Deep Learning (Peer Kart)

1. Introduction

Credit default data is extracted to work on the fraud model. The loans that did not pay a single EMI are marked as fraud. These are intentional defaults. There are 9,578 observations out of which 1,533 are fraud (16%) and 8,045 are fraud (84%) are not fraud.

- **Credit policy:** 1 if the customer meets the credit underwriting criteria and 0 otherwise
- **Purpose:** The purpose of the loan
- **Instalment:** The monthly installments owed by the borrower if the loan is funded
- **Log annual inc:** The natural log of the self-reported annual income of the borrower
- **FICO:** The FICO credit score of the borrower
- **Days with cr line:** The number of days the borrower has had a credit line
- **Revol bal:** The borrower's revolving balance (amount unpaid at the end of the billing cycle)
- **Revol util:** The borrower's revolving line utilization rate (the amount of the credit line used relative to total credit available)
- **Inq last 6mths:** The borrower's number of inquiries by creditors in the last 6 months
- **Pub rec:** The borrower's number of derogatory public records (bankruptcy filings, tax liens, or judgments)

Keras is an open-source library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

- **Activation:** The activation function of a node in an artificial neural network is a function that calculates the output of the node
- **Optimizer:** optimizers are algorithms that adjust the model's parameters during training to minimize a loss function.
    - **Stochastic Gradient Descent:** Using gradient descent on large data might not be the best option. Stochastic gradient descent solves the problem by randomly selecting the batches of data.
    - **Adam:** Adaptive Moment Estimation optimizer is an extension of the SGD algorithm and is designed to update the weights of a neural network during training.
- **Regularization:** weights are more important in neural networks. Hence regularization of weights is done using L2 (Ridge – forces weights to be closer to zero).
- **Drop out:** randomly removes nodes from the neural network during training.
- **Early stopping:** determines when the network is starting to overfit and stops training the model before this happens
- **Epochs:** The number of epochs is the number of complete passes through the training dataset.
- **Batch size:** The batch size is the number of samples processed before the model is updated.

Dataset: loan_borowwer_data.csv

2. Hyper Parameter Tuning

There are two parts to hyperparameter tuning. In the first part, the number of layers and number of nodes are decided. In the second part activation, optimizer, regularization, dropout, epochs, and batch size are decided.

K Fold Cross validation – The dataset is divided into 3 parts. The model is trained and tested 3 times and the average AUC is reported. For K = 1, the model is trained on the first & second sets and tested on the third set. For K = 2, the model is trained on the first & third sets and tested on the second set. For K = 3, the model is trained on the second & third sets and tested on the first set.

Set	First	Second	Third
Y = 1	First 500 observations	Between 501 and 1,000 observations	Between 1,001 and 1,533 observations
Y = 0	First 3,000 observations	Between 3001 and 6000 observations	Between 6,001 and 8,045 observations

First part – Option 2 (2 layers) is selected as the AUC is much higher than Option 3 (1 layer) and only slightly lower than Option 1 (3 layers).

Inputs	Option 1	Option 2	Option 3
number of layers	3	2	1
number of nodes	10 per layer	10 per layer	10 per layer
activation	'relu', 'tanh', 'sigmoid'	'relu', 'tanh', 'sigmoid'	'relu', 'tanh', 'sigmoid'
optimizer	'adam', 'sgd'	'adam', 'sgd'	'adam', 'sgd'
regularization	0.001, 0.1	0.001, 0.1	0.001, 0.1
dropout	0.001, 0.1	0.001, 0.1	0.001, 0.1
epochs	100, 1000	100, 1000	100, 1000
batch size	100, 1000	100, 1000	100, 1000

Outputs	Option 1	Option 2	Option 3
activation	'tanh'	'tanh'	'sigmoid'
optimizer	'adam'	'adam'	'adam'
regularization	0.001	0.001	0.001
dropout	0.001	0.001	0.001
epochs	100	1000	1000
batch size	100	100	100
AUC	**0.659**	**0.657**	**0.641**

Second part – Each iteration is decided based on the optimal selection in the previous iteration. Hyperparameters identified in iteration 3 are used in the final model.

Inputs	Iteration 1	Iteration 2	Iteration 3
number of layers	2	2	2
number of nodes	10 per layer	10 per layer	10 per layer
activation	'tanh'	'tanh'	'tanh'
optimizer	'adam'	'adam'	'adam'
regularization	0.0001,0.001, 0.01	0.001	0.001
dropout	0.0001,0.001, 0.01	0.0001,0.0005,0.001	0.0005
epochs	500, 1000, 5000	500, 1000, 5000	100,200,500,700,1000
batch size	50, 100, 500	50, 100, 500	50,100,200

Outputs	Iteration 1	Iteration 2	Iteration 3
regularization	0.001	0.001	0.001
dropout	0.0001	0.0005	0.0005
epochs	500	1000	500
batch size	50	50	100
AUC	**0.659**	**0.659**	**0.659**

3. Model Evaluation

```
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.callbacks import EarlyStopping
from keras.regularizers import l2
from sklearn.metrics import roc_auc_score, confusion_matrix
```

```
model = Sequential()
model.add(Dense(10, activation='tanh', input_dim=10, kernel_regularizer=l2(0.001)))
model.add(Dropout(0.0005))
model.add(Dense(10, activation='tanh', kernel_regularizer=l2(0.001)))
model.add(Dropout(0.0005))
model.add(Dense(1, activation='sigmoid', kernel_regularizer=l2(0.001)))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics='AUC')
model.fit(x,y,epochs=500,batch_size=100,verbose=0,callbacks=EarlyStopping(monitor='loss',patience=3))
```

```
pred_values = model.predict(x)
pred_values = pd.DataFrame(pred_values)[0]
print('AUROC:',np.round(roc_auc_score(y, pred_values), 3))
```

AUC or ROC curve measures the proportion of true positives versus the proportion of false positives. **The model has an AUC of 0.67**

Accuracy measures the number of correct predictions made divided by the total number of predictions made. **The model has an accuracy of 66.5%**

The F1 Score is calculated as the harmonic mean of the precision and recall scores. **The model has an F1 Score of 0.349.**