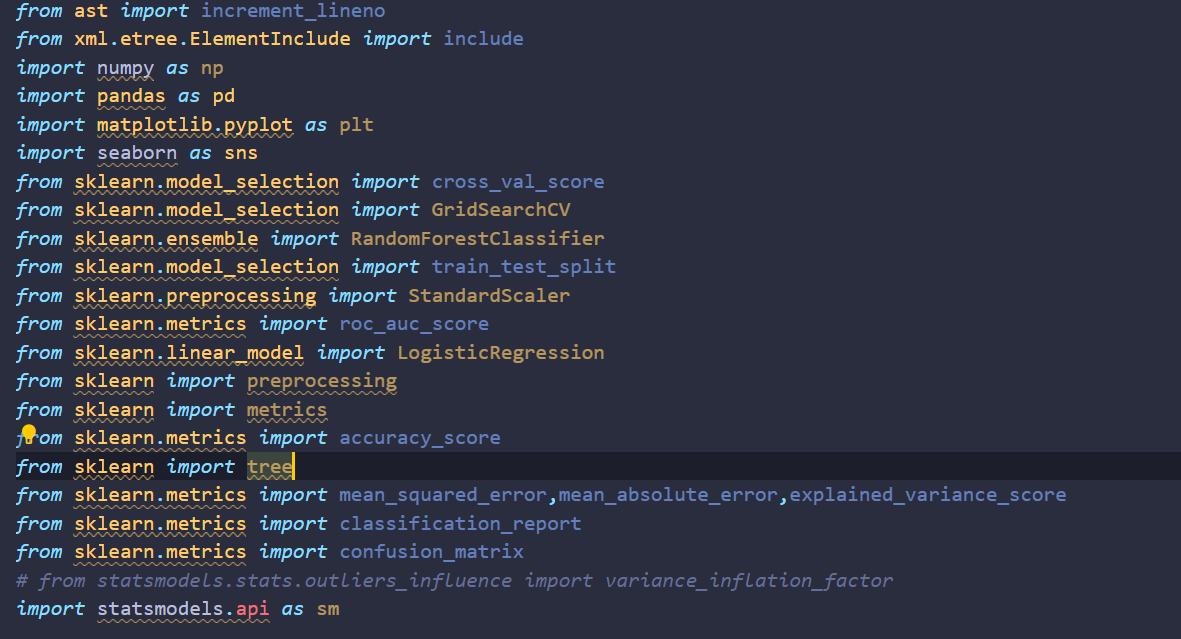
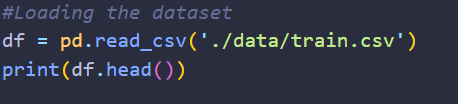
**Installing & Importing Libraries in Python Script**

1. Install and import all the required libraries in Python



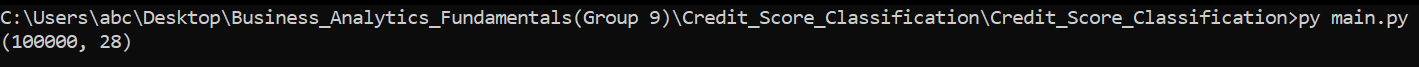
1. Loading the dataset



1. Check the data quality

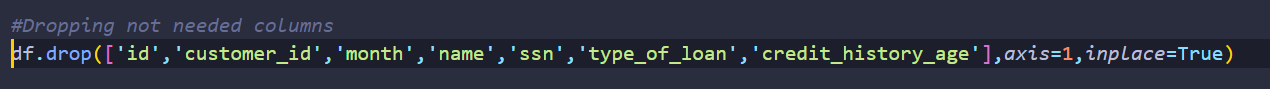


Output:



It says we have 100,000 rows and 28 columns in our dataset.

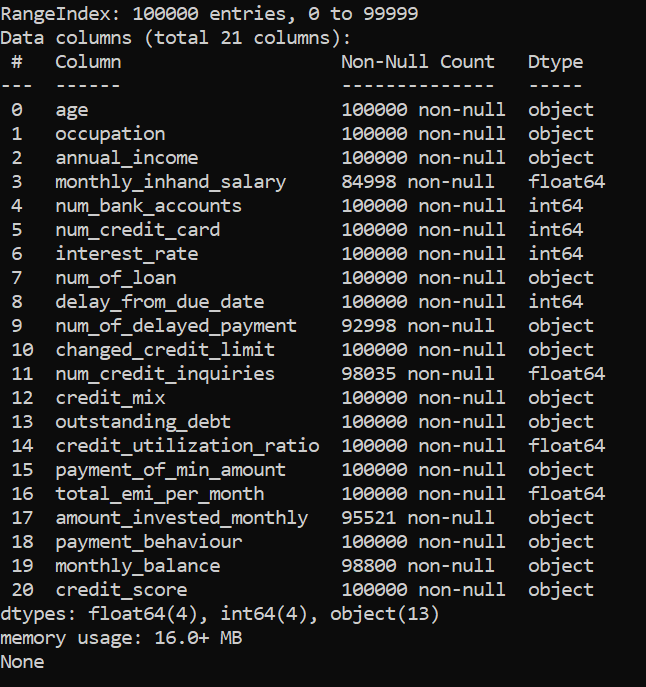
1. We would then drop all the columns that are not useful for analysis



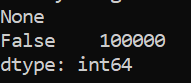
1. A useful command to get some details on the features



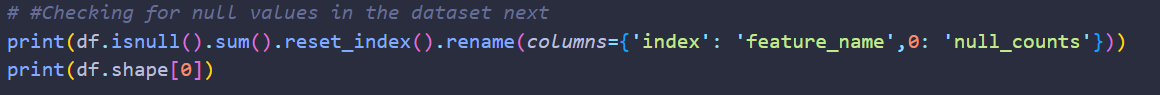
Output:



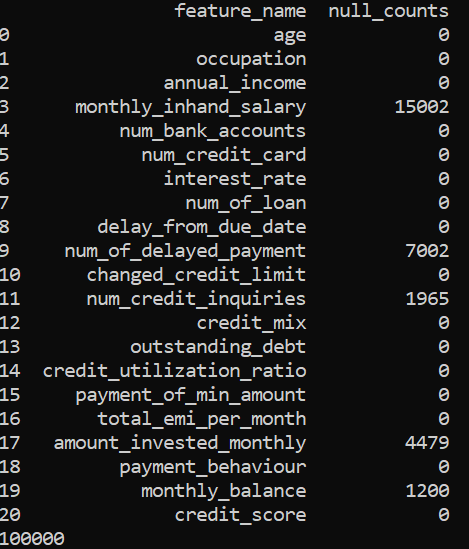
1. Check for duplicate Values



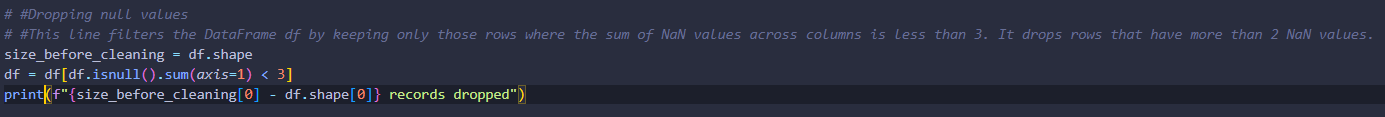
1. Check for null values in the df



Output:

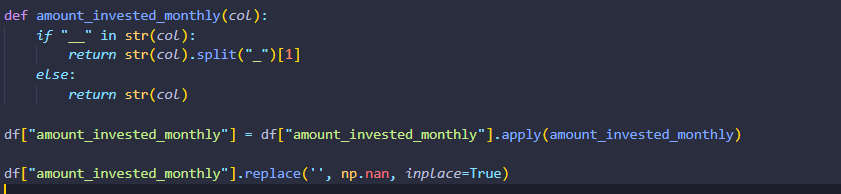


1. Check across columns in a particular row if there are NaN values and see if that count is 3 or more than 3.We will remove those rows from our dataset.

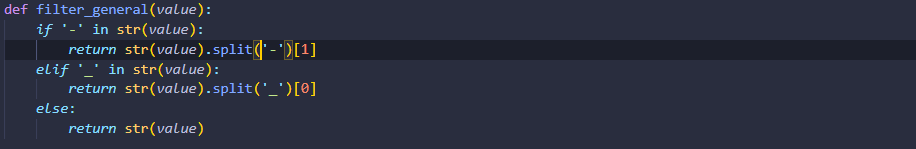


**Data Cleaning**

1. We will create new functions that will help us clean features, some of the functions would be feature specific like the one below:

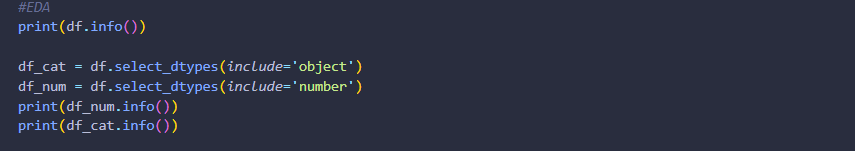


1. Others would be common for some of the features:



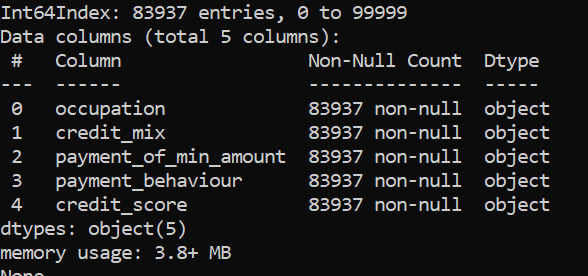
**EDA**

1. Its time for Exploratory Data Analysis now. We will first split the dataframe into 2 separate dataframes.One for numerical data and the other one for categorical data.

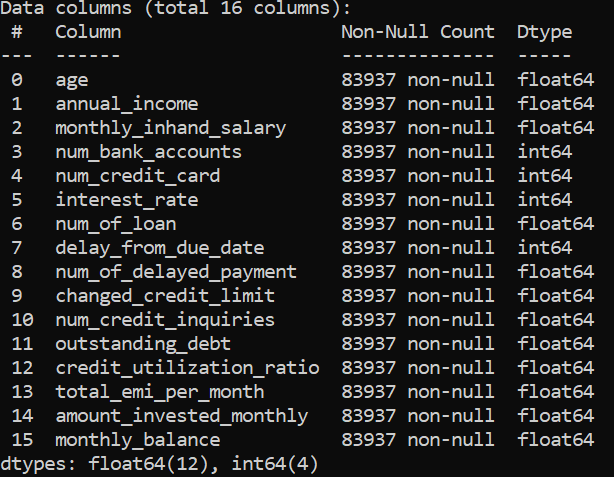


Output:

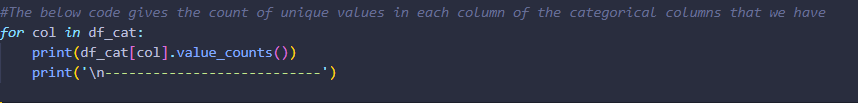
1. Categorical Data



1. Numerical Data

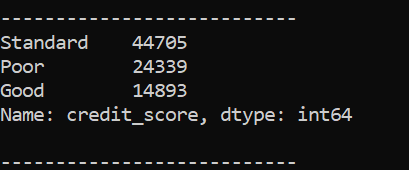


1. For categorical data, we will count all the unique values and plot it with matplotlib library.

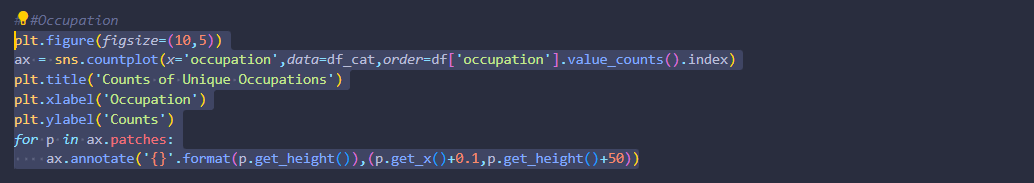


Output:

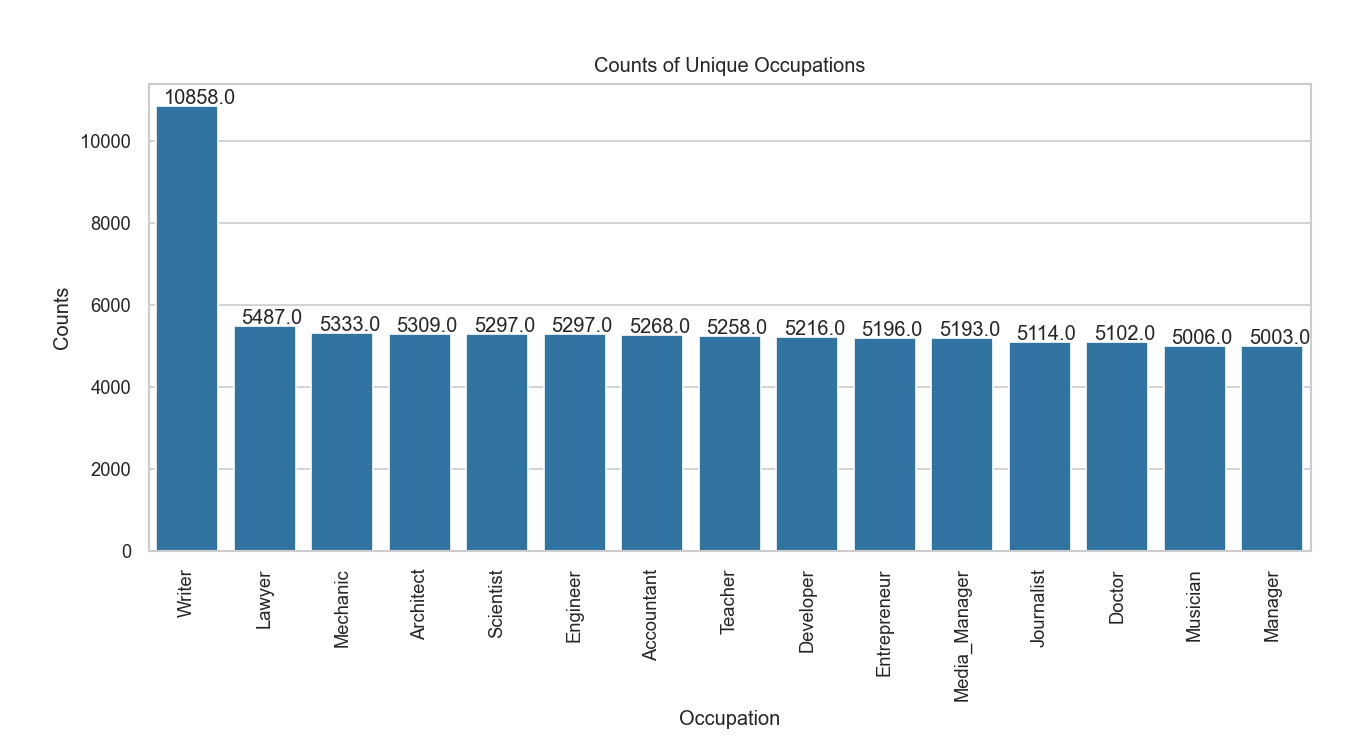
We get 5 more tables similar to the one below as the output. The graphs plotted with matplotlib do a better job of analysing the results.



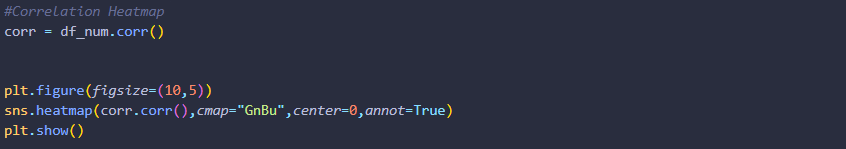
1. We will use the categorical data with the below code. We will do this for all the 5 features which are categorical to analyse the data.



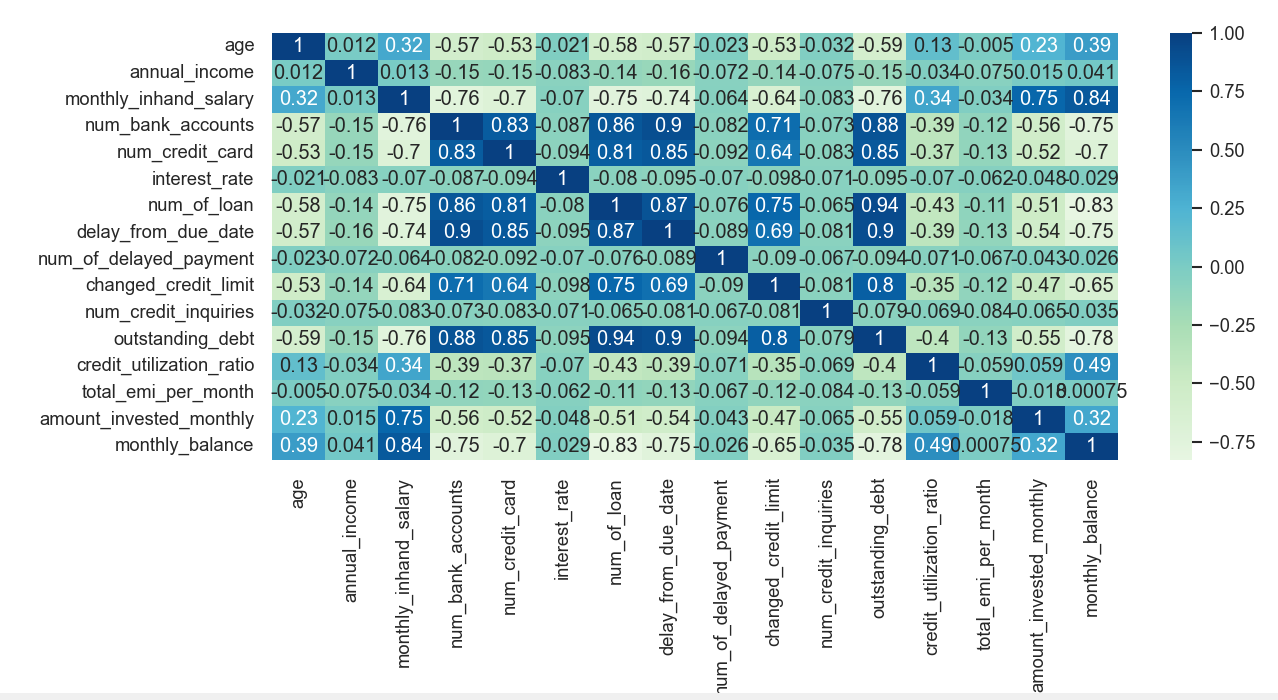
Output:



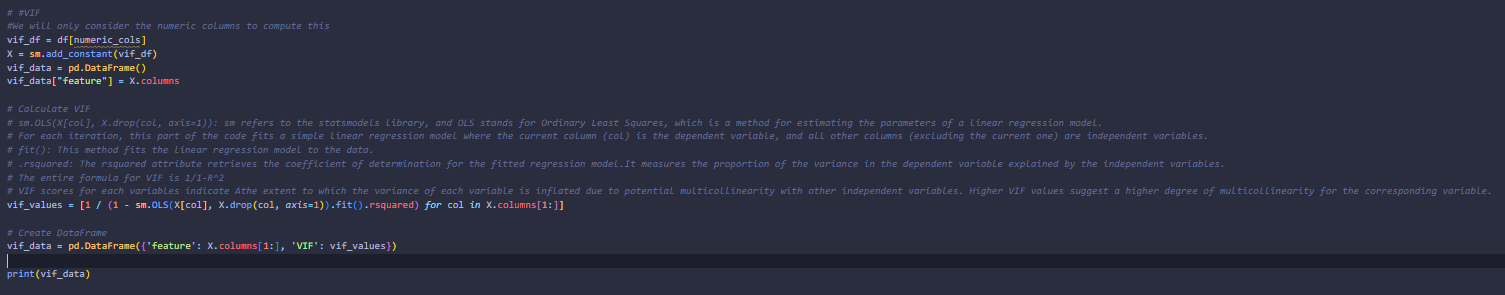
1. We will then plot the correlation for the numerical data with the seaborn library using a heatmap.



Output:

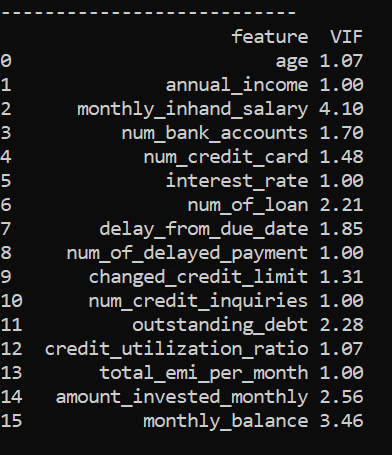


1. Calculating the VIF to negate multicollinearity issues. We will now remove the features with ViF greater than 2 as values above 2 are seen as a signal that there may be multicollinearity issues. The formula for VIF calculation is 1/1-R^2.We have kept 2 as the threshold for our data. Data analysts/scientists/researchers select this threshold based on the data available for analysis. We get rid of the features that have VIF > 2.





Output:



**Data Preparation**

1. We would drop the target variable i.e. Credit score feature and assign the remaining features of our df to variable ‘X’. We store the values of the target variable in ‘y’. We will then split the data into test and train data, assigning 20% of total data to test and the remaining 80% as train data.

X-train : Train data from the dataframe with all the features

X-test : Test data from the dataframe with all the features

Y\_train: Target variable data from the training data which is 80%

Y\_test: Target variable data from the training data which is 20%

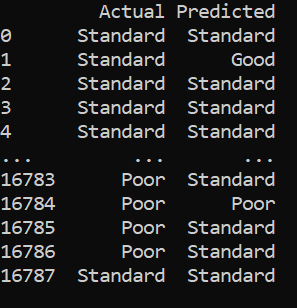
1. We will assign the numeric features to numeric\_cols\_train variable.The categorical data in the test and train data should be one-hot encoded i.e. converted to binary values of 1 or 0.
2. We than use standardscaler from scikitlearn library. The fit\_transform method is called on the scaler, which fits the scaler to the training data (X\_train\_encoded[numeric\_cols]) and then transforms it.
3. This means it calculates the mean and standard deviation of the numeric columns in X\_train\_encoded and applies the transformation to standardize these columns.
4. The result is assigned back to the corresponding columns in X\_train\_encoded.



**Model Selection & Results**

We have used Logistic Regression to classify the credit score of the borrowers.

1. We fit the regression line with the training data first.
2. Then we predict credit score using the predict function in LogisticRegression class.
3. The results\_df can be used to draw a comparison between the actual classification data(y\_test) and predicted data that we derived with the .predict function(y\_pred)



1. We can calculate the accuracy of the model's predictions by comparing the predicted labels (y\_pred) with the actual labels (y\_test)





1. We generate a text report with various classification metrics, such as precision, recall, and F1-score, for each class in your classification problem. It provides a more detailed view of the model's performance than accuracy alone.

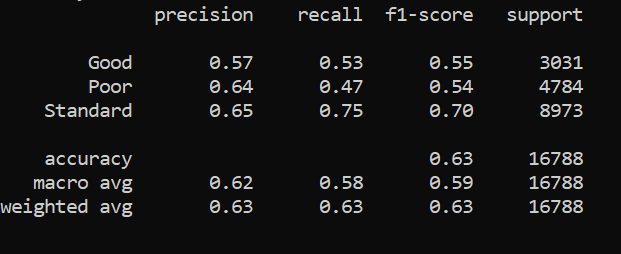
**Precision** - Precision checks how many predicted positives are actually positive?

**Recall** - Recall checks how many actual positives did we capture in our predictions

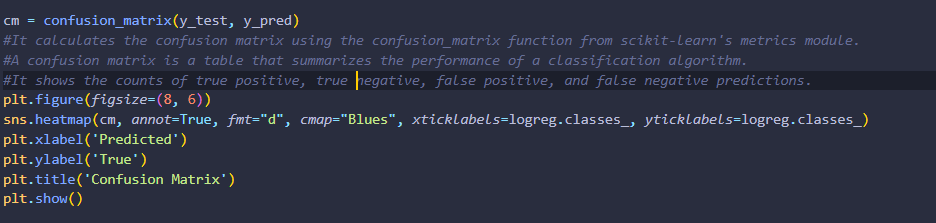
**F1-score** - 2 \* (Precision \* Recall) / (Precision + Recall).It is the harmonic mean of recall and precision.F1-score considers both false positives and false negatives, providing a single score that balances precision and recall

**Support**- Support is the number of actual occurrences of the class in the specified dataset





1. Last step would be to visualise the results of the model.



Output:

