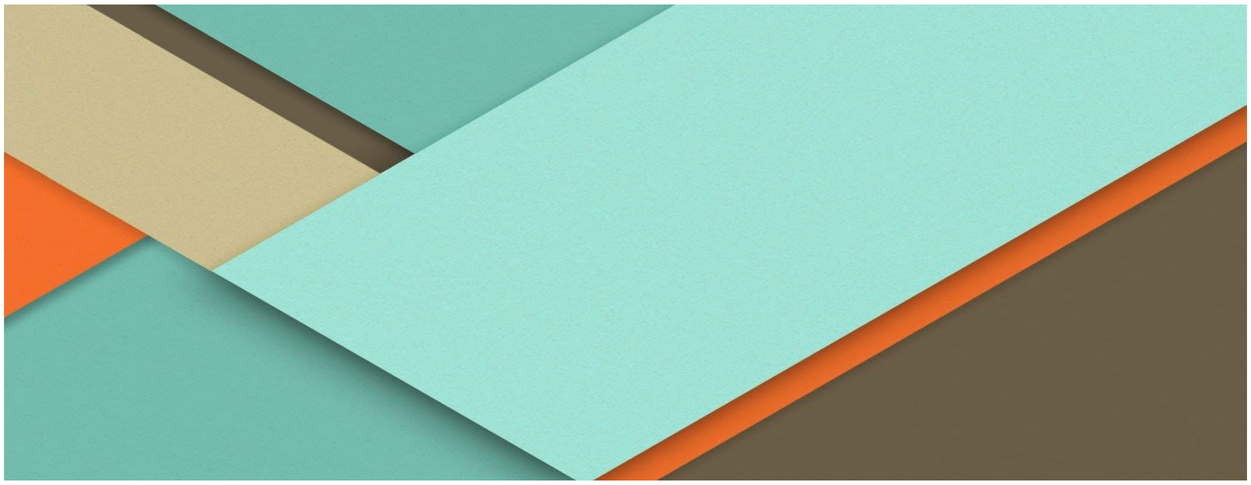


Loan Default Prediction

MS4610 Project Report

Team Members

Adarsh Subramanian(AE19B021)

Kshitij Shah(CS19B027)

Abhyudit Singh Manhas(ME18B088)

Mehul Shrivastava(CE19B066)

Abhinav Nahata(AE19B020)

Khushi Jaiswal(CE19B059)

Aditya S Vallakkudath(ME18B087)

CONTENTS

Overview

We have come up with 2 different models for the Loan prediction model, one using XGBoost, and one using Neural Networks.

We got an accuracy of 98.65% using the XGBoost Model, and an accuracy of 98.59% using the Neural Networks Model using a test train split.

We have divided the project process into 4 steps:

1. Data Analysis, Imputing Data
2. Choosing the best dataset using Cross-validation
3. Model selection / Model Definition and Training for Neural Network
4. Parameter Optimisation / Hyper Parameter Tuning for Neural Network

Note: (N) denotes that a few other observations are in the notebook and only the most crucial ones are mentioned here in the report.

XGBoost Model

Data Analysis, Imputing Data

Attached File: Data_Analysis.ipynb

- We first plot a basic **correlation matrix** and look at a few interesting connections:
    a. **Score5 and Expense** have perfect correlation
    b. There is a strong correlation (>0.5) between eleven combinations as well. (N)

  Note - For this process and ahead we split the Occupation type into three columns - X,Y,Z since our model requires Quantitative Data
- We then plot a number of **boxplots** to explore the correlations between Categorical and Quantitative variables . From these we see that there is a distinct cutoff between **Score2 and Age** and a strong relation between Score2 and loan type and Score4 and Occupation types. A few other observations are also made(N)
- The next step is plotting a number of **scatter plots** used to explore the correlations between Quantitative and Quantitative variables . From these we see that there is a perfect relation between Score5 and Expense. and a strong relation between **Score2 and Score4** and Score3 and Score5/Expense. A few other observations are also made(N) and we use regression for Score1 and Income imputation.
- Finally we use **frequency tables** to explore the correlations between Categorical and Categorical variables. From these we see that there is a strong relation between Age and Loan type. A few other observations are also made(N).

Dataset Choice

Attached file: Dataset_Choice_Using_CV+Validation.ipynb

- Now using the several methods that we have looked up we use best subset selection according to **Cross-Validation** to choose our dataset - Quite a few subsets were tried and 5 of them which had the highest accuracy are shown in the notebook
- For testing the cross validation data, we use a XGBClassifier. This is based on a **random forest model** and we do this for three reasons-

a. Since we have a **high number of categorical variables** (4 in effect becoming 7 due to the Occupation type split)

b. Since the computational power required to use cross validation on any other complex model was extremely large and we could not use SVMs.

c. In hindsight, we realize the XGBoost also gives the **highest accuracy** as we will see in the next step and thus we decided to choose a dataset suited for the highest accuracy on the same.

Model Selection

Attached file: Model_Analysis_Parameter_Optimisation.ipynb

- As mentioned above cross validation required a very high computational power, so we shifted to using test train split to compare the accuracy of two models.
- The **seven** models tried were:
  1. Support Vector Machines (4 kernel types)
  2. Random Forests
  3. Linear Discriminant Analysis
  4. Quadratic Discriminant Analysis
  5. Boosting Classifier
  6. Logistic Regression
  7. XGBoost Model
- On the basis of the above accuracies as well as the fact that a decision tree regressor fits this data best since we have a large number of categorical variables and plotting scatter plots such as the one highlighted at the end of the Data Analysis Notebook (N) where we notice how the **labels are distributed heterogeneously** within each group of the categorical variables.
- This is the basis for choosing a decision tree regressor. Since the XGB has a higher accuracy than Random Forests, we choose that.
- Also we note that while the test set had a majority of Label 0s we now have a majority of label 1s. This is why we have chosen to consider the '**Precision**' and 'Accuracy' metrics since accuracy would be pretty high for a model which predicted all 0s as well but precision might be a more sobering measure since the number of 1s is higher in the predicted test set.

Parameter Optimisation

Attached file: Model_Analysis_Parameter_Optimisation.ipynb

- Since we have now chosen our required model, we work on experimenting with the parameters of the model. Since XGBoost allows us to calculate the cross validation score, we use that to compare whether any changes in the models are helping the **precision** of the model.

We have calculated the best model and attached the file called 'pred_y_XGB.csv' which contains the final predictions using the XGBoost.

Final precision on XGB using cross validation (CV=5) : 94.73%

Final accuracy on XGB using test-train split (80/20) : 98.65%

Neural Network Model

Attached file: MS4610_Project_Neural_Network.ipynb

Data Analysis, Imputing the Data

- We are using the KNN Algorithm for data imputation. Each sample's missing values are imputed using the mean value from 'n_neighbours' nearest neighbours (in our model, we have set 'n_neighbours' to 10) found in the training set.

- The basic logic is to consider two samples close, if the features that neither is missing are close. We are using **KNNImputer()**, a library function in the "impute" module of sklearn.

- For imputing the feature matrix(train_x), we first measure the **Euclidean distance** to find the closest neighbours(i.e. The data points similar to the one considered), and the missing values can be estimated by averaging the values of neighboring observations, corresponding to the missing value.

- We then use the completed feature matrix to impute and fill any missing labels.

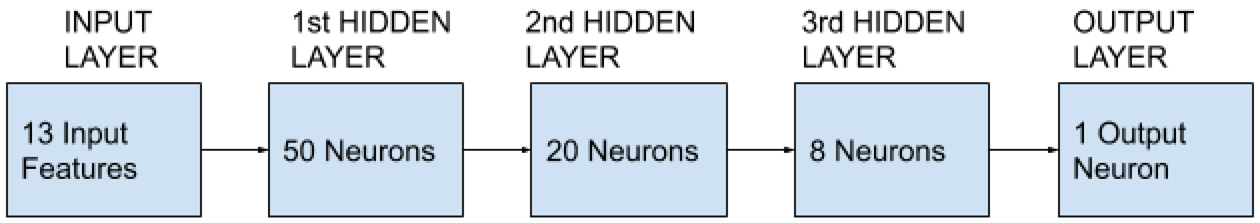
- Hence, we finally get a completed feature matrix and label vector, which can be used to train the model.

Model Definition and Training

- We are using Keras from Tensorflow for defining and training the model **(Sequential Neural Network).**

- We have divided the total dataset, 75% for training and 25% for testing the model. We used `train_test_split(),` a library function in sklearn, for this purpose. We are using 10% of the training data(75% of the total dataset) for cross validation.

- We tried many types of neural networks, by changing the number of hidden layers, and the number of neurons in each layer to find the best combination, based on the cross validation accuracy. Through repeated checking, we have concluded that a **3-layer neural network, with the first layer having 50 neurons, 2nd having 20, and the 3rd having 8 neurons, all layers densely connected,** works the best for the application.



- Since the dataset given is highly imbalanced (around 94% of the data is of label ‘Not Default’, i.e. label=0), a class weight is introduced. This will give a higher penalty for wrong classification of the minority class

- The model is trained using the training data.

- Since we finally want a binary classification model, the loss has been defined to be **binary_crossentropy.**

Hyper-Parameter Tuning and Validation

- **We have tuned all the hyper parameters, by considering the cross validation and test data accuracy**(the 25% of the data that was taken from the total dataset for test). Using this concept, we have concluded that the following combination gives the best accuracy(both cross validation and test):
    - kernel_initializer = 'uniform’
    - optimizer="adam"
    - batch_size=10
    - validation_split=0.1
        - 10% of the training data

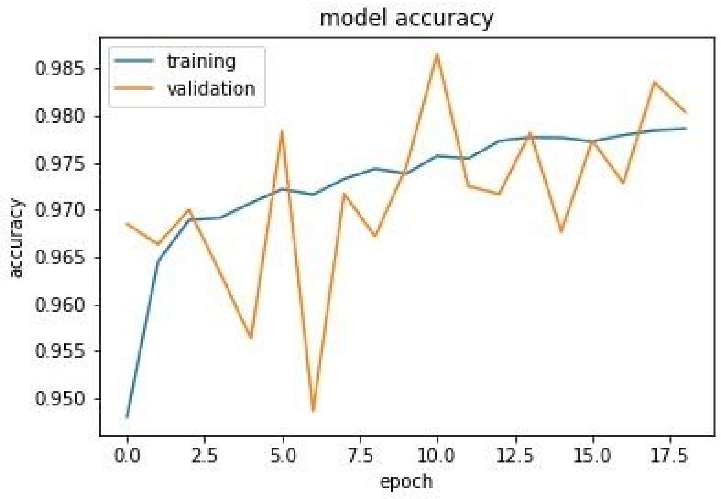
          (75% of the total dataset) for cross validation
    - activity_regularizer=l1 (0.005)
        - L1 regularization is similar to Lasso Regularization.
        - (0.005 is the regularization parameter)

    - The activation function for all 3 layers is ‘ReLu’
    - The activation function for the output layer has to be ‘sigmoid’ since it is a binary classifier.
    - A Dropout layer with dropout rate = 0.1 has been used.
    - Early Stopping was also used so that the training is time efficient.

- **Finally, after the fine tuning, the model reached the highest accuracy of 98.59%. The predictions for x_test (the data given, for which we need to predict the labels) have been attached in the mail as “pred_y_Neural_Network.csv”.**

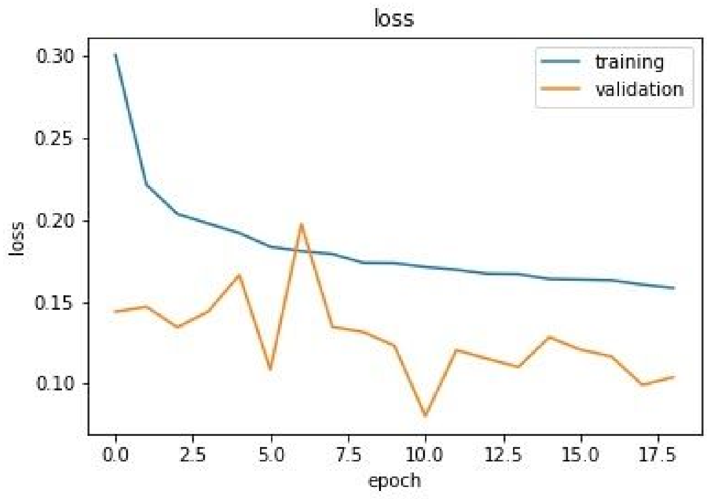
➔ **Accuracy and Validation Accuracy, plotted against the number of epochs**

(For the highest accuracy of 98.59%)



➔ **Loss and Validation Loss, plotted against the number of epochs:**

(For the highest accuracy of 98.59%)


loss
training
validation
loss
epoch