

Form4: Results and Conclusion

1.Team: 09

2. Project Title: Loan Approval Estimation Deploying Deep Learning and Blockchain Technologies

3. Experiment Environment:

Deep Learning Model:

Dataset: Here we are using the Kaggle dataset "credit_risk_dataset" for your deep learning model and working with libraries like NumPy, Pandas, Matplotlib, ScikitLearn, TensorFlow, and Pickle. Here we use various tools and technologies, from text editors like Visual Studio Code to web extensions like MetaMask.

Application:

Text Editors/IDEs: Visual Studio Code, RemixIDE

Package Managers: npm (Node Package Manager), pip (Python Package Installer)

Web Extensions: MetaMask

Frontend Libraries and Frameworks: React.js, Ethers.js

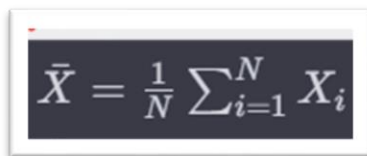
Backend Frameworks: Flask

Development Language: JavaScript, Python, Solidity

Testnet: Sepolia Testnet

Formulas used:

Mean:

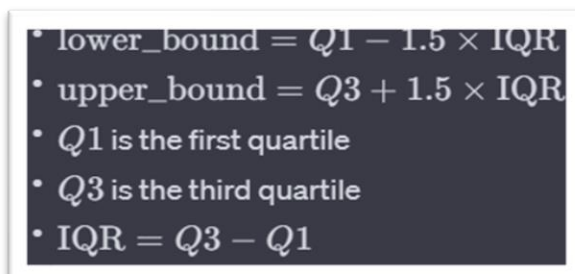

$$\bar{X} = \frac{1}{N} \sum_{i=1}^N X_i$$

\bar{X} = Mean

N = Number of values in the dataset

X_i = Input value

Inter Quartile Range:



- $\text{lower_bound} = Q1 - 1.5 \times \text{IQR}$
- $\text{upper_bound} = Q3 + 1.5 \times \text{IQR}$
- Q1 is the first quartile
- Q3 is the third quartile
- $\text{IQR} = Q3 - Q1$

Normalisation:

$$X_{\text{normalized}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

X = Original value you want to normalize

Xmin = minimum value in the dataset

Xmax = maximum value in the dataset

Xnormalised = normalised value

ReLU:

$$f(x) = \max(0, x)$$

f(x) = output value after function is performed

x = input value

Linear:

$$f(x) = x$$

f(x) = output value after function is performed

x = input value

Mean Absolute Error:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

n = Total number of samples or data points

y_i = Actual value for the ith sample

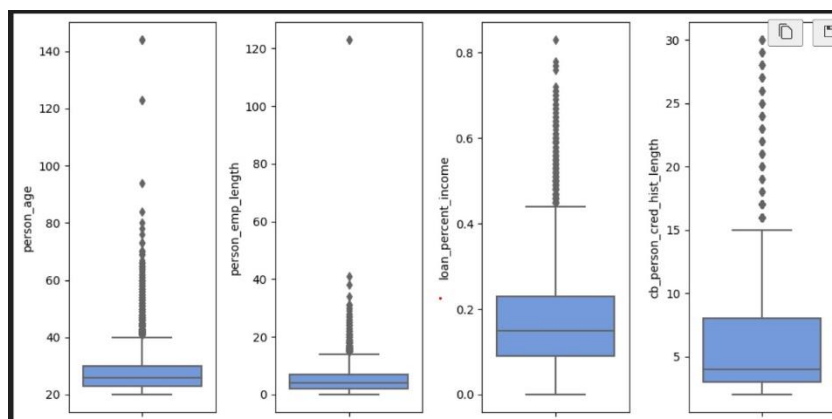
\hat{y}_i = Predicted value for the ith sample

4.Experiment:

Deep Learning Model

- ➔ Data collection: Kaggle dataset
- ➔ Preprocessing:
 - ➔ - Imputation: replace the null values with a mean of the column.
 - Handling outliers: IQR method
 - Transfer categorical to ordinal data
 - Convert required columns to integer datatype
 - Normalization of data
- ➔ Split into train and test dataset
- ➔ Utilize the sequential neural network architecture with 4 hidden layers for with activation function of ReLU with neurons 128, 256,256, and 256 respectively, linear activation for output.
- ➔ Train the model on train data sets uprisng historical known data including approved and denied cases.

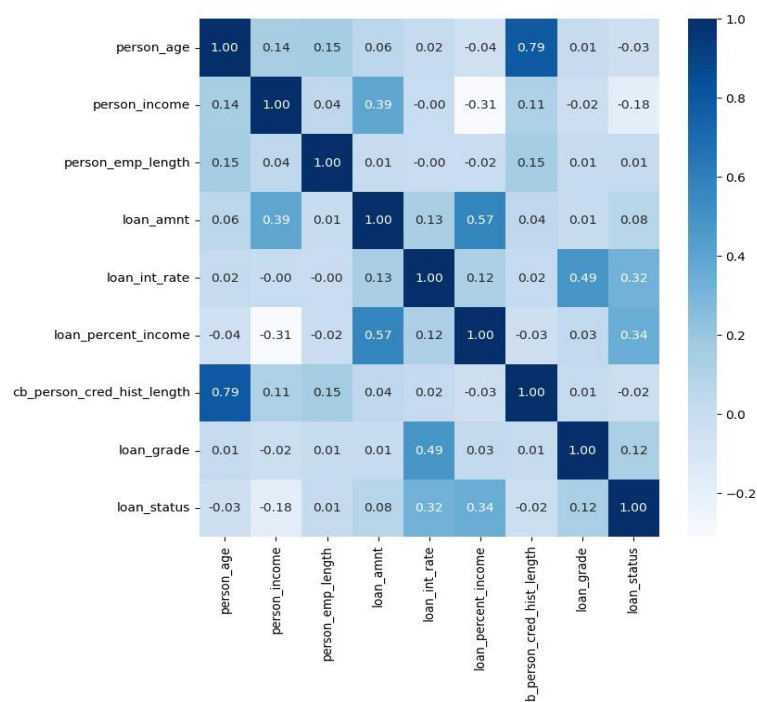
Analysis of outliers:



In addressing outliers within our dataset, we opted for the IQR (Interquartile Range) method. This method involves pinpointing the range that encapsulates the middle 50% of our data points. Any data points falling outside this range are flagged as potential outliers. By concentrating on the dispersion of the central portion of our data, we strive to mitigate the influence of extreme values that might distort our analysis. This approach ensures that our model is trained on a more accurate and dependable set of data, ultimately enhancing its reliability and performance.

Correlation matrix:

We also leveraged a correlation matrix as part of our analysis. A correlation matrix provides insights into the relationships between different variables in our dataset. By calculating correlation coefficients, we could determine the degree of association between pairs of variables. This step was crucial for identifying potential dependencies or multicollinearity among features. Understanding the correlation between variables allowed us to make informed decisions during the model-building process. For instance, highly correlated features might carry redundant information, and in such cases, we could consider excluding one of them to streamline the model and reduce complexity.



Feature Analysis and selection:

SNO	Number of features	Feature names	Accuracy
1.	2 features	loan_grade loan_percent_income	0.8375
2.	2 features	loan_percent_income loan_int_rate	0.8305
3.	3 features	loan_percent_income loan_int_rate loan_grade	0.8438
4.	4 features	loan_percent_income loan_int_rate loan_grade	0.8540

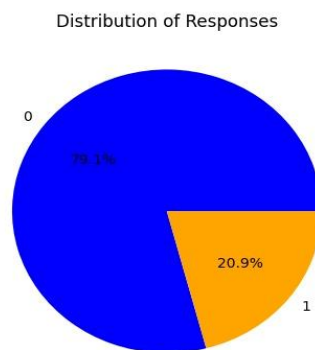
		loan_amnt	
5.	5 features	loan_percent_income loan_int_rate loan_grade loan_amnt person_emp_length	0.8576
6.	6 features	loan_percent_income loan_int_rate loan_grade loan_amnt person_emp_length cb_person_cred_hist_length	0.8569
7.	6 features	loan_percent_income loan_int_rate loan_grade loan_amnt person_emp_length person_age	0.8520
8.	7 features	loan_percent_income loan_int_rate loan_grade loan_amnt person_emp_length person_age cb_person_cred_hist_length	0.8523
9.	8 features	loan_percent_income loan_int_rate loan_grade loan_amnt person_emp_length person_age cb_person_cred_hist_length person_income	0.8537

Table: Analysis of feature selection

The model utilizing five features achieves the highest accuracy of 0.8576, the analysis reveals that the model employing eight features attains a commendable accuracy of 0.8537 while also incorporating all the requisite columns: **loan_percent_income**, **loan_int_rate**, **loan_grade**, **loan_amnt**, **person_emp_length**, **person_age**, **cb_person_cred_hist_length**, and **person_income**. Therefore, the model with eight features is selected as it balances accuracy with the inclusion of necessary features for a comprehensive analysis.

Pie chart of loan status:

The pie chart visually represents the distribution of responses, breaking them down into two parts. The larger portion, comprising 79.1%, is represented by the blue section, indicating a substantial majority of responses falling into this category. In contrast, the yellow section accounts for a smaller share, specifically 20.9%, highlighting its comparatively lesser representation in the overall responses. The chart clearly conveys the significant dominance of the blue category in the dataset.



Model Selection:

Our model architecture featured four hidden layers with ReLU activation functions, employing neuron counts of 128, 256, 256, and 256 respectively after experimenting with different layers. For the output layer, a linear activation function was utilized. Training encompassed historical data, including both approved and denied cases. Fine-tuning the hyperparameters was a meticulous process, involving rigorous experimentation aimed at optimizing accuracy.

Experiment with change in neuron layers

Number of Layers	Number of neurons in each layer	Accuracy
2 layers	128,256	0.8437
2 layers	256,256	0.8500
3 layers	128, 256, 256	0.8519
3 layers	265, 256, 256	0.8500
3 layers	265, 128, 256	0.8503
4 layers	128, 265, 256, 256	0.8553
4 layers	256, 256, 256, 256	0.8533
5 layers	128, 256,256,256,256	0.8499
5 layers	256,256,256,256,256	0.8550

Table : Analysis with different neuron layers

Experiments with changing Epoches

Number of Epoch	50	100	200	1000
Accuracy	0.8557	0.8495	0.8460	0.8249
Precision	0.6757	0.6675	0.6512	0.5880
F1	0.6012	0.5637	0.5736	0.5543
recall	0.6363	0.6112	0.6099	0.5706

Table : Analysis with different epoches

Experiment with changing learning rate and Epoches

Learning rate	Epoch=50	Epoch=100
0.1	0.8551	0.8519
0.01	0.8515	0.8537
0.001	0.8539	0.8499
0.0001	0.8498	0.8522

Table : Analysis with different learning rate and epoches

Experiment using different optimizers:

Stochastic Gradient Descent (SGD):

- The basic form of gradient descent where the model parameters are updated based on the gradient of the loss function with respect to the parameters.

Adams: (Adaptive Moment Estimation):

- An adaptive learning rate optimization algorithm that combines the benefits of both momentum and RMSprop

RMSprop (Root Mean Square Propagation):

- An optimization algorithm that adapts the learning rate for each parameter based on the average of recent squared gradients.

Adagrad (Adaptive Gradient Algorithm):

- An optimization algorithm that adapts the learning rates of each parameter based on the historical sum of squared gradients.

Nadam:

- An optimizer that incorporates Nesterov Accelerated Gradient (NAG) with the benefits of Adam.

Optimizers	Accuracy
Stochastic Gradient Descent (SGD):	0.7901
Adams: (Adaptive Moment Estimation):	0.8520
RMSprop (Root Mean Square Propagation):	0.8501
Adagrad (Adaptive Gradient Algorithm):	0.7901
Nadam	0.7901

Table : Analysis with different optimisers

This comprehensive approach aimed to develop a robust deep learning model capable of effectively handling the intricacies of the dataset and delivering improved predictive performance.

Smart Contract

We have deployed a smart contract named “dataStorage.sol” on the RemixIDE and copied the deployed address and ABI and connected with the application

Application

- ➔ The converted pickle file is deployed in Flask web framework
- ➔ The Frontend is connected to the Flask and predicted output is reflected on frontend
- ➔ Construct and deploy a smart contract for data storage
- ➔ Using the ether.js library we can connect to the blockchain environment
- ➔ When a user clicks on “save data” it connects to MetaMask and requests to implement a smart contract function
- ➔ Once the request is approved the information is successfully stored on the blockchain
- ➔ Currently the application is runned on the Sepolia test network.

Results:

Deep Learning Model

- ➔ The accuracy of the model is 0.8557855985181755
- ➔ The Precision of the model is 0.6848118279569892
- ➔ The recall of the model is 0.5620518477661335
- ➔ The f1-score of the model is 0.6173886700999698

Application:

- ➔ The model which we have created is successfully integrated with frontend
 - ➔ The Smart contract has been successfully deployed
 - ➔ The application and the blockchain environment are successfully Integrated
- Hence, we have an end-to-end application for loan approval estimation

5.Parameters and comparison Table:

Parameter	Previous method	Proposed method
End to end application	The integration of the frontend for the previous method would focus on streamlining traditional loan processes with features like user registration, form submission, status tracking, and customer support.	We will try to execute the incorporating advanced technologies such as deep learning and blockchain into the frontend, thereby enhancing the loan approval process with improved accuracy, efficiency, transparency, and data integrity."
Integration of block chain	Integration of was not used in the previous methods	We have successfully integrated blockchain technology into our loan approval process, ensuring enhanced security and transparency, while maintaining the immutability and integrity of all decisions and transactions. This implementation fosters trust among all parties involved by creating an impenetrable ledger for recording loan-related activities
Deep learning techniques	The previous methods in the loan approval process involve traditional approaches and mostly machine learning algorithms that suffer from challenges such as long wait timesand reliance on old systems. We can see that deep learning models give more performance than machine learning algorithms	We proposed deep learning model to enhance loan approval faster processing times and we used the sequential neural network architecture with 4 hidden layers with activation function of ReLU with neurons 128, 256,256, and 256 respectively, linear activation for output.

6. Final Conclusion

To build a model for predicting loan approvals, implement a secure data storage system using blockchain, and create a user-friendly interface for smooth interaction, right. It's all about using deep learning to predict loan approvals, blockchain for secure data storage, and a user-friendly interface for easy interaction. It's like making everything work together seamlessly

We are using a Kaggle dataset for data collection and implementing various preprocessing techniques like imputation, handling outliers, and normalization. how you're utilizing a sequential neural network architecture with ReLU activation and fine-tuning the hyperparameters to improve accuracy. It's all about using deep learning and blockchain to make those predictions as accurate as possible.

Signature Supervisor
Dr. G. Balakrishna