s1

December 5, 2023

0.0.1 Packages

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

# read csv/xlsx
previous_application = pd.read_csv("./data/previous_application.csv")
application_data = pd.read_csv("./data/application_data.csv")
description = pd.read_excel("./data/columns_description.xlsx")
```

0.0.2 Data Cleaning

Prior to commencing our analysis, we conducted an initial assessment of the dataframe and identified the presence of NaN values. This initial data cleaning phase serves the dual purpose of streamlining our dataset. It involves the removal of superfluous columns, addressing specific subtopics, and mitigating substantial discrepancies arising from missing data.

[2]:
```
# Count the number of null values in each column of the 'application_data'␣
↪DataFrame
application_data.isnull().sum()
```

[2]:
```
SK_ID_CURR                       0
TARGET                           0
NAME_CONTRACT_TYPE               0
CODE_GENDER                      0
FLAG_OWN_CAR                     0
                               …
AMT_REQ_CREDIT_BUREAU_DAY    41519
AMT_REQ_CREDIT_BUREAU_WEEK   41519
AMT_REQ_CREDIT_BUREAU_MON    41519
AMT_REQ_CREDIT_BUREAU_QRT    41519
AMT_REQ_CREDIT_BUREAU_YEAR   41519
```

```
Length: 122, dtype: int64
```

[3]:
```python
# Count the number of null values in each column of the 'application_data'␣
↪DataFrame
null_data = application_data.isnull().sum() * 100/len(application_data)
missing_data_columns = null_data[null_data >= 40] # for greater then 40% of nan␣
↪values allocate them in this variable
missing_data_columns # print value
```

[3]:
```
OWN_CAR_AGE                      65.990810
EXT_SOURCE_1                     56.381073
APARTMENTS_AVG                   50.749729
BASEMENTAREA_AVG                 58.515956
YEARS_BEGINEXPLUATATION_AVG      48.781019
YEARS_BUILD_AVG                  66.497784
COMMONAREA_AVG                   69.872297
ELEVATORS_AVG                    53.295980
ENTRANCES_AVG                    50.348768
FLOORSMAX_AVG                    49.760822
FLOORSMIN_AVG                    67.848630
LANDAREA_AVG                     59.376738
LIVINGAPARTMENTS_AVG             68.354953
LIVINGAREA_AVG                   50.193326
NONLIVINGAPARTMENTS_AVG          69.432963
NONLIVINGAREA_AVG                55.179164
APARTMENTS_MODE                  50.749729
BASEMENTAREA_MODE                58.515956
YEARS_BEGINEXPLUATATION_MODE     48.781019
YEARS_BUILD_MODE                 66.497784
COMMONAREA_MODE                  69.872297
ELEVATORS_MODE                   53.295980
ENTRANCES_MODE                   50.348768
FLOORSMAX_MODE                   49.760822
FLOORSMIN_MODE                   67.848630
LANDAREA_MODE                    59.376738
LIVINGAPARTMENTS_MODE            68.354953
LIVINGAREA_MODE                  50.193326
NONLIVINGAPARTMENTS_MODE         69.432963
NONLIVINGAREA_MODE               55.179164
APARTMENTS_MEDI                  50.749729
BASEMENTAREA_MEDI                58.515956
YEARS_BEGINEXPLUATATION_MEDI     48.781019
YEARS_BUILD_MEDI                 66.497784
COMMONAREA_MEDI                  69.872297
ELEVATORS_MEDI                   53.295980
ENTRANCES_MEDI                   50.348768
FLOORSMAX_MEDI                   49.760822
```

```
FLOORSMIN_MEDI                67.848630
LANDAREA_MEDI                 59.376738
LIVINGAPARTMENTS_MEDI         68.354953
LIVINGAREA_MEDI               50.193326
NONLIVINGAPARTMENTS_MEDI      69.432963
NONLIVINGAREA_MEDI            55.179164
FONDKAPREMONT_MODE            68.386172
HOUSETYPE_MODE                50.176091
TOTALAREA_MODE                48.268517
WALLSMATERIAL_MODE            50.840783
EMERGENCYSTATE_MODE           47.398304
dtype: float64
```

[4]:
```
# dropping the above columns from the analysis as they contain a large␣
↪percentage of Nan values
application_data_df = application_data.drop(columns=missing_data_columns.index)
```

[5]:
```
application_data_df.shape

# Calculate the proportion of missing values in each row
missing_rows = application_data_df.isnull().sum(axis=1)/application_data_df.
↪shape[1]

# Assert that there are no rows with more than 50% NaN values
assert len(missing_rows[missing_rows > 50]) == 0

# Print a message indicating that none of the rows have more than 50% NaN values
print("we see that none of the rows have more than 50% nan values. so we will␣
↪proceed with further checks.")

# Display rows with more than 50% NaN values (if any)
missing_rows[missing_rows > 50]
```

```
we see that none of the rows have more than 50% nan values. so we will proceed
with further checks.
```

[5]:
```
Series([], dtype: float64)
```

Now that we removed the values greater then 50%, now we can take care of the null values, but will try to find the values with which it can be imputed at later point in time

[6]:
```
# Filter columns with missing data counts between 1 and 15
minor_missing_data_col = null_data[(null_data <= 15) & (null_data > 0)].
↪sort_values(ascending=False)

# Display columns with minor missing data
minor_missing_data_col
```

[6]:
```
AMT_REQ_CREDIT_BUREAU_HOUR    13.501631
AMT_REQ_CREDIT_BUREAU_DAY     13.501631
AMT_REQ_CREDIT_BUREAU_WEEK    13.501631
AMT_REQ_CREDIT_BUREAU_MON     13.501631
AMT_REQ_CREDIT_BUREAU_QRT     13.501631
AMT_REQ_CREDIT_BUREAU_YEAR    13.501631
NAME_TYPE_SUITE                0.420148
OBS_30_CNT_SOCIAL_CIRCLE       0.332021
DEF_30_CNT_SOCIAL_CIRCLE       0.332021
OBS_60_CNT_SOCIAL_CIRCLE       0.332021
DEF_60_CNT_SOCIAL_CIRCLE       0.332021
EXT_SOURCE_2                   0.214626
AMT_GOODS_PRICE                0.090403
AMT_ANNUITY                    0.003902
CNT_FAM_MEMBERS                0.000650
DAYS_LAST_PHONE_CHANGE         0.000325
dtype: float64
```

[7]:
```python
# Filter columns with missing data counts greater than 13 from
↪'minor_missing_data_col'
tmep_columns = minor_missing_data_col[minor_missing_data_col > 13].index.values

# Display information about the selected columns in 'application_data_df'
application_data_df[tmep_columns].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 6 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   AMT_REQ_CREDIT_BUREAU_HOUR  265992 non-null  float64
 1   AMT_REQ_CREDIT_BUREAU_DAY   265992 non-null  float64
 2   AMT_REQ_CREDIT_BUREAU_WEEK  265992 non-null  float64
 3   AMT_REQ_CREDIT_BUREAU_MON   265992 non-null  float64
 4   AMT_REQ_CREDIT_BUREAU_QRT   265992 non-null  float64
 5   AMT_REQ_CREDIT_BUREAU_YEAR  265992 non-null  float64
dtypes: float64(6)
memory usage: 14.1 MB
```

[8]:
```python
# Iterate over columns in 'tmep_columns' and print unique values count for each
↪column
for col in tmep_columns:
    print(f"{col} unique values: {application_data_df[col].nunique()}")
```

```
AMT_REQ_CREDIT_BUREAU_HOUR unique values: 5
AMT_REQ_CREDIT_BUREAU_DAY unique values: 9
AMT_REQ_CREDIT_BUREAU_WEEK unique values: 9
AMT_REQ_CREDIT_BUREAU_MON unique values: 24
```

```
AMT_REQ_CREDIT_BUREAU_QRT unique values: 11
AMT_REQ_CREDIT_BUREAU_YEAR unique values: 25
```

[9]:
```
# Display the column names of the 'application_data_df' DataFrame
application_data_df.columns
```

[9]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'FLAG_MOBIL',
       'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE',
       'FLAG_EMAIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS',
       'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
       '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',
       'ORGANIZATION_TYPE', 'EXT_SOURCE_2', 'EXT_SOURCE_3',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3',
       'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6',
       'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9',
       'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12',
       'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15',
       'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18',
       'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object')
```

[10]:
```
# This columns have no information we can gain from keeping these flag documents
application_data_df.drop(columns=['FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3',
                           'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5',␣
 ↪'FLAG_DOCUMENT_6',
                           'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8',␣
 ↪'FLAG_DOCUMENT_9',
                           'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11',␣
 ↪'FLAG_DOCUMENT_12',
                           'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14',␣
 ↪'FLAG_DOCUMENT_15',
                           'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17',␣
 ↪'FLAG_DOCUMENT_18',
```

```
                              'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20',␣
 ↪'FLAG_DOCUMENT_21',], inplace=True)
```

[11]:
```
application_data_df.columns
```

[11]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'FLAG_MOBIL',
       'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE',
       'FLAG_EMAIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS',
       'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
       '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',
       'ORGANIZATION_TYPE', 'EXT_SOURCE_2', 'EXT_SOURCE_3',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', 'AMT_REQ_CREDIT_BUREAU_HOUR',
       'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK',
       'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT',
       'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object')
```

[12]:
```
#merging the application_data with previous application data
all_data_df = pd.merge(left=application_data,␣
 ↪right=previous_application,how='inner', on='SK_ID_CURR',suffixes='_x')
```

Overall Research Topic: Factors Affecting People and Risk Involved in Loaning out Money for a Financial Institution: An Analysis of Risk, And Pricing and Profitability. - Subtopic 1 family status - Subtopic 2 employment, income conditions

0.1 Subtopic 1.1

- **Financial institutions, like banks, engage in lending activities that involve risk. So to minimize the likelihood of a default or a risky investment it is essential to understand the factors contributing to potential risks.**

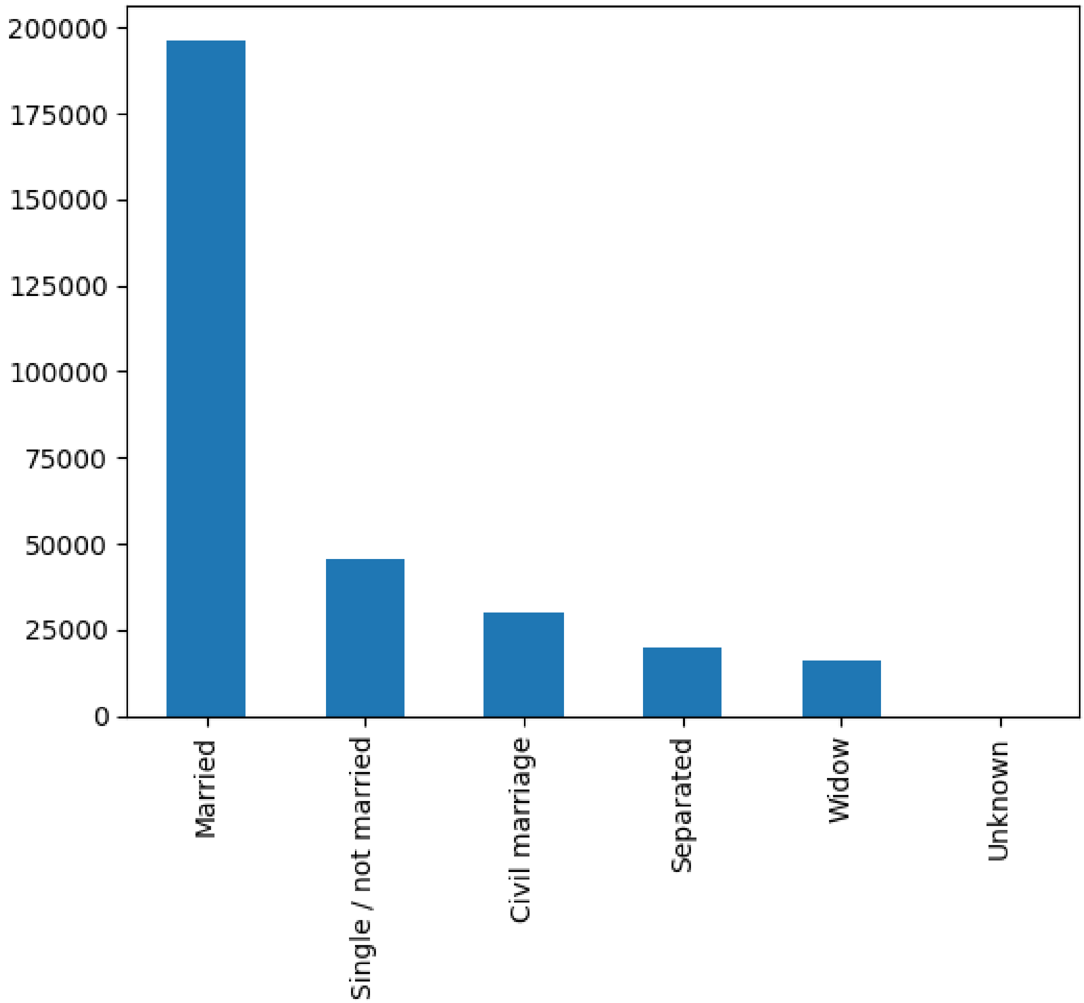
Hypothesis: People without children are greatly represented in clients with On-Time payments and people with children and providing for people other then themselves.

Background/Introduction of problem: One of the main questions we are trying to answer is whether there is a relationship between family status and loan payments, and loan application towards loan providing companies. We hypothesize that people without any children are greatly represented in clients with On-Time payments and the people with children and providing for people are greatly represented in clients with late-Time payments.

Outlier Detection First before we can find if this hypothesis is true we must indicate the existence of possible outliers within our data. Given if they exist. pruning them would be our course of action as we do not want skew our data analysis for unlikely configuration.

[13]:
```
#
application_data_df["NAME_FAMILY_STATUS"].value_counts().plot(kind="bar")
```

[13]: <AxesSubplot: >

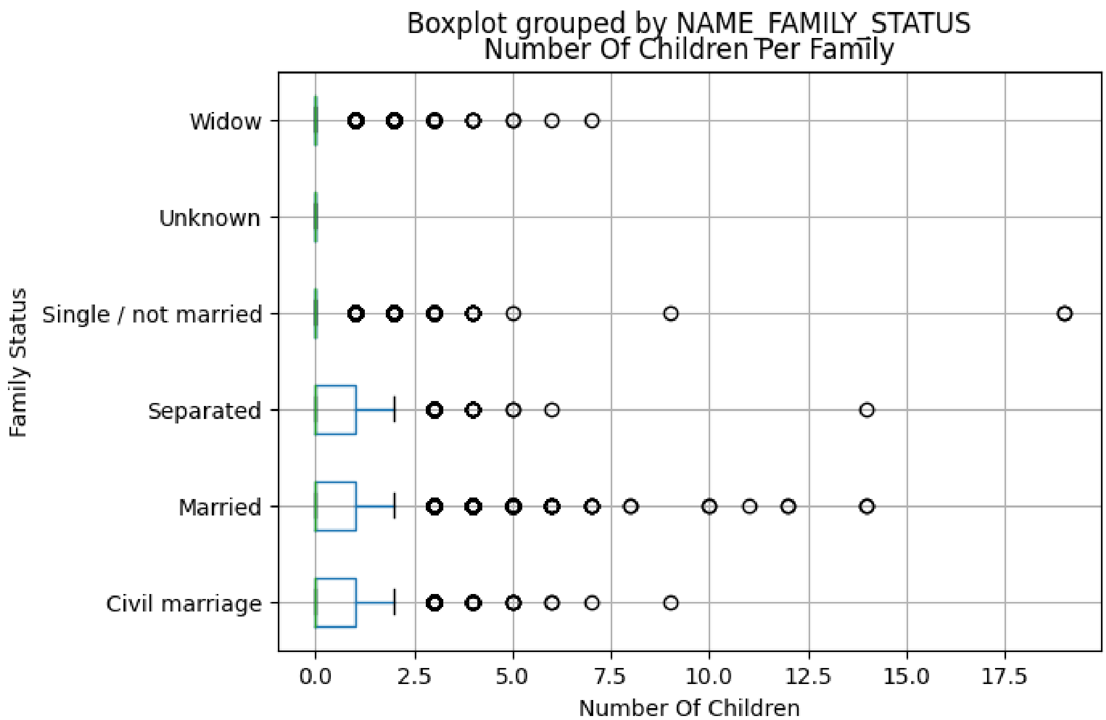


I can analysis that our data, that a majority of our data contains married clients.

[14]:
```
# box plot distribution of family status
application_data_df.boxplot(column="CNT_CHILDREN", by="NAME_FAMILY_STATUS",␣
    ↪vert=False)
plt.xlabel("Number Of Children")
plt.ylabel("Family Status")
```

```
plt.title("Number Of Children Per Family")
```

[14]: Text(0.5, 1.0, 'Number Of Children Per Family')

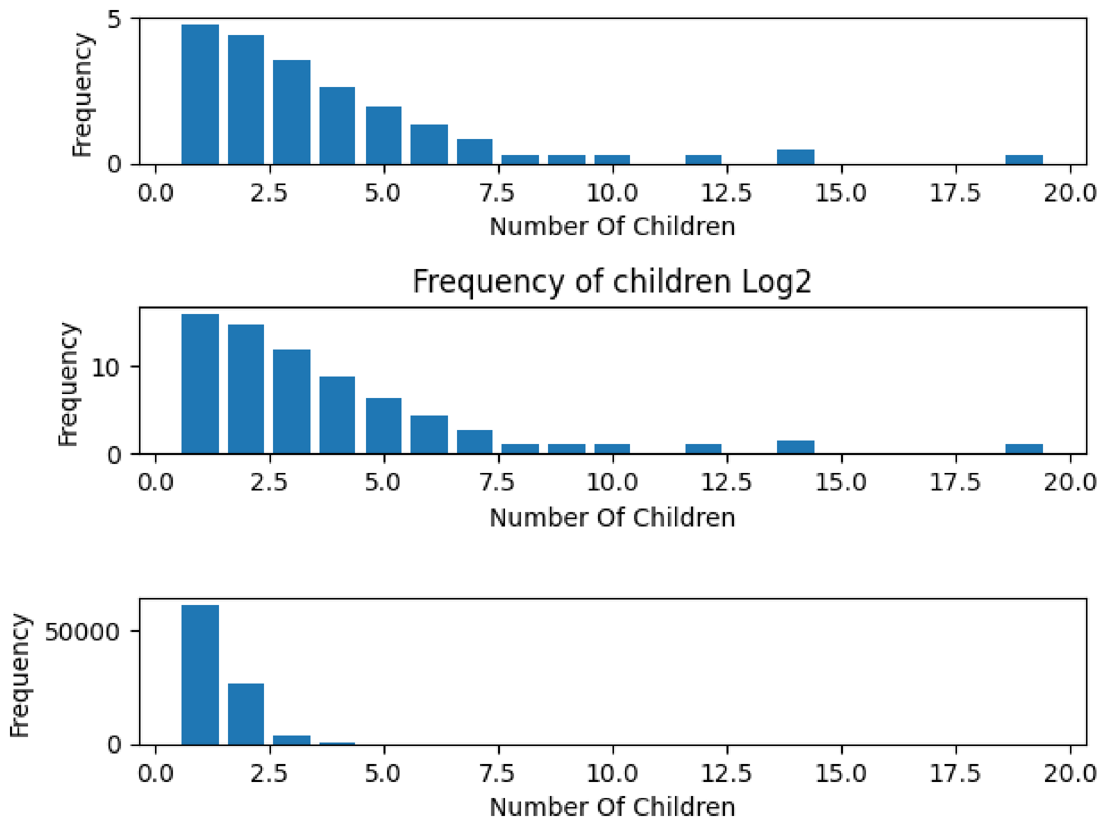


Within our data we see a high level of outliers, for a number of children the median while the mean amount of children per Family Status is 0 with a max contains around 0-1 children while beyond 2 we find that there exist a large group of outliers. within all Family Statuses given there outliers as we want to focus on the central tendency.

[15]:
```
# create two variables containing people with no children in the␣
↪application_data_df
no_children = application_data_df[application_data_df["CNT_CHILDREN"] == 0]
with_children  = application_data_df[application_data_df["CNT_CHILDREN"] > 0]
```

[16]:
```
f, (ax1, ax2, ax3) = plt.subplots(3, 1)

# log base 10 normalization
data = with_children["CNT_CHILDREN"].value_counts()
ax1.bar(data.index, np.log10(data))
ax1.set_xlabel("Number Of Children")
ax1.set_ylabel("Frequency")

# log base 2 normalization
ax2.bar(data.index, np.log2(data))
```

```
ax2.set_xlabel("Number Of Children")
ax2.set_ylabel("Frequency")
ax2.set_title("Frequency of children Log2")

ax3.bar(data.index, data)
ax3.set_xlabel("Number Of Children")
ax3.set_ylabel("Frequency")

plt.tight_layout()
```



[17]:
```
# for the sake of non generalization it is in our best interest to remove these outliers i.e families
# with 5 children or higher
with_children  = with_children[application_data_df["CNT_CHILDREN"] < 5]
assert (with_children["CNT_CHILDREN"] >= 5).sum() == 0, "Not all families with 5 children or over have been removed"
```

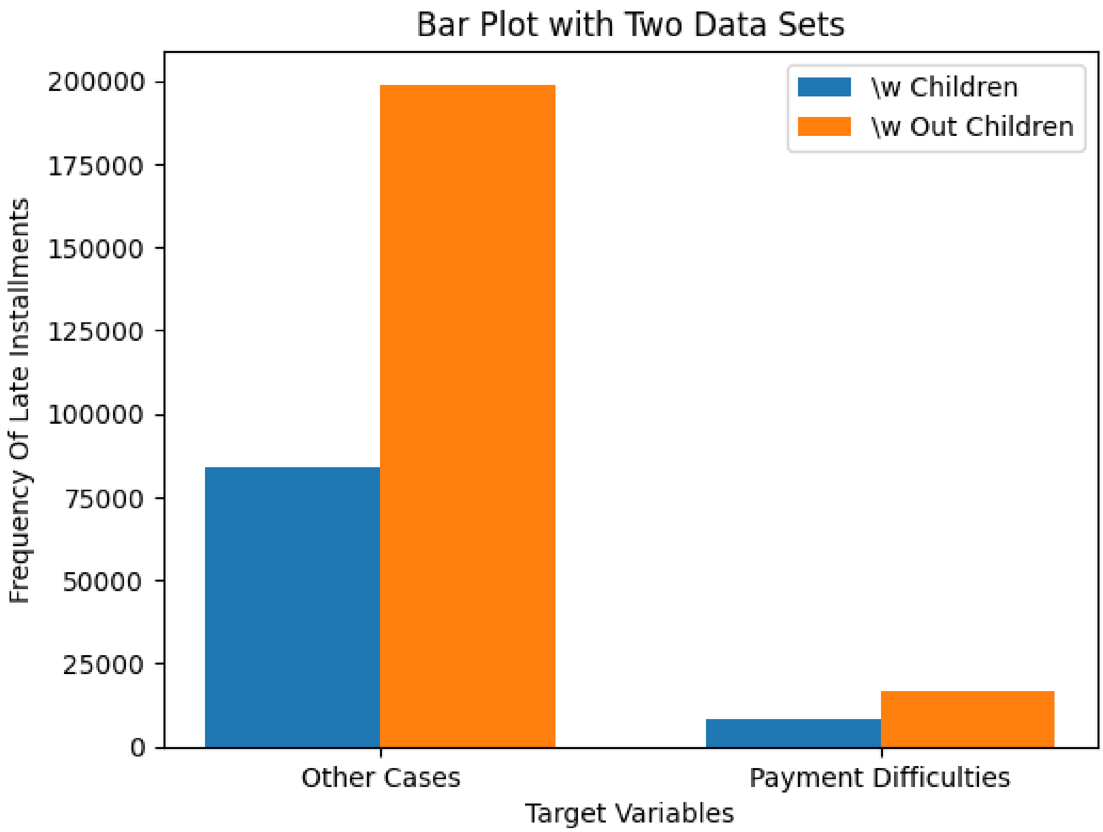
0.1.1 Analysis

Now that we have remove possible outliers within our data we can analysis our hypothesis proposed above. using a bar graph we can see between the two groups we separated within our application data on how they compare with payment difficulties by looking columns targets with respect to people with children, and people without.

[18]:
```
# get the description of the TARGET
target = description[description["Row"] == "TARGET" ]
print("TARGET:", target["Description"][1])
```

```
TARGET: Target variable (1 - client with payment difficulties: he/she had late
payment more than X days on at least one of the first Y installments of the loan
in our sample, 0 - all other cases)
```

[19]:
```
categories = ['Other Cases', 'Payment Difficulties']
# count the targets within both groups
target_child = with_children.groupby("TARGET")["TARGET"].value_counts().
 ↪to_list()
target_no_child = no_children.groupby("TARGET")["TARGET"].value_counts().
 ↪to_list()
bar_width = 0.35
index = np.arange(len(categories))

# Create bar plot
plt.bar(index, target_child, bar_width, label='\w Children')
plt.bar(index + bar_width, target_no_child, bar_width, label='\w Out Children')

# Customize plot
plt.xlabel('Target Variables')
plt.ylabel('Frequency Of Late Installments')
plt.title('Bar Plot with Two Data Sets')
plt.xticks(index + bar_width / 2, categories)
plt.legend()

# Show plot
plt.show()
```



[20]:
```python
# Describe target distribution of people who have no children
no_children["TARGET"].describe()
```

[20]:
```
count    215371.000000
mean          0.077118
std           0.266779
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           1.000000
Name: TARGET, dtype: float64
```

[21]:
```python
# Describe target distribution of people who have children
with_children["TARGET"].describe()
```

[21]:
```
count    92014.000000
mean         0.089117
std          0.284914
min          0.000000
```

```
25%          0.000000
50%          0.000000
75%          0.000000
max          1.000000
Name: TARGET, dtype: float64
```

As observed in the graph, individuals without children appear to encounter fewer difficulties with late installments. However, asserting this with absolute certainty is challenging due to differences in the sample sizes. To rigorously examine this pattern, we will conduct a chi-squared test, framing our hypotheses as follows:

- Null Hypothesis (H_0): The distribution of difficulties is the same in both groups.
- Alternative Hypothesis (H_1): The distribution of difficulties is different in the two groups.

The chi-squared test will help us determine if there is a statistically significant difference in the distribution of difficulties between individuals with and without children. Should the test reject the null hypothesis, we can infer that there is indeed a significant distinction, providing evidence contrary to our initial observation. It would suggest that individuals with children tend to experience fewer difficulties with installment payments, offering valuable insights into the relationship between family status and payment challenges.

[22]:
```
data = [target_child, target_no_child]
chi2, p, _, _ = chi2_contingency(data)
# Print results
print(f"Chi-square statistic: {chi2}")
print(f"P-value: {p}")

# Make a decision based on the p-value and significance level
alpha = 0.05
if p < alpha:
    print("Reject the null hypothesis - There is a significant difference in␣
 ↪distributions.")
else:
    print("Fail to reject the null hypothesis - No significant difference in␣
 ↪distributions.")
```

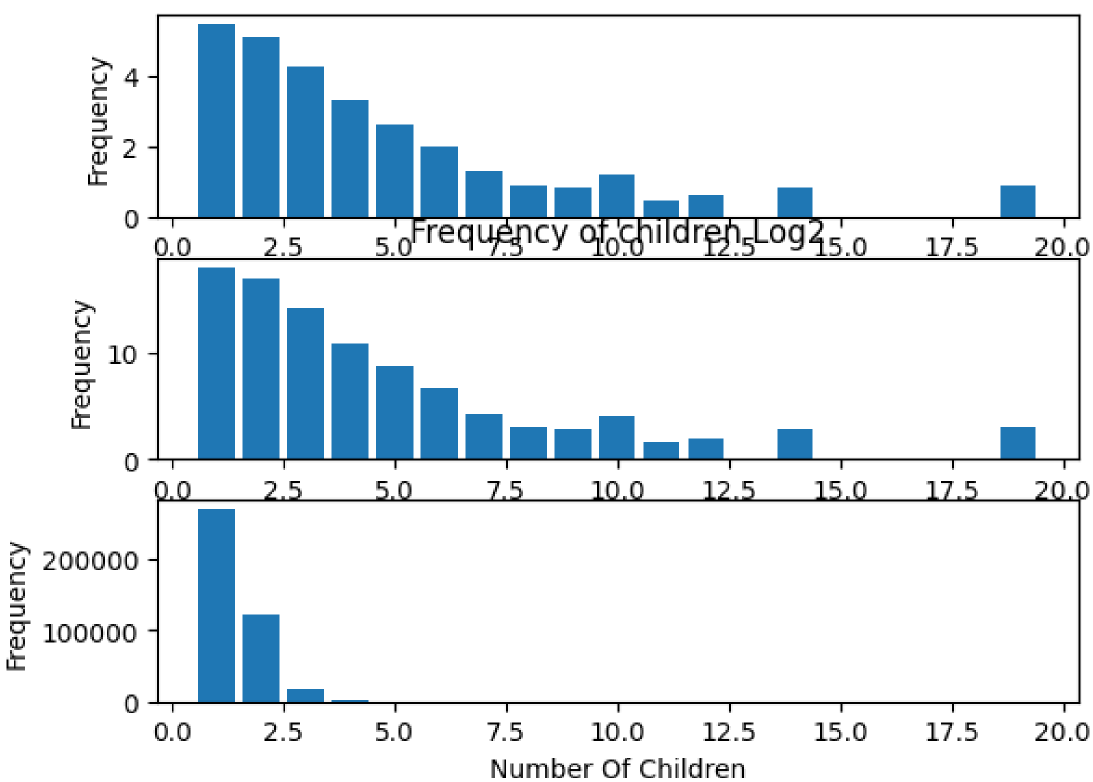
```
Chi-square statistic: 124.93731291617436
P-value: 5.252814292481338e-29
Reject the null hypothesis - There is a significant difference in distributions.
```

Given the results, you can conclude that there is a significant difference in the distribution of difficulties with installment payments between individuals with and without children. This supports your initial observation that people with children tend to have a different distribution of payment difficulties compared to those without children. As Families with children tend to find more difficulty in payments

To obtain results of non approved loan contracts we need to take our joined application with the previous application using that joint dataframe we can look how children effect the approval process of a given application and see if it may be correlated

[23]:
```python
# Our same logic applies to this joint dataframe
# 1. Seperate into two groups \w children, \w out children
no_children_joint = all_data_df[all_data_df["CNT_CHILDREN"] == 0]
with_children_join  = all_data_df[all_data_df["CNT_CHILDREN"] > 0]

f, (ax1, ax2, ax3) = plt.subplots(3, 1)

data = with_children_join["CNT_CHILDREN"].value_counts()
ax1.bar(data.index, np.log10(data))
ax1.set_xlabel("Number Of Children")
ax1.set_ylabel("Frequency")

ax2.bar(data.index, np.log2(data))
ax2.set_xlabel("Number Of Children")
ax2.set_ylabel("Frequency")
ax2.set_title("Frequency of children Log2")

ax3.bar(data.index, data)
ax3.set_xlabel("Number Of Children")
ax3.set_ylabel("Frequency")
```

[23]:
```
Text(0, 0.5, 'Frequency')
```



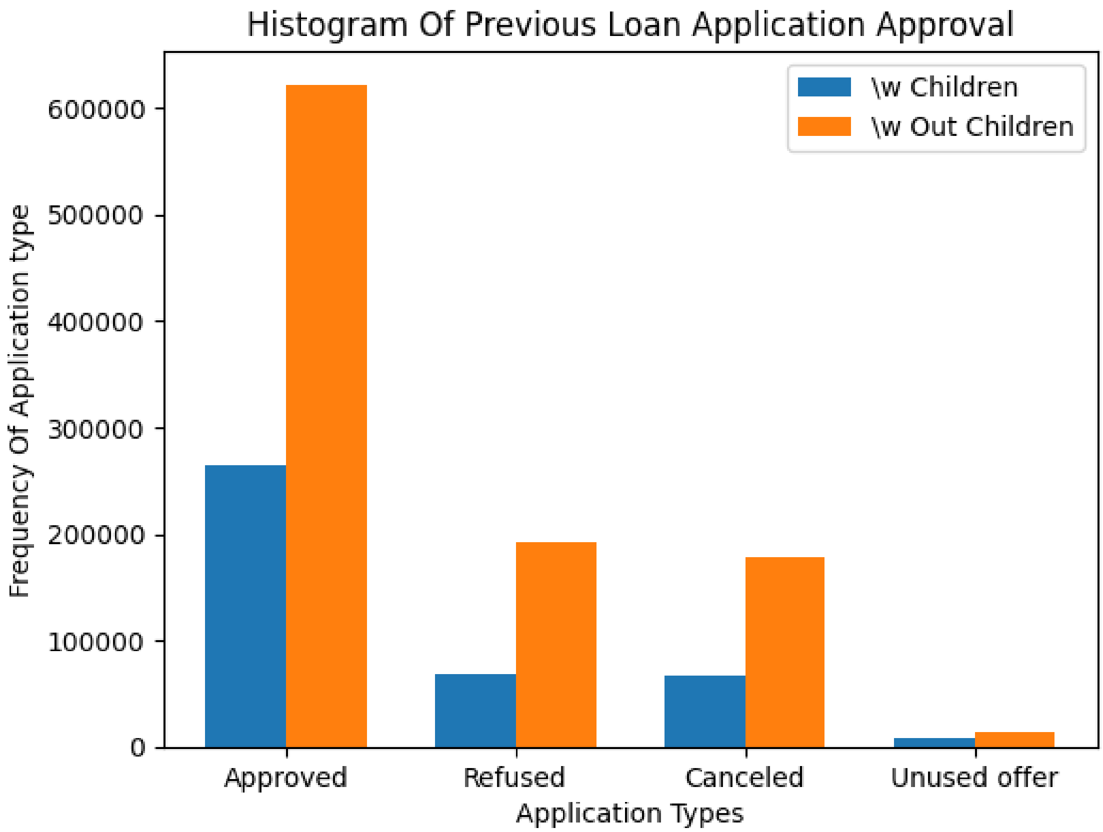
[24]:
```
# same thing occurs so we can threshold the families greater then 5
# 2. Remove outliers within our assumption we choose 5
with_children_join  = with_children_join[with_children_join["CNT_CHILDREN"] < 5]
assert (with_children_join["CNT_CHILDREN"] >= 5).sum() == 0, "Not all families␣
↪with 5 children or over have been removed"
```

Of these Now we can plot a histogram the reason why, is due to it's useful nature for showing the frequency (or count) of values within specific intervals or bins. When you have a dataset and want to understand how values are distributed across different ranges, a histogram provides a visual summary of the data's overall shape. this same idea applies in our search for families with, and with out children on there payment difficulties.

[25]:
```
with_children_join["NAME_CONTRACT_STATUS"].value_counts()
```

[25]:
```
Approved        263845
Refused          68184
Canceled         67503
Unused offer      8725
Name: NAME_CONTRACT_STATUS, dtype: int64
```

[26]:
```
categories = ['Approved', 'Refused', 'Canceled', 'Unused offer']
bar_width = 0.35
index = np.arange(len(categories))

target_child = with_children_join["NAME_CONTRACT_STATUS"].value_counts().
↪tolist()
target_no_child = no_children_joint["NAME_CONTRACT_STATUS"].value_counts().
↪tolist()

# Create bar plot
plt.bar(index, target_child, bar_width, label='\w Children')
plt.bar(index + bar_width, target_no_child, bar_width, label='\w Out Children')

# Customize plot
plt.xlabel('Application Types')
plt.ylabel('Frequency Of Application type')
plt.title('Histogram Of Previous Loan Application Approval')
plt.xticks(index + bar_width / 2, categories)
plt.legend()

# Show plot
plt.show()
```



[27]:
```python
data_with_children = [target_child, target_no_child]
chi2_stat, p_value, _, _ = chi2_contingency(data_with_children)

# Output the results
print(f"Chi-square statistic: {chi2_stat}")
print(f"P-value: {p_value}")

# Make a decision based on the p-value and a significance level (e.g., 0.05)
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The distribution is significantly␣
 ↪different.")
else:
    print("Fail to reject the null hypothesis: The distribution is not␣
 ↪significantly different.")
```

```
Chi-square statistic: 2438.4462486696584
P-value: 0.0
Reject the null hypothesis: The distribution is significantly different.
```

[28]:
```
green_color_code = '\033[92m'
red_color_code = '\033[91m'
reset_color_code = '\033[0m'

print(f"People with children average loan␣
↪{green_color_code}approval{reset_color_code}: {target_child[0]/
↪sum(target_child):.2f}")
print(f"People with NO children average loan␣
↪{green_color_code}approval{reset_color_code}: {target_no_child[0]/
↪sum(target_no_child):.2f}")

print(f"People with children average loan␣
↪{red_color_code}rejection{reset_color_code}: {target_child[1]/
↪sum(target_child):.2f}")
print(f"People with NO children average loan␣
↪{red_color_code}rejection{reset_color_code}: {target_no_child[1]/
↪sum(target_no_child):.2f}")
```

```
People with children average loan approval: 0.65
People with NO children average loan approval: 0.62
People with children average loan rejection: 0.17
People with NO children average loan rejection: 0.19
```

After a thorough analysis across different categories, a discernible pattern emerges, indicating that individuals without children often garner significantly less approval compared to their counterparts who are parents. This observation prompts consideration of an underlying financial risk, particularly highlighted by our initial histogram, revealing an elevated likelihood of missed payments among individuals with children, thus presenting a reversed dynamic.

0.1.2 Problem With Our Analysis

Sample Size Discrepancy - The sizes of the two groups (individuals with and without children) may not be balanced. A significant difference in sample sizes can impact the statistical power of the chi-squared test and affect the reliability of the results.

Confounding Variables - Other relevant factors that could influence payment difficulties may not have been considered. For example, income, employment status, or other demographic variables might confound the relationship between having children and payment difficulties.

Causation vs. Correlation - Statistical association does not imply causation. While you may observe a significant difference in distributions, it doesn't necessarily mean that having children causes payment difficulties. There could be other external factors like economic downturns, losing of ones Job, Divorce, etc influencing the observed patterns.