Credit card project

Project - To predict their clients' repayment abilities—

Table of content:-

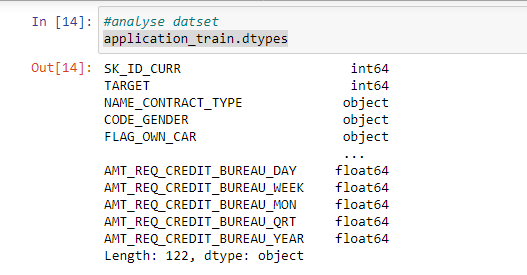
* Import necessary libraries in python notebook.
* Import the csv Data set in notebook

Import Necessary Libraries in python notebook

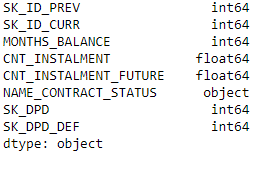
* For analysis using python we have to use pandas library which efficient in data frame
* Matplotlib library is used to plot charts for the data set.
* Numpy arrays are faster and more compact than python lists
* Import pandas as pd
* Import matplotlib.pyplot as plt
* Import as numpy as np
* Import the csv data set in notebook
* We have been given 8 csv files of credit card details
* To import csv files we hve to use command as
* application\_test = pd.read\_csv("application\_test.csv")
* application\_train = pd.read\_csv("application\_train.csv")
* bureau= pd.read\_csv("bureau.csv")
* bureau\_balance = pd.read\_csv("bureau\_balance.csv")
* installments\_payments = pd.read\_csv("installments\_payments.csv")
* POS\_CASH\_balance= pd.read\_csv("POS\_CASH\_balance.csv")
* previous\_application = pd.read\_csv("previous\_application.csv")
* credit\_card\_balance = pd.read\_csv("credit\_card\_balance.csv")

Check Data

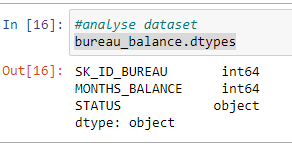
* to check the data types of a column in a data frame we use dtypes.
* application\_train.dtypes



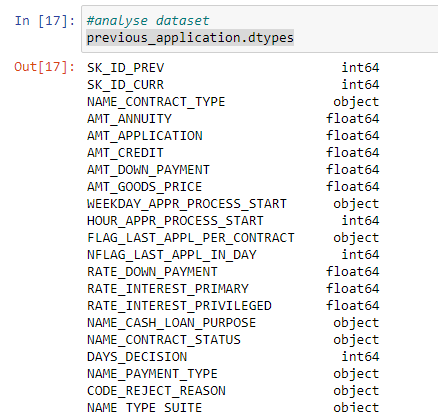
* POS\_CASH\_balance.dtypes



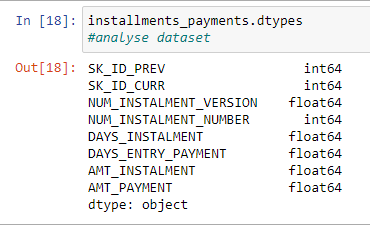
bureau\_balance.dtypes



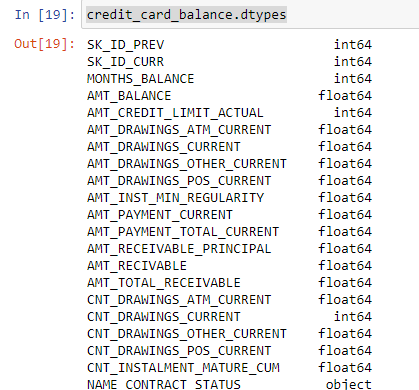
previous\_application.dtypes



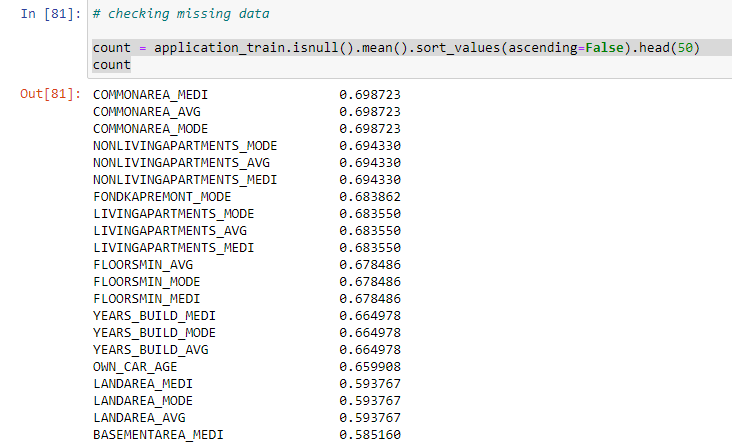
installments\_payments.dtypes



credit\_card\_balance.dtypes



* Check for missing data
* **isnull()**. values. any() will work for a DataFrame object to indicate if any value is missing , in some cases it may be useful to also count the number of missing values across the entire DataFrame.
* . count = application\_train.isnull().mean().sort\_values(ascending=False).head(50)
* count



# checking missing data

POS\_CASH\_balance.isna().sum()

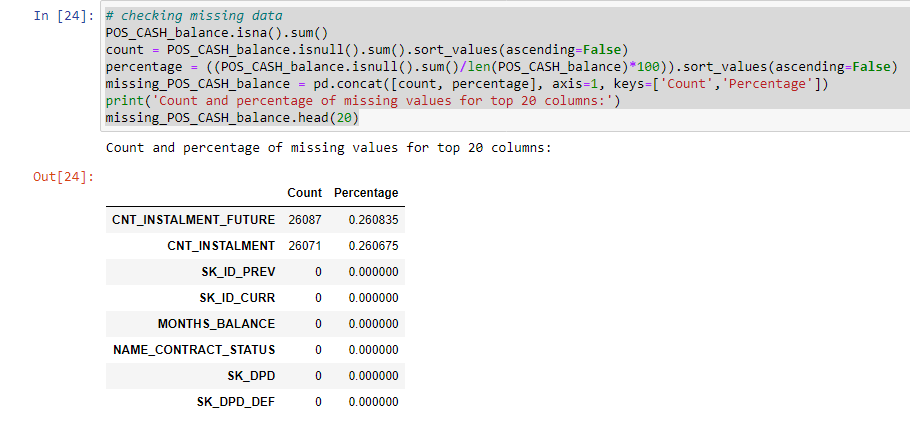
count = POS\_CASH\_balance.isnull().sum().sort\_values(ascending=False)

percentage = ((POS\_CASH\_balance.isnull().sum()/len(POS\_CASH\_balance)\*100)).sort\_values(ascending=False)

missing\_POS\_CASH\_balance = pd.concat([count, percentage], axis=1, keys=['Count','Percentage'])

print('Count and percentage of missing values for top 20 columns:')

missing\_POS\_CASH\_balance.head(20)



# checking missing data

# checking missing data prev\_app

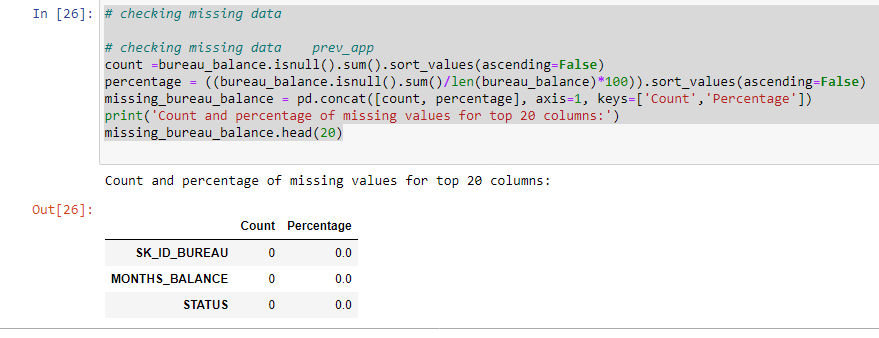
count =bureau\_balance.isnull().sum().sort\_values(ascending=False)

percentage = ((bureau\_balance.isnull().sum()/len(bureau\_balance)\*100)).sort\_values(ascending=False)

missing\_bureau\_balance = pd.concat([count, percentage], axis=1, keys=['Count','Percentage'])

print('Count and percentage of missing values for top 20 columns:')

missing\_bureau\_balance.head(20)



# checking missing data

previous\_application.isna().sum()

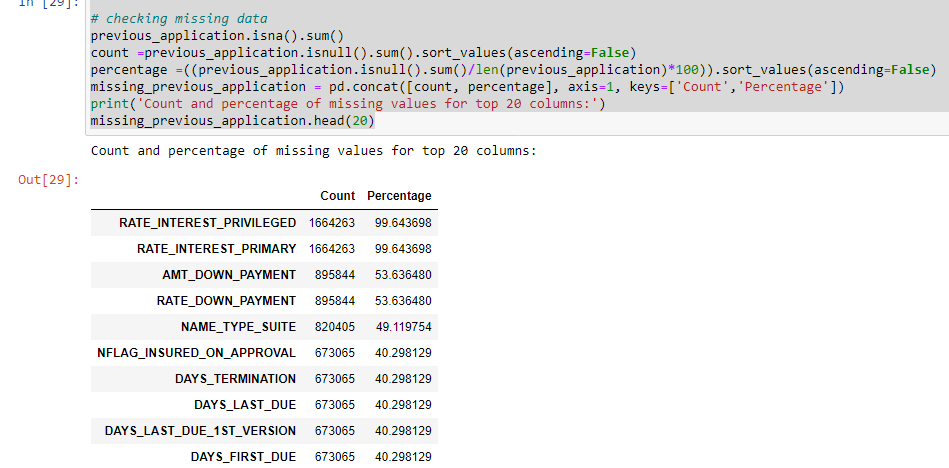
count =previous\_application.isnull().sum().sort\_values(ascending=False)

percentage =((previous\_application.isnull().sum()/len(previous\_application)\*100)).sort\_values(ascending=False)

missing\_previous\_application = pd.concat([count, percentage], axis=1, keys=['Count','Percentage'])

print('Count and percentage of missing values for top 20 columns:')

missing\_previous\_application.head(20)



# checking missing data

installments\_payments.isna().sum()

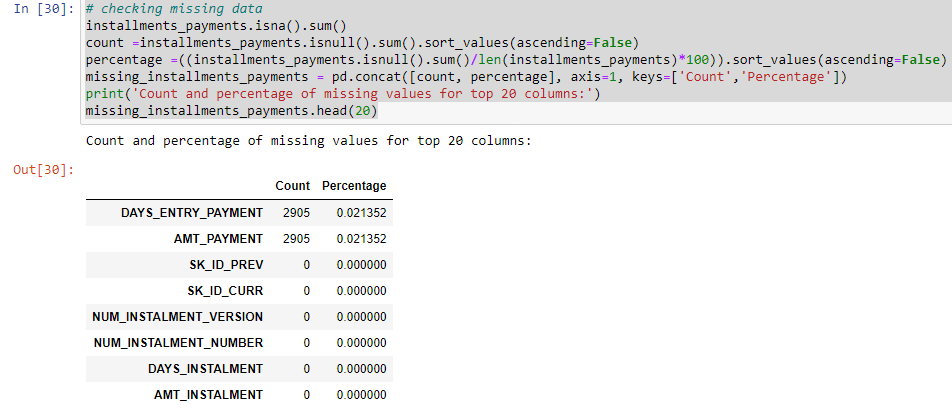
count =installments\_payments.isnull().sum().sort\_values(ascending=False)

percentage =((installments\_payments.isnull().sum()/len(installments\_payments)\*100)).sort\_values(ascending=False)

missing\_installments\_payments = pd.concat([count, percentage], axis=1, keys=['Count','Percentage'])

print('Count and percentage of missing values for top 20 columns:')

missing\_installments\_payments.head(20)



# checking missing data

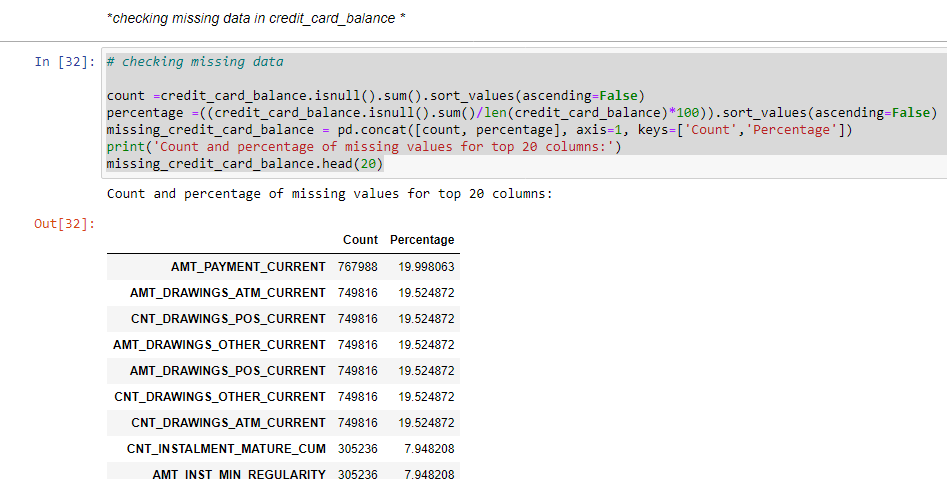
count =credit\_card\_balance.isnull().sum().sort\_values(ascending=False)

percentage =((credit\_card\_balance.isnull().sum()/len(credit\_card\_balance)\*100)).sort\_values(ascending=False)

missing\_credit\_card\_balance = pd.concat([count, percentage], axis=1, keys=['Count','Percentage'])

print('Count and percentage of missing values for top 20 columns:')

missing\_credit\_card\_balance.head(20)



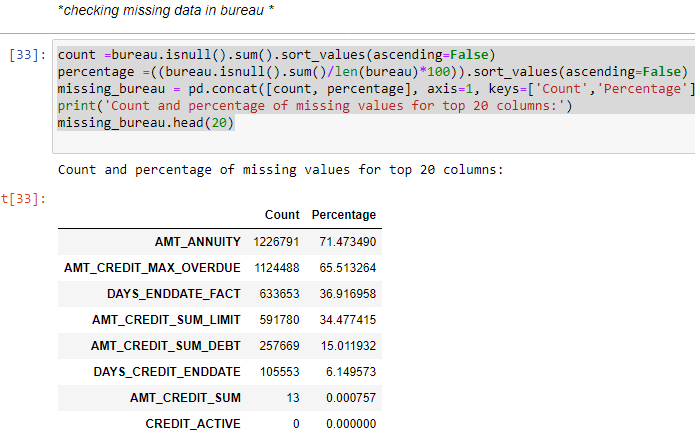
count =bureau.isnull().sum().sort\_values(ascending=False)

percentage =((bureau.isnull().sum()/len(bureau)\*100)).sort\_values(ascending=False)

missing\_bureau = pd.concat([count, percentage], axis=1, keys=['Count','Percentage'])

print('Count and percentage of missing values for top 20 columns:')

missing\_bureau.head(20)

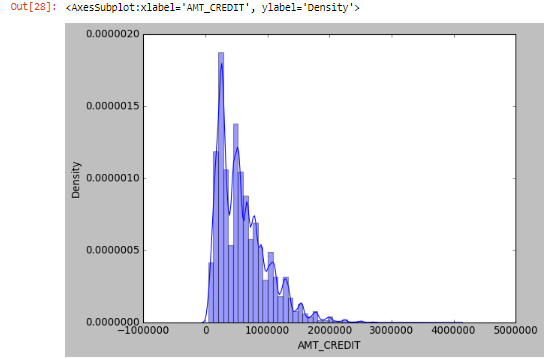


* Data exploration
* Data exploration is a key aspect of data analysis and model building.
* Data exploration techniques include both manual analysis and automated data exploration software solutions that visually explore and identify relationships between different data variables.

**Distribution of AMT\_CREDIT**

plt.style.use("classic")

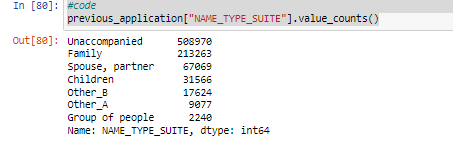
sns.distplot(application\_train["AMT\_CREDIT"],bins = 50)



* Who accompanied client when applying for the application

#code

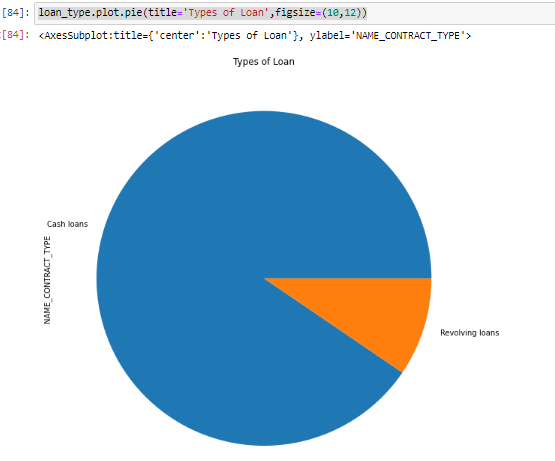
previous\_application["NAME\_TYPE\_SUITE"].value\_counts()



## Types of loan

* Rovolving loans : Arrangement which allows for the loan amount to be withdrawn, repaid, and redrawn again in any manner and any number of times, until the arrangement expires. Credit card loans and overdrafts are revolving loans. Also called evergreen loan

loan\_type.plot.pie(title='Types of Loan',figsize=(10,12))



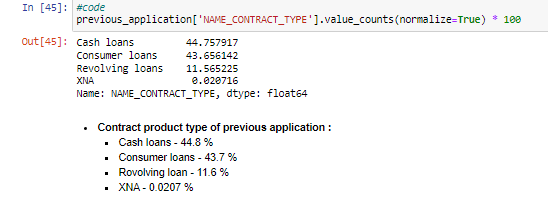
# Exploartion of previous application data

## Contract product type of previous application

## We use value\_counts() function. it is used to get a Series containing counts of unique values.

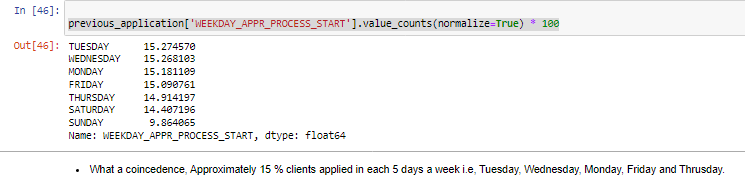
## With normalize set to True , returns the relative frequency by dividing all values by the sum of values

## We use unique () to get unique values of Series object. Uniques are returned in order of appearance.



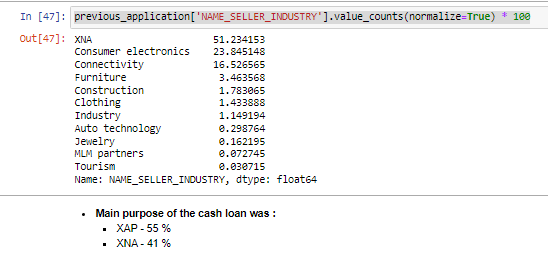
**On which day highest number of clients applied in prevoies application**

previous\_application['WEEKDAY\_APPR\_PROCESS\_START'].value\_counts(normalize=True) \* 100



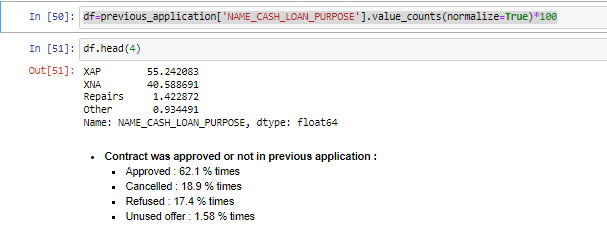
## Purpose of cash loan in previous application

previous\_application['NAME\_SELLER\_INDUSTRY'].value\_counts(normalize=True) \* 100



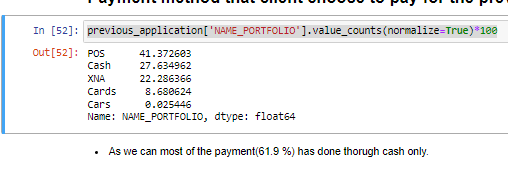
## Contract was approved or not in previous application

df=previous\_application['NAME\_CASH\_LOAN\_PURPOSE'].value\_counts(normalize=True)\*100



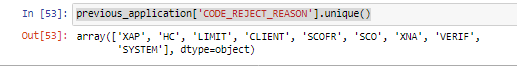
## Payment method that client choose to pay for the previous application

previous\_application['NAME\_PORTFOLIO'].value\_counts(normalize=True)\*100



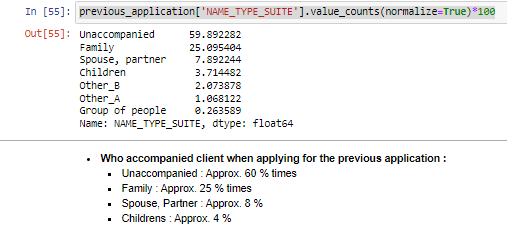
## Why was the previous application rejected ?

previous\_application['CODE\_REJECT\_REASON'].unique()



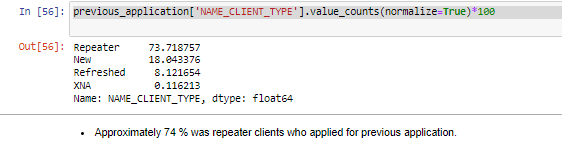
## Who accompanied client when applying for the previous application

previous\_application['NAME\_TYPE\_SUITE'].value\_counts(normalize=True)\*100



## Was the client old or new client when applying for the previous application

previous\_application['NAME\_CLIENT\_TYPE'].value\_counts(normalize=True)\*100



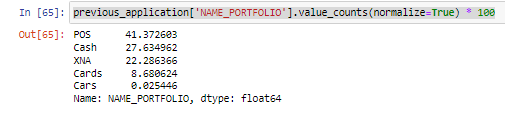
## What kind of goods did the client apply for in the previous application

## previous\_application['NAME\_GOODS\_CATEGORY'].value\_counts(normalize=True) \* 100



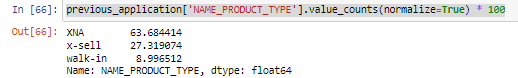
## Was the previous application for CASH, POS, CAR, …

previous\_application['NAME\_PORTFOLIO'].value\_counts(normalize=True) \* 100



## Was the previous application x-sell or walk-in ?

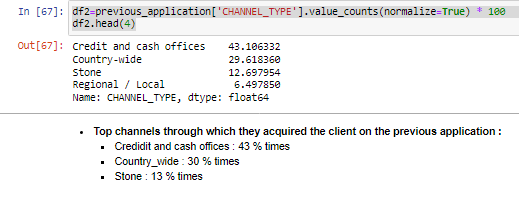
previous\_application['NAME\_PRODUCT\_TYPE'].value\_counts(normalize=True) \* 100



## Top channels through which they acquired the client on the previous application

df2=previous\_application['CHANNEL\_TYPE'].value\_counts(normalize=True) \* 100

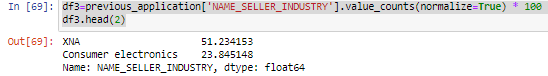
df2.head(4)



## Top industry of the seller

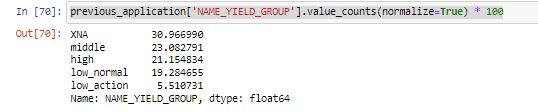
df3=previous\_application['NAME\_SELLER\_INDUSTRY'].value\_counts(normalize=True) \* 100

df3.head(2)



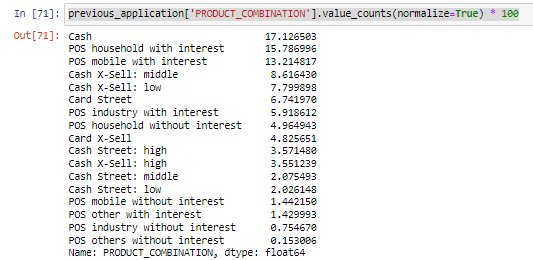
## Grouped interest rate into small medium and high of the previous application

previous\_application['NAME\_YIELD\_GROUP'].value\_counts(normalize=True) \* 100



## Top Detailed product combination of the previous application

previous\_application['PRODUCT\_COMBINATION'].value\_counts(normalize=True) \* 100



## Did the client requested insurance during the previous application

previous\_application['NFLAG\_INSURED\_ON\_APPROVAL'].value\_counts(normalize=True) \* 100

