LOAN DEFAULT PREDICTION USING NEURAL NETWORKS

By-

Mrunmay Angaitkar, BTech Year III,

Department of Mechanical and Industrial Engineering,

IIT Roorkee.

Contents

1. **Abstract**

Banks earn a major part of their revenue by lending loans. But, with lending comes a great financial risk due to the possibility of borrower defaulting on loan. The traditional methods of mitigating this issue involved setting up a panel of experts who would analyze the data concerning applicant's employment records, annual income, interest rate, credit history, etc. However, with the increase in computational power and the recent advancements in machine learning, it has been a topic of interest whether ML algorithms can perform better classification. Our aim is to analyze a dataset pertaining to loan borrowers in the past and build a strong ML model to classify if any borrower is likely to default or not. Firstly, the data has been preprocessed and examined to identify the features that might be good predictors of defaulting. A fully connected deep neural network has been used for this purpose. The evaluation metric used for comparing is F1-score since defaulting is a greater risk than a missed loan.

2. **The Dataset**

The dataset consists of 5,32,428 entries with 45 columns inclusive of the 'loan status' column which is our target variable. The description of columns is as follows.

- **member_id** - unique ID allotted to each member in the dataset
- **loan_amnt** - Loan Amount ($) Applied by the Member
- **funded_amnt** - Loan Amount ($) Sanctioned by the Bank
- **funded_amnt_invt** - Loan Amount ($) Sanctioned by the Investors
- **term** - Term of Loan Taken or Availed (in Months)
- **batch_enrolled** - Batch Numbers Allotted to Members
- **int_rate** - Interest Rate (%) charged by the Bank on Loan
- **grade** - Grade Assigned by the Bank
- **sub_grade** - Grade Assigned by the Bank
- **emp_title** - Job / Employer Title of Member
- **emp_length** - Employment Length where 0 means Less than One Year & 10 means ten or more years
- **home_ownership** - Status of Home Ownership
- **annual_inc** - Annual Income ($) stated by the Member
- **verification_status** - Status of Income Verified by the Bank
- **pymnt_plan** - Any Payment Plan that has started against Loan
- **desc** - Loan Description given by the Member
- **purpose** - Purpose of the Loan/Reason of Taking Loan
- **title** - Loan Title Provided by the Member
- **addr_state** - State of Residence of the Member
- **dti** - Ratio of member's Total Monthly Debt repayment excluding Mortgage Divided by Self-Reported Monthly Income
- **delinq_2yrs** - Number of 30+ Days Delinquency (Non-Payment) in Past 2 Years
- **inq_last_6mths** - Count of Inquiries in Last 6 Months
- **mths_since_last_delinq** - Number of Months since Last Delinquency
- **mths_since_last_record** - Number of Months since Last Public Record
- **open_acc** - Number of Open Credit Line in Member's Credit Line (Search Line of Credit on Google for More)
- **pub_rec** - Number of Derogatory Public Records
- **revol_bal** - Total Credit Revolving Balance
- **revol_util** - Credit Amount used by a member in relation to revol_bal
- **total_acc** - Total No of Credit Lines Available in Members Credit Line
- **initial_list_status** - Unique Listing Loan Status - W(Waiting), F(Forwarded)
- **total_rec_int** - Interest Received Till Date
- **total_rec_late_fee** -Late Fee Received Till Date
- **recoveries** -Post Charge off(Settlement) Gross Recovery
- **collection_recovery_fee** - Post Charge off (Settlement) Recovery Fee
- **collections_12_mths_ex_med** - No of Collections in last 12 months excluding Medical Collections
- **mths_since_last_major_derog** - Months since most recent 90 day or Worse Rating
- **application_type** - Indicates when the Member is an Individual or Joint
- **verification_status_joint** - Indicates if the Joint Members Income Verified by the Bank
- **last_week_pay** - Indicates how long (in weeks) a member has paid EMI after batch enrolled
- **verification_status_joint** - Number of Accounts (Case of Multiple Loan Accounts) on which the member is Delinquent (Unable to Pay or Not Paying)
- **tot_coll_amt** - Total Collection Amount ever Owed
- **tot_cur_bal** - Total Current Balance of all Accounts
- **total_rev_hi_lim** - Total Revolving Credit Limit
- **loan_status** - Status of Loan Amount, 1 = Defaulter, 0 = non-Defaulters

The dataset has been taken from Kaggle data repositories, link for the same has been provided in the references.

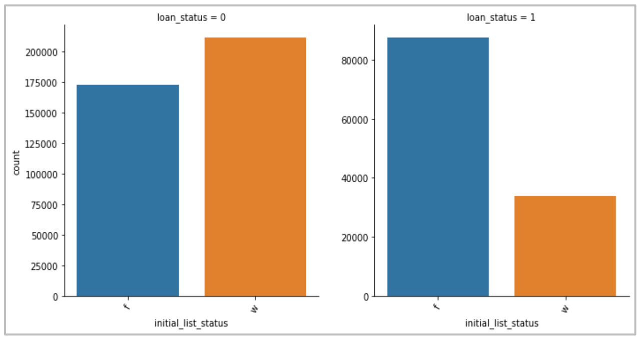
3. **Data Cleaning and Insights**

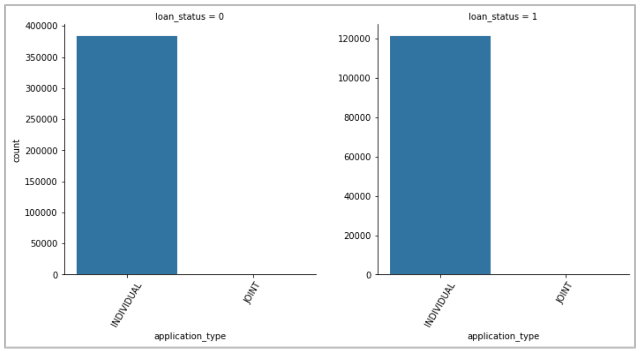
Firstly, columns that are irrelevant to the problem statement and hardly hold any significance in our analysis have been removed. These are 'batch_enrolled', 'member_id', 'zip_code' and 'addr_state'.

	missing_value_count
desc	456829
verification_status_joint	532123
mths_since_last_major_derog	399448
mths_since_last_delinq	272554
mths_since_last_record	450305

Moreover, the columns 'mths_since_last_record', 'mths_since_last_major_derog', 'mths_since_last_delinq' and 'verification_status_joint' have missing values for more than 50% of the records and thus contain very less information. So, they have been removed as well.

Now, four features namely 'application_type', 'initial_list_status', 'term' and 'pymnt_plan' were identified as binary (having only two unique entries) Its frequency plot for both target categories suggested that 'pymnt_plan' and 'application_type' hardly show any different values with respect to target categories (evident from the graphs below). Hence, they have been removed. The remaining two namely 'initial_list_status' and term have been encoded with labels '0' and '1'.





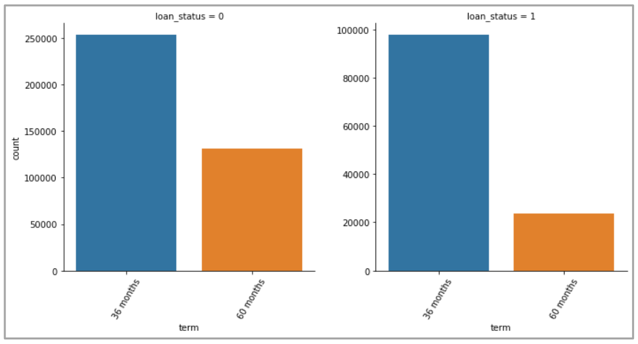
The column 'grade' and 'subgrade' inform the category assigned to each applicant by the bank based on their credit score. The categories are 'A', 'B', 'C', 'D', 'E', 'F' and 'G' with subcategories as 'A1', 'A2', …., 'A5', 'B1' and so on. To handle these, we came up with an encoding where the 'grade' will determine the integer part of the code whereas the 'subgrade' the decimal part. Here we have assuming the categories have certain hierarchy. (A<B<C<D<E<F<G and A1<A2<A3<A4<A5). A will be assigned 1, B assigned 2 and so on. A1 will be assigned 1.0, A2: 1.2, A3: 1.4, A4: 1.6 and A5 : 1.8 and then B1 : 2.0 and so on.

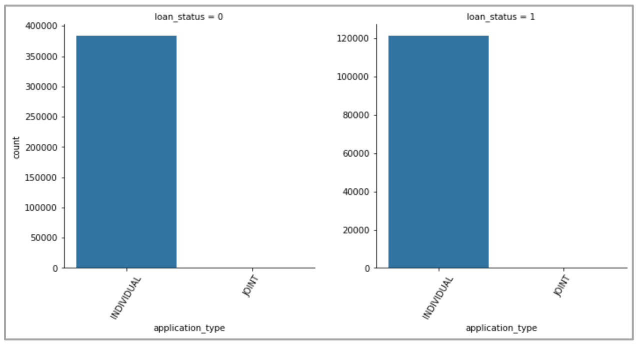
The column 'emp_title' is also very important but it contains text data with nearly 1,90,000 unique entries. So, it was inconvenient to convert this to numerical data. Also, we assumed that this factor was considered while assigning the grades. So, this column was dropped.

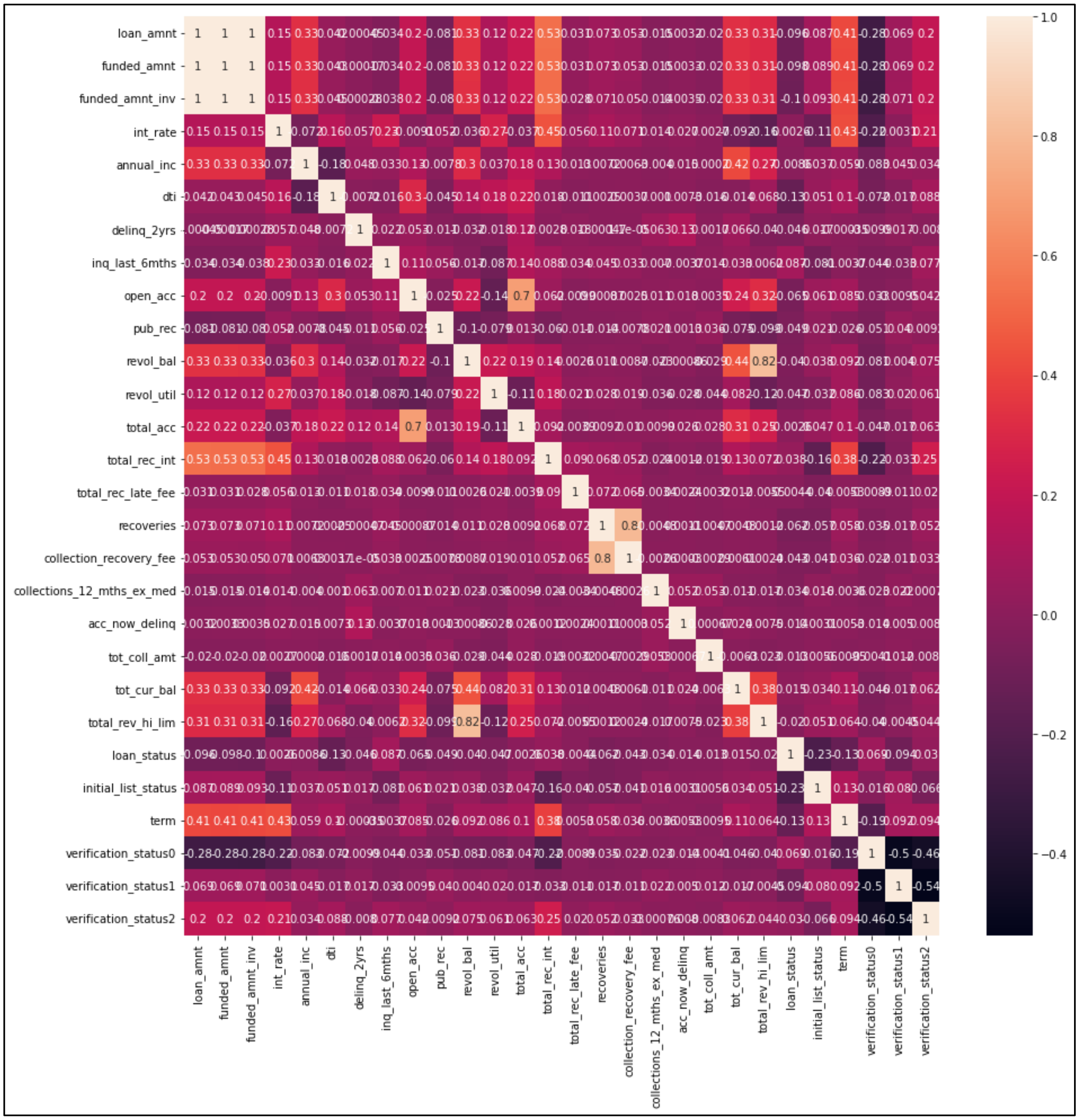
The column 'title' also had a similar problem with around 35000 unique entries. This was also dropped.

From the heatmap plotted with Pearson Correlation coefficient for all numeric columns, we came to know that the columns 'funded_amnt_inv', 'funded_amnt', 'loan_amnt' have very strong positive correlation. Although they have different meanings, they provide redundant information to our ML model and so, the features 'funded_amnt_inv' and 'loan_amnt' were removed and only 'funded_amnt' was kept.

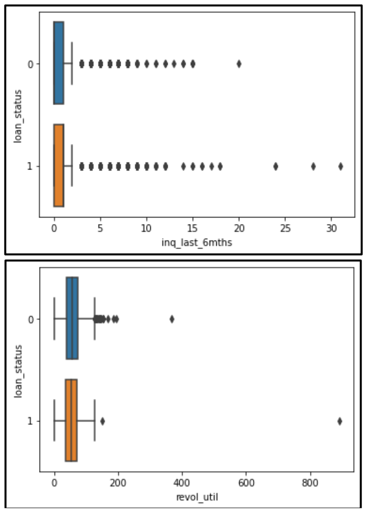
For numeric features, box plotted were plotted and among them, we saw that 'revol_util' and 'inq_last_6mths' show almost same variation irrespective of the target category. Thus, they are not good predictors of default. They have been dropped.



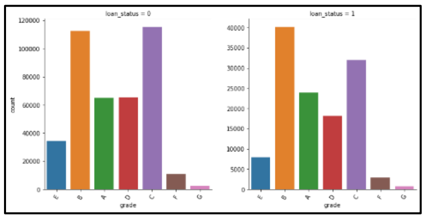




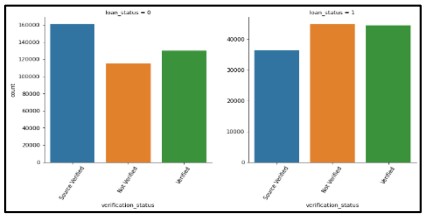
Heatmap showing correlation of features



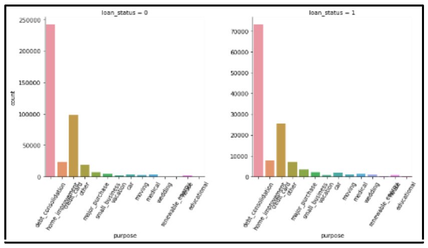
The box plots for 'inq_last_6mths' and 'revol_util'



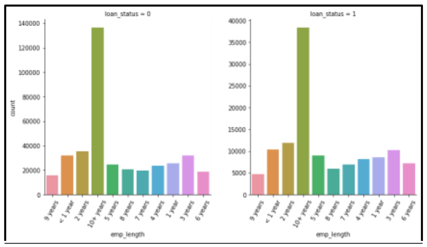
Histogram for 'grade' for loan_status 0 and 1



Histogram for 'verification_status' for loan_status 0 and 1



Histogram for 'purpose' for loan_status 0 and 1



Histogram for 'emp_length' for loan_status 0 and 1

The columns ‘purpose’, ‘emp_length’, ‘verification_status’ and ‘home_ownership’ show more than 2 unique entries and can’t be encoded with 0 and 1. Here from the histogram plots we observe that ‘emp_length’, ‘purpose’ and ‘home_ownership’ have same distribution for both 0s (Non-defaulters) and 1s (Defaulters). So, they are not good predictors of defaulting and have been dropped. However, the other column ‘verification_status’ has been encoded using One-Hot-Encoding. To handle NaN values in these columns, a separate ‘others’ category has been added.

The missing values have been handled by filling them with the mean values of the respective columns.

Now the dataset has been cleaned and ready for ML models to be applied. It has 59 features and the same number of records (rows) as earlier. (We avoided deleting rows owing to the possibility of losing important information.

4. Splitting and Scaling

4.1 Splitting the data.

The data has been split into 3 sets, viz., Training set (70% records), Cross-validation set (15% records) and Test Set (15% records). The model will training set and one cross validation set was used to continuously monitor the performance of model on unseen data. The cross-validation accuracy was also used to tweak the hyperparameters of the model whenever required like learning rate, optimizers, etc.

4.2 Standardising the data

The numeric features of all 3 sets of data were scaled using scikit-learn StandardScaler so that all columns may have unit variance and zero mean.

5. The neural network hyperparameters.

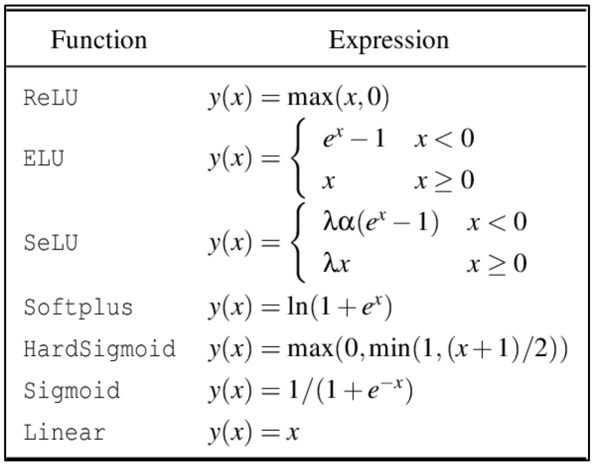
Training a neural network needs a lot of things to be decided, viz., the number of hidden layers, the size of each hidden layer, the activation functions, the optimizer, the loss functions and most importantly the learning rate. If not properly tuned, the model may fail to perform at its best.

5.1 Hidden layers and their sizes

During our analysis, we came across the fact that an ANN with just one hidden layer can theoretically model even the most complex functions provided it has enough neurons. However deep networks are observed to perform much better than shallow networks and the number of neurons per layer required is exponentially reduced. However too much hidden layers and too many neurons might lead to the problem of overfitting and neural networks are prone to overfitting. So, we decided to keep 8 hidden layers (to make the network deep) however keep number of neurons per layer to 50.

5.2 Activation Functions

For the output layer neurons, we used Softmax activation because our output layer has 2 neurons, and their outputs would be probabilities and we want their sum to be equal to 1. For the hidden layer neurons, various options are available, viz., ‘tanh’, ‘relu’, ‘selu’, ‘sigmoid’, etc.



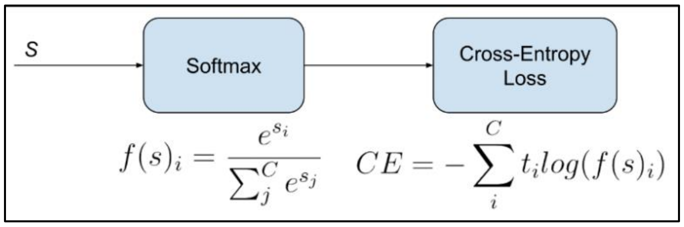
Function	Expression
ReLU	$y(x) = \max(x, 0)$
ELU	$y(x) = \begin{cases} e^x - 1 & x < 0 \\ x & x \geq 0 \end{cases}$
SeLU	$y(x) = \begin{cases} \lambda\alpha(e^x - 1) & x < 0 \\ \lambda x & x \geq 0 \end{cases}$
Softplus	$y(x) = \ln(1 + e^x)$
HardSigmoid	$y(x) = \max(0, \min(1, (x+1)/2))$
Sigmoid	$y(x) = 1/(1 + e^{-x})$
Linear	$y(x) = x$

Source: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.semanticscholar.org%2Fpaper%2FLearning-Task-specific-Activation-Functions-using-Basirat-

We will use SELU (Scaled Exponential Linear Unit) due to its self-normalizing property which helps to avoid the vanishing /exploding gradients problem.

5.3 The Loss function and Optimizer

The loss function chosen is the standard sparse-categorical cross-entropy loss used for classification tasks when your input is encoded with integers. It is different from its parent ‘categorical-crossentropy’ which is used when the output is one-hot encoded. However, since it is a binary classification both integer encoding and one-hot encoding are the same.



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

For a binary classification, the expression of sparse categorical cross entropy and categorical cross entropy are same. Also known as ‘Softmax Loss’, it is softmax function combined with cross-entropy loss.

To get to the optimal weights and biases, we use the Adam optimizer. Adam which stands for adaptive moment estimation combines the ideas of momentum optimization and RMSProp. Further discussion about Adam optimizer is beyond our scope.

Learning rate is one of the most important hyperparameter. Set it too high and the algorithm will start diverging and never arrive at a solution. However, keeping it too low will increase the computational cost of training. The default learning rate of 0.001 for Adam optimizer seemed to be working well for our model after trying out different learning rates.

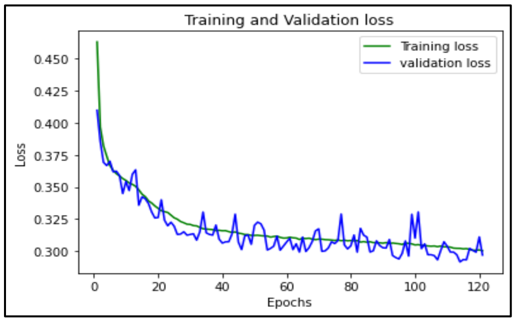
5.4 Kernel initializer

Kernel Initializer refers to the method by which the weights and biases are initialized for first iteration. Depending on where the deep learning model starts in the training process, it can converge to any of the possible local minima in the irregular loss surface. You might think that initializing all weights to zero or some random initialization should be a good starting point, but as we'll find out, the more we think about this, the more interesting the problem becomes. Mishkin and Matas (2015) showed that arbitrary initialization can slow down or even stall the training.

In LeCun-Normal Initialization we initialise our weights and biases from a Gaussian distribution, $N(0, \frac{1}{n_{in}})$, where n_{in} is the number of input features. The aim is for the activation output to have the same variance as the input features. It is usually used when the activation function is 'SELU'.

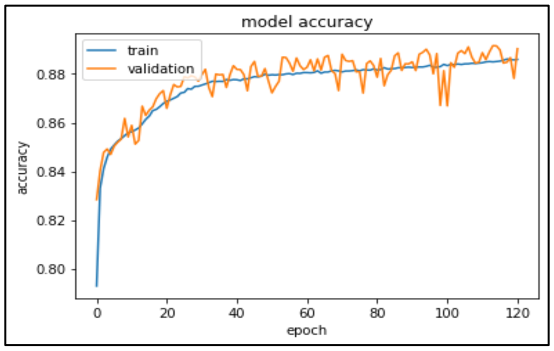
6. Results

The number of epochs is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset. The model was trained for 121 epochs with a batch size of 45. The batch size is the number of samples after which the model updates its parameters. The cross-validation loss and training loss was plotted against the number of epochs to get the following results.



The plot of training set loss and validation set loss against the number of epochs

The curve flattens out as the number of epochs increase suggesting proximity to global minima. Similarly, the model's accuracy over training and cross validation data was plotted to yield the following graph.



The plot of training set accuracy and validation set accuracy against the number of epochs

The training set accuracy was found to be 88.6%. The confusion matrix for cross-validation set along with the classification report has been presented below.

	0 (Predicted)	1(Predicted)
0(Actual)	57754	3236
1(Actual)	5518	13356

Cross-Validation confusion matrix

```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93     60990
           1       0.80      0.71      0.75     18874

    accuracy                           0.89     79864
   macro avg       0.86      0.83      0.84     79864
weighted avg       0.89      0.89      0.89     79864
```

Cross-Validation classification report

The cross-validation set was used for hyperparameter tuning and continuously tracking the performance of the network on unseen data. The cross-validation accuracy was found to be 89.04% whereas the F1-score was found to be 0.75 for the positive defaulter class which is very important since **predicting a defaulter as a non-defaulter (Type2 error) is a greater risk for the bank than missed loan (Type1 error).**

Now we can report the performance of the model on totally unseen data, i.e., our test-set. The test-set has 79865 records. The following results were obtained.

	0 (Predicted)	1(Predicted)
0(Actual)	57855	3136
1(Actual)	5476	13398

Test-set confusion matrix

```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93     60991
           1       0.81      0.71      0.76     18874

    accuracy                           0.89     79865
   macro avg       0.86      0.83      0.84     79865
weighted avg       0.89      0.89      0.89     79865
```

Test-Set classification report

The accuracy over test-set was found to be 89.22%. The model yielded satisfactory results over both seen and unseen data.

7. Acknowledgements

This research was supported by Department of Management Studies, IIT Roorkee. We thank our colleagues who provided insight and expertise that greatly assisted the research.

We would also like to thank Prof. Sumit Kumar Yadav for his guidance and comments that greatly improved the project.

8. References

- Link to dataset: Bank Fears Loanliness | Kaggle
- Default: What It Means, What Happens When You Default, Examples (investopedia.com)
- What Does it Mean to Default on a Loan? What Happens When You Default? - ValuePenguin
- ML | One Hot Encoding to treat Categorical data parameters - GeeksforGeeks
- Why One-Hot Encode Data in Machine Learning? - MachineLearningMastery.com
- seaborn.heatmap — seaborn 0.12.1 documentation (pydata.org)
- Pearson correlation coefficient - Wikipedia
- tf.keras.callbacks.History | TensorFlow v2.11.0
- tf.keras.metrics.sparse_categorical_crossentropy | TensorFlow v2.11.0
- tf.keras.optimizers.Adam | TensorFlow v2.11.0
- Guide To Tensorflow Keras Optimizers (analyticsindiamag.com)
- Cross-Entropy Loss Function. A loss function used in most… | by Kiprono Elijah Koech | Towards Data Science
- https://www.tensorflow.org/api_docs/python/tf/keras/initializers/LecunNormal
- A Gentle Introduction To Weight Initialization for Neural Networks – Weights & Biases (wandb.ai)
- [1511.06422] All you need is a good init (arxiv.org)
- Display Deep Learning Model Training History in Keras - MachineLearningMastery.com
- Aurelien Geron, Hands-On Machine Learning with Scikit-Learn and Tensorflow, 2nd Edition.