**Introduction**

**Problem statement and objective**

**Requirement specifications**

The requirements needed to do this project are given below.

Hardware requirements:

Software requirements:

**Exploratory data analysis**

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns in the data, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

This is a very important step in our model building because we can identify which data is the most influential on the output data. We also can determine unnecessary data, we can clean the data in later parts of the process.

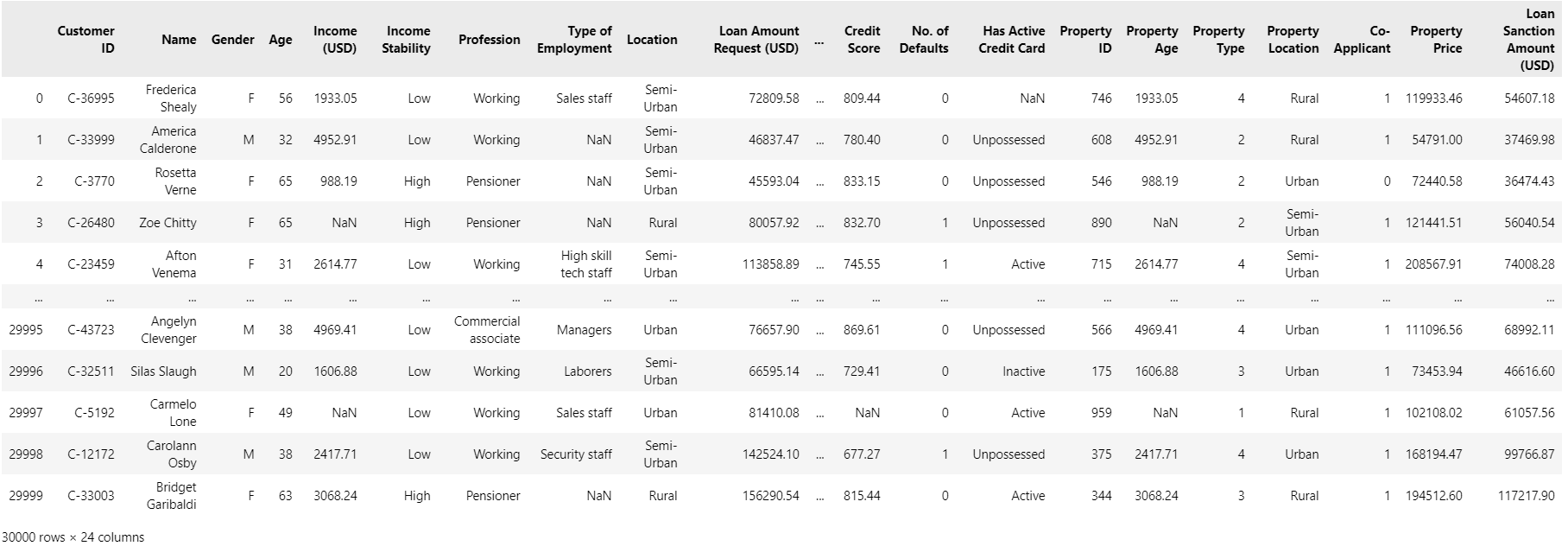
To do the EDA part, we are using **Pandas** library and to visualize the data, we are using **Matplotlib** and **Seaborn** libraries.

Here, we start by importing all the necessary libraries required to do the EDA.

We then import our dataset and store it in a variable.



If we call the variable like shown above, we get the following output.

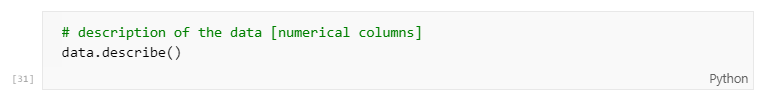


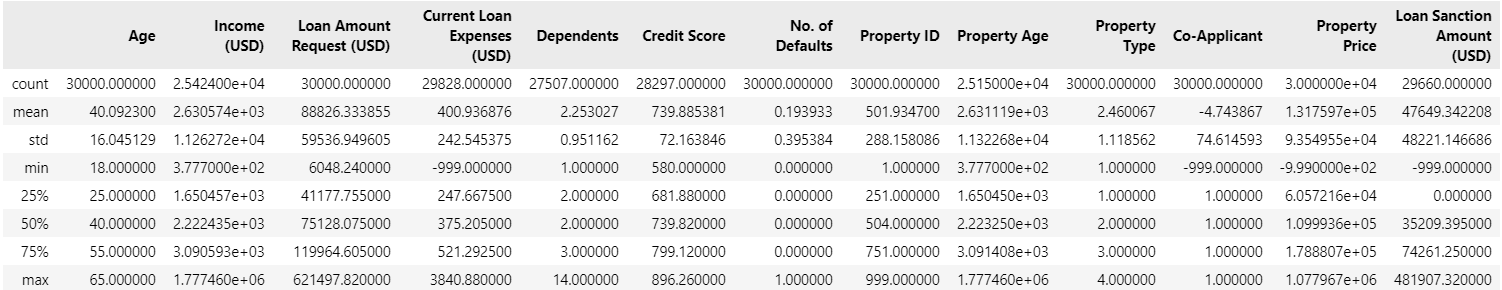
Here, we can see that the dataset consists of 30000 rows and 24 columns.

It also consists of many columns such as Customer ID, Income, Profession etc.

We will analyze this data and perform operations on the data to further explore it and clean it such that it is good enough to be used to train a machine learning model.

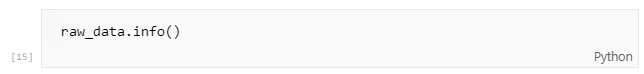
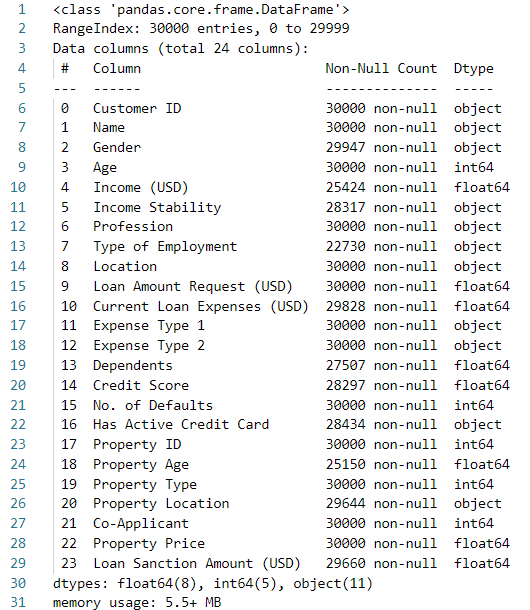
Af





Description of the data shows us the description of the numerical columns of the dataset. We can see the count, mean, 25%, max, etc.

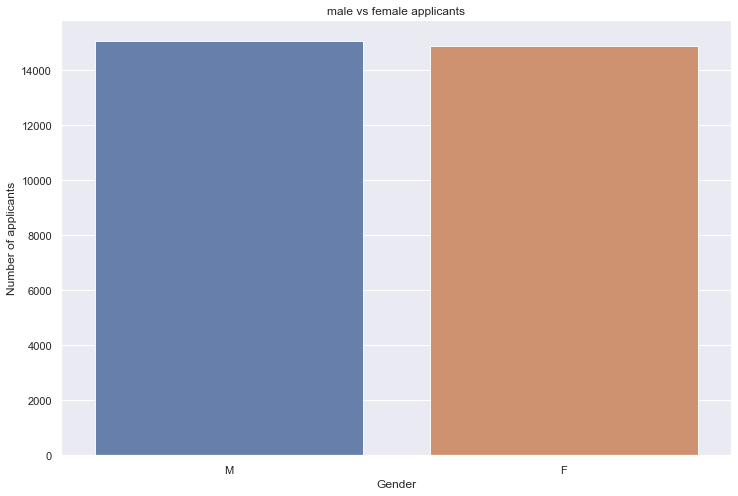
From the above result, we can conclude the following:

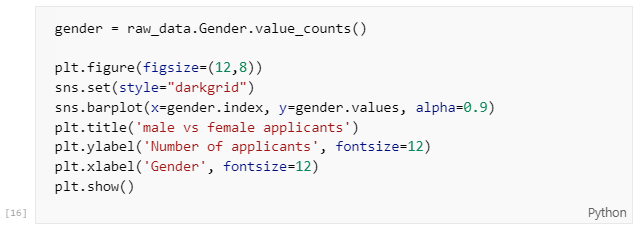
* 4 columns have a negative min value, which have to be cleaned.
* 25% of the rows in the Loan Sanction Amount (USD) column have a value of 0, which needs to be properly handled.
* Over 75% of the applicants have an age of about 55 years, 50% of them are about 40 years, and about 25% of them are about 25 years of age.
* The applicants have dependents ranging from about 1 to 14 people.
* 

If we call data.info(), we get the following result showing all the columns in our dataset, the number of non-null values, and the dtype of the columns.

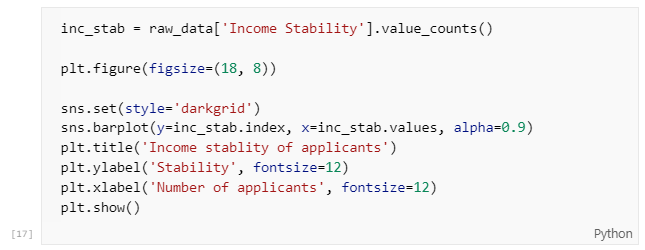
We also get a count of each dtype present in the dataset.

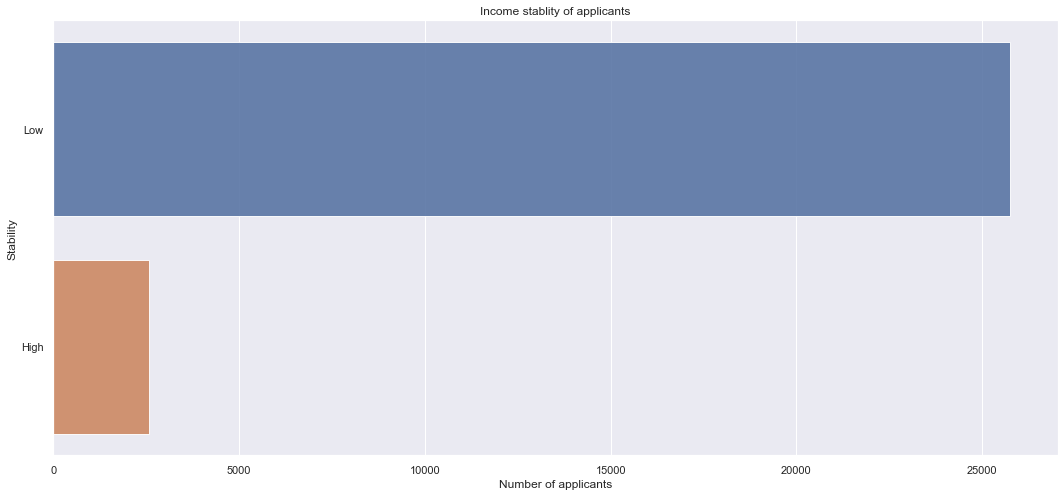
**Male vs female applicants**

****

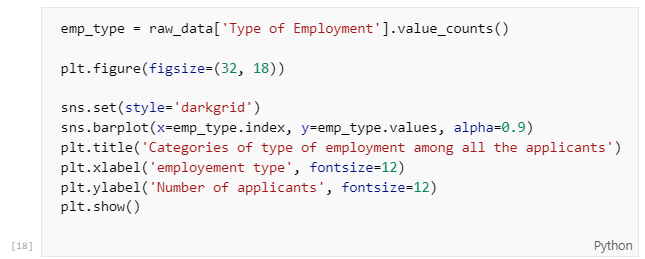
****

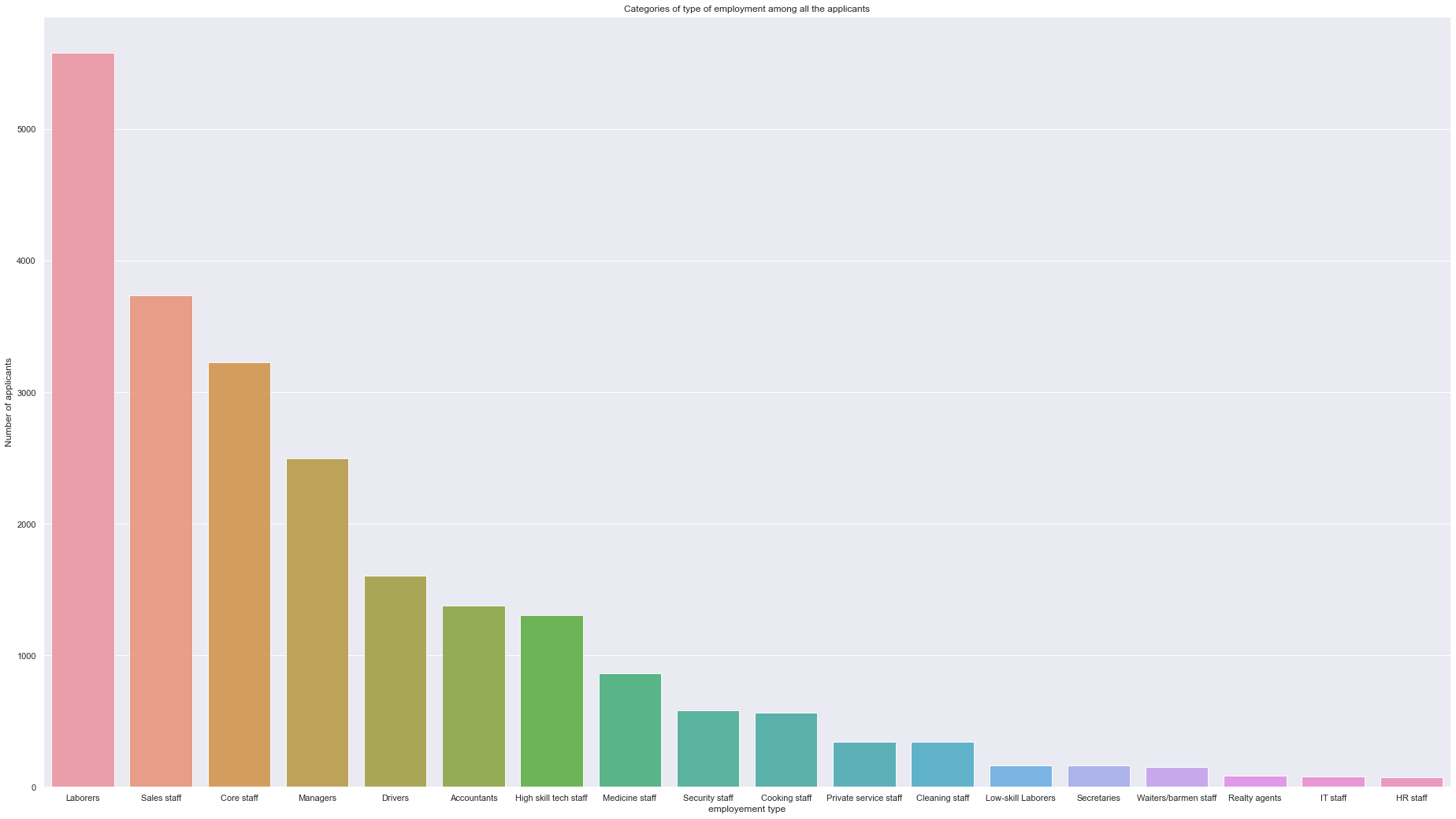
Plotting a barplot of male vs female applicants shows that male applicants constitute of just above 50% and female applicants constitute just below 50% of all the applicants.

**Income Stability**



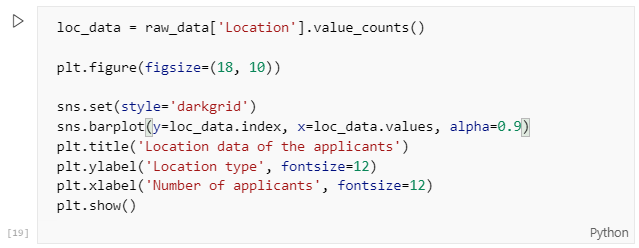
Plotting a barplot of the income stability shows that the majority of the applicants have a low income stability whereas only a handful of them have a high income stability.

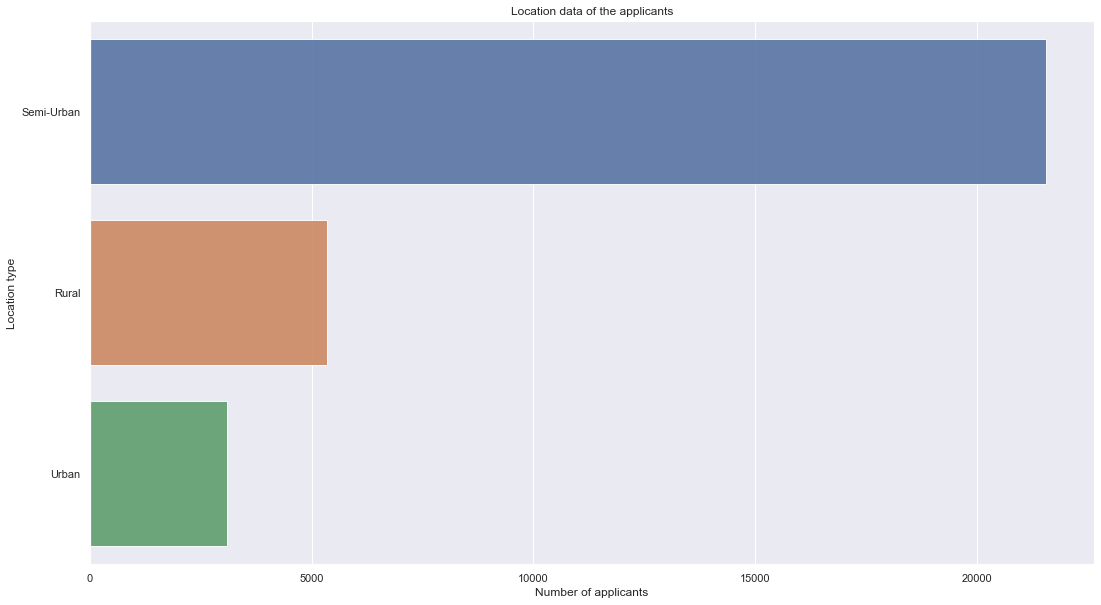
**Categories of type of employment of the applicants**



Plotting the graph of the employment type of the applicants, we can conclude that the majority of the applicants are laborers, sales staff, core staff and managers. This explains the low income stability of the applicants.

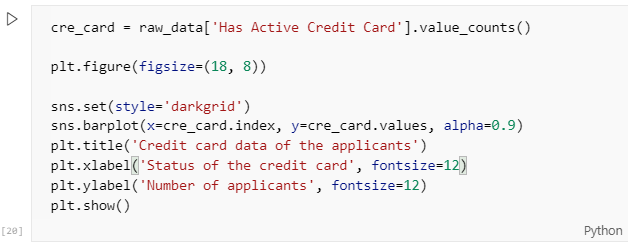
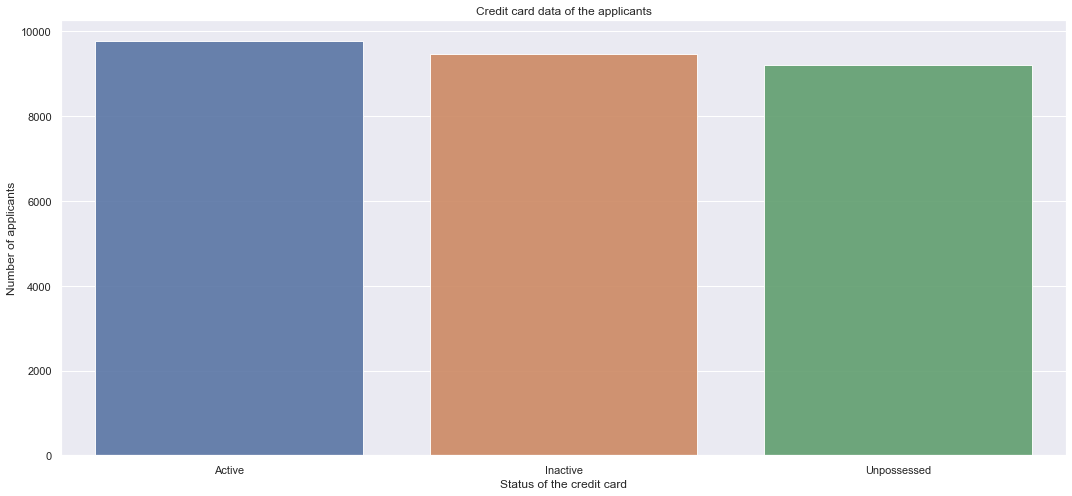
**Location data of the applicants**

****

****

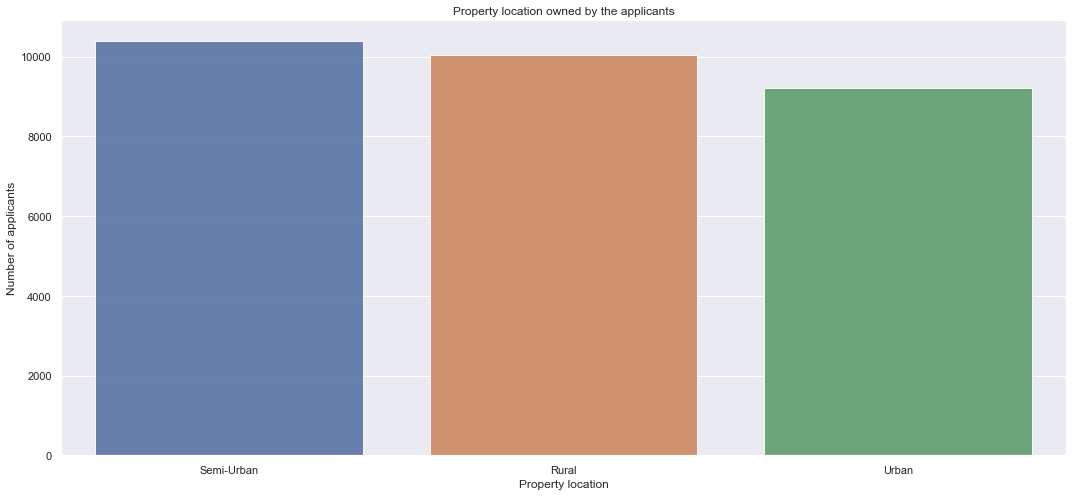
Plotting a graph of the location data given, we can see that people from a semi urban area applied the most, while people from urban areas applied the least for a loan.

**Credit card data**



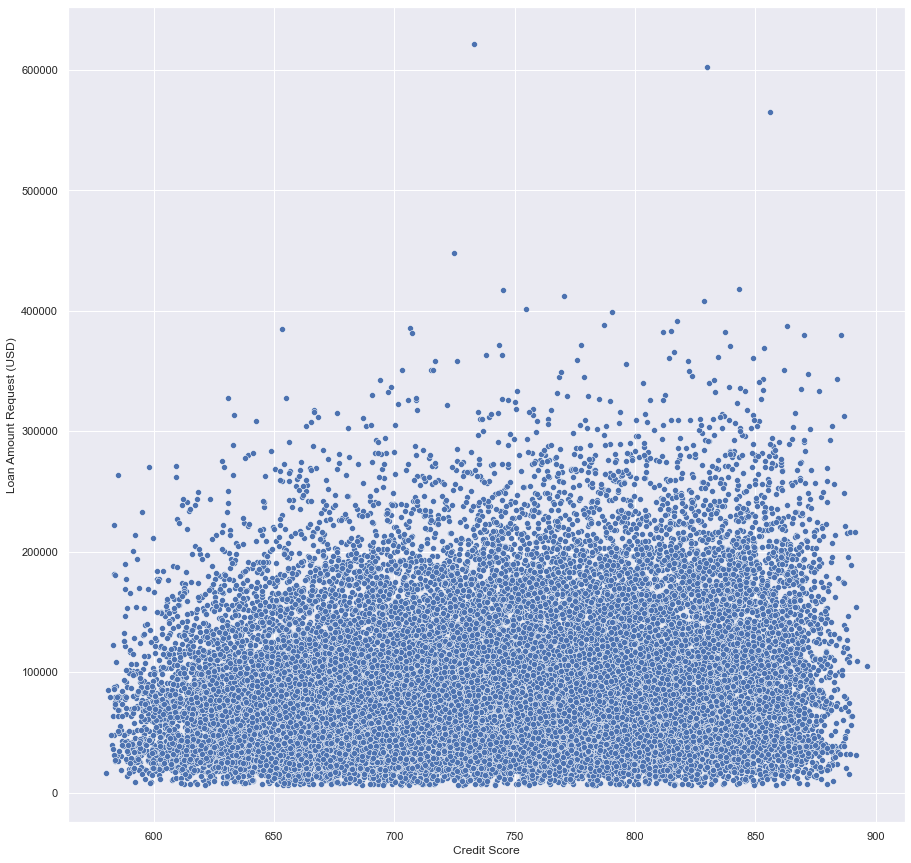
From the graph above, we can see that the number of applicants having an active credit card, the ones who have an inactive card and the ones who don’t have a credit card are almost equal, but the number of credit card holding applicants is slightly higher than the other two categories.

**Property location owned by the applicants**



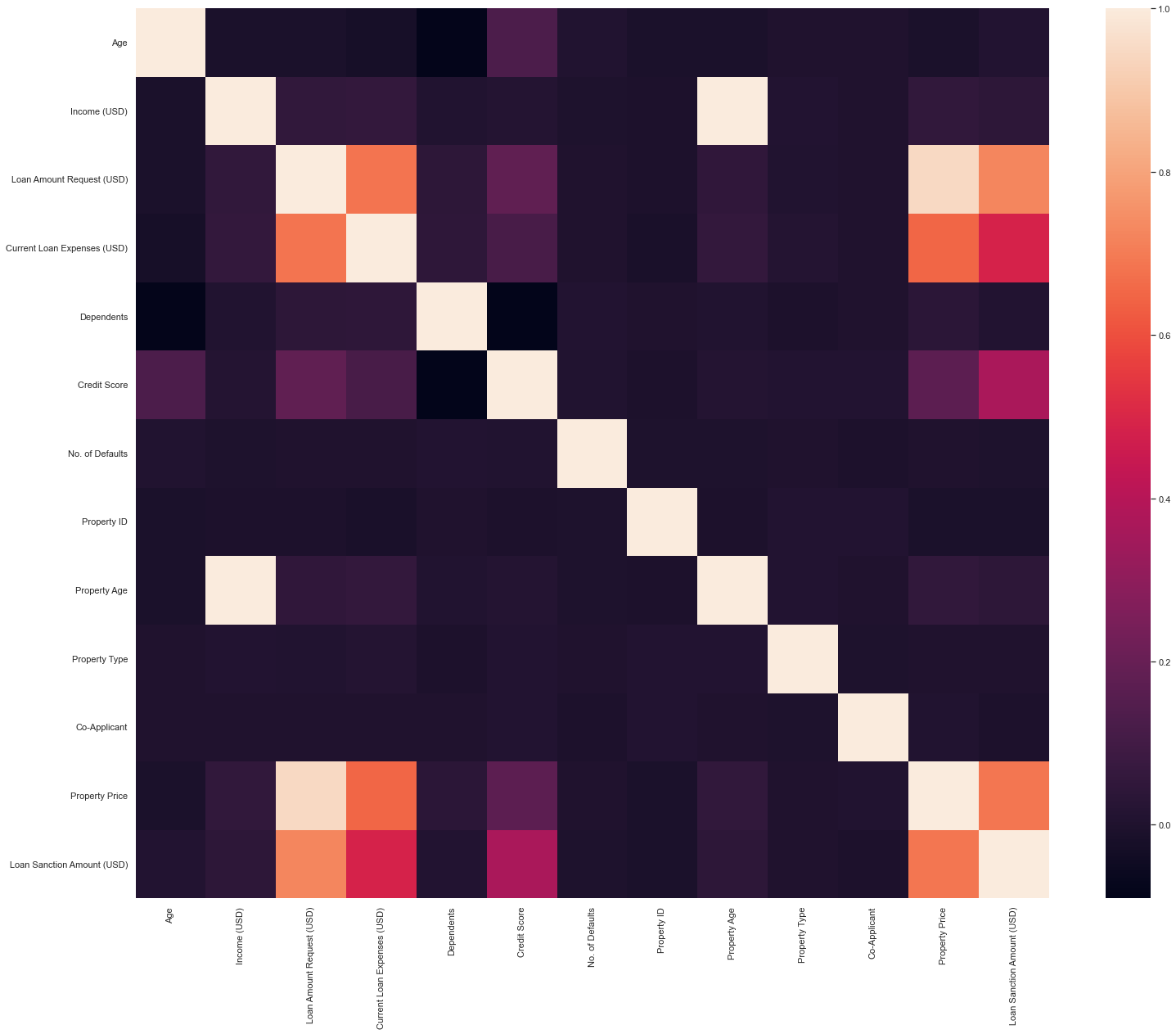
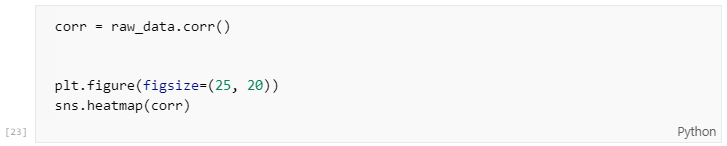
From the above plot, we can conclude that the number of applicants having a property in a semi-urban area is a little higher than the ones having a property in a rural area. The applicants having a property in an urban area are the least among the three.

**Credit score vs Loan Amount Request (USD)**



Plotting a scatterplot of the credit score vs loan amount request in USD shows us the following:

* Over 95% of all the applicants requested a loan amount less than 200000USD.
* The remaining applicants requested for more than 200000USD.
* There are a few anomalies such as one person requesting a loan of over 600000USD.

**Heatmap of all the data**

From the above heatmap, we can observe that, the factors such as Loan Amount Request, Current Loan Expenses and Property price matter the most and also Credit score has an impact on the Loan Sanction Amount.

**Preparing the machine learning model**

In this section, we will prepare the machine learning model. The steps required are:

* Data cleaning / Data preprocessing.
* Identifying the type of task and choosing the proper machine model.
* Training the model with the prepared data.
* Testing the trained model.

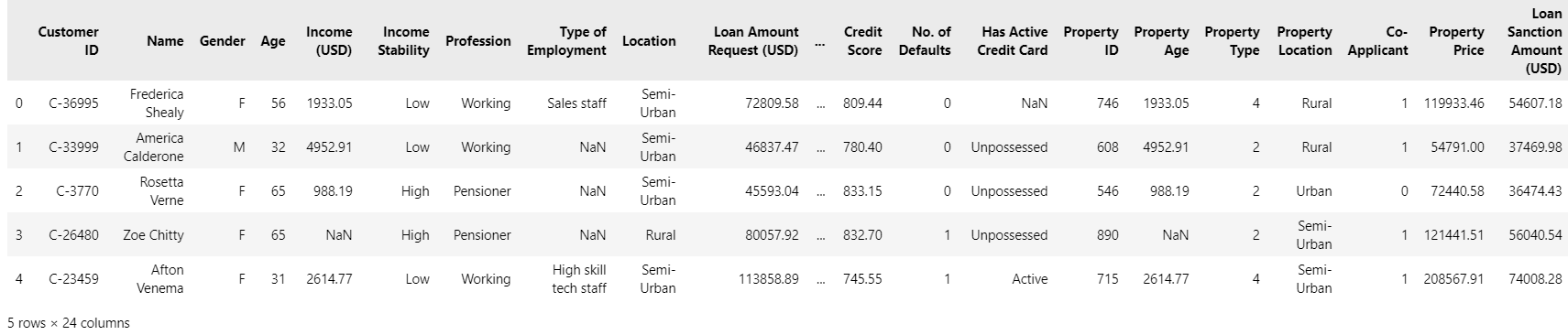
**Data cleaning/Data preprocessing**

We start by cleaning the data by the following steps:

* Identifying the data that needs to be dropped.
* Identifying and handling null values.
* Identifying and handling negative values and inconsistency in the data.
* Encoding the categorical data.

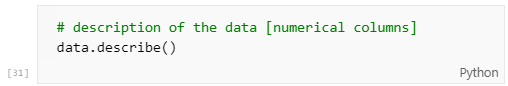
This is a very important step in the process because, without cleaning and preparing the data properly, training a machine learning model will result in the model not giving accurate predictions. The model may not be even trained in some instances because the data is of a different data type other than the one that the model accepts.

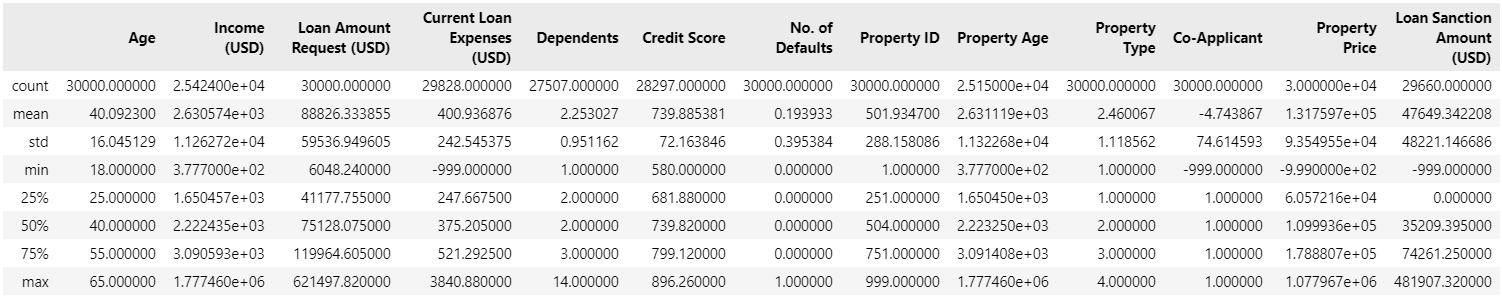




The above code displays the first 5 rows of the dataset. As we can see, there are 24 columns in the dataset, many of them are not necessary and need to be dropped and many of them are categorical and need to be encoded.

Unique data such as names, IDs etc. are not necessary to train the machine learning model and thus need to be dropped from the dataset. Categorical data such as Income Stability, Type of employment etc. has to be encoded into a numerical data type so that it can be fed into the machine learning model.

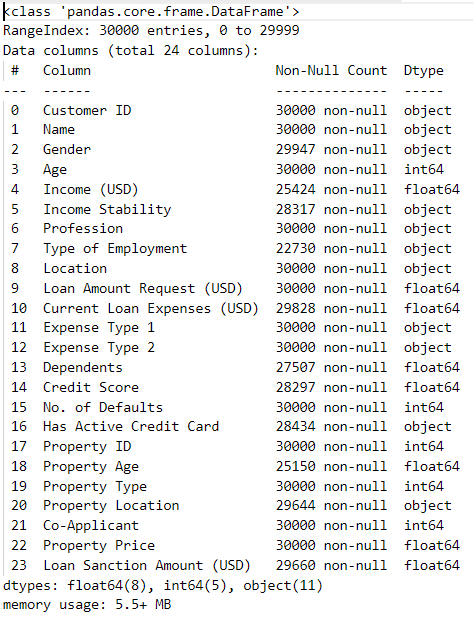




When we call data.describe(), it gives a description of all the numerical columns of the dataset. From this, we can see that the columns Current loan expenses, Co-applicant, Property price and Loan Sanction Amount have a negative value as minimum. This does not correlate to the real world where a loan sanction amount is a positive value, not a negative one. So, these columns must be handled properly.

The next step is to call data.info() to get all the columns, their data types so that we can decide which columns have unique values like IDs etc. so that we can drop them from our dataset.





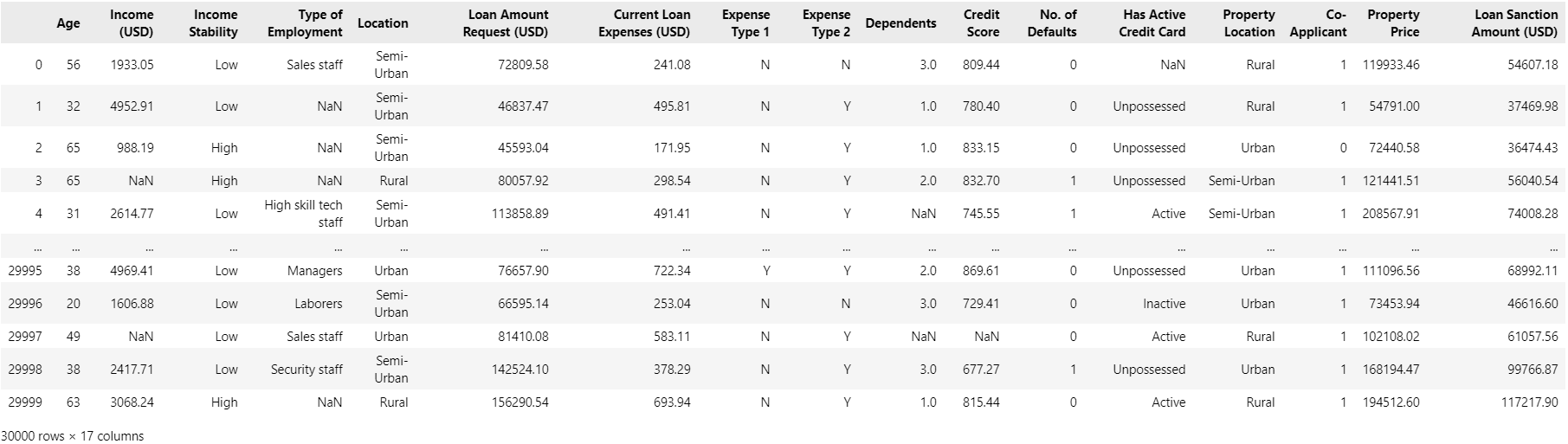
Calling data.info() shows that there are many columns consisting of unique values that need to be dropped. It also tells us that there are 8 columns of float64 data type, 5 columns of int64 data type.

Columns such as Customer ID, name, Profession, Property ID, Property type have unique values. So they need to be dropped from the dataset. We are also going to drop the Gender column from the dataset as we are trying to prepare a machine learning model that is not gender biased. Also in the EDA part, we observed that the difference between male and female applicants is miniscule.



Executing the above code drops all the columns specified in the columns list.

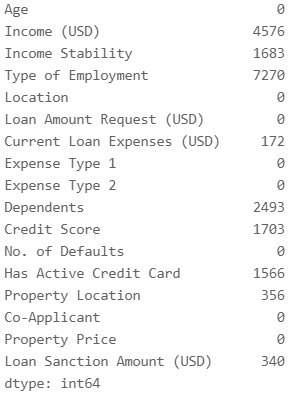
So, we dropped all the unique and unnecessary data from the dataset. inplace=True ensures that the original dataset variable is altered.



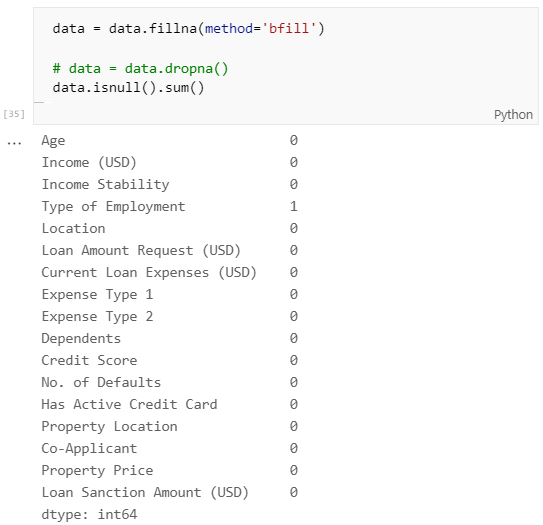
The result obtained above shows us that the specified columns have been dropped. The dataset is reduced from 24 columns to 17 columns.

**Finding and handling NaN values**

****



From the result above, we can see that there are null values in many fields and they need to be handled properly. Many of the numerical columns and categorical data have null values which means that they are not available. Proper handling of NaN/null values is the difference between an accurate model and an inaccurate one, so this step is important.

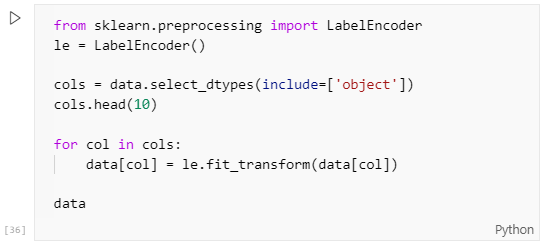


We are handling the NaN values by filling them with another value. We can go the route of dropping the rows with NaN values but that will result a huge loss of data as there are many rows with NaN values.

Here, we are fill the data by specifying the method as ‘bfill’, i.e backfill which fills the NaN value with the next valid observation.

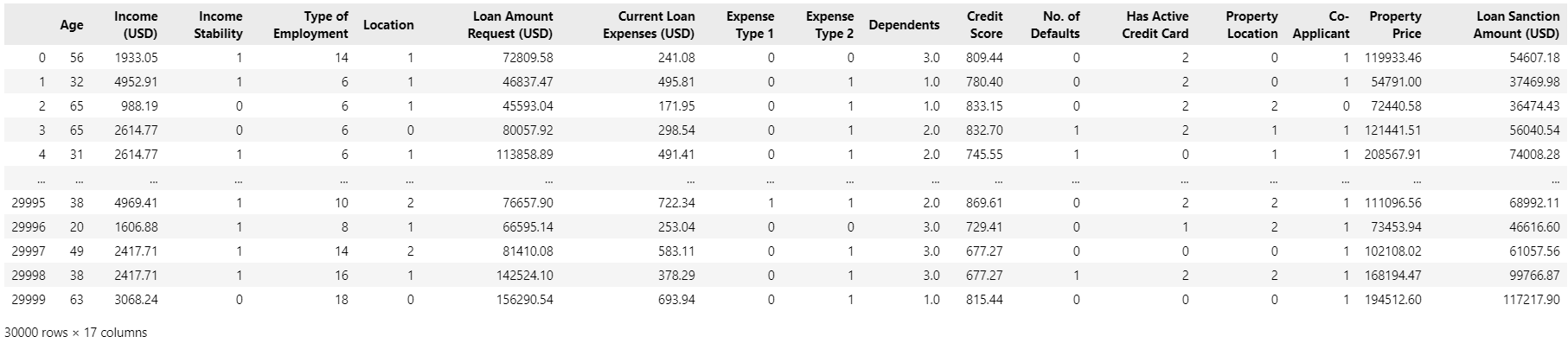
After filling all the columns with NaN values with the next valid observations, we get the above result. Notice that ‘type of employment’ column has one NaN value. This is because of the applicants who are pensioners don’t have a type of employment. This will be also encoded by the encoder that we use.

**Encoding categorical data columns**

****

Encoding is a process of converting the unique values in a column into integer data such that each value in the column corresponds to a integer.

To encode our data we use LabelEncoder from SKLearn. This will encode target labels from values ranging from 0 to n – 1 where n is the number of unique values in the column.



After encoding operation is done, the above is the result we get. We can see that all the categorical data has been encoded into an integer ranging from 0 to n-1.

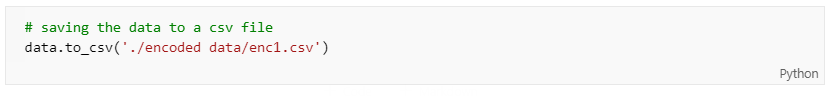
**Handling negative values**

****

We saw earlier that some columns had negative values that had to be processed. For this dataset and the given scenario, having negative values is not ideal because it does not reflect the real world where those values will never be negative. So we will handle these values by first converting all the negative values into null/NaN values and then dropping them.

After the conversion of negative values to null values has been done, we can see that there are a lot of negative values. Especially we need to look at Loan Sanction Amount column which has 8302 rows with negative values. This data is simply incorrect as there is never a negative loan sanctioned in the real world.



After the conversion of negative values into null values, we will drop all the null values by executing the above code. This will get rid of all the rows with negative values which were converted into null values.

After all the data cleaning and encoding operation has been done, we will save the processed data into a new csv file. We will call the file as ‘enc1.csv’.

**Training the machine learning models**

After the data has been processed, we can use that data to train the machine learning models. The steps taken while training a model are:

* Identifying the type of task given i.e classification or regression.
* Choosing the appropriate models.
* Identifying the dependent and independent variables.
* Splitting the data into training and testing data.
* Training the model.
* Using the test data to predict the output.
* Evaluating the trained model’s accuracy by comparing with the test output and the predicted output.

**Identifying the task given**

Here, our task is to predict the amount of loan to be sanctioned to an applicant by the bank. This falls under the category of regression as we are predicting the actual amount to be sanctioned. We can use regression models for our application.

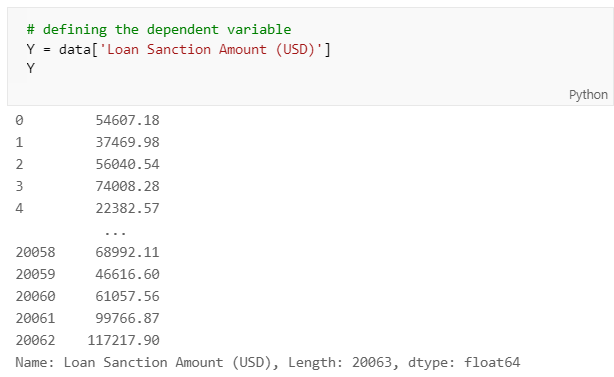
**Choosing the appropriate models**

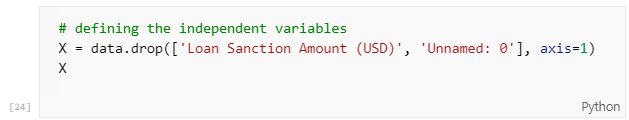
Choosing the correct model is a crucial step in the model training and testing because it will decide the accuracy of the trained model. Not all models are the same and each model is built for a specific set of tasks. So, we need to choose the appropriate models for our task to maximize the accuracy of our trained models.

For this application, we will choose to train 3 regression models available in SKLearn:

* Linear regression: Linear Regression fits a linear model with coefficients w = (w1, …, wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.
* Bayesian Ridge: Bayesian regression allows a natural mechanism to survive insufficient data or poorly distributed data by formulating linear regression using probability distributors rather than point estimates. The output or response ‘y’ is assumed to drawn from a probability distribution rather than estimated as a single value.
* Decision Tree Regressor: Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.

**Identifying the dependent and independent variables**

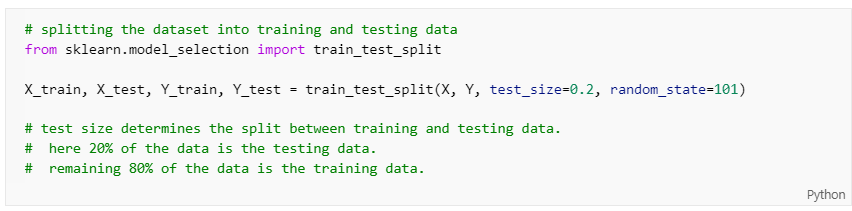
We need to identify the dependent variables and independent variables in our data to train the model. Here, the dependent variable is ‘Loan Sanction Amount (USD)’. We can isolate it and store it in a variable.

 The independent variables are all the columns except the dependent variable in this case. So, we isolate the dependent variables and store them in a variable.

**Splitting the data into training and testing**

After we decide which models to train, we need to split the data into training and testing data so that most of the data is used to train the model and the remaining data can be used to test the model for it’s accuracy.

To split the data, we will use ‘train\_test\_split’ from sklearn’s model selection. Then we pass the dependent and independent variables to the train\_test\_split method along with a test size and random\_state. Test\_size determines the percentage of the data to be used as test data. Random\_state controls the shuffling applied to the data before applying the split.



Train\_test\_split returns 4 values: X\_train, X\_test, Y\_train, Y\_test. The \*\_train data is used to train the model and the \*\_test data is used to test the model after training it.

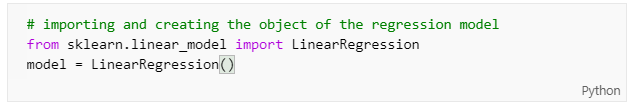
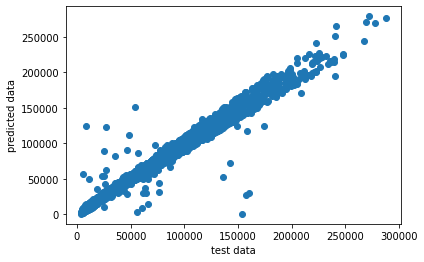
**Training, testing and evaluating the models**

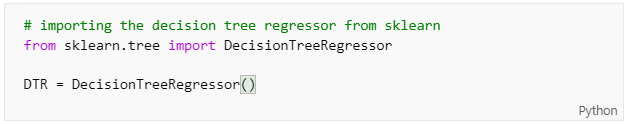
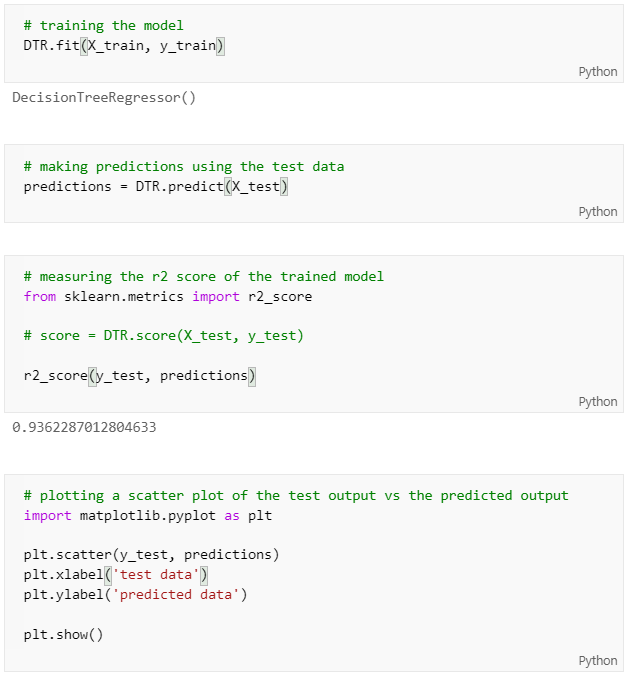
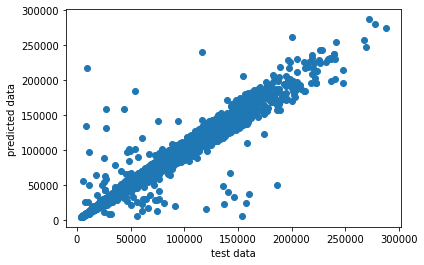
After the train and test data is ready, we can move on to training and testing the models. The process of training and testing is as follows:

* The model is imported from the library and instantiated.
* Then it is trained using the training data.
* The trained model is used to predict the output using the test input.
* Then the model is evaluated by analyzing the r2 score by using the predicted output and the test output.

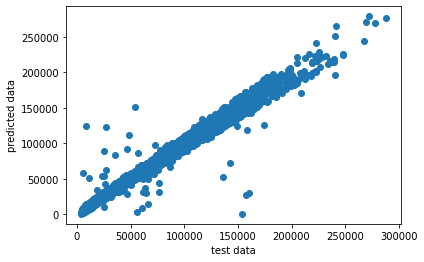
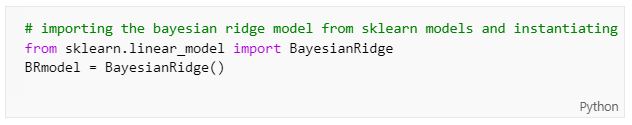
All the 3 selected models are trained, tested and evaluated according the steps mentioned above.

**Model 1: Linear Regression**

Linear regression model is successfully trained, tested and evaluated. The r2 score of the trained model is 0.9693971794711593. The scatter plot of the test output vs predicted output can be seen above.

**Model 2: Decision Tree Regressor**

Decision Tree Regressor model is successfully trained, tested and evaluated. The r2 score of the trained model is 0.9362287012804633. The scatter plot of the test output vs predicted output can be seen above.

**Model 3: Bayesian Ridge**

Bayesian Ridge model is successfully trained, tested and evaluated. The r2 score of the trained model is 0.9694010419387209. The scatter plot of the test output vs predicted output can be seen above.

**Machine Learning model chart**

|  |  |  |
| --- | --- | --- |
| **Sl no.** | **Algorithm name** | **r2 score** |
| 1 | Bayesian Ridge | 0.9694010419387209 |
| 2 | Linear Regression | 0.9693971794711593 |
| 3 | Decision Tree Regressor | 0.9362287012804633 |

**Conclusion**

**Bibliography**

Loan sanction amount prediction using Machine Learning

[Document subtitle]

[School]

[Course title]