University of Missouri–St. Louis

Department of Computer Science

A Project Report on

Customer Churn Prediction

Submitted by:
Sachina Koirala
Student ID: 18281333

Submitted to:
Dr. Badri Adhikari

December 5, 2024

Contents

Abstract

Customer churn prediction is a critical challenge for the banking industry, as it directly affects customer retention and revenue generation. This project explores the application of machine learning models, particularly neural networks and logistic regression, to predict customer churn with a focus on maximizing predictive accuracy. The analysis is based on a Kaggle dataset comprising 175,028 records, with eight predictive features and a target variable indicating churn status.

The study systematically evaluated various neural network architectures to identify the optimal model while addressing challenges such as data imbalance, overfitting, and determining the appropriate level of model complexity. Key processes included data preprocessing, feature normalization, and iterative experimentation with multiple models. Techniques like feature importance analysis and dimensionality reduction were employed to enhance model efficiency and performance.

This report outlines the evaluation of different models used to predict customer churn using the TensorFlow library. The main goal in this phase was to maximize the accuracy on the validation set after splitting the data into training and validation sets. The following models were trained and evaluated:

1. **Random Baseline Classifier:** The model that always predicts the majority class.

2. **Logistic Regression Model:** A simple linear model used as the baseline for comparison with neural networks.

3. **Multi-layer Neural Networks:** Several neural network models with increasing complexity to evaluate how accuracy changes with more neurons and layers.

1 Introduction

“Churn” is the term used to describe when a customer stops using a certain organization’s services. Customer churn, specially in the banking industry, is a critical issue as it directly affects customer retention and revenue. Predicting churn involves analyzing multiple factors, such as customer demographics, account activities, product engagement, and financial behavior.

With an interest in applying Artificial Intelligence to real-world problems, I aim to explore how neural networks can be utilized to predict customer churn in banks. This project focuses on building a neural network model to identify customers at risk of leaving.The insights gained will be beneficial in designing personalized customer retention strategies.For example: Offer a gift voucher or any promotional pricing and lock them in for an additional year or two to extend their lifetime value to the company.

2 Phase 1 : Data Analysis and Preparation

The dataset was obtained from Kaggle website called the ”Binary Classification with Bank Churn Dataset”. The dataset consists of 175,028 rows (customers) and 9 columns (features), including both input features and the target column (Exited), which indicates whether the customer churned.

2.1 Dataset Description

Following are the columns in this dataset:

- CreditScore: Customer’s credit score, indicating financial reliability.
- Age: Customer’s age in years.
- Tenure: Number of years the customer has been with the bank.
- Balance: Customer’s account balance in monetary terms.
- NumOfProducts: Number of bank products used by the customer.
- HasCrCard: Whether the customer owns a credit card (1 = Yes, 0 = No).
- IsActiveMember: Whether the customer is an active bank member (1 = Yes, 0 = No).
- EstimatedSalary: Estimated annual salary of the customer.
- Exited: Target variable indicating if the customer churned (1 = Yes, 0 = No).

This dataset also includes both numerical and categorical features:

- Numerical Features: ’CreditScore’, ’Age’, ’Tenure’, ’Balance’, ’EstimatedSalary’
- Categorical Features: ’NumOfProducts’, ’HasCrCard’, ’IsActiveMember’

2.2 Input Data Visualization

The histogram plot of every input features showing their maximum and minimum value as well as how they are distributed can be seen in the figures below.

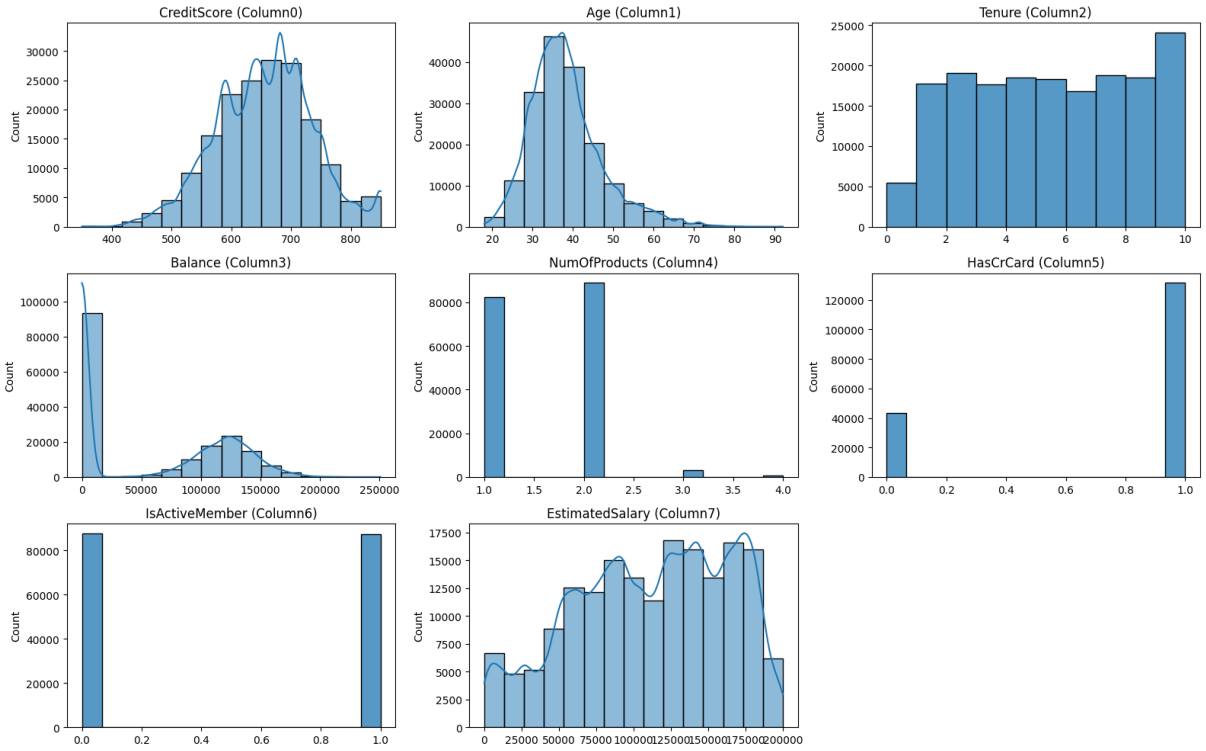

Figure 1: Input Data Distributions

Table 1: Statistics of Input Features

Feature	**Min**	**Max**	**Mean**	**Std**
CreditScore	350.00	850.00	656.11	81.15
Age	18.00	92.00	38.17	8.97
Tenure	0.00	10.00	5.02	2.81
Balance	0.00	250898.09	56676.77	62982.24
NumOfProducts	1.00	4.00	1.55	0.55
HasCrCard	0.00	1.00	0.75	0.43
IsActiveMember	0.00	1.00	0.50	0.50
EstimatedSalary	11.58	199992.48	111863.30	50814.97

2.3 Output Data Visualization

Following is the distribution of output labels:

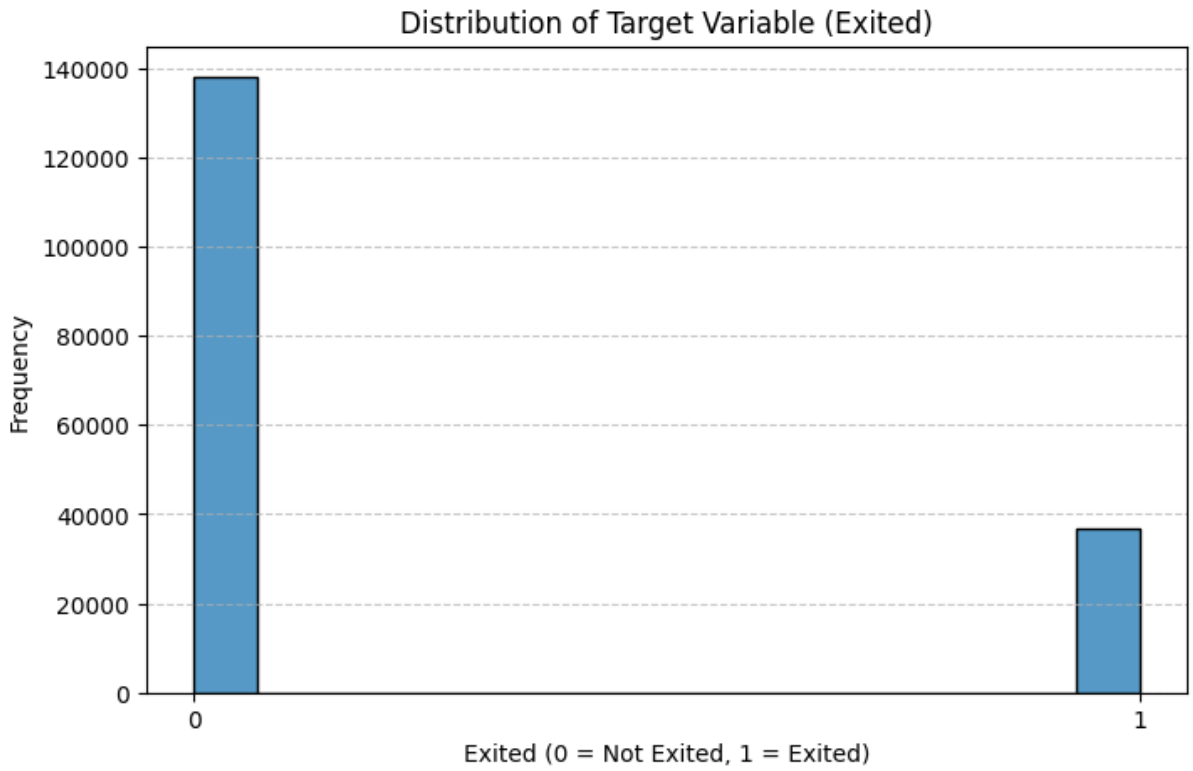

Figure 2: Output Data Distributions

2.4 Baseline Accuracy

The baseline accuracy is a useful metric that helps to evaluate the effectiveness of any machine learning model built to predict customer churn. For this project, the baseline accuracy is **78.89%** , which means that if we always predict that a customer will not churn (i.e., always predict 0), the model would be correct **78.89%** of the time.

When evaluating the model's performance, it's important to compare the model's accuracy with this baseline. If the accuracy of the model is only slightly higher than **78.89%** , it indicates that the model might not be effectively capturing the patterns in the data. In such cases, other metrics like precision, recall, and F1-score should also be considered, especially if the focus is on correctly predicting churned customers.

3 Phase 2 : Build a model to overfit the entire dataset

The goal of this phase was to experiment and find the network size required to overfit the entire dataset. In this phase, no data splitting was performed, and the entire dataset was used for training. The objective was to determine how big an architecture is needed to overfit the data completely, achieving close to 100% accuracy.

The starting point was a Logistic Regression model, which was trained for 128 epochs with a batch size of 512. Despite the extended training, the accuracy stabilized around **82.23%** and did not increase further, indicating that the model's single neuron was insufficient to fully capture the complexity of the data and overfit it.

To improve performance, the model was expanded to a multi-layer neural network with additional neurons. Multiple experiments were conducted using a larger architecture with layer sizes $[64, 32, 16, 8, 4, 2]$. This model was trained for 256 epochs with an early stopping mechanism, and although the model's complexity increased, the accuracy still stabilized at around 85.8%.

These results indicate that even with a larger architecture and extended training, the model could not fully overfit the data. This implies that the data may contain inherent noise or complexities, or that further tuning of the model's architecture and hyperparameters is required.

4 Phase 3: Model selection and evaluation

The main goal in this phase was to maximize the accuracy on the validation set after splitting the data into training and validation sets.

4.1 Data Splitting and Preparation

The data was split into a training set and a validation set, with the rows shuffled before splitting to ensure randomness. The primary metric used for comparison was accuracy on the validation set, and additional metrics such as precision, recall, and F1-score were also used to evaluate model performance.

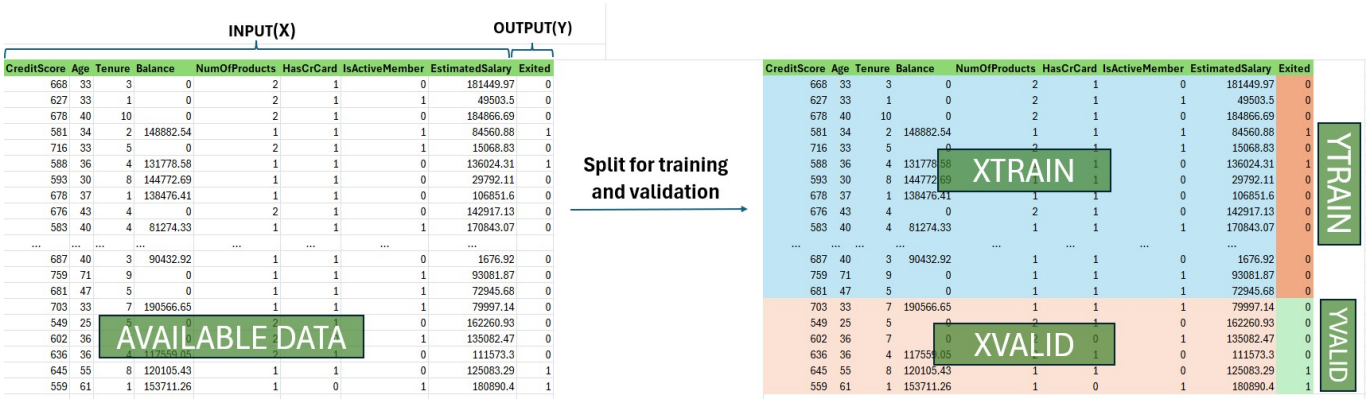

Figure 3: Data Splitting

35005 rows will goes as validation set and the remaining (which is 140023 rows) goes to the training.

The following figure shows the distribution of output variable of the training and validation data after splitting process.

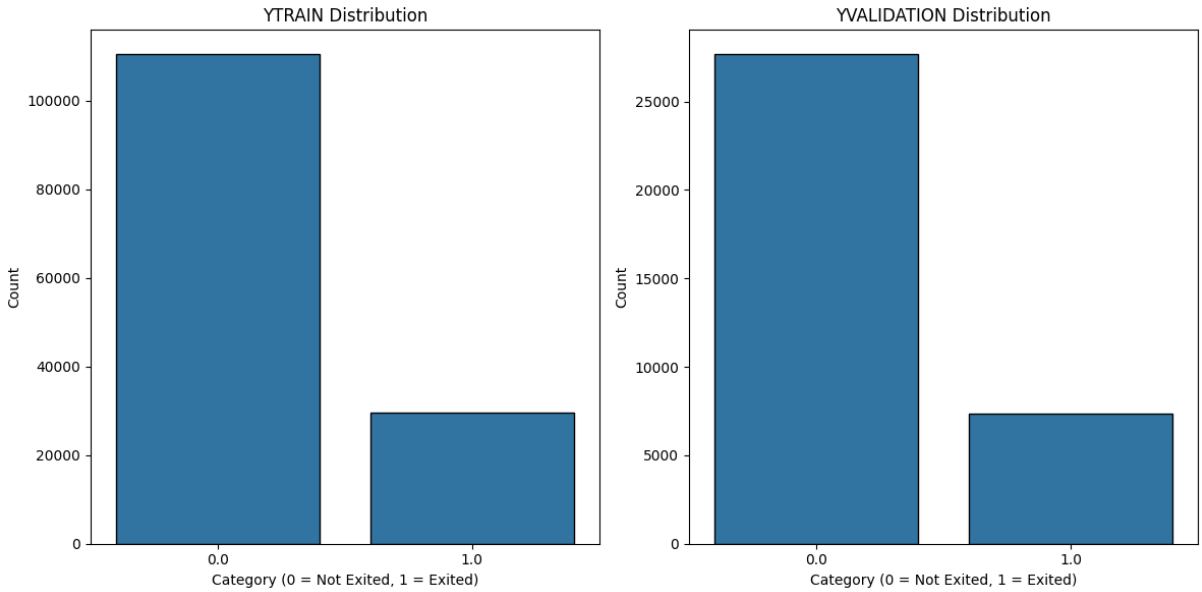

Figure 4: Data Distributions after splitting

4.2 Data Normalization

Normalization is done in this project to ensure that all input features have a similar scale. This helps the model to converge faster and prevents features with larger magnitudes from dominating the learning process.

In this project, the numerical features were normalized using the Z-score normalization technique. This transformation ensures that the numerical columns have a mean of 0 and a standard deviation of 1. The formula applied is:

$$X_{\text{normalized}} = \frac{X - \text{mean}(X)}{\text{std}(X)}$$

This method centers the data around zero with a standard deviation of one. The following features were normalized:

- CreditScore
- Age
- Tenure
- Balance
- NumOfProducts
- EstimatedSalary

The training set was normalized using its own mean and standard deviation, while the validation set was normalized using the same statistics from the training set.

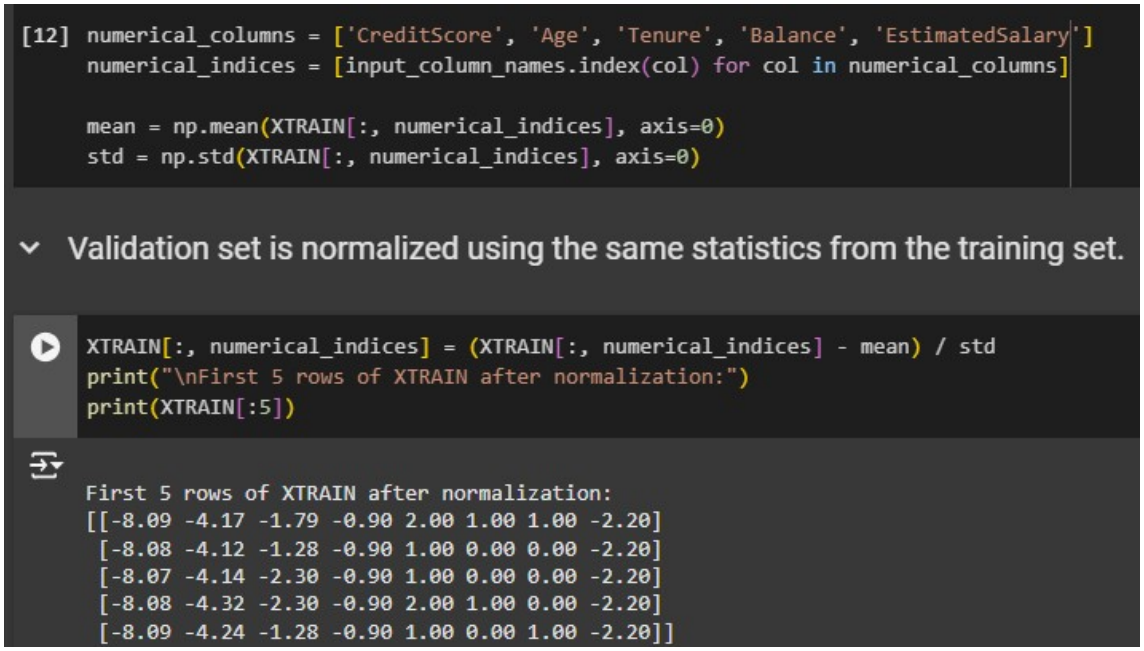

Figure 5: Code implementation and output of Normalization

4.3 Modeling

Different models were evaluated to predict customer churn using the TensorFlow library. The main goal in this phase was to maximize the accuracy on the validation set after splitting the data into training and validation sets. The following models were trained and evaluated:

1. **Random Baseline Classifier**: The model that always predicts the majority class.
2. **Logistic Regression Model**: A simple linear model used as the baseline for comparison with neural networks.
3. **Multi-layer Neural Networks**: Several neural network models with increasing complexity to evaluate how accuracy changes with more neurons and layers.

4.4 Logistic regression Model

A Logistic regression model for predicting customer churn was build first. The model was created using Keras's `Sequential` API, consisting of a single dense layer with one neuron and a sigmoid activation function, which is appropriate for binary classification. The training process stopped after 30 epochs as the validation loss did not improve further.

The table given below shows the performance of the logistic regression model.

Table 2: Model Performance Metrics

Training Accuracy (%)	Validation Accuracy (%)	Training Loss	Validation Loss
82.14%	82.76%	0.4166	0.4178

Additional Metrics

- **Precision**: 68.30%
- **Recall**: 37.56%
- **F1-Score**: 0.48

The given graph below shows the learning curve for the logistic regression model.

4.4.1 Learning Curve for Logistic Model

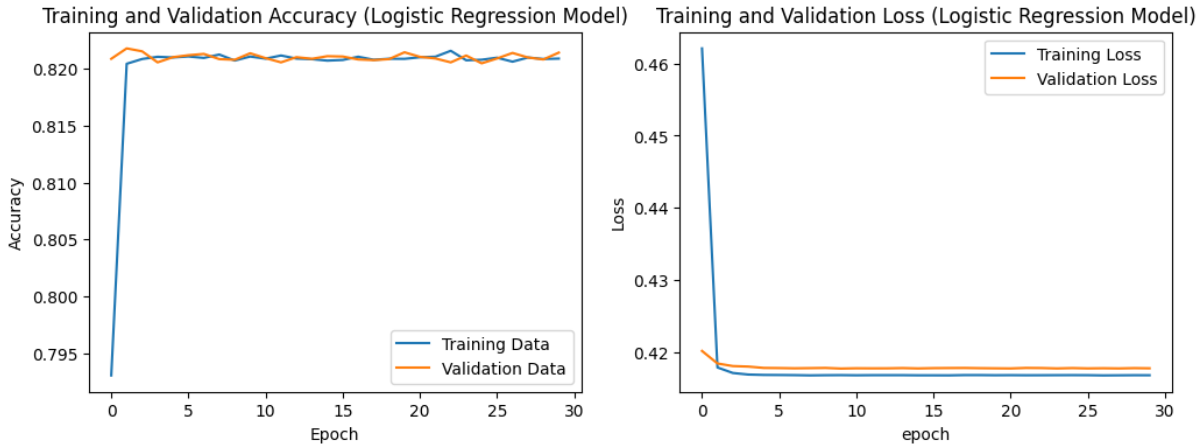

Figure 6: Comparison of training and validation metrics for Logistic Model

4.5 Varying Neural Network Architectures

A neural network model based on layers, known as ANN (Artificial Neural Network) is build. It looks a bit like the following diagram.

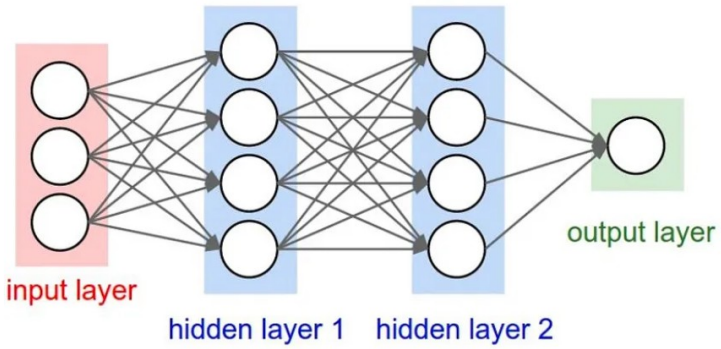

Figure 7: Neural Network architecture

4.6 Model 1 (2-1)

This model consisted of two layers: one hidden layer with 2 neurons and an output layer with 1 neuron. The hidden layer used the ReLU activation function to introduce

non-linearity, while the output layer used a sigmoid activation for binary classification.

The model used early stopping, which halted training at epoch 56, restoring the weights from the best epoch (epoch 36) to prevent overfitting.

The performance measure is shown as:

Table 3: Model Performance Metrics

Training Accuracy (%)	Validation Accuracy (%)	Training Loss	Validation Loss
84.65%	84.75%	0.3577	0.3565

4.6.1 Learning Curve for Model 1

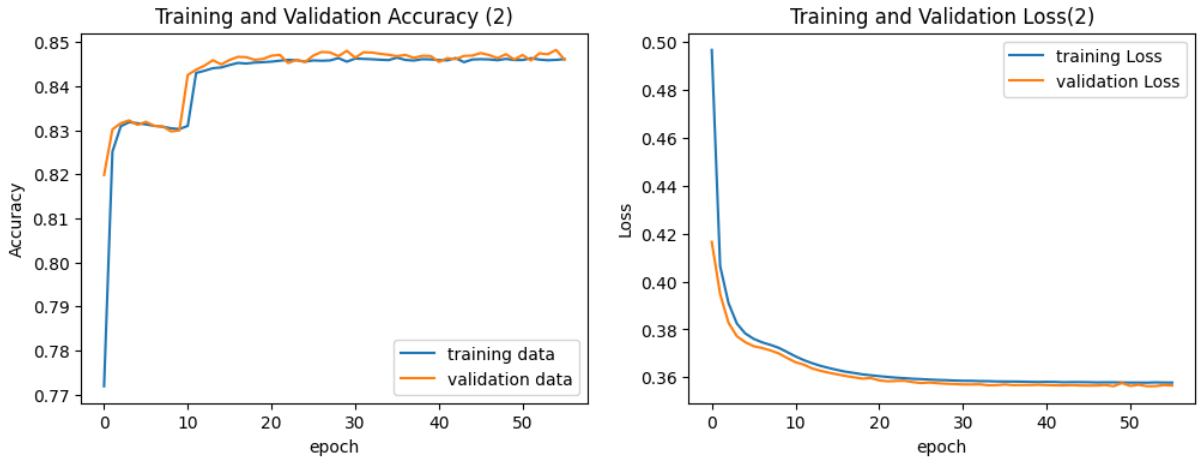

Figure 8: Comparison of training and validation metrices for Model 1

Additional Metrics

- **Precision**: 71.22%
- **Recall**: 47.48%
- **F1-Score**: 0.57

The accuracy is relatively high; however, the recall and F1-score suggest the model struggles with imbalanced data or misclassifies minority classes.

4.7 Model 2 (4-1)

This model consisted of two layers: one hidden layer with 4 neurons and an output layer with 1 neuron. The performance measure is as shown as:

Table 4: Model Performance Metrics

Training Accuracy (%)	Validation Accuracy (%)	Training Loss	Validation Loss
85.32%	85.55%	0.3473	0.3449

Additional Metrics

- **Precision**: 72.14%
- **Recall**: 49.36%
- **F1-Score**: 0.59

4.7.1 Learning Curve for Model 2

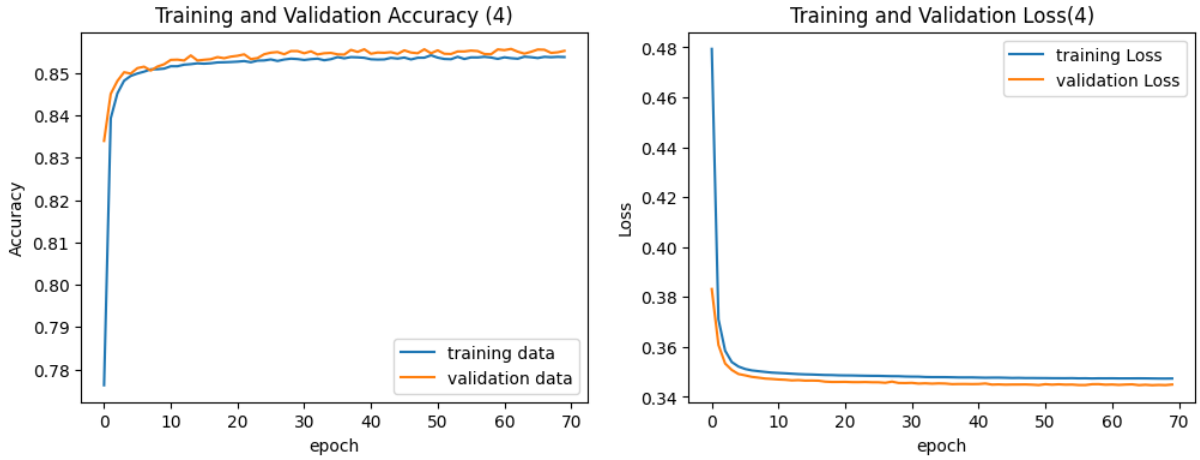

Figure 9: Comparison of training and validation metrices for Model 2

4.8 Model 3 (8-1)

This model consisted of two layers: one hidden layer with 8 neurons and an output layer with 1 neuron. The performance measure is as shown as:

Table 5: Model Performance Metrics

Training Accuracy (%)	Validation Accuracy (%)	Training Loss	Validation Loss
85.25%	85.64%	0.3425	0.3499

Additional Metrics

- **Precision**: 73.90%
- **Recall**: 50.22%
- **F1-Score**: 0.60

4.8.1 Learning Curve for Model 3

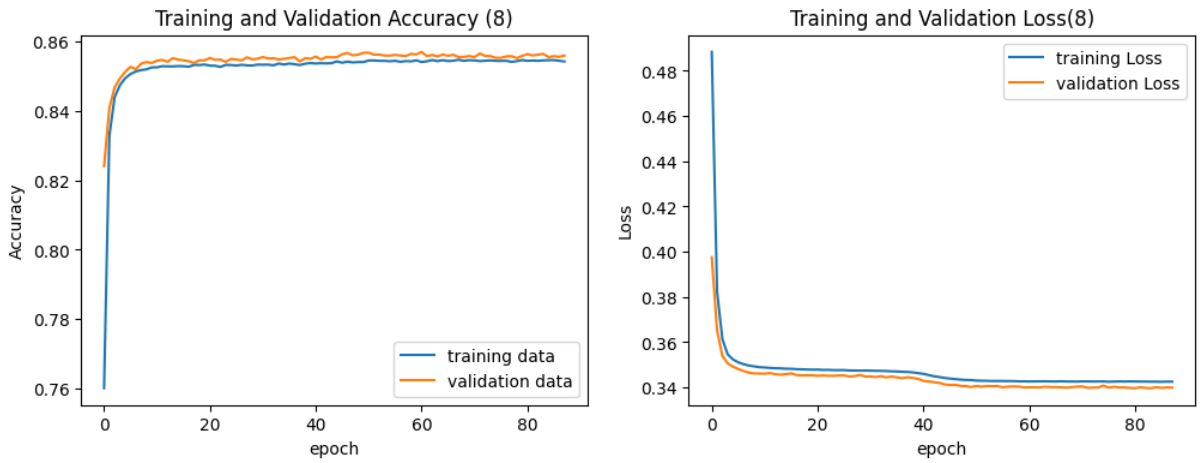

Figure 10: Comparison of training and validation metrices for Model 3

4.9 Model 4 (16-8-1)

This model takes input features and passes them through two hidden layers, the first with 16 neurons and the second with 8 neurons, both using activation functions to introduce non-linearity. Finally, the output layer with 1 neuron produces the prediction.

The performance model is as shown as:

Table 6: Model Performance Metrics

Training Accuracy (%)	Validation Accuracy (%)	Training Loss	Validation Loss
85.64%	85.72%	0.3403	0.3376

Additional Metrics

- **Precision**: 74.87%
- **Recall**: 49.52%
- **F1-Score**: 0.60

4.9.1 Learning Curve for Model 4

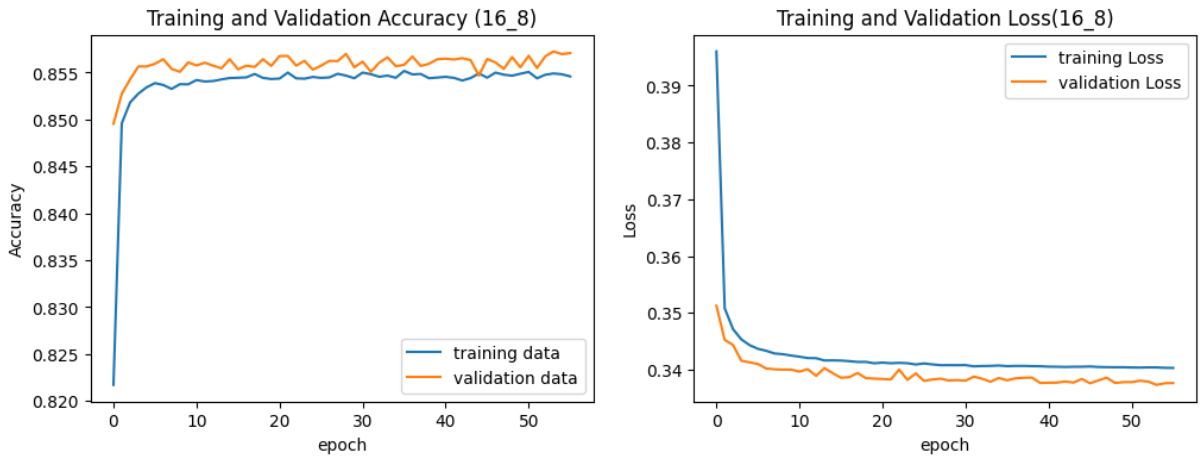

Figure 11: Comparison of training and validation metrices for Model 4

4.10 Model 5 (32-16-8-1)

In this model, the first layer has 34 neurons followed by layers with 16, and 8 neurons, all using the ReLU activation function.The final layer has 1 neuron with a sigmoid activation function. The performance measure is shown as:

Table 7: Model Performance Metrics

Training Accuracy (%)	Validation Accuracy (%)	Training Loss	Validation Loss
85.32%	85.62%	0.3369	0.3379

Additional Metrics

- **Precision**: 73.01%
- **Recall**: 51.40%
- **F1-Score**: 0.60

4.10.1 Learning Curve for Model 5

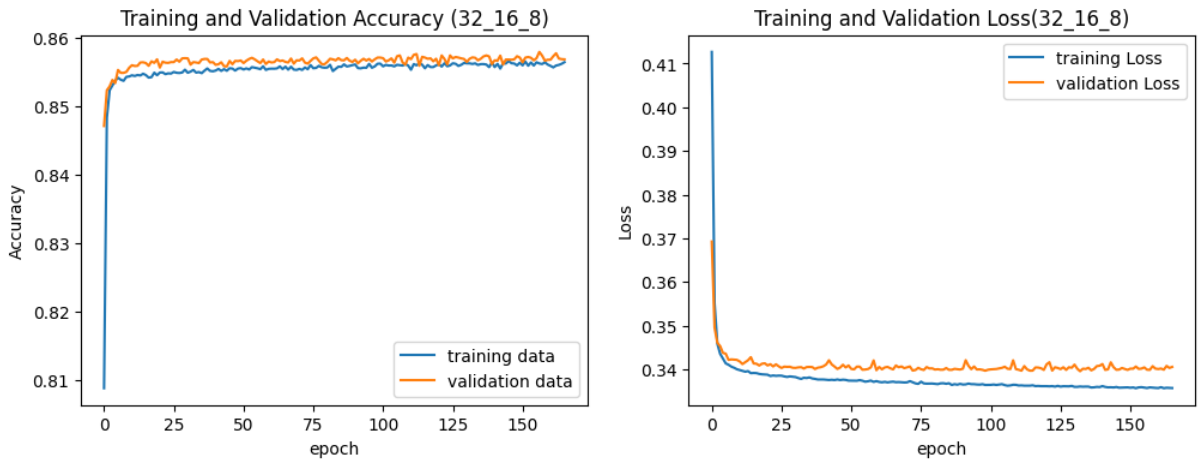

Figure 12: Comparison of training and validation metrices for Model 5

4.11 Model 6 (64-32-16-8-1)

In this model, the first layer has 64 neurons followed by layers with 32, 16, and 8 neurons, all using the ReLU activation function.The final layer has 1 neuron with a sigmoid activation function. The performance measure is shown as:

Table 8: Model Performance Metrics

Training Accuracy (%)	Validation Accuracy (%)	Training Loss	Validation Loss
85.32%	85.54%	0.3266	0.3471

Additional Metrics

- **Precision**: 74.06%
- **Recall**: 49.30%
- **F1-Score**: 0.59

4.11.1 Learning Curve for Model 6

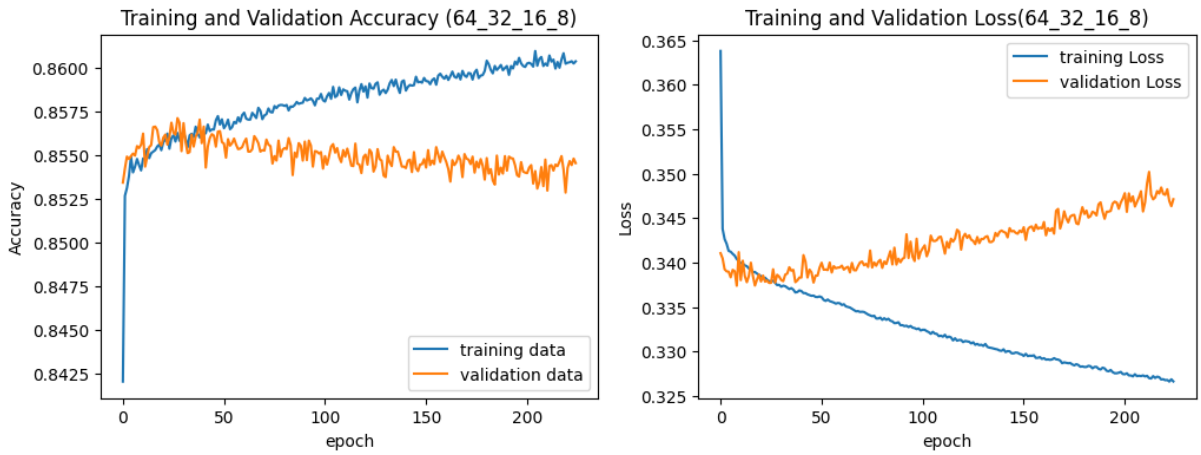

Figure 13: Comparison of training and validation metrics for Model 6

This model is **overfitting**. It performs very well on the training data but poorly on the validation data as training progresses. This happens because the model is learning the noise and specific details of the training set rather than general patterns.

4.11.2 Model Evaluation

Below is a tabular summary of the performance of different models used in this project:

Model	Training Accuracy (%)	Validation Accuracy (%)	Precision (%)	Recall (%)	F1-Score
Random Baseline Classifier	78.89	78.89	-	-	-
Logistic Regression	82.14	82.76	68.30	37.56	0.48
Neural Network (2-1)	84.65	84.75	71.22	47.48	0.57
Neural Network (4-1)	85.32	85.55	72.14	49.36	0.59
Neural Network (8-1)	85.25	85.64	73.90	50.22	0.60
Neural Network (16-8-1)	85.64	85.72	74.87	49.52	0.60
Neural Network (32-16-8-1)	85.32	85.62	73.01	51.40	0.60
Neural Network (64-32-16-8-1)	85.32	85.54	74.06	49.30	0.59

Table 9: Model Performance Metrics

Comparing the above metrics, the neural network (8-1) model is the best-performing model due to its balanced performance and simplicity. It achieves a high validation accuracy of 85.64%, with a precision of 73.90%, recall of 50.22%, and an F1-score of 0.60, outperforming simpler models while avoiding overfitting seen in more complex architectures.

5 Phase 4: Feature Importance and Reduction

The project focused on identifying and eliminating non-informative features from the dataset to simplify the neural network model without compromising its predictive performance. Below is a detailed explanation of the feature selection and reduction process implemented, along with the rationale behind each step.

For each feature in the dataset, a separate neural network model is trained using only that feature as input.

Model Architecture for Each Feature:

- **Input Layer:** Accepts a single feature.
- **Hidden Layer:** One Dense layer with 2 neurons and ReLU activation.
- **Output Layer:** One Dense layer with 1 neuron and sigmoid activation.

Table 10: Validation Accuracies for Individual Features

Feature Index	**Validation Accuracy**
Feature 0	0.7883
Feature 1	0.7981
Feature 2	0.7883
Feature 3	0.7883
Feature 4	0.7883
Feature 5	0.7883
Feature 6	0.7883
Feature 7	0.7883

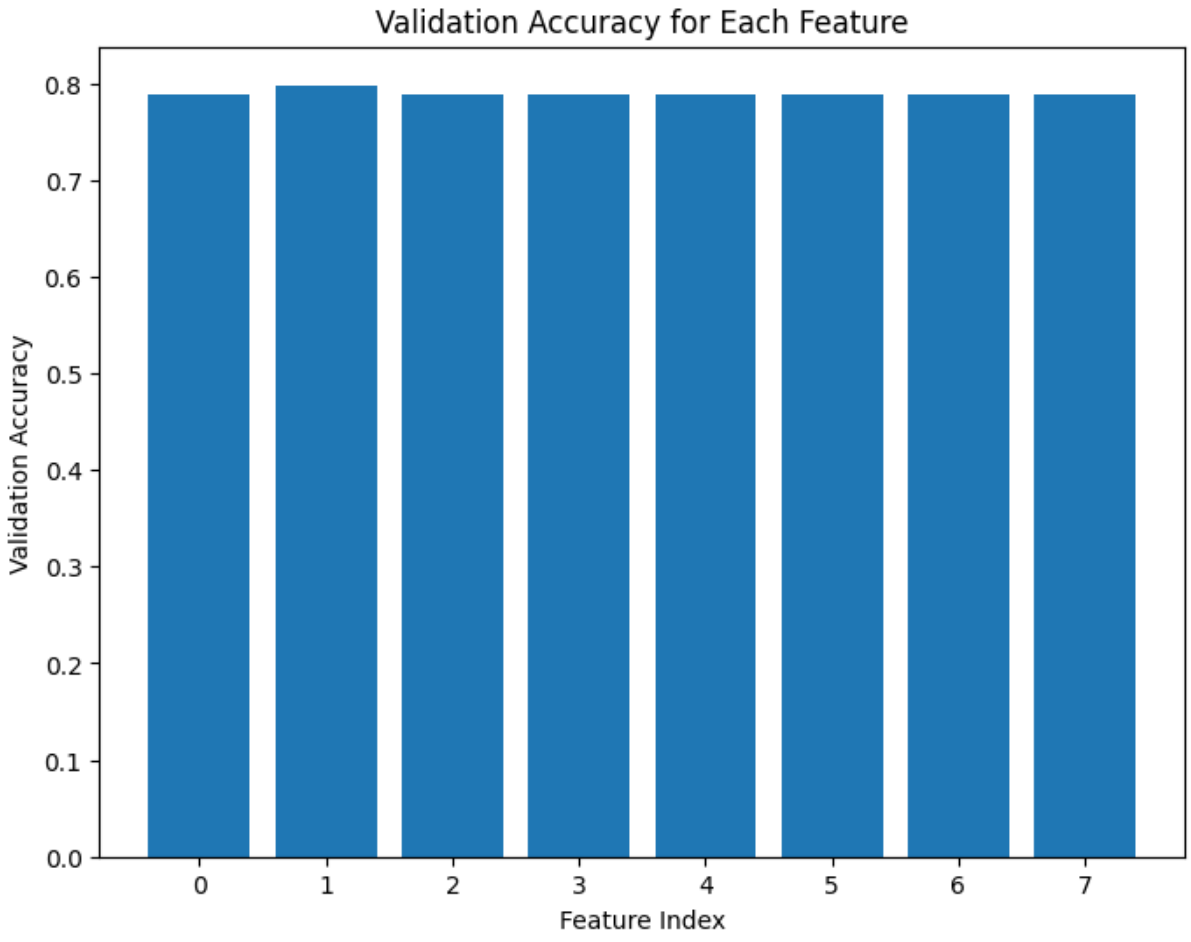

Figure 14: Validation Accuracy with each features

In above figure, Feature 1 which is **'Age'** column, achieved the highest validation accuracy, making it the most informative feature. All other features demonstrated identical validation accuracies.

After that,an iterative feature removal process was conducted, starting with the least important features, to evaluate their impact on the model's validation accuracy.

Table 11: Feature Reduction Process and Validation Accuracy

Removed Features	**Validation Accuracy**
None	0.8350
{Feature 7}	0.8470
{Features 6, 7}	0.8207
{Features 5, 6, 7}	0.8040
{Features 4, 5, 6, 7}	0.8008
{Features 3, 4, 5, 6, 7}	0.7978
{Features 2, 3, 4, 5, 6, 7}	0.8072
{Features 0, 2, 3, 4, 5, 6, 7}	0.7997

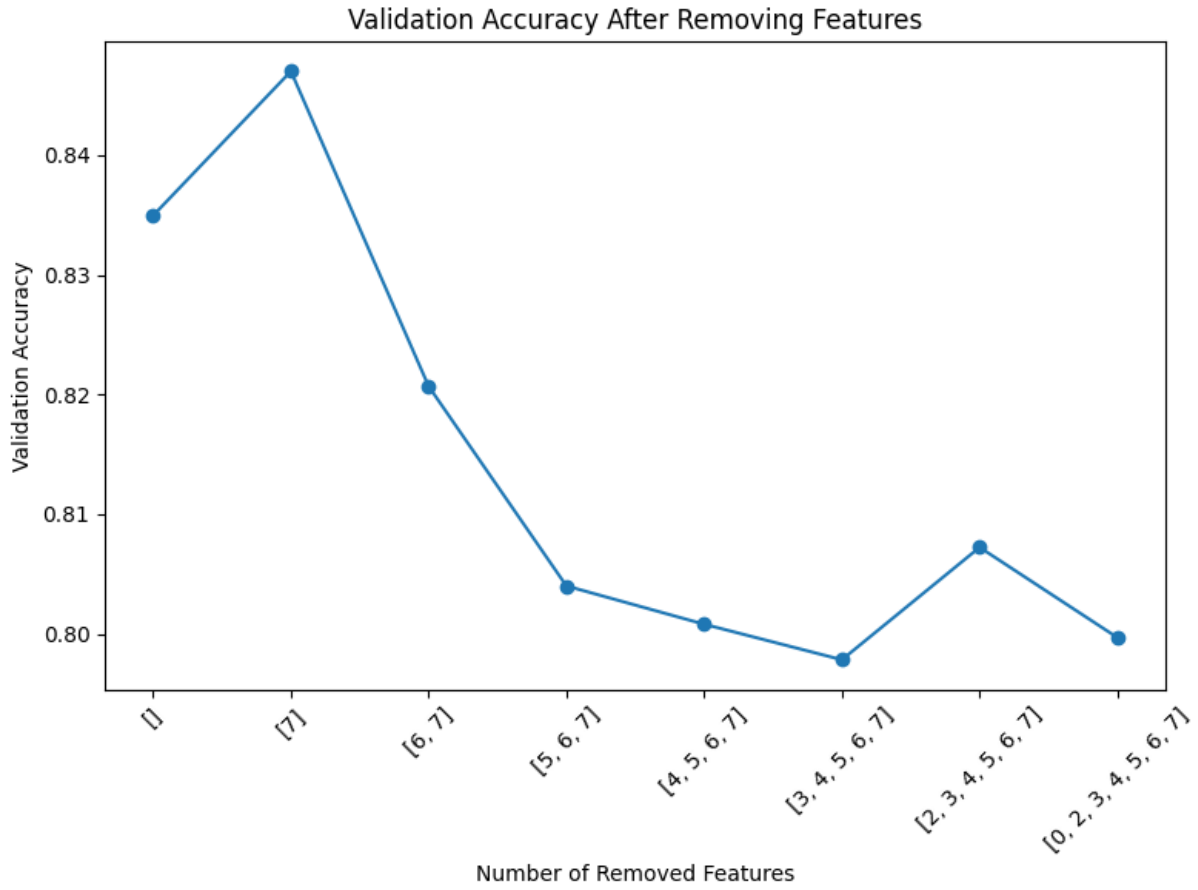

Figure 15: Validation Accuracy after removing features

When no features were removed, the validation accuracy was 0.8350.

However, removing Feature 7 which is **'EstimatedSalary'** resulted in an increase in validation accuracy to 0.8470, suggesting that Feature 7 might introduce noise. This analysis provides clear guidance for feature reduction to improve model efficiency without significant loss of performance.

6 Challenges Faced

1. **Data Imbalance:** One of the main challenge was the imbalance in the dataset, where the majority of customers did not churn. This made it difficult for the model to correctly predict churned customers, leading to biased predictions.

2. **Overfitting:** Complex models with multiple layers and neurons showed a tendency to overfit, capturing noise in the training data and failing to generalize to the validation set.

3. **Finding the Right Architecture:** Extensive experimentation was required to determine the ideal number of layers and neurons. Achieving a balance between model complexity, accuracy, and avoiding overfitting was difficult.

4. **Training Time:** Larger models and long training durations increased computational time, making it challenging to quickly test and refine different architectures.

5. **Feature Selection:** Identifying less important features required iterative experimentation.

7 Future Improvements

1. **Hyperparameter Tuning:** Further hyperparameter optimization (e.g., batch size, number of epochs) could be conducted to achieve better performance.

2. **Data Augmentation:** Balancing the dataset through data augmentation or synthetic data generation could help improve the model's ability to predict the minority class (i.e., churned customers).

8 Conclusion

This project demonstrated the effectiveness of machine learning techniques, particularly logistic regression and neural networks, in predicting customer churn in the banking sector. Starting with a baseline accuracy of 78.89%, logistic regression achieved moderate performance with a validation accuracy of 82.76%, but it struggled with imbalanced data, as reflected in lower recall and F1-score. Neural networks outperformed logistic regression, with the (8-1) architecture emerging as the best model. This model achieved a validation accuracy of 85.64% and a balanced F1-score of 0.60, demonstrating its capability to effectively capture complex patterns in the data.

AFeature analysis identified key predictors such as *Credit Score* and *Age*, simplifying the model while maintaining performance. Simplifying the model by focusing on these

features maintained performance while improving computational efficiency. Additionally, removing less informative features, such as *Estimated Salary*, contributed to further improvements. While deeper architectures, like (16-8-1), showed marginal gains in accuracy (up to 85.72%), they introduced higher computational costs and overfitting risks, highlighting the importance of finding an optimal balance between complexity and performance.

Overall, the project underscores the potential of neural networks for customer churn prediction and retention strategies. The findings provide a strong foundation for future improvements, such as addressing data imbalance, optimizing hyperparameters, and integrating additional data sources to enhance predictive power. These insights pave the way for developing actionable and efficient strategies to minimize customer churn and improve business outcomes.

9 References

Michael (2024) Bank customer churn prediction using machine learning, Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2022/09/bank-customer-churn-prediction-using-machine-learning/ (Accessed: December 2024).

Mayur (2023) Customer churn prediction using Artificial Neural Network, Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/10/customer-churn-prediction-using-artificial-neural-network/ (Accessed: November 2024).

Abhikumar Patel, Amit G Kumar, "Predicting Customer Churn In Telecom Industry: A Machine Learning Approach For Improving Customer Retention".

Grammarly: An online tool that helps with grammar, punctuation, and style

Kaggle: Customer Churn Prediction: Bank Dataset