Thomas Gardner

210541874

Applied Machine Learning

Using machine learning to classify borrowers as constituting a good or bad risk

The Data

Importing and describing the data using the CSV File

To begin, I installed the Pandas library, which gives us the functions required to read as well as manipulate and view the Comma Separated Value file (CSV). This is one of many libraries that I will use throughout.

Now I will import Pandas as pd meaning I can access all the functions of the pandas library by using dot notation on pd instead of the full name of the library (pandas). This is demonstrated by using the read_csv function.

Next I can print the first rows of my data to get a clean general overview of the dataset using the .head pandas function. The following method by default prints 5 rows including all of the columns [2]. I am using the german_credit_data.csv dataset [3] which has 1000 rows (instances or people who have requested a loan) and 11 columns (which consists of 10 features and 1 column that classifies the borrowers as a good or bad risk).

How many missing values are there?

After calling isnull on my data, there is a large amount of missing information. Of the 1000 entries in Savings accounts and Checking account 183 and 394 respectively are missing.

After this step, I identified the most frequently occurring value in both the Saving accounts and Checking account column using SciPy's mode function as this ignores NaN values. This established that the most common value was ‘little’ for both the Saving and Checking accounts.

I can assume these features are quite vital in predicting the reliability of loans, as if a borrower was wealthy there would be no feasible reason for them defaulting, meaning it must have a strong correlation to being a ‘good’ or ‘bad’ risk. As such deleting the entire row is not possible meaning I will have to impute the missing values. I did so initially by replacing the null values with the Mode.

The reason I chose the mode is that it will fill in the empty values with the lowest value (or in this case the most common financial situation) of a person taking out a loan. In addition to this it is safe to assume that I would prefer the worst case scenario (low disposable income and only a small amount of savings) as this would increase False Negatives and reduce False Positives. This is desirable as banks would usually want to reduce the Risk of unpaid loans, to ensure the greatest return.

Pre-processing steps

The first 5 column and row values and the dimensionality of the data is shown by data.shape. I have 10 features and 1 target variable. They are nearly all in text form meaning I will need to convert this into numeric form before I input it into a machine learning algorithm that requires numeric inputs and outputs.

I chose to use LabelEncoder as it takes the amount of unique attributes of a feature and replaces them with a numeric/integer value (starting from 0). In the ‘Sex’ column LabelEncoder took our 'Male' attribute and replaced it with 1 and took the 'Female' attribute and replaced it with 0. I then needed to insert this as the 'Sex' column and remove the original 'Sex' column which used Strings as values. I did so by replacing the entire column. I then repeated this procedure for every column that had non-numerical data resulting in the dataset containing only numbers [4].

Why is this dataset interesting and what real-world problem could be solved by analysing it?

This dataset provides an insight into why consumers utilise credit, and what they use that credit for. As you can see the data provided creates a succinct but comprehensive picture of each individual's finances. Using this data and whether they were able to repay the loan allows me to extrapolate this to new cases, determining prior to loaning money whether the person is a high risk or not. This could help to reduce underwriting expenses which is a clear ‘goal for any company’ in order to have the ‘highest net income possible’ [1]. This would be very useful in the financial domain, ensuring only safe loans were approved while reducing operating costs.

Is there any skew in the representation of the different attributes?

As this is a Classification problem the balance of our Risk class (class value) needs to be checked to see if additional data processing is required in order to artificially even out the balance.

I had a somewhat imbalanced dataset here, as 70% of all the instances are classified as 'good' (a low amount of risk) and only 30% as bad (a high amount of risk). This needs to be addressed when preprocessing the data which comes in a later section.

Do the attributes conform to a Gaussian distribution?

At this point in the coursework I was unsure as to what model I would select, but standard distribution of our features is often a required prerequisite for many useful tools in Machine Learning. As such, I called skew on my data to check the skew of my attributes and many were skewed quite significantly. In order to clarify this I used pyplot to graph the distributions of all the features [5]

In these graphs I can see that all of the features are quite far from a Gaussian distribution, and as such to utilise useful functions, such as Pearson's Correlation Coefficient, I have to normalise the data. In addition to this, the skewed data contains many outliers which will result in models that are not as accurate.

Constructing and selecting features

Separating the features from the target variable.

Here I separated the data into two separate arrays one of which contains all of the data about our instances and the features that pertain to them (X) and another contains how that data is labelled (Y).

Using a wrapper filter technique - Recursive Feature Evaluation (RFE)

From a brief overview of the data it appeared that some features that are included are somewhat unnecessary (like 'Purpose') to predict the target variable of 'good' or 'bad' credit risk.

RFE will recursively remove these features, testing the model accuracy each time on a model of our choice to assess importance. An important note is that after I apply RFE I need to renormalise the data as certain features will have been removed. I chose Linear Regression to select features and chose to initially be quite aggressive with my removal of features passing the RFE function 4 (to select just 4 features).

Building the Machine Learning algorithms

Training and Test split

I created a training and test split on the unprocessed data first passing this data to a Logistic Regression model which achieved a higher accuracy than with the processed data.

Using K - fold cross validation

As mentioned, when I used a Simple Train Test split (using our unprocessed X values/features) my measure of accuracy has gone up but there are various other approaches we can use that reduce the variance of our estimated accuracy, such as K - three fold cross validation. I used three folds as the dataset is relatively small and did not want data to be too sparse by dividing it too much.

The effect of preprocessing

The preprocessing used means the model is less computationally expensive (as RFE has removed 6 features from the dataset meaning less factors for the model to consider) but it is also less accurate. But there was also much less variance in my results as the deviation from the mean was reduced. As this dataset is relatively small, arguably I should prioritise accuracy over the reduction of computational expense.

As mentioned before, one issue I believed I had with my data had been caused by my method of filling in the NaN values with the mode as opposed to using a more advanced method such as Backfill, arguably this created too much bias causing more instances to be predicted as 'bad' risk. This is a desirable characteristic given the use case of this model but I wanted to check if this was the case using a Confusion Matrix on both the processed and unprocessed data [6].

The Confusion Matrix is split into quarters. The top left is the True Positives, the top right is the False Positives, the bottom left is the False Negatives and the bottom right is the True Negatives. My attempts at preprocessing have had the desired effect (although only slightly) as they have reduced the False Positives. As previously stated, I then try to oversample the minority class - which I hoped would result in more bias towards classifying new data as bad as opposed to good (a desirable result).

When using the oversampled data our percentage of False Positives went down but so did our accuracy. I then trained two other models on the processed data (Random Forest and Decision Tree) and got similar results with the Decision Tree reducing False Positives the most.

At this time in my modelling I had been sacrificing accuracy for the sake of reducing False Positives and increasing False Negatives - which as you can see from the Confusion Matrix has been somewhat successful. As such, a bank that used this model would have a low chance of unintentionally loaning money to someone who would be less likely to pay back the money but a higher chance of not loaning money to someone who does constitute a good risk - thus minimising risk but also hindering profits. To rectify this I changed the process I selected for filling in null values as I believed it was too rudimentary. In addition to this I also changed the aggressiveness of RFE in order to determine if I am excluding valuable features

Adapting the preprocessing

Next I created a new array that replaces all NaN data with Backfill as opposed to using the Mode. In addition to this I increased the amount of selected features to seven as opposed to four and finally oversampled the minority class in order to reduce bias. I compared the algorithms in a much more succinct way than used before and included an additional one for good measure.

The accuracy of all our models improved [7] after I adapted the RFE, normalised the data subsequent to RFE and oversampled the minority class. This results in a much more satisfactory model with the DTC and the RFC achieving a much higher accuracy than the other models whilst also keeping False Positives much lower than False Negatives. After this test I clearly established that our two candidate algorithms will be a Decision Tree Classifier and a Random Forest Classifier. These are both examples of supervised learning as I have labelled data that determines each instance class with the algorithm target being to categorise new data as a ‘bad’ or ‘good’ risk [8].

Evaluating models and analysing the results

Using a Confusion Matrix and a Classification Report to justify my choice of models.

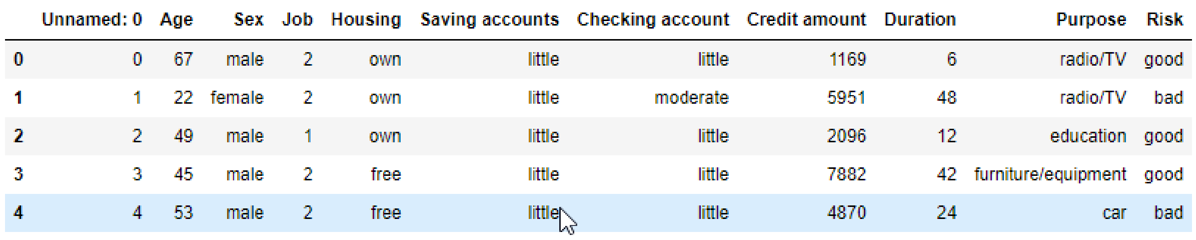
The classification report provides us with a simple overview of our models and demonstrates that it is performing quite well. The most important figure is the F1 score weighted average which effectively summarises how well the model performed highlighting that our models achieved a good score (the closer to 1 the better). I also saw that the issue of our class imbalances was rectified as 0 and 1 both had similar counts of samples. Finally we can see in the Confusion Matrix that my attempts at balancing accuracy whilst minimising False Positives have been somewhat successful, hopefully making this model useful for its intended purpose. The model also generalises well to the test data meaning it is not overfitted as it can predict accurately when provided with unseen data.

Trade-offs between the two selected models

The Decision Tree is very intuitive with SciKit-learn using ‘binary trees’ with each node only having ‘two children’ (two options to fulfil). Whether or not an instance is within a certain boundary determines which child node the result goes to from the root node. Each result will reach a certain depth in the decision tree which will confirm whether it is categorised as ‘good’ or ‘bad’. The decision tree is called a ‘white box algorithm’ as we are able to fully understand its internal workings. Random Forests on the other hand are ‘black box algorithms’ so despite its benefit to the accuracy of our model, it is near impossible to comprehensively explain how my model internally comes to a classification of new data [9].

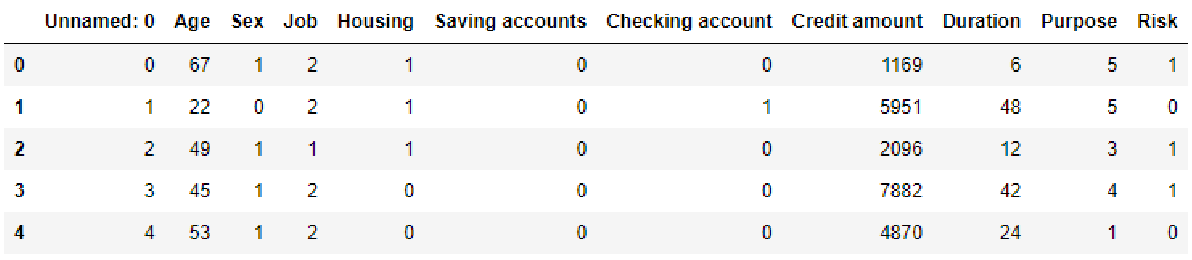
Appendix

[1]
J. Kagan, “Underwriting Expenses,” *Investopedia*, 2022.
https://www.investopedia.com/terms/u/underwriting-expenses.asp#:~:text=Underwriting%20expenses%20include%20all%20expenses,the%20highest%20net%20income%20possible. (accessed May 31, 2022).
[2]



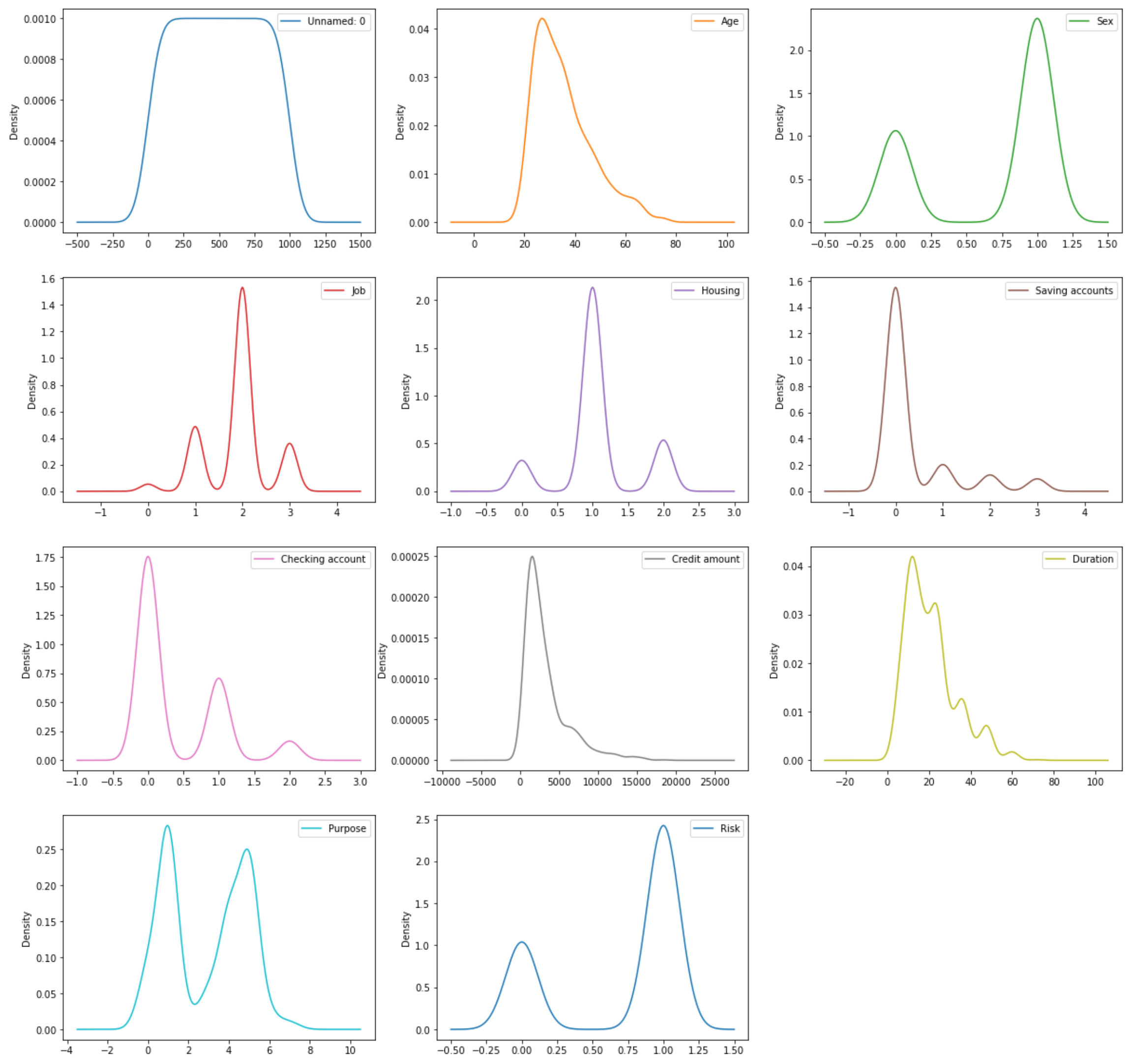
	Unnamed: 0	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose	Risk
0	0	67	male	2	own	little	little	1169	6	radio/TV	good
1	1	22	female	2	own	little	moderate	5951	48	radio/TV	bad
2	2	49	male	1	own	little	little	2096	12	education	good
3	3	45	male	2	free	little	little	7882	42	furniture/equipment	good
4	4	53	male	2	free	little	little	4870	24	car	bad

[3]
UCI Machine Learning, “German Credit Risk,” *Kaggle.com*, 2016.
https://www.kaggle.com/datasets/uciml/german-credit (accessed Jun. 06, 2022).
[4]



	Unnamed: 0	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose	Risk
0	0	67	1	2	1	0	0	1169	6	5	1
1	1	22	0	2	1	0	1	5951	48	5	0
2	2	49	1	1	1	0	0	2096	12	3	1
3	3	45	1	2	0	0	0	7882	42	4	1
4	4	53	1	2	0	0	0	4870	24	1	0

[5]

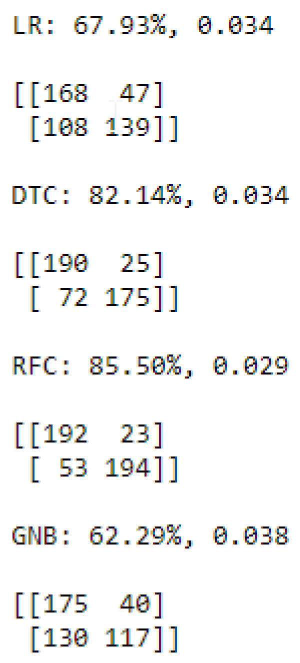


[6]

```
[[ 33  60]
 [ 31 206]]

[[ 38  55]
 [ 29 208]]
```

[7]



```
LR: 67.93%, 0.034

[[168  47]
 [108 139]]

DTC: 82.14%, 0.034

[[190  25]
 [ 72 175]]

RFC: 85.50%, 0.029

[[192  23]
 [ 53 194]]

GNB: 62.29%, 0.038

[[175  40]
 [130 117]]
```

[8]
The, “1. The Machine Learning Landscape,” O’Reilly Online Learning, 2022. https://learning.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ch01.html#what_is_machine_learning (accessed Jun. 16, 2022).
[9]
6. Decision Trees, “6. Decision Trees,” O’Reilly Online Learning, 2022. https://learning.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ch06.html#idm45022163459208 (accessed Jun. 16, 2022).