Credit Punctuation Model

Developed by: Emiliano Mena González and Jorge Alberto Hernández Hernández

ID: 728407 and 730342

University: Instituto Tecnológico y de Estudios Superiores de Occidente (ITESO)

Degree: Finanacial Engineer

Semester: 10th

Date: 20/02/2024

ITESO, Universidad Jesuita de Guadalajara

In [ ]:
```python
# Libraries and dependencies
import main as m
import visualizations as v
```

Introduction

Main objective

The main objective of this project is to create a Personal Credit Punctuation Model that classify every new record on one of three possible scores(Poor, Standard or Good).

Data

The data we are taking to do the model is the dataset contained in the csv file called "train.csv", this dataset has 28 columns (variables) and 100,000 rows (records). The last column of the dataset is the Credit_Score variable that will be the reference point to see the accuracy of the model. Below we have the first 5 record of the dataset to see what variables we have, what type of value they are.

In [ ]:
```python
m.df.head(5)
```

Out[ ]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	...	Credit_Mix	Outstanding_Debt	Credit_Utilization_Ratio	Credit_Hist
0	0x1602	CUS_0xd40	January	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	...	_	809.98	26.822620	22 Yea
1	0x1603	CUS_0xd40	February	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	NaN	3	...	Good	809.98	31.944960	
2	0x1604	CUS_0xd40	March	Aaron Maashoh	-500	821-00-0265	Scientist	19114.12	NaN	3	...	Good	809.98	28.609352	22 Yea
3	0x1605	CUS_0xd40	April	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	NaN	3	...	Good	809.98	31.377862	22 Yea
4	0x1606	CUS_0xd40	May	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	...	Good	809.98	24.797347	22 Yea

5 rows × 28 columns

To see more clearly the type of data we will work with we have a little dataset that has some valious information to know what proccesses we will need to do in order to clean the data. The info we have is first of all the type of data, if they are null values, the present values and the unique values.

In [ ]:
```python
m.data_info
```

Out[ ]:

Variable_name	Type	NA_values	Present_values	Unique_values
ID	object	0	100000	100000
Customer_ID	object	0	100000	12500
Month	object	0	100000	8
Name	object	9985	90015	10139
Age	object	0	100000	1788
SSN	object	0	100000	12501
Occupation	object	0	100000	16
Annual_Income	object	0	100000	18940
Monthly_Inhand_Salary	float64	15002	84998	13235
Num_Bank_Accounts	int64	0	100000	943
Num_Credit_Card	int64	0	100000	1179
Interest_Rate	int64	0	100000	1750
Num_of_Loan	object	0	100000	434
Type_of_Loan	object	11408	88592	6260
Delay_from_due_date	int64	0	100000	73
Num_of_Delayed_Payment	object	7002	92998	749
Changed_Credit_Limit	object	0	100000	4384
Num_Credit_Inquiries	float64	1965	98035	1223
Credit_Mix	object	0	100000	4
Outstanding_Debt	object	0	100000	13178
Credit_Utilization_Ratio	float64	0	100000	100000
Credit_History_Age	object	9030	90970	404
Payment_of_Min_Amount	object	0	100000	3
Total_EMI_per_month	float64	0	100000	14950
Amount_invested_monthly	object	4479	95521	91049
Payment_Behaviour	object	0	100000	7
Monthly_Balance	object	1200	98800	98792
Credit_Score	object	0	100000	3

Data cleaning

Delete the data it won´t be used

To start with the data cleaning, first we delete the data that won´t be used on the creation of the punctuation model. The data deleted is: - ID: It is only a descriptive variable of the person who is being punctuated. - Month: The model is the same in every moment of the year so it doesn´t matter the month of the record evaluated. - Age: The age doesn´t has a correlation with the final score. - SSN: Is another descriptive variable of the person. - Occupation: It doesn´t matter on what the person evaluated works. - Num_Bank_Accounts: This info is not necessary considering the dataset has already a Credit_Mix variable. - Num_Credit_Card: This info is not necessary considering the dataset has already a Credit_Mix variable. - Num_of_Loan: This info is not necessary considering the dataset has already a Credit_Mix variable. - Type_of_Loan: This info is not necessary considering the dataset has already a Credit_Mix variable. - Payment_of_Min_Amount: There are more relevant variables of payment. - Payment_Behaviour: There are more relevant variables of payment.

Null values

After deleting the variables that doesn´t have relevance on the model, the next step is to fill the null values. After seeing the variables that have null values, we separate them in two groups. The first group are the variables that can be filled with the values of other records from the same person, to do this we consider the Customer_ID to assign the same values. The second group are the variables that need to be filled with 0, this is because are variables that change on every record and are considered as 0. Below we can see a table that shows which variables are going on both fill methods.

Filled with customer info	Filled with 0
Name	Num_of_Delayed_Payment
Annual_Income	Num_Credit_Inquiries
Monthly_Inhand_Salary	Amount_invested_monthly
Interest_Rate	Monthly_Balance
Credit_Mix	
Credit_History_Age	

Numeric values

Now that we have a dataset with all the present values we are only missing to assign the correct format to every variable. The data has multiple variables that are numbers but of string type, to be able to be used the number variables we need to changed them to a numerical type, in this case float. This variables are: - Annual_Income - Num_of_Delayed_Payment - Changed_Credit_Limit - Outstanding_Debt - Amount_invested_monthly - Monthly_Balance

Change age format

In this part we need to change the Credit_History_Age variable format from "NN Years and NN Months" to only the number of months. Now we can see the data cleaned and ready to be used on the model creation.

Outliers

It is necessary to see if the selected variables have any outliers that can difficult the model. Down we have a table with the quartiles, maximum and minimum of the variables to see if there are any outliers.

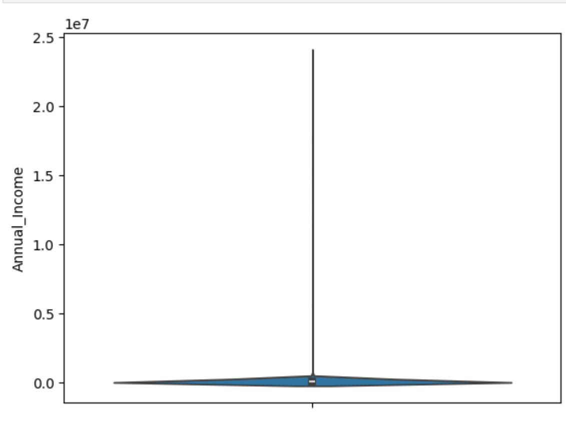
In [ ]:
```python
m.data_stadistics
```

Out[ ]:

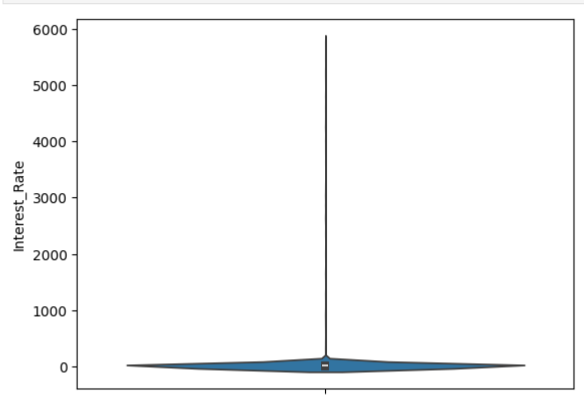
	Annual_Income	Monthly_Inhand_Salary	Interest_Rate	Delay_from_due_date	Num_of_Delayed_Payment	Changed_Credit_Limit	Num_Credit_Inquiries	Outstanding_Debt	Credit_Utilization_Ratio	C
count	1.000000e+05	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	
mean	1.616206e+05	4188.592303	73.213360	21.068780	28.779410	10.246841	27.208880	1426.220376	32.285173	
std	1.297796e+06	3180.036303	468.665823	14.860104	218.114813	6.768353	191.308723	1155.129026	5.116875	
min	7.005930e+03	303.645417	1.000000	-5.000000	0.000000	0.000000	0.000000	0.230000	20.000000	
25%	1.945333e+04	1624.937917	8.000000	10.000000	8.000000	4.990000	3.000000	566.072500	28.052567	
50%	3.757238e+04	3087.595000	14.000000	18.000000	13.000000	9.250000	5.000000	1166.155000	32.305784	
75%	7.269021e+04	5947.364167	20.000000	28.000000	18.000000	14.660000	9.000000	1945.962500	36.496663	
max	2.383470e+07	15204.633333	5789.000000	67.000000	4397.000000	36.970000	2597.000000	4998.070000	50.000000	

Some possible variables with outliers are: - Annual_Income - Interest_Rate - Num_of_Delayed_Payment - Num_Credit_Inquiries - Total_EMI_per_month - Amount_invested_monthly - Monthly_Balance But to be sure we have the violinplots of this variables to see it more clearly.

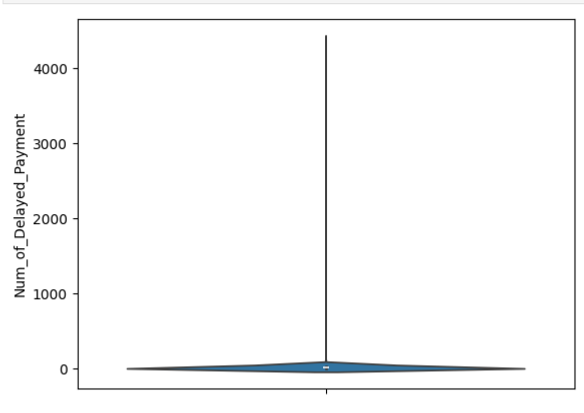
In [ ]:
```python
v.violinplots(m.df2['Annual_Income'])
```



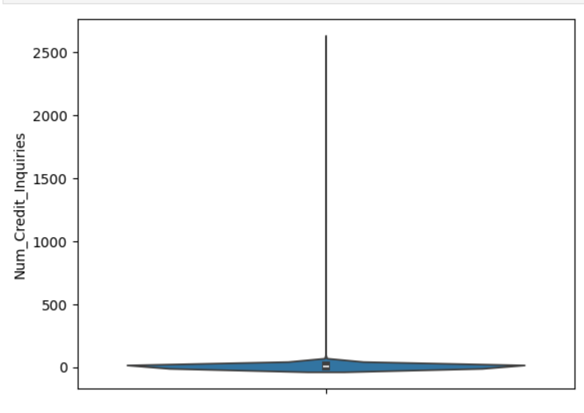
In [ ]:
```python
v.violinplots(m.df2['Interest_Rate'])
```



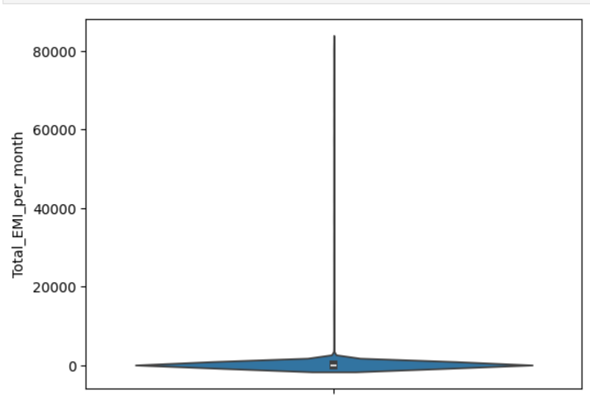
In [ ]:
```python
v.violinplots(m.df2['Num_of_Delayed_Payment'])
```



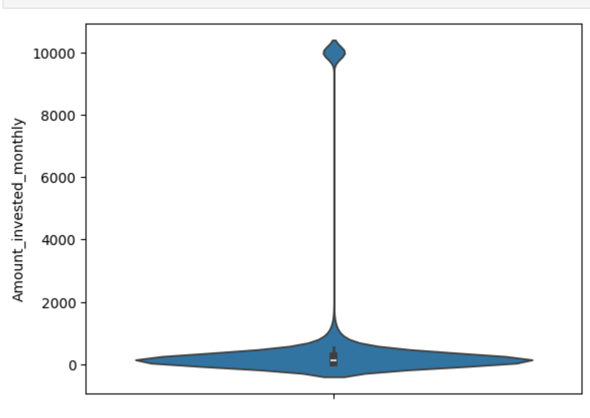
In [ ]:
```python
v.violinplots(m.df2['Num_Credit_Inquiries'])
```



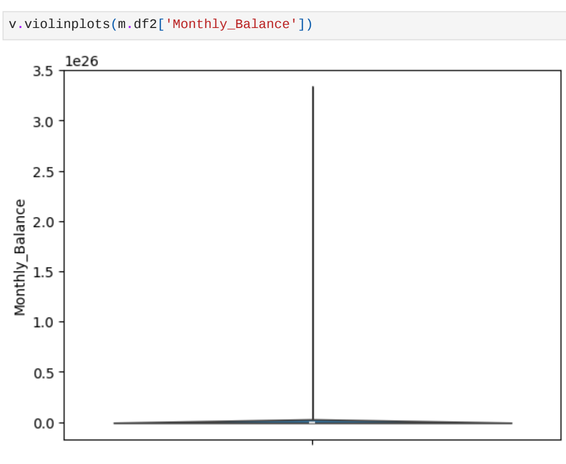
In [ ]:
```python
v.violinplots(m.df2['Total_EMI_per_month'])
```



In [ ]:
```python
v.violinplots(m.df2['Amount_invested_monthly'])
```



In [ ]:
```python
v.violinplots(m.df2['Monthly_Balance'])
```



As we expected, the previous variables contain visible outliers. We replaced this outliers with the value of the same person before (take the last value of the same Customer_ID). We finally have our cleaned data available to be used in the model creation, the data look like:

In [ ]:
```python
m.df3.head(5)
```

Out[ ]:

	Customer_ID	Name	Annual_Income	Monthly_Inhand_Salary	Interest_Rate	Delay_from_due_date	Num_of_Delayed_Payment	Changed_Credit_Limit	Num_Credit_Inquiries	Credit_Mix	Outstanding_D
0	CUS_0xd40	Aaron Maashoh	19114.12	1824.843333	3.0	3	7.0	11.27	4.0	Good	809
1	CUS_0xd40	Aaron Maashoh	19114.12	1824.843333	3.0	-1	0.0	11.27	4.0	Good	809
2	CUS_0xd40	Aaron Maashoh	19114.12	1824.843333	3.0	3	7.0	0.00	4.0	Good	809
3	CUS_0xd40	Aaron Maashoh	19114.12	1824.843333	3.0	5	4.0	6.27	4.0	Good	809
4	CUS_0xd40	Aaron Maashoh	19114.12	1824.843333	3.0	6	0.0	11.27	4.0	Good	809

Model

Model bases

The developed credit punctuation model is a traditional one that takes it principal inspiration on the FICO model, there are multiple traditional models (FICO, Vantage, Credit Karma, Equifax, etc.) but we selected the FICO as a base because it´s the most used on the United States and multiple other models are based on this one. So to select the variables and the percentage of the final score they will have we looked on the FICO model pillars: 1. Payment history: essentially how frequently a borrower makes payments on time. Accounts for missed or late payments. 2. Amounts owed: how much a borrower owes, includes credit card balances, loans, and mortgages. 3. Credit history: how long a borrower has had credit accounts. Including age of their oldest account and the average age of all their accounts. 4. Credit mix: examines types of credit accounts a borrower has—credit cards, loans, and mortgages. 5. New credit: evaluates the number of recently opened credit accounts and credit applications. Also looks at the borrower’s overall credit history. It is not known for sure how they are weighted every one of this pillars on the final score of the model, but lot of sources say that the distribution is the next:

Pillar	Percentage
Payment history	35%
Amounts owed	30%
Length of credit history	15%
Credit mix	10%
New credit	10%

We add a sixth pillar to our model, this pillar is the Credit utilization. It is important because if a person is using properly his credit line it could be increased and vice versa. Below we can see the final conformation of the pillars and the variables selected for the model structure:

Pillar	Percentage	Variables
Payment history	20%	Delay_from_due_date
	15%	Num_of_Delayed_Payment
Amounts owed	30%	Outstanding_Debt
Lengh of credit history	10%	Credit_History_Age
Credit mix	15%	Credit_Mix
New credit	10%	Num_Credit_Inquiries

After seting the punctuation variables with their respective percentages, we gave the final score to every record. Down below we have some of the scores on a table, this table contains: - Customer_ID and Name: Descriptive data of the person that is being scored. - Model_Punctuation: The final punctuation the model assigned for every person, the range of the punctuation is between 170 and 1000. - Model_Score: The final score that the model gave, if the punctuation was under 600 the score is Poor, if it is between 600 and 800 is Standard and if it is above 800 it is Good. - Original_Score: The original score that the dataset had.

In [ ]:
```python
m.results
```

Out[ ]:

	Customer_ID	Name	Model_Punctuation	Model_Score	Original_Score
0	CUS_0xd40	Aaron Maashoh	930	Good	Good
1	CUS_0xd40	Aaron Maashoh	960	Good	Good
2	CUS_0xd40	Aaron Maashoh	930	Good	Good
3	CUS_0xd40	Aaron Maashoh	880	Good	Good
4	CUS_0xd40	Aaron Maashoh	860	Good	Good
...	...	...	...	...	...
99995	CUS_0x942c	Nicks	820	Good	Poor
99996	CUS_0x942c	Nicks	820	Good	Poor
99997	CUS_0x942c	Nicks	820	Good	Poor
99998	CUS_0x942c	Nicks	850	Good	Standard
99999	CUS_0x942c	Nicks	820	Good	Poor

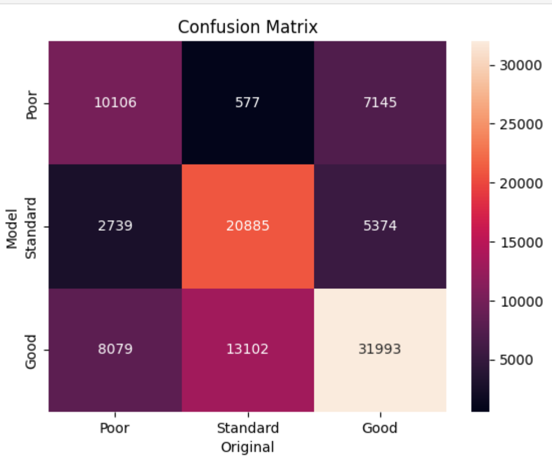
100000 rows × 5 columns

In [ ]:
```python
print(f'The accuracy of the model was of {round(m.accuracy*100,2)} %')
```

```
The accuracy of the model was of 62.98 %
```

Now we can see the Confusion Matrix that the majority of the model errors are on the Good model assigned that are on reality Poor. The model assigned Good are the scores with more accuracy.

In [ ]:
```python
v.heatmap(m.results['Original_Score'],m.results['Model_Score'],['Poor', 'Standard','Good'])
```



Conclusions

After we developed the model and get the final scores we can say that: 1. The variables and parameters that are generally used, are completely arbitrary, so in this case you’ll have to believe that we have chosen the best ones, as we do with FICO. 2. The variables that we finally used where the ones that gave us a best accuracy, this is interesting considering that the other models we were trying they considered other variables that are out of the FICO pillars. So despite we weren´t able to get a more precise accuracy, the variables are not the problem, the problem may be the ranges of values that we are taking as reference to give the punctuation. 3. Using non complex models (with added variables) can lead us to a model accuracy maybe above 50, but can’t take us to an optimal accuracy, so this is not an optimal way to found our desired model; probably using machine learning could get us where we want to be.