Critical analysis report

**Albina Maharjan**

**Student ID: 48045977**

Table of Contents

[1. Load Dataset and Clean Dataset 2](#_Toc149857676)

[2. Data Analytics and Classification 3](#_Toc149857677)

[3. Data Preprocess 4](#_Toc149857678)

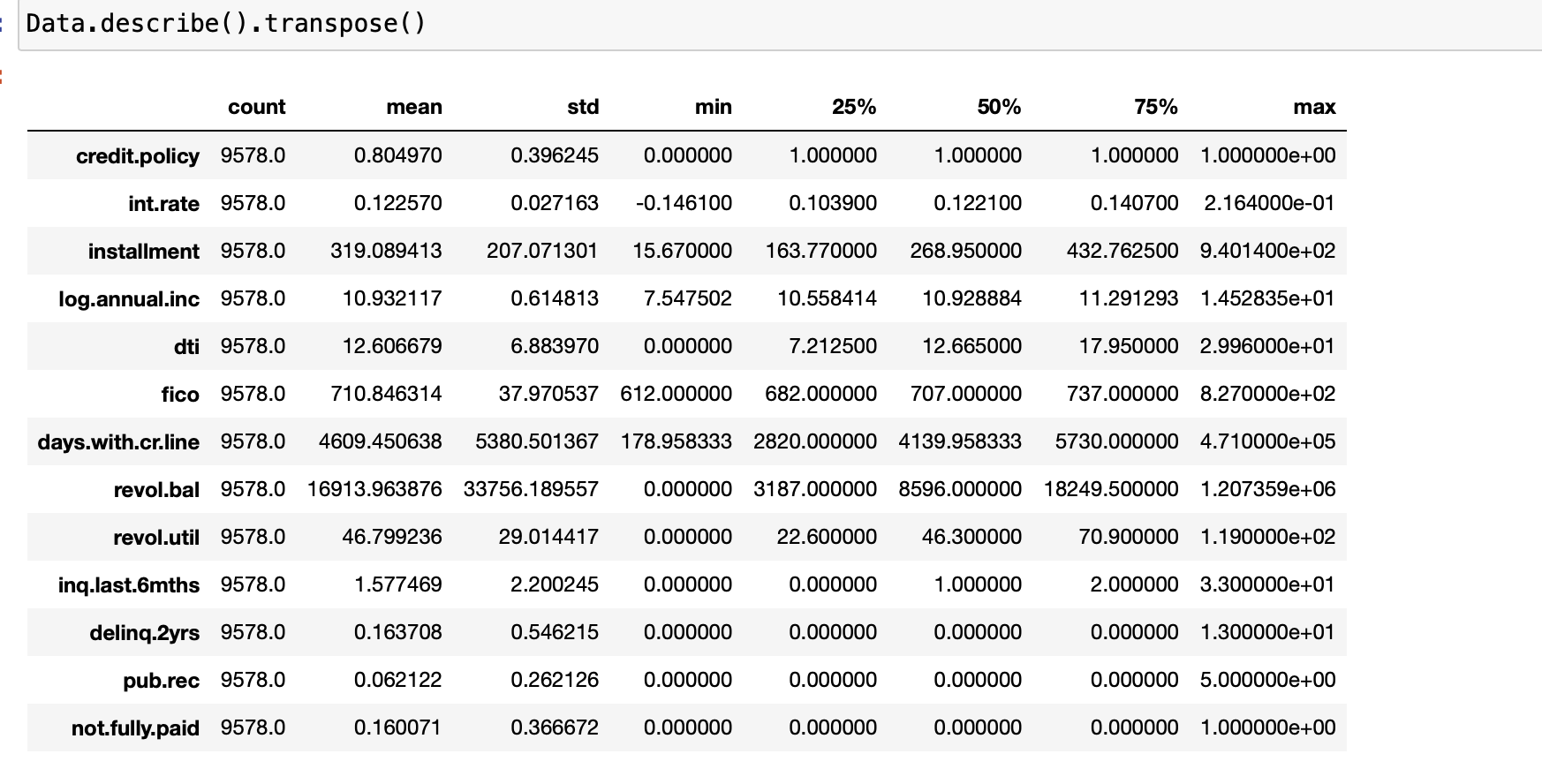
[**Process of Object data type** 4](#_Toc149857679)

[4. Dataset Splitting 5](#_Toc149857680)

[5. Evaluation with test set 6](#_Toc149857681)

# Load Dataset and Clean Dataset

**Given Code:**

****

**Amemded Code:**

**A screenshot of a computer code

Description automatically generated**

****

**Analysis**

* Before Data Cleaning & Scaling:

Some features had abnormal values. For instance, the int.rate had a minimum value of less than 0, which is impossible in practice.

The days.with.cr.line had a maximum value equivalent to approximately 1,290.41 years, which is clearly an extreme value.

The range of values for each feature varies widely. For example, the mean of int.rate is 0.122570, while the mean of revol.bal is 16,913.963876.

* After Data Cleaning & Scaling:

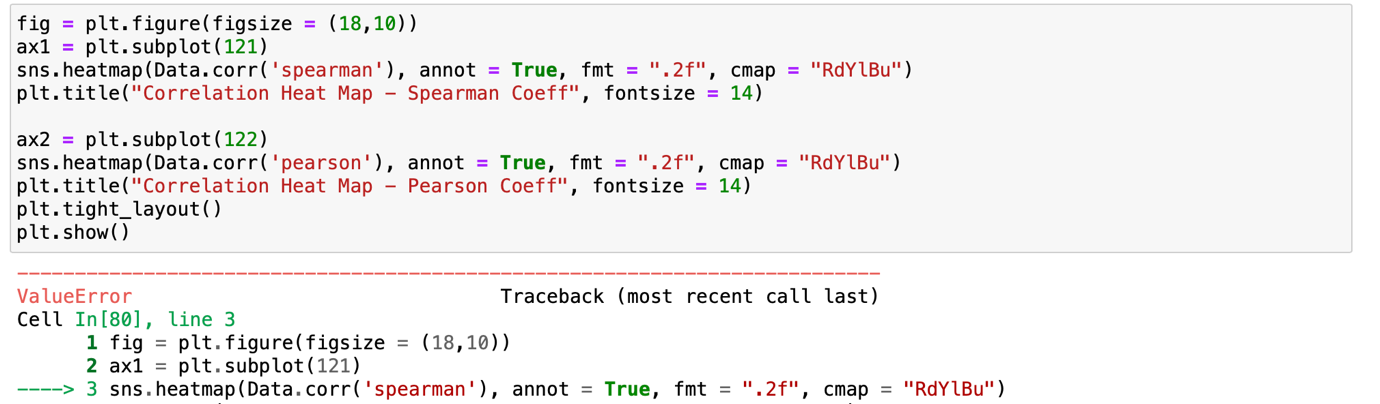
The abnormal values in int.rate have been addressed. Its minimum value is now 0.060000, which is reasonable.

The extreme value in days.with.cr.line has been addressed. Its maximum value is now reduced to about 176.4 years (64,730 days).

The data has been scaled using the MinMaxScaler, which brings the values of all numeric columns to a range between 0 and 1. This can be observed in the minimum and maximum values for each feature. For instance, int.rate now has a range from 0.060000 to 0.216400

# Data Analytics and Classification

**Given Code:**

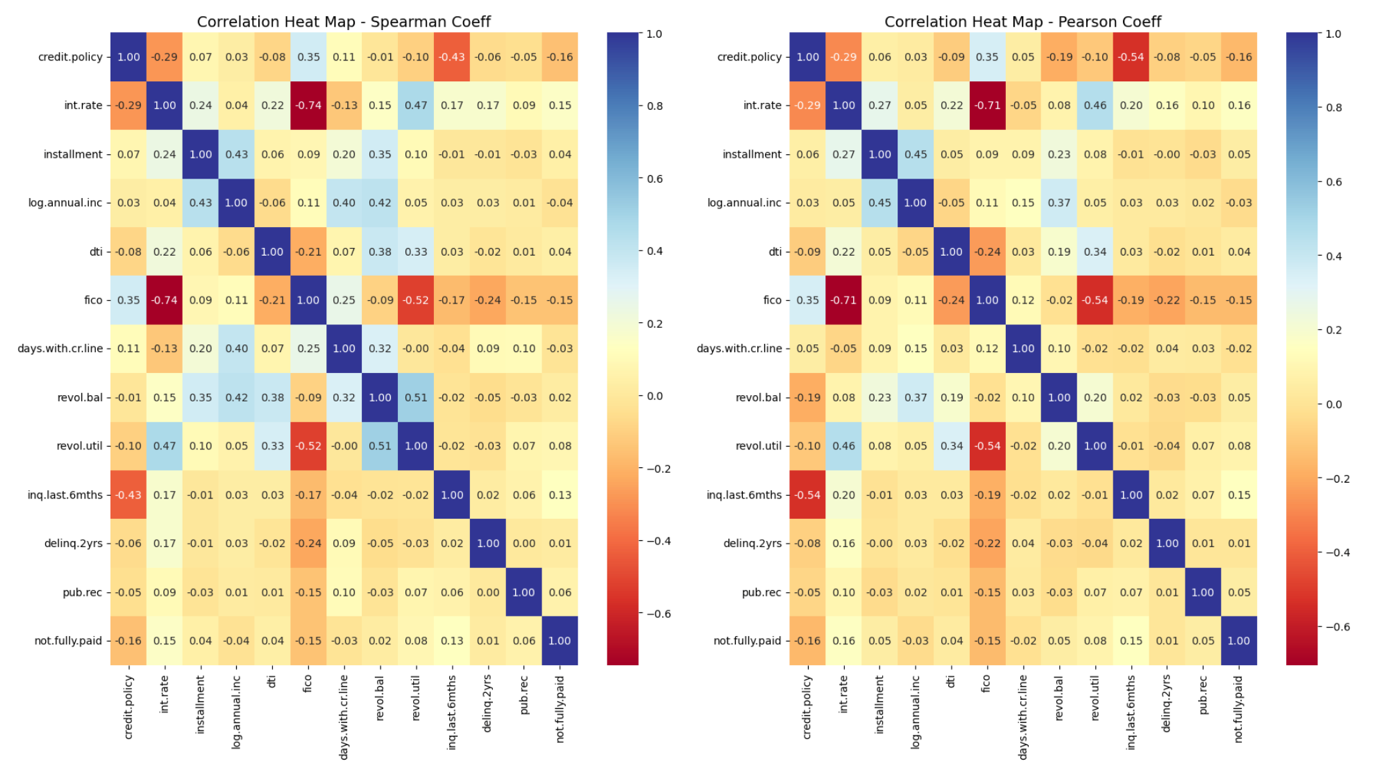
****

I was encountering *Valueerror* in this code i.e. could not convert string to float: 'debt\_consolidation'. To encounter this error, I have excluded non-numeric columns to generate heatmap.

Code:

A screenshot of a computer program

Description automatically generated

The Result: 

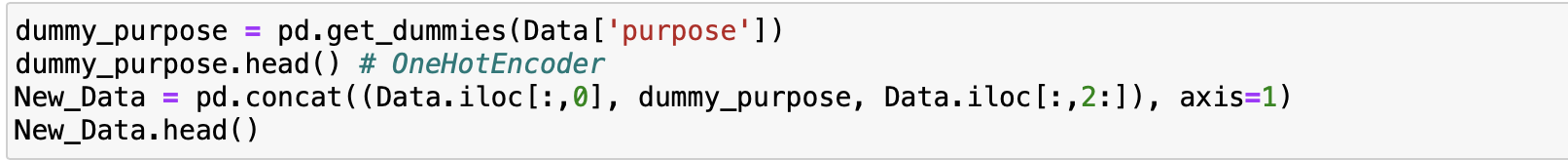
From the heatmaps, we can find different correlations between each feature and 'credit.policy'. We only reserve features that have positive correlations with 'credit.policy' by removing all features, i.e., 'int.rate', 'revol.bal', 'inq.last.6mths' and 'not.fully.paid' which are negatively correlated with 'credit.policy'. We believe that features with negative correlations are useless for model training.

# Data Preprocess

Process of Object data type

The logistic regression model cannot well process the object data type. We convert this data type with OneHotEncoder such that this feature can be handled by the logistic regression model.

Given code:



Amended Code:



Result:

The table below the code demonstrates the results of this process, where each unique value in the "purpose" column gets its own column in the dataframe, with binary values (0 or 1) indicating the presence of that category for each row.

A screenshot of a survey

Description automatically generated

# Dataset Splitting

We Split all data records into training set (80%), validation set (10%) and test set (10%) so that we can determine hyper-parameters with k-cross validation.

Sort all records by an ascending order of 'credit.policy'. and select the first 90% as the training and validation sets. The rest 10% will be used as the test set.

Given Code:

A screenshot of a computer code

Description automatically generated

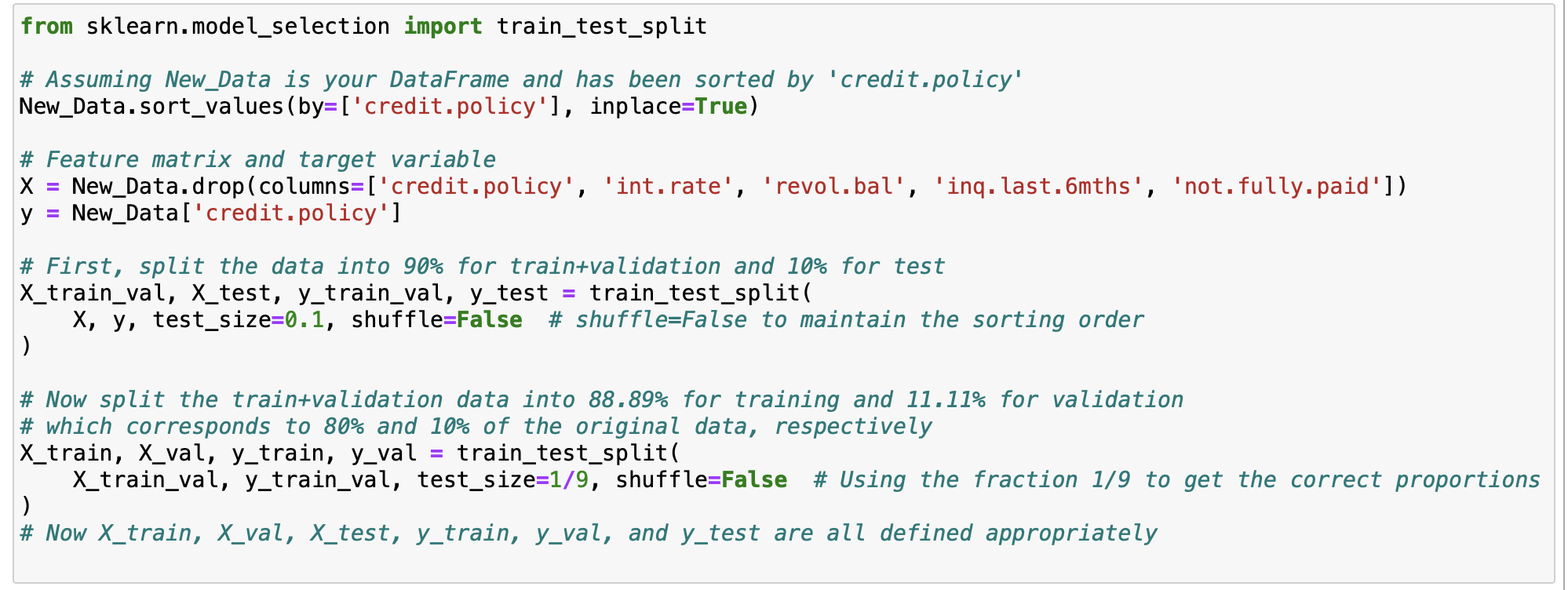
Analysis:

The data splitting described is a manual process that divides the dataset into two parts: a combined training and validation set (90%) and a test set (10%). This approach is slightly unconventional for few reasons and they are:

* The entire dataset is sorting by credit.policy field before splitting. This may lead to biased training, validation, and test sets.
* The use of fixed split i.e. 1st is 90% for training d validation and 10% for testing. This ensures that the set is comprised of records with the highest credit.policy values.
* Instead of using built in functions like train-test-split from sklearn, it has used manual splitting.

For better result

We can do train-test-split



# Evaluation with test set

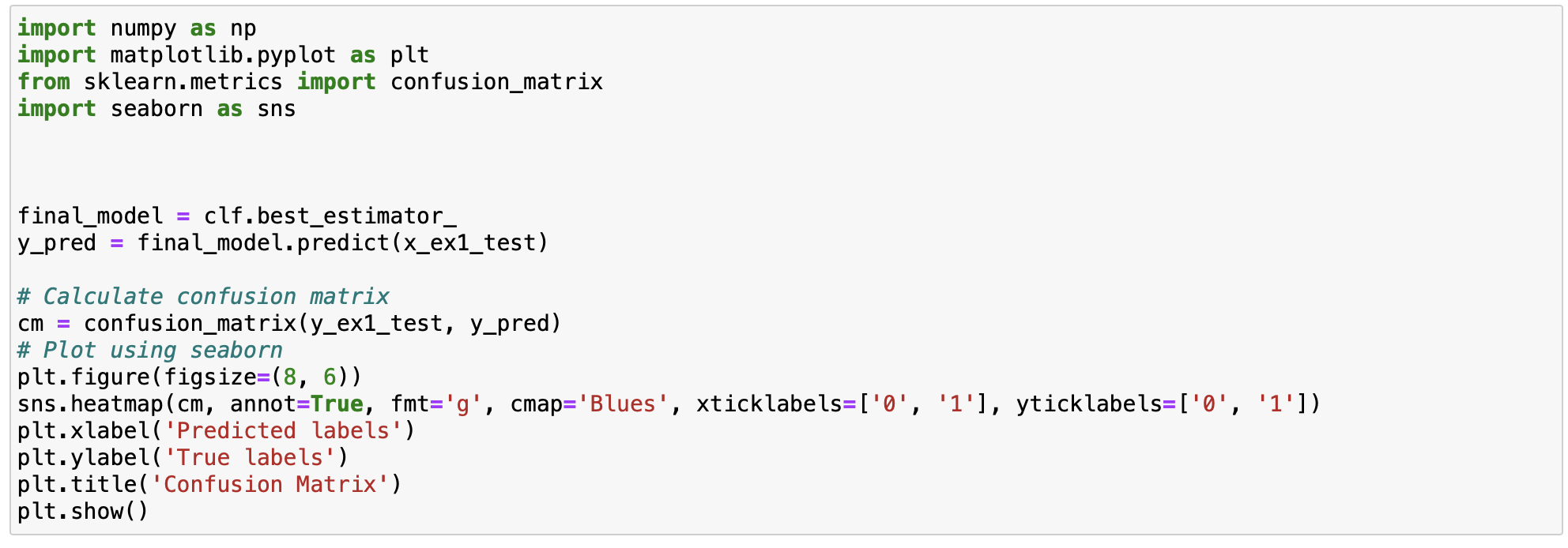
This is the last step. We further evaluate the performance of the decision tree model on the test set.

Given Code:

A screenshot of a computer program

Description automatically generated

Amemded Code:



Result:

A blue square with white text

Description automatically generated