKDAY_APPR_PROCESS_START',
                   'INCOME_TYPE', 'EDUCATION_TYPE', 'FAMILY_STATUS', 'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
                    'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'OCCUPAT
```

```
variable_name = 'EXT_SOURCE_3'  # Replace 'target_variable' with the actual variable name
top_correlated_variables = find_top_correlated_variables(temp_df, variable_name, nominal_columns, top_n=10)
top_correlated_variables

```

```
import dython
```

```
---------------------------------------------------------------------------
ContextualVersionConflict                 Traceback (most recent call last)
<ipython-input-74-141924438240> in <cell line: 1>()
----> 1 import dython

                                6 frames
/usr/local/lib/python3.10/dist-packages/pkg_resources/__init__.py in
_resolve_dist(self, req, best, replace_conflicting, env, installer, required_by,
to_activate)
    871             # Oops, the "best" so far conflicts with a dependency
    872             dependent_req = required_by[req]
--> 873             raise VersionConflict(dist, req).with_context(dependent_req)
    874         return dist
    875

ContextualVersionConflict: (numpy 1.22.4 (/usr/local/lib/python3.10/dist-
packages), Requirement.parse('numpy>=1.23.0'), {'dython'})
```

```
from sklearn.impute import KNNImputer
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

# Select the variables for imputation
var_list = top_correlated_variables.index.tolist()

# Create a copy of the training data with selected variables
X_train_selected = X_train[var_list]

# Identify the categorical variables
categorical_cols = X_train_selected.select_dtypes(include=['object']).columns

# Create a column transformer to handle categorical variables
column_transformer = ColumnTransformer([
    ('encoder', OneHotEncoder(), categorical_cols)
], remainder='passthrough')

# Fit and transform the selected variables with column transformer
X_train_transformed = column_transformer.fit_transform(X_train_selected)

# Convert the transformed data to a dense numpy array
X_train_dense = X_train_transformed.toarray()

# Create a KNNImputer object
imputer = KNNImputer(n_neighbors=5)

# Fit and transform the dense data with the imputer
X_train_imputed = imputer.fit_transform(X_train_dense)

# Replace the imputed values in the original dataframe
X_train[var_list] = X_train_imputed[:, -len(var_list):]

```

```
EXT_SOURCE_2                0.098052
```

```
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.impute import KNNImputer
from sklearn.ensemble import RandomForestClassifier

var_list = top_correlated_variables.index.tolist()
X_train_imp = X_train[var_list]
y_train_imp = X_train['EXT_SOURCE_3']

# Define the range of hyperparameters for KNNImputer
k_values = [1, 3, 5, 7, 9, 15, 18, 21]

# Define the model
model = RandomForestClassifier()

# Evaluate each hyperparameter
for k in k_values:
    # Define the imputer for EXT_SOURCE_3
    imputer = KNNImputer(n_neighbors=k)

    # Impute missing values in EXT_SOURCE_3 using other variables
    X_train_imp_filled = X_train_imp.copy()
    X_train_imp_filled['EXT_SOURCE_3'] = imputer.fit_transform(X_train_imp, y_train_imp)

    # Define the pipeline
    pipeline = Pipeline(steps=[('m', model)])

    # Evaluate the pipeline
    cv = cross_val_score(pipeline, X_train_imp_filled, y_train_imp, cv=5, scoring='accuracy')

    # Report performance
    print('k=%d, Accuracy: %.3f' % (k, cv.mean()))

```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-72-9e9ef4e13d31> in <cell line: 17>()
     21     # Impute missing values in EXT_SOURCE_3 using other variables
     22     X_train_imp_filled = X_train_imp.copy()
---> 23     X_train_imp_filled['EXT_SOURCE_3'] =
imputer.fit_transform(X_train_imp, y_train_imp)
     24
     25     # Define the pipeline

                                    6 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in __array__(self,
dtype)
   2068
```

```python
from sklearn.cluster import KMeans
from imblearn.over_sampling import SMOTE
import numpy as np
import pandas as pd

class SimplifiedKMeansSMOTE:
    def __init__(self, k_clusters=100, k_neighbors=5):
        self.k_clusters = k_clusters
        self.k_neighbors = k_neighbors

    def fit_resample(self, X, y):
        # Perform K-means clustering
        kmeans = KMeans(n_clusters=self.k_clusters)
        kmeans.fit(X)

        # Identify clusters
        cluster_labels = kmeans.labels_

        # Initialize SMOTE
        smote = SMOTE(sampling_strategy='minority', k_neighbors=self.k_neighbors)

        # Apply SMOTE within each cluster
        X_resampled = pd.DataFrame()
        y_resampled = pd.Series()

        for cluster in np.unique(cluster_labels):
            X_cluster = X[cluster_labels == cluster]
            y_cluster = y[cluster_labels == cluster]

            if len(np.unique(y_cluster)) < 2:
                # If the cluster contains only one class, no need to resample
                X_resampled = pd.concat([X_resampled, X_cluster])
                y_resampled = pd.concat([y_resampled, y_cluster])
            else:
                X_res, y_res = smote.fit_resample(X_cluster, y_cluster)
                X_resampled = pd.concat([X_resampled, pd.DataFrame(X_res, columns=X.columns)])
                y_resampled = pd.concat([y_resampled, pd.Series(y_res)])

        return X_resampled, y_resampled

```

```python
# For numerical features, use an advanced imputation method to handle missing values
num_cols = X_train.select_dtypes(include=['int64', 'float64']).columns
imputer = KNNImputer(n_neighbors=5)
X_train[num_cols] = imputer.fit_transform(X_train[num_cols])
X_val[num_cols] = imputer.transform(X_val[num_cols])
X_test[num_cols] = imputer.transform(X_test[num_cols])

# Feature engineering on numerical features
# Add your feature engineering steps here

# One hot encoding on categorical features
cat_cols = X_train.select_dtypes(include=['object']).columns
encoder = OneHotEncoder(drop='first', sparse=False)
X_train = pd.concat([X_train.drop(cat_cols, axis=1), pd.DataFrame(encoder.fit_transform(X_train[cat_cols]))], axis=1)
X_val = pd.concat([X_val.drop(cat_cols, axis=1), pd.DataFrame(encoder.transform(X_val[cat_cols]))], axis=1)
X_test = pd.concat([X_test.drop(cat_cols, axis=1), pd.DataFrame(encoder.transform(X_test[cat_cols]))], axis=1)

# Use SimplifiedKMeansSMOTE
kmeans_smote = SimplifiedKMeansSMOTE(k_clusters=100, k_neighbors=5)
X_train_res, y_train_res = kmeans_smote.fit_resample(X_train, y_train)

```

```
# Now you can use X_train_res and y_train_res to train your model

```

Genetic Algorithm

```

```