Loan Approvals

A Neural Network Approach

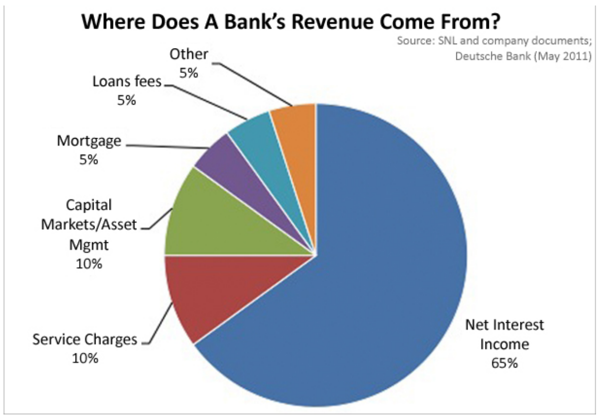
Joshua Lake 12576930

Utsav Patel 13253366

Video Pitch Available at:
https://youtu.be/KsmavWi7MzY

1. Introduction

1.1 Background and Problem Statement



In the current commercial and housing lending markets, banks are continuously seeking to grow their profit margins whilst maintaining client relationships and creating shareholder value without increasing risk. Historically, banks have relied on loans as their number one source of income (net interest income), emphasizing the importance of the loan approval process.

Most recently, the royal commission into banking in Australia saw how banks were assessing would-be homebuyers for a loan flagged as one of the biggest issues. This saw a tightening of lending standards by APRA in 2017 and late 2018 and a number of new rules to be enforced in the loan approval process. With so much emphasis now on the loan approval process, it is important that the banks get it right. Historically, manual and paper-based processes seemed to be largely outdated and as a result slowed decision accuracy and speed. Growing complexity in the data attributable to each loan has also seen spreadsheets become largely redundant.

Specifically, loan officers are continuing to rely on traditional means to evaluate loan status, and there is a number of issues with this.

→ These decisions can be subjective
→ The process is often slow
→ Bias can arise in deciding whether or not the applicant is successful.

1.2 Goal of the Project and why our problem is worth doing

The underlying concept that can be built on is the accuracy in identifying risky loan applicants. Risk is by far the most important concern in a bank's decision to loan to a customer and with hundreds of new applications every day, there is still a large amount of loans being approved that result in loans defaulting. In order to build on this process, automation has long been a goal and a challenge of the industry. Specifically, machine learning allows for time to be saved and processes to be made more efficient. If the machine learning process can predict loan approvals more accurately, credit losses will be made smaller for banks, there will be fewer servicing costs, higher revenues and a higher percentage of quality loans.

For our algorithm implementation, we have chosen a binary classification problem. This problem is extremely relevant to banks, since loans are the most important factor of banking. To maintain the integrity of banking, businesses have started to find methods to predict customer behaviour regarding loan approvals. Machine learning algorithms have been used by banks to predict the approval decision of the bank. In the dataset, it is very hard to predict when a person will be granted the loan and what person won't be granted the loan due to different income levels, applicant incomes and loan amounts. This problem is also extremely relevant in real-life since many banks can then identify whether that person can receive the loan or not.

Since the dataset we are using contains a binary classification problem, we need to predict the status of the loan for all people. To determine the loan status, we will use Neural Networks classifier which helps us to evaluate the attribute results. Artificial Neural Networks are promising tools for dealing with noisy and incomplete data for different business problems. It also helps with arranging incomplete data that may have not been entered. In addition to that, the network can learn from past gathered data in the dataset and provide a better output.

The use of neural networks (as also proven through Python Code), has been vital for the classification problem as it has given us a good accuracy for prediction of the results. It has also enabled for this classifier to be used as an effective tool in classifying loan status at a high accuracy standard. Therefore, Neural networks are an effective tool for classifying binary problems and ultimately helps for banks to predict these results.

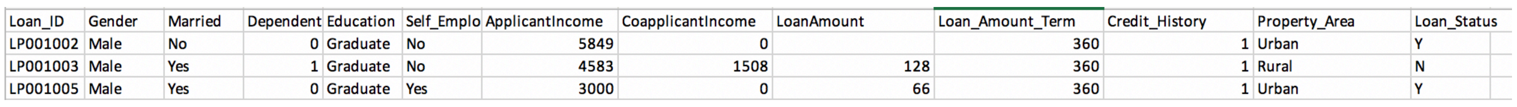
1.3 The Algorithm

In order to best solve this problem, it is proposed that a Neural Network be developed for the classification of loan applicants. The application of new technologies in the business sector can improve the businesses competitiveness and lead to higher performance. Neural networks have been used in business regularly in modern years. The particular Neural Network proposed is a Multi-layer Perceptron and will be compared to the performance of a simple decision tree classifier and logistical regression classifier to demonstrate its performance and emphasise the beneficial results it will have in the banking industry. The algorithm will explore the benefits of a neural network in providing accurate loan decisions and will look to overcome the issue of multivariate normality as seen in logistical regression.

The Neural network is able to handle complex relationships and nonlinear data and as a result poses as a great tool for loan classification. It will take an input layer of x variables, transform them through hidden layers and in turn predict the y variable as an output layer. The neural network will lean on the input data provided by the loan client and will gain information on patterns and relationships in order to predict the output.

2. Data Exploration

2.1 The dataset



Loan_ID	Gender	Married	Dependent	Education	Self_Emplo	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
LP001002	Male	No	0	Graduate	No	5849	0		360	1	Urban	Y
LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y

The dataset contains a number of customer loan applications that include a variety of attributes, with the classification being the "Loan Status" column. The attributes are explained below:

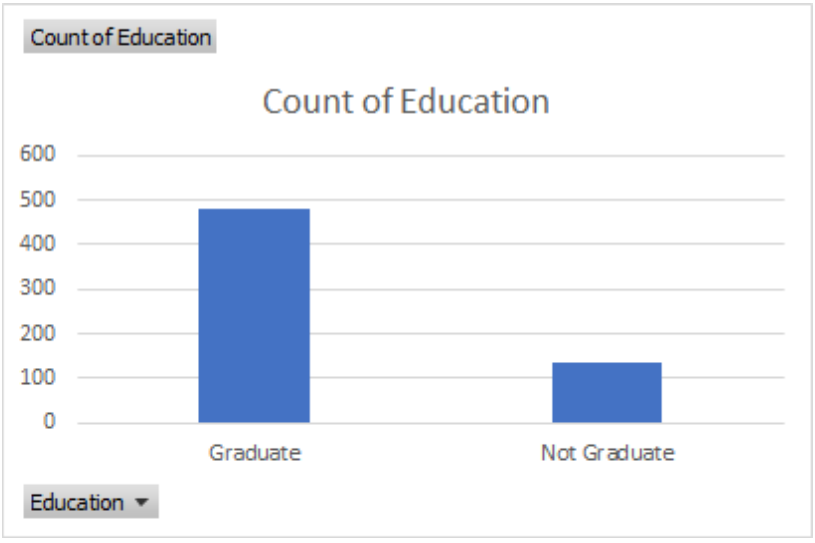
Attribute	Explanation	Type
Loan ID	The applicants unique loan identification number	Numeric
Gender	The gender of the applicant (Male or Female)	Nominal
Married	The marital status of the applicant (Married (Yes or No))	Nominal
Dependant	Dependants (Number of children)	Numeric
Education	Education level of the applicant (Graduate or non-Graduate)	Nominal
Self Employed	Whether or not the applicant is self-employed or not (Yes or No)	Nominal
Application Income	The income of the applicant	Numeric
Coapplication Income	The income of the applicants spouse	Numeric
Loan Amount	The amount the applicant is requesting	Numeric
Loan Amount Term	The period of time the loan is for	Numeric

Credit History	Whether the applicant has a good or bad credit history (1 or 0)	Nominal
Property Area	The location of the applicants dwelling (Urban, Semi-Urban or Rural)	Nominal
Loan Status	Whether the loan is approved or not (Yes or No)	Nominal

For this assignment, we used a "Machine Learning Classification" problem regarding the Prediction of Loan Approval. Our objective is to estimate whether a loan application can be accepted or not. We used an online dataset for training our data and tried to pre-process it to understand it concisely. From eight attributes, there were seven attributes with Categorical data and three main numerical data.

- **Categorical Attributes:** Gender, Married, Dependent, Education, Self Employed, Credit History, Property area
- **Continuous:** Applicant Income, Coapplicant Income, Loan amount

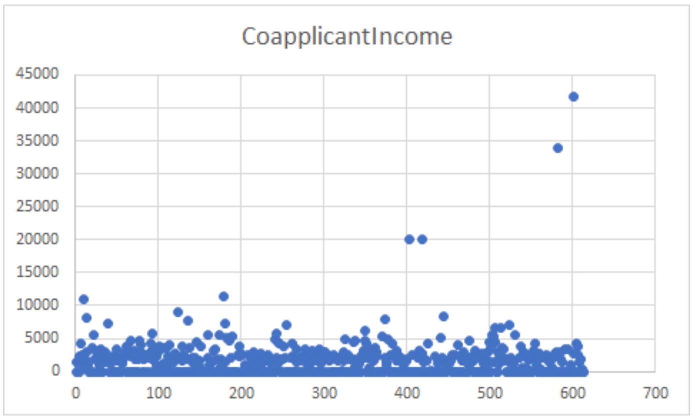
<u>Education</u>:



Education	Count of Education
Graduate	480
Non-Graduate	134

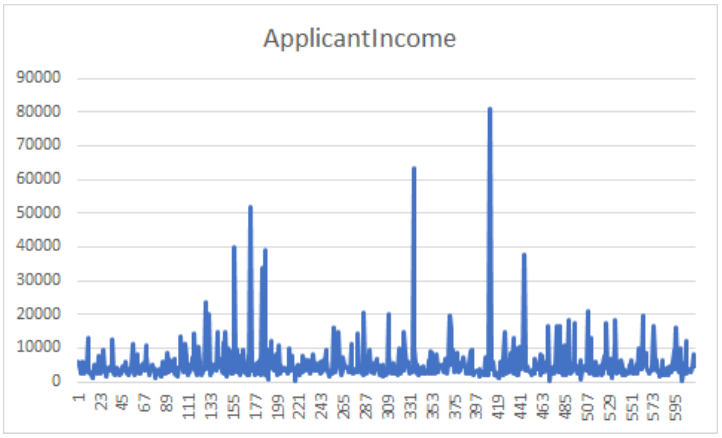
This column is about the education levels of the applicants. Most of the people were "Graduated" so that it increases their chances of obtaining a loan. However, this does not imply that only graduate people will get a loan but will in turn enhance their chances.

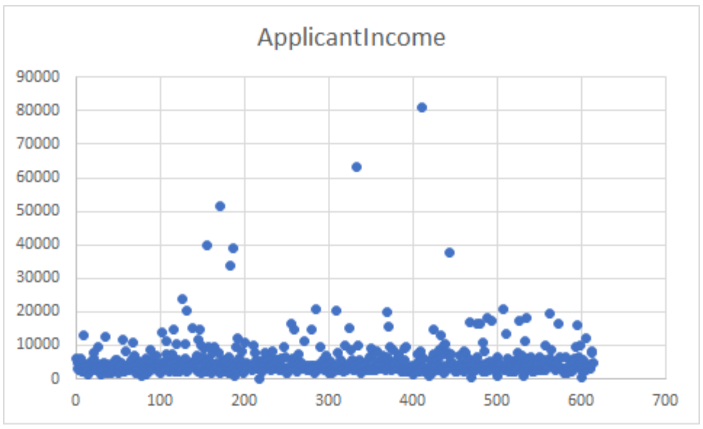
<u>Coapplicant Income</u>:



<u>Coapplicant Income</u>	
Mean	1621.246
Standard Deviation	118.0938

Applicant Income:
This column refers to the amount of income the applicant has. The graph below represents the income of all individuals.





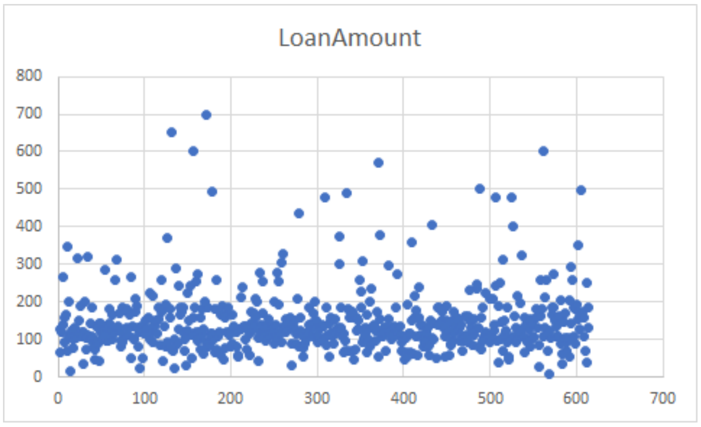
Applicant Income	
Mean	*5403.549*
Standard Deviation	*6109.042*

According to the graph above, most of the data is skewed between 150 to 81000. There are 2 main outliers located in the middle top of the graph. The income levels of both outliers are significantly higher which affects the mean of this data.

However, the low "Standard Deviation" for this attribute signifies that the applicant's income is clustered together and not widely spread. The clustering of data represents that the data is close to the mean and hence follows a bell curve.

The outliers are easier to identify on the scatter plot; hence it is a better graph for this attribute.

Loan Amount:

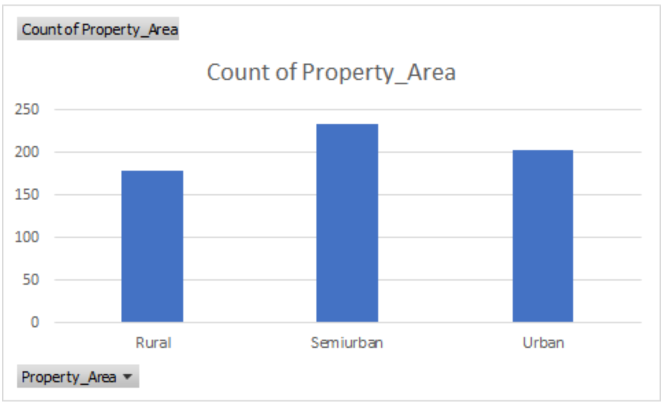


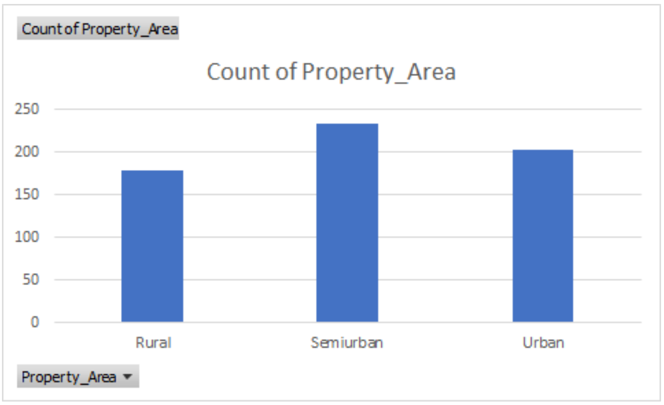
Loan Amount	
Mean	1621.246
Standard Deviation	118.0938

This column shows the number of loan individuals has got from the bank. From the graph above we can determine that the mean for these data points is 1621.246. The standard deviation is also relatively far from the mean indicating that the distribution is not to closely packed and spread apart.
From the graph above, anything with or over the value of 600 is considered as outliers.

<u>Property Area:</u>

This attribute explains the type of "Property Area" that the applicants live in. This has an impact on their local approval status because of their locations.



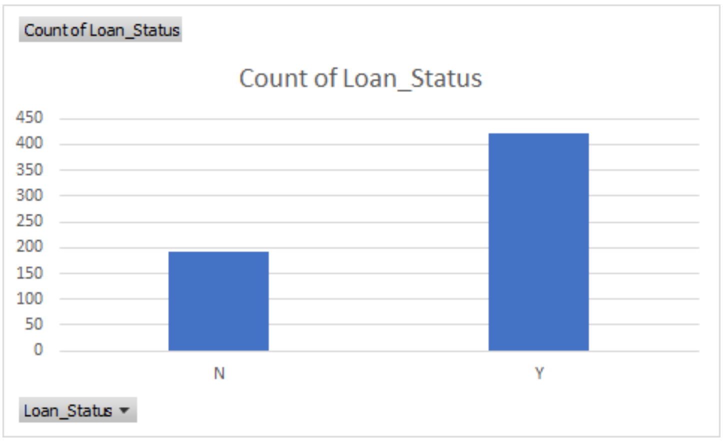


<u>Property Area</u>	**<u>Count of Property Area</u>**
Rural	179
Semiurban	233
Urban	202

From the Bar chart above, it can be observed that most applicants lived in the Semiurban area, followed by Urban and finally Rural. The

<u>Loan Status:</u>

Loan _Status	**Count of Loan_Status**
N	192
Y	422

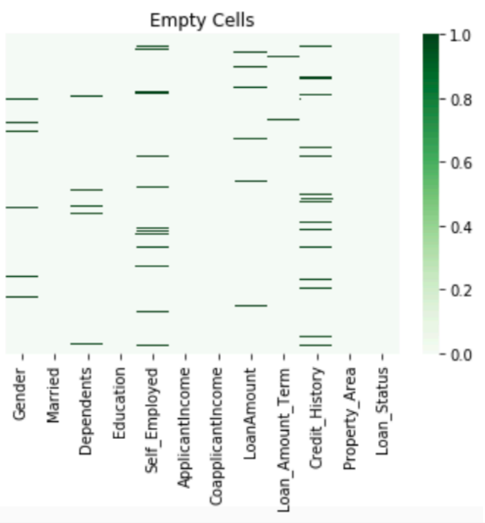


This attribute has only 2 options; Yes or No. It displays the status of the applicants regarding whether they have the loan or not. Most people were granted the loan depending on several factors and less than half the people were rejected a Loan. In total, 422 people received a loan whereas 192 people didn't get it.

2.2 Data Challenges

The process of loan approval originates with the bank requesting a variety of information from the applicant, and assigning them a loan ID. This information legally must be legitimate and help the bank answer a number of questions in regard to how much risk they are taking on. This data acquisition however may result in issues of completeness, formatting and relevance. The acquisition of information is in both paper and digital format in the form of application forms that also require personal identification information, however this identification information is not relevant for a classification of whether or not the applicants loan is approved.

Missing Values:
When retrieving the data from the loan applicant, some information may or may not be specified, resulting in blank cells in the data table. Whether this be through mistake or lack of information available, it has a negative influence on the outcome of loan status. The blank records have been identified using visual techniques in python and handled accordingly. The first line of code identifies the sum of blank cells in each column, whereas the second visualizes the empty cells across the entire data set



```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
```

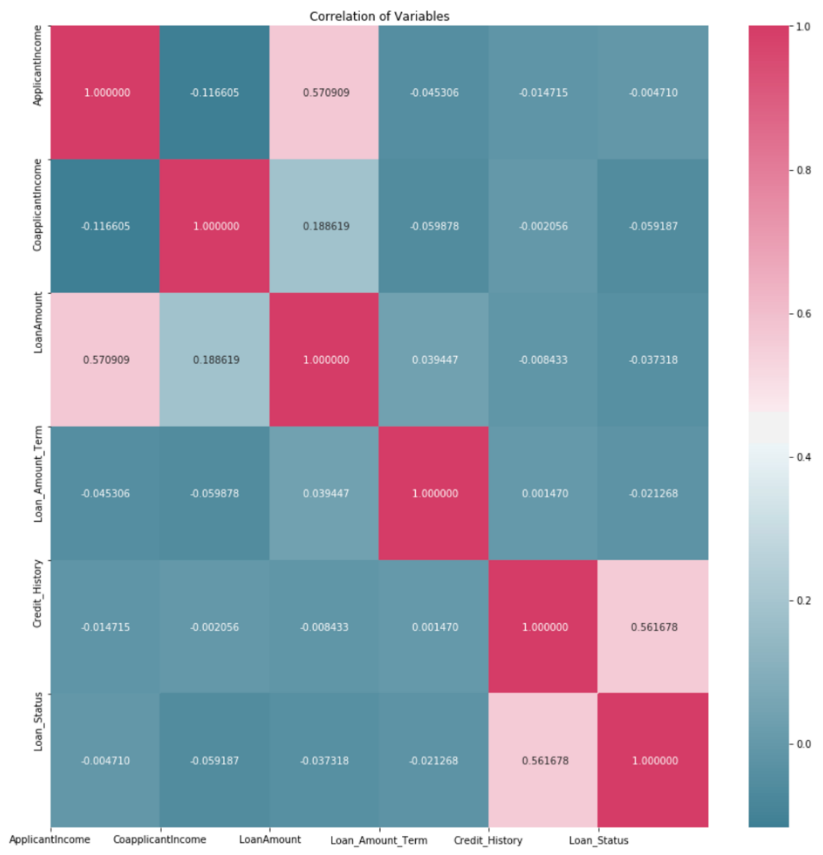
```
print (data.isnull().sum())
sns.heatmap(data.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

(Identification of null values; Visual representation of null values)

When a numerical blank cell has been identified in the dataset, the cell has been replaced with the median of the column to limit the number of columns removed. This has been implemented on the Coapplicant Income and Loan Amount data columns. As there tends to be outliers in the data due to the nature of loan requests being widespread, the median has been used to fill the cells as opposed to the mean due to the mean being heavily influenced by outliers, resulting in replacing the cell with data that is not truly reflective of the dataset.

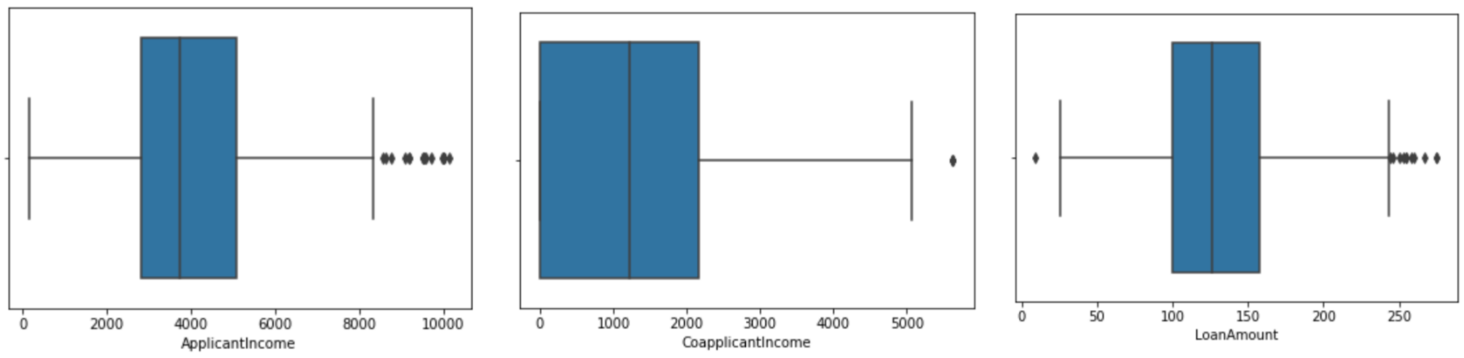
Similarly, for the Gender, Married, Dependents, Self-employed and loan_amount_term, the data missing has been replaced with the mode. The mode is used here to identify the most occurring result in the given column and treat the empty cell as 'common' occurrences of a particular value. Replacing data is much more beneficial than dropping the entire data row as it prevents data(information) loss and preserves the quality of the initial dataset.

Due to the correlation between credit history and loan status being higher than any other two variables, it has been decided that this is a sensitive feature in terms of prediction and rather than replace with the mean, median or mode, these rows will be dropped completely. When a bank assesses someone's credit history, this is a very personal and sensitive matter that is influenced by a number of variables. It would do no justice to the prediction to fill this cell, and although reduces sample size, it is necessary for the real-world applicability of the model.



(Correlation between variables and Loan status)

Outliers:
Due to the large amount of loan applications received every day, a large amount of unusual and unique individuals will apply for unique loans with unique data. This information tends to be outlying from the general population of loans and as a result will not be beneficial in predicting the loan status of the majority of loan applicants. To identify these outliers, they have been both visualised and mathematically computed using the IQR before being removed. Due to the small number of outliers and the reasonably large size of the dataset, it has been decided that removing the outlier will not negatively affect the prediction and will improve the quality by removing irrelevant data.



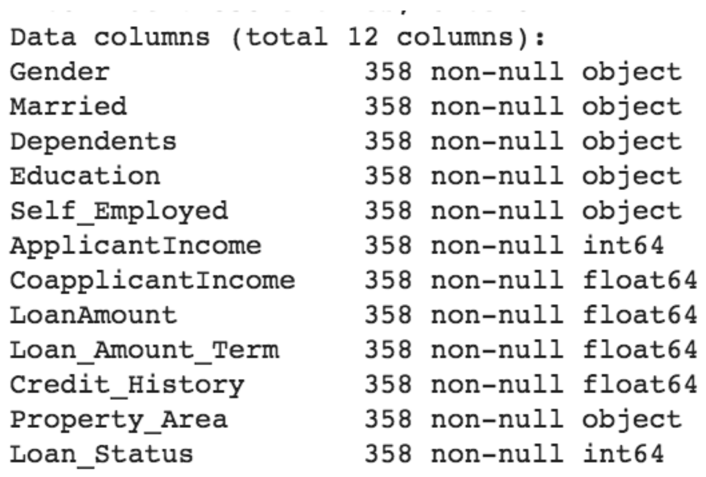
```
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3-Q1


(data < (Q1-1.5*IQR)) | (data > (Q3 + 1.5 * IQR))
data = data[~((data < (Q1 - 1.5 * IQR)) |(data > (Q3 + 1.5 * IQR))).any(axis=1)]
data.shape
```

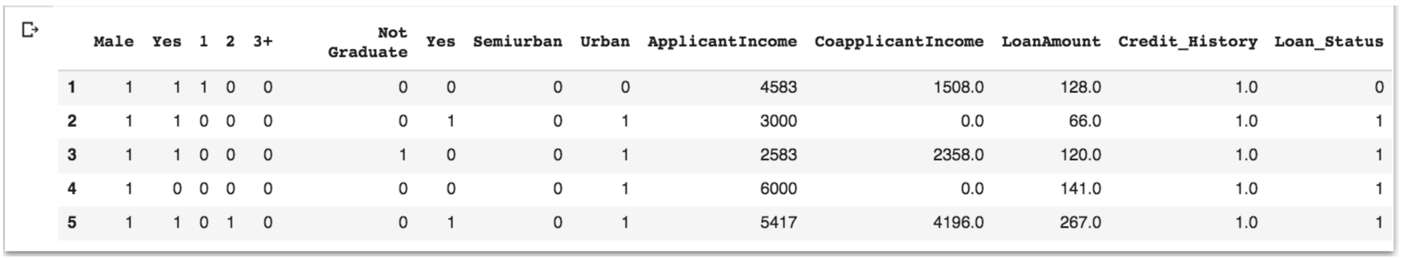
Conversion of Categorical data:

Due to the various forms of data required from the applicant in the sense that they can be both numerical and categorical, many models require data to be transformed into a consistent pattern first. Due to the low cardinality of the categorical features in the dataset and the need for numerical data in various algorithms, the dataset has been encoded with new columns appended and old columns dropped.

These columns are identified by the 'object' data type as seen to the right. To make the entire data set consistent and assist in the prediction of the y variable using x variables, the Loan_Status column has been converted from Yes and No to 1 and 0 respectively. The data frame now looks like this (below). This was achieved by binarizing the data, and dropping the original columns, replacing them with dummy variables.



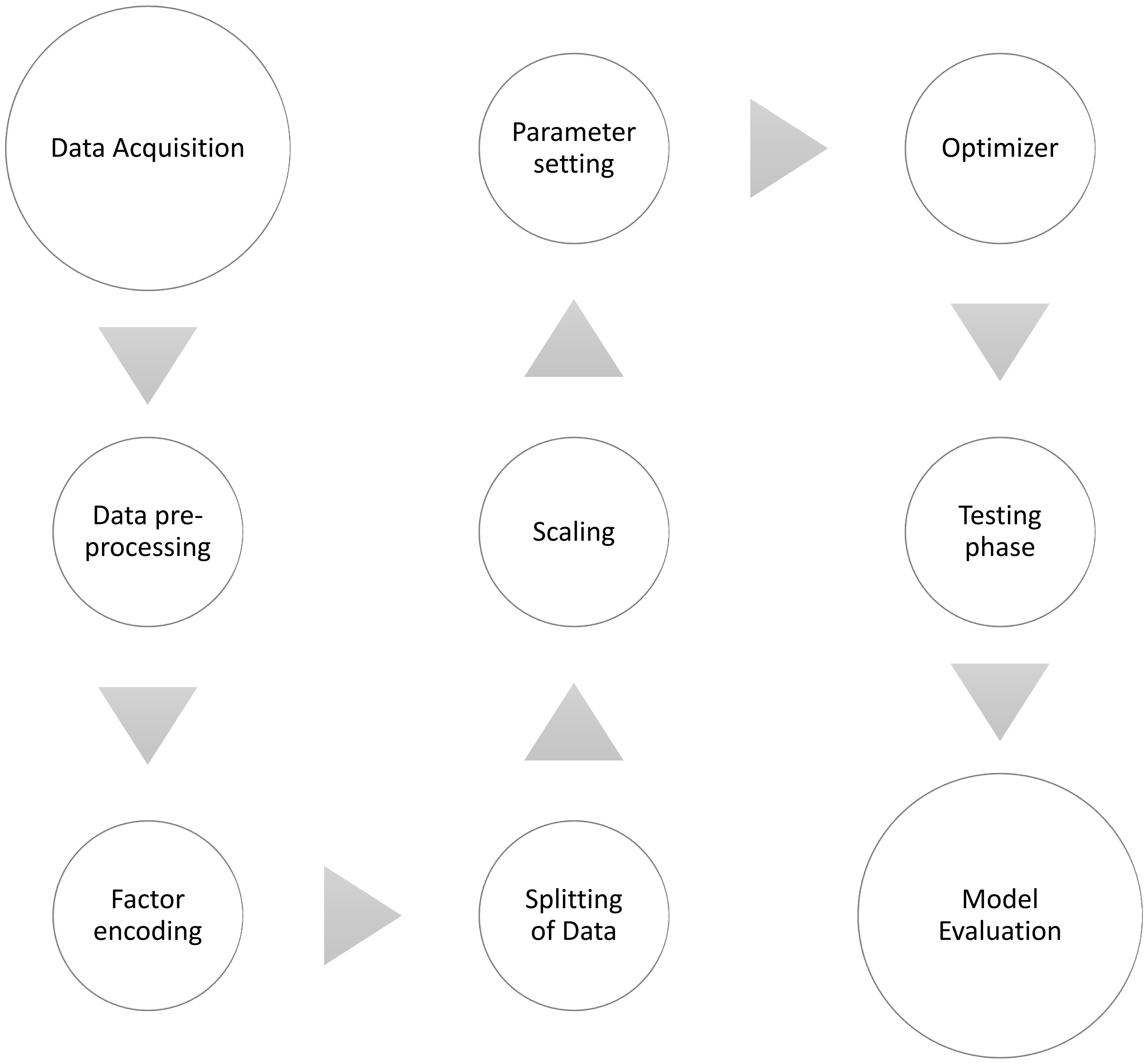
```
Data columns (total 12 columns):
Gender               358 non-null object
Married              358 non-null object
Dependents           358 non-null object
Education            358 non-null object
Self_Employed        358 non-null object
ApplicantIncome      358 non-null int64
CoapplicantIncome    358 non-null float64
LoanAmount           358 non-null float64
Loan_Amount_Term     358 non-null float64
Credit_History       358 non-null float64
Property_Area        358 non-null object
Loan_Status          358 non-null int64
```



	Male	Yes	1	2	3+	Not Graduate	Yes	Semiurban	Urban	ApplicantIncome	CoapplicantIncome	LoanAmount	Credit_History	Loan_Status
1	1	1	1	0	0	0	0	0	0	4583	1508.0	128.0	1.0	0
2	1	1	0	0	0	0	1	0	1	3000	0.0	66.0	1.0	1
3	1	1	0	0	0	1	0	0	1	2583	2358.0	120.0	1.0	1
4	1	0	0	0	0	0	0	0	1	6000	0.0	141.0	1.0	1
5	1	1	0	1	0	0	1	0	1	5417	4196.0	267.0	1.0	1

2.3 Design Data structure/Plan data models and Tests
2.3.1 The Data Model

The experiment will be implemented using the following structure, with the model evaluation consisting of a comparison to two other classifiers, demonstrating alternatives for the banking industry.



1. **Data Acquisition:** The dataset acquired consists of (originally) 615 rows of data and 13 columns. It has been acquired from a financial organisation for an educational purpose. It is set to replicate the data that is acquired on a daily basis by those working in the banking sector and seeking to approve or disapprove loan applications.

2. **Data Pre-Processing:** There is a number of stages to this step, in order to get the data prepared for the neural network. These techniques have been discussed above and include filling missing values, removing outliers, converting columns and dropping certain data. The purpose of this is to allow the loan approver to transform their data in such a matter that the most information can be taken from the raw data, and as a result can be used in the neural network for information extraction and classification.

3. **Factor encoding:** The encoding of categorical data using dummy variables, resulting in new columns being appended and redundant columns being dropped, allows for a consistent numerical set of

data that is necessary for the neural network. The conversion of both the x and y variables will allow for the input to successfully be transformed into output.

4. **Splitting of data:** Cross validation will be used to split the data into training and test sets. This is appropriate as the loan industry has a large sample size, and both the test and training sample have relatively similar distributions. The random nature of this split will ensure the neural network has a low chance of bias. There are 13 x variables that in turn are used to predict the y variable.

5. **Scaling:** The normalization is necessary for the MLP classifier as the input layer variables need to be activated to a smaller number in order to optimize results. This converts inputs into a comparable range and removes geometrical bias towards the data vectors.

6. **Parameter Setting:** A multilayer perceptron will be used, containing multiple hidden layers. The input layer takes in a bias node as well as all X variables, and at this stage no computations occur so the inputs into the hidden layer are these input nodes. The hidden layers will compute linear and nonlinear transformations on the data, before the output layer transforms the hidden activations into a usable format. A number of repetitive experiments will be conducted within the optimizer to determine which set of parameters are best.

   The number of neurons that was most effective was deemed to be 14 across the 4 hidden layers.

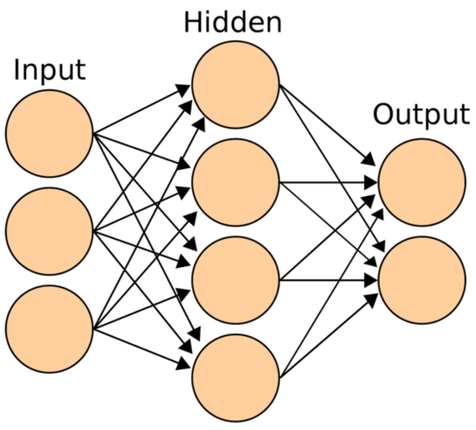
7. **Optimizer:** The optimizer will allow the experiment to be conducted on a variety of parameters to determine the best algorithm set up, that will result in the highest level of prediction

8. **Testing phase:** The testing phase will compare the predicted values to the original y-values in order to determine the accuracy of the model

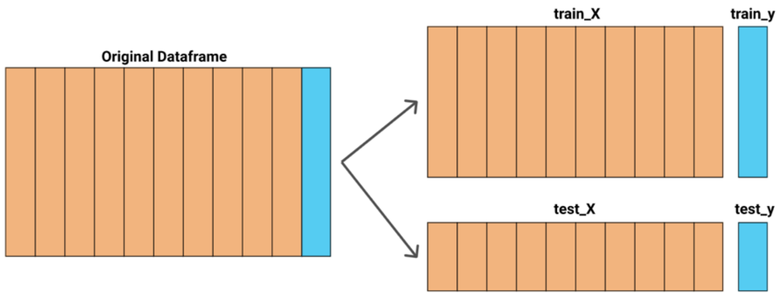
9. **Evaluation:** The model will be evaluated through a comparison with a logistical regression model and a decision tree model, before being visualized on a variety of graphics.

3. Methodology: Implementing the Algorithm

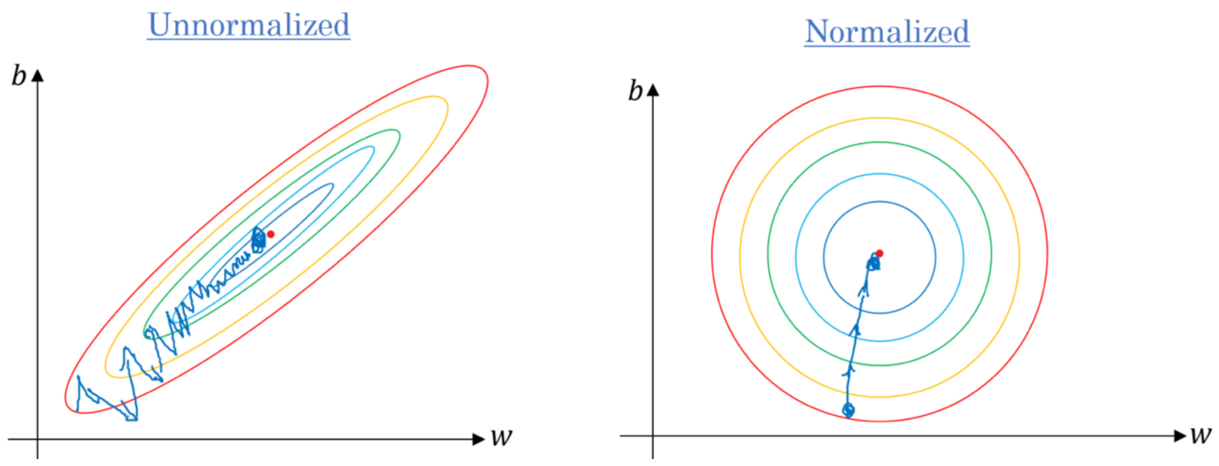
The MLP neural network was implemented with the following process and outcomes. Following the data pre-processing and preparation, the MLP is now ready to be implemented. The first stage of this is to implement the necessary libraries. Following this, the data is split into 4 variables under the categories of training data and testing data using the train_test_split function.



This splits the x variable from the y variable at a test size of 85% training and 15% testing and this abides by the rule to never train on test data. Resulting from this is 4 variables; X_train, X_test, y_train and y_test. The training data and test data will be used for their respective duties, however the y_test will be compared against the predictions of the classifier to determine the accuracy of the classifier.



Following the splitting of the data, the X_train and X_test data is normalized using a scaler, in this case, the standard scaler. Mathematically, the gradient based optimization of a neural network requires the data points to be normalized in order to transform the data "shape" into a more concentrated circle, forming diagonals of covariance.



Now that the data is prepared for the MLP, the parameters must be set. The hidden layer sizes, activation method, solver method, learning rate and alpha are optimized using the grid search CV. This grid search CV filters through several different parameter settings to determine the optimal parameter space that will achieve the highest prediction. This optimal parameter space included:

- 4 hidden layers
- A hidden layer size of 14
- A constant learning rates
- And the Adam solver

The initiation of the neural network will set the networks "weights", that are adjusted with each training set, learning as it is processed. The number of neurons comprising in the input layer is equal to that of the number of features in the data (x-variables). The neural network then trains the data and predicts on a test set the loan status of the applicants, extracting the results to a confusion matrix, scatter plot and accuracy score.

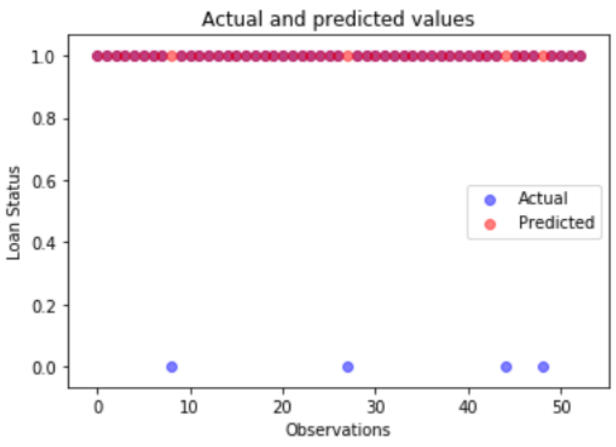
4. Evaluation

The neural network was executed on 53 test rows, classifying 4 incorrectly and 49 correctly for an accuracy of 92% as seen in the classification report. This testing was the result of the optimizer testing a variety of constraints and the optimal parameters being set as aforementioned.

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         4
           1       0.92      1.00      0.96        49

    accuracy                           0.92        53
   macro avg       0.46      0.50      0.48        53
weighted avg       0.85      0.92      0.89        53
```



The execution speed was high, resulting in a quick classification of the test set. (Y variable) In order to demonstrate alternative methods and their results for a comparative study, a decision tree model and logistic regression model were used for comparison.

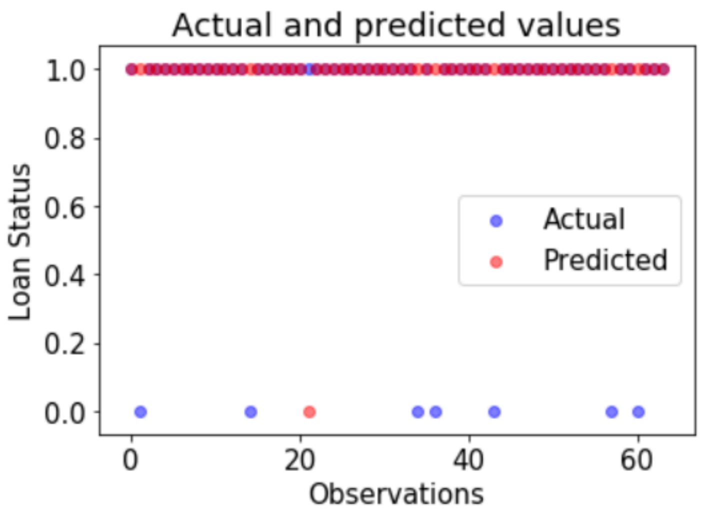
Logistical Regression

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         7
           1       0.89      0.98      0.93        57

    accuracy                           0.88        64
   macro avg       0.44      0.49      0.47        64
weighted avg       0.79      0.88      0.83        64

Accuracy: 0.875
```



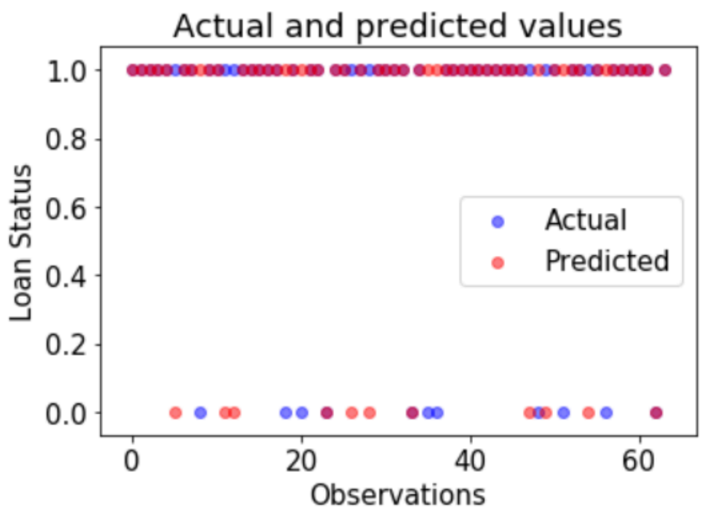
The logistical regression model yielded an accuracy of 87% classification correction when operating under the same pre-processing as the neural network.

Decision Tree

```
              precision    recall  f1-score   support

           0       0.27      0.27      0.27        11
           1       0.85      0.85      0.85        53

    accuracy                           0.75        64
   macro avg       0.56      0.56      0.56        64
weighted avg       0.75      0.75      0.75        64
```



The decision tree yielded an accuracy of 75% classification correction when operating under the same pre-processing as the neural network.

Evaluation Report

Faced with a variety of issues ranging from slowness in execution to bias and subjectivity in the decision-making process, leading to incorrect classification of loans and financial loss. This neural network not only provides a high classification rate exceeding 90%, it removes the emotional and individual bias that can get in the way of highly delicate financial decisions at a high speed. It will not only save the banks money in terms of only making loans that are profitable, but it will also reduce man hours spent with clients. An overall improvement of the classification percentage will make the bank more reliable and will see it attract more customers, also improving profits. This demonstrates that the neural network will be a great tool to use in the finance sector moving forward.

5. Conclusion

Furthermore, this research project has demonstrated that although faced with a number of different issues in the loan classification process, there is in fact technology available to improve the system. The use of a neural network will look to build on the loan decision process and improve the productivity, efficiency and profitability of banking firms. Going forward, this research could be implemented alongside feature engineering to optimize performance and attract information of clients that is most useful in the decision-making process. Due to the performance of the neural network, it is also possible to now take on larger amounts of variables that can assist in classifying due to the complex data handling ability of the neural network.

6. Ethical

In machine learning, the ethical and social issues must be addressed due to vast amounts of security threats that it offers. Ethics are moral principles that govern the behaviour of a an individual or business. An example of ethical issues includes the ease at users can duplicate copyright content, however ethics consider that it is unethical to do that without the author's permission.

In terms of Machine learning, classification algorithms are used in most scenarios. Classification algorithms clarify a set of rules to place data into specific groups. Models are created through programming that allow regressions and analysis.

The **Ethical challenge** is in the creation of the data models. This is largely due to the training required to train the models while a validation dataset is given to the algorithm that represents the environment.

Social concern regarding algorithm decisions develops as models start to use data. It is extremely important to understand the standards for data accuracy that have significant consequences. Hence, it is very important to consider these when an unsupervised method is used.

In terms of Neural Networks, one serious ethical issue is the ability of machine learning to teach other machine wrong things. One example of this is ‘Inadvertent discrimination’, where network learns about racism or something similar. It may be used to develop a representation of a customers’ ethnicity and can be linked in a negative way. The results are structured due to factors in society that are racist.

Similarly, our “Loan Approval” classification has certain challenges like the “selling of information about the client’s income details”, “tracking a client based on which area they live in”, “confidentiality of data”, and many more.

However, according to the Kantian duty-based approach, the approach ethics console that some things should never be done, regardless of its consequences. This method is a duty-based system that focuses on giving resect to all humans, thus portraying the essential human rights. This method also provides “certainty”, which ensures that no-one is certain about what results will be output from the problem as individuals and organisations can focus on their efforts rather than predicting the results.

References

Azure AI Gallery. (2019). Loan Granting Binary Classification. [online] Available at: https://gallery.azure.ai/Competition/Loan-Granting-Binary-Classification-1 [Accessed 17 Sep. 2019].

Britz, D. (2019). Implementing a Neural Network from Scratch in Python – An Introduction. [online] WildML. Available at: http://www.wildml.com/2015/09/implementing-a-neural-network-from-scratch/ [Accessed 20 Sep. 2019].

Budd, C. (2018). Machine learning: Is it ethical?. [online] plus.maths.org. Available at: https://plus.maths.org/content/what-can-we-use-machine-learning [Accessed 24 Sep. 2019].

Brownlee, J. (2019). Crash Course On Multi-Layer Perceptron Neural Networks. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/neural-networks-crash-course/ [Accessed 20 Sep. 2019].

Copycoding.com. (2018). Machine learning project in python to predict loan approval (Part 6 of 6). [online] Available at: https://copycoding.com/d/machine-learning-project-in-python-to-predict-loan-approval-prediction-part-6-of-6- [Accessed 17 Sep. 2019].

Docs.scipy.org. (2019). NumPy User Guide — NumPy v1.17 Manual. [online] Available at: https://docs.scipy.org/doc/numpy/user/ [Accessed 11 Sep. 2019].

GeeksforGeeks. (2019). Decision tree implementation using Python - GeeksforGeeks. [online] Available at: https://www.geeksforgeeks.org/decision-tree-implementation-python/ [Accessed 24 Sep. 2019].

Kumar, V. (2019). Applying Logistic Regression: Classifying Loans based on the risk of defaulting. [online] Medium. Available at: https://medium.com/swlh/classifying-loans-based-on-the-risk-of-defaulting-using-logistic-regression-9bd9c6b44640 [Accessed 22 Sep. 2019].

Li, M., Mickel, A. and Taylor, S. (2018). "Should This Loan be Approved or Denied?": A Large Dataset with Class Assignment Guidelines. [online] Taylor & Francis. Available at: https://www.tandfonline.com/doi/full/10.1080/10691898.2018.1434342 [Accessed 20 Sep. 2019].

Mahanta, J. (2017). Introduction to Neural Networks, Advantages and Applications. [online] Medium. Available at: https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207 [Accessed 20 Sep. 2019].

McCaffrey, J. (2015). Neural Network Binary Classification -- Visual Studio Magazine. [online] Visual Studio Magazine. Available at: https://visualstudiomagazine.com/articles/2015/08/01/neural-network-binary-classification.aspx#targetText=McCaffrey%20looks%20at%20two%20approaches%20to%20implement%20neural%20network%20binary%20classification.&targetText=In%20a%20classification%20problem%2C%20the,one%20of%20several%20discrete%20values. [Accessed 21 Sep. 2019].

Ray, S. (2017). Essentials of Machine Learning Algorithms (with Python and R Codes). [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/ [Accessed 24 Sep. 2019].

Sevenpillarsinstitute.org. (2013). [online] Available at: https://sevenpillarsinstitute.org/kantian-duty-based-deontological-ethics/ [Accessed 20 Sep. 2019].

This Link Contains the repository that includes the video, the code and the original dataset. https://github.com/JoshLake154/Machine-Learning-A2

This Link is a direct link to the code (Self Link https://github.com/JoshLake154/Machine-Learning-A2/blob/master/MachineLearning_JoshLake_UtsavPatel.ip

This is a link to the video pitch. https://www.youtube.com/watch?v=KsmavWi7MzY

```
import pandas as pd
import io
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np


url = 'https://raw.githubusercontent.com/JoshLake154/Machine-Learning-A2/master/Data.csv?token=ANALJGASFS6HEBSM54VI6B25RFZF4'
data = pd.read_csv(url)
#Reads the data file and decodes the CSV
#The dataset is now stored as a dataframe

data.Loan_Status.replace(('Y', 'N'), (1, 0), inplace=True)

data.info()

data.head()
```

```
corr=data.corr()
fig,ax=plt.subplots(figsize=(15,15))
colormap=sns.diverging_palette(221,1, as_cmap=True)
sns.heatmap(corr, cmap=colormap, annot=True, fmt=" 2f")
plt.title('Correlation of Variables')
plt.xticks(range(len(corr.columns)), corr.columns);
plt.yticks(range(len(corr.columns)), corr.columns)
plt.show()

#Correlation Visualisation between the variables to determine if any variables need to be removed
```

```
data = data.drop(['Loan_ID'], axis=1)

# We drop the Loan_ID column as
# it is an identifier and does not offer any benefit in prediction.
```

```
print (data.isnull().sum())
#Prints a summary of the null cells in each column of the data

sns.heatmap(data.isnull(),yticklabels=False,cbar=True,cmap='Greens')
#Prints a heatmap of the null cells for identification of which column is most effected by the null values

plt.title('Empty Cells')
```

```
median = data['CoapplicantIncome'].median()
data['CoapplicantIncome'].fillna(median, inplace=True)
#Replaces the null cells in the CoapplicantIncome column with the median value of the column

sns.heatmap(data.isnull(),yticklabels=False,cbar=True,cmap='Greens')

#Prints a heatmap of the null cells, showing the Coapplicant Income column now has no empty cells.
```

```
median = data['LoanAmount'].median()
data['LoanAmount'].fillna(median, inplace=True)
#Replaces the null cells in the LoanAmount column with the median value of the column
```

```
sns.heatmap(data.isnull(),yticklabels=False,cbar=True,cmap='Greens')
#Prints a heatmap of the null cells, showing the Loan Amount column now has no empty cells.
```

```
data['Gender'].fillna(data['Gender'].mode()[0], inplace=True)
data['Married'].fillna(data['Married'].mode()[0], inplace=True)
data['Dependents'].fillna(data['Dependents'].mode()[0], inplace=True)
data['Self_Employed'].fillna(data['Self_Employed'].mode()[0], inplace=True)
data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].mode()[0], inplace=True)

#Here we are replacing the empty data cells with the mode
```

```
data.dropna(inplace=True)

#Dropping the rows with empty cells in the credit rating column due to the high correlation of credit rating
```

```
sns.boxplot(x=data['ApplicantIncome'])
#A visual plot of the outliers in the Applicant income column
```

```
sns.boxplot(x=data['CoapplicantIncome'])
#A visual plot of the outliers in the Coapplicant income column
```

```
sns.boxplot(x=data['LoanAmount'])
#A visual plot of the outliers in the Loan amount column
```

```
plt.rc("font", size=15)
table = pd.crosstab(data.Property_Area, data.Loan_Status)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True, lw=2,ec='black')
plt.title('Loan Status for Property Area')
plt.xlabel('Property Area')
plt.ylabel('Loan Status')
plt.savefig('prop_vs_ls_bar')
plt.savefig('prop_vs_ls_bar')

#A visual plot of the loan status outcome based on each column
```

```
table = pd.crosstab(data.Gender, data.Loan_Status)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True, lw=2,ec='black')
plt.title('Loan Status for Gender')
plt.xlabel('Gender')
plt.ylabel('Loan Status')
plt.savefig('gend_vs_ls_bar')
#A visual plot of the loan status outcome based on each column
```

```
table = pd.crosstab(data.Married, data.Loan_Status)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True, lw=2,ec='black')
plt.title('Loan Status for Marrital Status')
plt.xlabel('Marital Status')
plt.ylabel('Loan Status')
plt.savefig('ms_vs_ls_bar')
#A visual plot of the loan status outcome based on each column
```

```
table = pd.crosstab(data.Dependents, data.Loan_Status)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True, lw=2,ec='black')
plt.title('Loan Status for Number of Children')
plt.xlabel('Number of Children')
plt.ylabel('Loan Status')
plt.savefig('dep_vs_ls_bar')
#A visual plot of the loan status outcome based on each column
```

```
table = pd.crosstab(data.Education, data.Loan_Status)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True, lw=2,ec='black')
plt.title('Loan Status for Education')
plt.xlabel('Ecucation')
plt.ylabel('Loan Status')
plt.savefig('edu_vs_ls_bar')
#A visual plot of the loan status outcome based on each column
```

```
table = pd.crosstab(data.Self_Employed, data.Loan_Status)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True, lw=2,ec='black')
plt.title('Loan Status for Self Employment')
plt.xlabel('Self Employment')
plt.ylabel('Loan Status')
plt.savefig('se_vs_ls_bar')
#A visual plot of the loan status outcome based on each column
```

```
table = pd.crosstab(data.Credit_History, data.Loan_Status)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True, lw=2,ec='black')
plt.title('Loan Status for Credit History')
plt.xlabel('Credit History')
plt.ylabel('Loan Status')
plt.savefig('ch_vs_ls_bar')
#A visual plot of the loan status outcome based on each column
```

```
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3-Q1
(data < (Q1-1.5*IQR)) | (data > (Q3 + 1.5 * IQR))
data = data[~((data < (Q1 - 1.5 * IQR)) |(data > (Q3 + 1.5 * IQR))).any(axis=1)]

#Removal of outliers from the dataset using the IQR formula
```

```
gender = pd.get_dummies(data['Gender'],drop_first=True)
married = pd.get_dummies(data['Married'],drop_first=True)
dependents = pd.get_dummies(data['Dependents'],drop_first=True)
education = pd.get_dummies(data['Education'],drop_first=True)
self_employed = pd.get_dummies(data['Self_Employed'],drop_first=True)
property_area = pd.get_dummies(data['Property_Area'],drop_first=True)
data.drop(['Gender', 'Married', 'Dependents','Education','Self_Employed','Property_Area','Loan_Amount_Term'],axis=1, inplace =
data =pd.concat([data,gender,married,dependents,education,self_employed,property_area],axis=1)
```

```
#Transformation of continuous variables to categorical variables for classification
```

```
import matplotlib.pyplot as plotter
```

```
pieLabels = '1', '0'


figureObject, axesObject = plotter.subplots()
axesObject.pie(data['Loan_Status'].value_counts(),labels=pieLabels,autopct='%1.2f')

#Visual representation of the number of Yes and No loan classifications in the entire data set.
```

```
#Neural Network
from sklearn.metrics import accuracy_score

import pandas as pd
import io
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler

from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV


#Splitting the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(data.drop('Loan_Status', axis=1),data['Loan_Status'], test_size=0.15, rand


#Scaling the data (normalizing) for optimal classification between 0 and 1
scaler = StandardScaler()
scaler.fit(X_train)
StandardScaler(copy=True, with_mean=True, with_std=True)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)


#Setting the parameteres of the MLP classifier

mlp = MLPClassifier()

mlp.fit(X_train,y_train)
MLPClassifier(activation='tanh',
              alpha=0.0001,
              hidden_layer_sizes=(14, 14, 14, 14),
              learning_rate='constant',
              random_state=None,
              solver=['adam'])


#Optimizing the parameters of the MLP classifier to find the best performing model
parameter_space = {
              'hidden_layer_sizes': [(8, 8, 8,8),(13, 13, 13,13),(14, 14, 14,14),(15, 15, 15,15), (16, 16, 16,16)],
              'activation': ['tanh', 'relu'],
              'solver': ['sgd', 'adam'],
              'alpha': [0.00001, 0.0001, 0.005, 0.05],
              'learning_rate': ['constant','adaptive'],


}

clf = GridSearchCV(mlp, parameter_space, n_jobs=-1, cv=3)
clf.fit(X_train, y_train)

print('Best parameters found:\n', clf.best_params_)
```

```
predictions = mlp.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,predictions))

#The cconfusion matrix will represent the prediction accuracy
```

```
print(classification_report(y_test,predictions))

#Printing of the classification report
```

```
print("Accuracy:",accuracy_score(y_test, predictions))
```

```
#Printing of the accuracy score
```

```
import matplotlib.pyplot as plt
_, ax = plt.subplots()

ax.scatter(x = range(0, y_test.size), y=y_test, c = 'blue', label = 'Actual', alpha = 0.5)
ax.scatter(x = range(0, predictions.size), y=predictions, c = 'red', label = 'Predicted', alpha = 0.5)


plt.title('Actual and predicted values')

plt.xlabel('Observations')
plt.ylabel('Loan Status')
plt.legend()
plt.show()

#A graphical representation of the classification performance.
```

```
#Logistic Regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import GridSearchCV


#Splitting the training and test set
X_train, X_test, y_train, y_test = train_test_split(data.drop('Loan_Status', axis=1),data['Loan_Status'], test_size=0.18, rand
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)

#Defining the predictions through the logirithimic model
predictions = logmodel.predict(X_test)


#Setting the parameter space for optimization
parameter_space = {

              'random_state':[ 9, 10, 11] }

#Optimizing the performance of the log regression
```

```
clf = GridSearchCV(logmodel, parameter_space, n_jobs=-1, cv=3)
clf.fit(X_train, y_train)

print('Best parameters found:\n', clf.best_params_)
```

```
print(classification_report(y_test,predictions))
print("Accuracy:",accuracy_score(y_test, predictions))


#Printing of the accuracy score
```

```
import matplotlib.pyplot as plt
_, ax = plt.subplots()

ax.scatter(x = range(0, y_test.size), y=y_test, c = 'blue', label = 'Actual', alpha = 0.5)
ax.scatter(x = range(0, predictions.size), y=predictions, c = 'red', label = 'Predicted', alpha = 0.5)


plt.title('Actual and predicted values')

plt.xlabel('Observations')
plt.ylabel('Loan Status')
plt.legend()
plt.show()

#A graphical representation of the classification performance.
```

```
print(confusion_matrix(y_test,predictions))
#Printing of the classification report
```

```
#DecisionTree Classifier
from numpy import loadtxt

from sklearn.metrics import accuracy_score


import io
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

#Splitting of the data into training and test sets

X = data.drop('Loan_Status', axis=1)
y = data['Loan_Status']
X_train, X_test, y_train, y_test = train_test_split(data.drop('Loan_Status', axis=1),data['Loan_Status'], test_size=0.18, stra

data.head()

dtc = DecisionTreeClassifier()

dtc = dtc.fit(X_train, y_train)
predictions = dtc.predict(X_test)

print("Accuracy:",accuracy_score(y_test, predictions))
```

```
import matplotlib.pyplot as plt
_, ax = plt.subplots()

ax.scatter(x = range(0, y_test.size), y=y_test, c = 'blue', label = 'Actual', alpha = 0.5)
ax.scatter(x = range(0, predictions.size), y=predictions, c = 'red', label = 'Predicted', alpha = 0.5)


plt.title('Actual and predicted values')

plt.xlabel('Observations')
plt.ylabel('Loan Status')
plt.legend()
plt.show()

#A graphical representation of the classification performance.
```

```
print(classification_report(y_test,predictions))
#Printing of the classification report
```