*Nipun Chauhan (NXC220056)*

**Project Report**

**Bank Customer Churn Prediction**

***Executive Summary***

*Project Objective:*

The goal of this project was to build and compare different machine learning models to predict customer churn. By analyzing customer data, we aimed to identify patterns and factors contributing to customer attrition, with the ultimate objective of enhancing customer retention strategies.

*Overview:*

The project commenced with data exploration of the 'Churn\_Modelling.csv' dataset, which included customer demographic and financial attributes. After preprocessing, which involved feature encoding and scaling, three models were developed and evaluated: a neural network using Keras and TensorFlow, a logistic regression, and a decision tree classifier. The performance of each model was compared based on accuracy, precision, recall, and F1 score.

*Results:*

The neural network achieved an accuracy of 84.85% on the test set after being trained for 100 epochs, showcasing an improvement over the base accuracy of predicting the majority class at 79.63%. The logistic regression and decision tree classifier models resulted in lower accuracies of 80.65% and 78.85%, respectively. Precision-recall and ROC curves were generated for the neural network, illustrating the trade-offs between true positive rate and false positive rate, as well as between precision and recall.

*Conclusions:*

The neural network outperformed the other models in terms of overall accuracy and is a promising tool for predicting customer churn. The model can be improved by further tuning hyperparameters, feature engineering, and utilizing more advanced ensemble methods. The results can inform the bank's retention strategies by identifying high-risk customers for proactive engagement.

***Report***

**Introduction:**

In this project, we focused on developing a model to predict when customers might stop using a bank's services—a problem known as 'customer churn'. By predicting churn, banks can try to keep their customers by addressing their concerns before they leave.

**Data Exploration:**

We started by looking at a big set of data from the bank, specifically a file called 'Churn\_Modelling.csv', which contains details on 10,000 customers. Each customer's data had 14 variables, like their credit score, where they live, and how much money they have in their account.

To get a better idea of what we were working with, we took a random peek at 10 customers from the data. We noticed some variables like 'CustomerId', 'Surname', and 'RowNumber' that wouldn't help us predict churn, so we decided to remove them from our analysis. This left us with 11 types of information for each customer that we thought would be more useful.

We also wanted to see if there was a pattern in how long customers stayed with the bank before they left. So, we made a histogram that showed us the number of customers who stayed (loyal) and those who left (churned) based on how long they had been with the bank. We colored the loyal customers in red and the ones who left in blue, and we included this as a visual to help us understand customer behavior better.

A screenshot of a computer

Description automatically generated

The histogram shows us how long customers usually stick with the bank before some of them decide to leave. What stands out is that in the 10th year, a lot of customers decide to go—this is the time when we see the most churn. But at the same time, the 10th year is also when many customers are choosing to stay. This tells us that the 10th year is super important; we need to pay extra attention to customers during this time to keep them from leaving, even though it's also a year when a lot of customers are happy to stay.

Once we had a clearer view of the data and removed what wasn't needed, we were ready to move on to preparing the data for our models.

**Feature Selection and Data Preprocessing:**

Before we could start building our models to predict churn, we needed to get our data in shape. First, we chose which pieces of customer information (or features) to use in our model. We needed features that would really show us who's likely to leave the bank. So, we took out things like names and customer IDs because they don't tell us anything about a customer's behavior.

Next, we turned our attention to the type of data we had. Some of the features were categories like 'Male' or 'Female' for gender and different countries for geography. We know that computers like numbers, not words, so we had to change these into numbers. For gender, we just switched 'Female' to 1 and 'Male' to 0. For the countries, we used 'one hot encoding', which creates new features to represent the categories with 1s and 0s.

**Data Cleaning:**

After that, we made sure our data was clean, meaning no weird or missing values that could cause problems to our model. If there were any missing numbers, we got rid of those rows entirely.

With everything clean, we changed some of the features like how much money customers have or how old they are into a common scale, between 0 and 1 using MinMaxScaler. This helps the model treat all the features fairly, without giving too much attention to the big numbers.

Lastly, we checked the base accuracy. We checked how many customers left the bank (churned) versus how many stayed. It turns out about 80%(roughly) stayed, which gave us a starting point for our model—if it guessed that everyone stayed, it would be right 80% of the time. But we can do better than that, and that's what we aimed for with our models.

**Dataset after preprocessing and cleaning:**

A screenshot of a computer

Description automatically generated

**Model Building and Initial Evaluation:**

**Neural Network Model:**

Our first attempt at predicting customer churn was with a neural network, a kind of artificial brain that learns from data. We used Keras and TensorFlow, two powerful tools that help us create and teach these networks.

Our network was built with layers, like a sandwich. The first layer had 48 'neurons', or tiny processing units that work together to find patterns in our data. We fed it all 12 pieces of customer data we had processed earlier. We used 'relu', a common function, to help each neuron decide what to pass on to the next layer. The second layer had 24 neurons, also using the relu function. Finally, the last layer had just one neuron with a 'sigmoid' function, which squishes the output between 0 and 1—perfect for guessing whether a customer will churn (closer to 1) or not (closer to 0).

To prepare our network for learning, we had to compile it. This step involved choosing 'adam', a smart way to adjust how the network learns as it sees more data, and 'binary\_crossentropy', which measures how far off our network's guesses are from the actual answers.

A screenshot of a computer

Description automatically generated

**Model Training and Validation:**

We split the customer data into two groups: 80% for training our network and 20% to test it later. Training a network is a bit like teaching a child; we show it examples and tell it when it's right or wrong. We did this for a few rounds, known as epochs. In the first five epochs, we saw our network getting better at predicting churn, its accuracy climbing from about 78% to almost 84%. We kept this up for 100 epochs in total, watching the accuracy improve to almost 88%.

Training a neural network is tricky; it can get too fixated on the training data, a problem known as overfitting. It's like memorizing answers without understanding the questions. We kept an eye out for this as we trained, making sure our network was learning the right way.

After all that training, we checked how well the network learned by using our test data, and it scored about 85% accuracy. We also looked at the individual predictions it made to get an idea of when it was guessing right or wrong. This gives us clues on how to make our network smarter.

**Evaluation Metrics:**

To understand how well our models were doing, we used a few different measurements. Accuracy tells us the percentage of predictions our model got right. Precision measures how many of the customers our model thought would churn actually did churn. Recall checks how many of the actual churners our model managed to catch. And the F1 score is a mix of precision and recall, giving us a single score that balances both. These metrics give us a full picture because just knowing the accuracy isn't enough—if a model simply guesses that no one churns, it might seem accurate, but it wouldn't be useful.

**Model Comparison:**

We trained three different models: a neural network, logistic regression, and a decision tree. The neural network did the best, with about 85% accuracy, and it was pretty good at precision and recall too. The logistic regression was a bit less accurate, around 81%, and wasn't as good at catching churners. The decision tree had the lowest accuracy at around 79% and was less precise but caught more actual churners than logistic regression.

***Classification report for ANN***A number of numbers in a row

Description automatically generated with medium confidence

***Confusion matrix for ANN***

**A graph of a number of different colored squares

Description automatically generated with medium confidence**

***Classification report for Logistic regression***

***A number of numbers in a row

Description automatically generated with medium confidence***

***Classification report for Decision Trees***

***A number of numbers in a row

Description automatically generated with medium confidence***

**Advanced Evaluation Techniques:**

Then we dug deeper with some advanced techniques. The ROC curve plots out the trade-off between catching all the churners (true positive rate) and mistakenly identifying loyal customers as churners (false positive rate). The area under this curve (AUC) tells us, in one number, how good the model is—the closer to 1, the better. The Precision-Recall curve does something similar but focuses on how many of the predictions of churn are correct (precision) as it tries to catch more churners (recall). The area under this curve also helps us measure performance, especially when the classes (like churn/no churn) are imbalanced.

Both these curves help us see beyond just accuracy and understand how our model performs in different scenarios, like when we're more or less strict about predicting churn.

A graph of a positive and negative curve

Description automatically generated with medium confidence

By looking at all these metrics and curves, we got a clear idea of where each model shines and where it could use some improvement. It's all about finding the right balance for what the bank needs—whether that's not missing any churners or making sure not to bother customers who are happy with their service.

**Conclusions and Future Work:**

After comparing the models, the Artificial Neural Network (ANN) emerged as the leader with the highest accuracy at 85%. It's a well-rounded model, doing a good job of balancing both the detection of customers who will churn and those who will stay. The Decision Tree came in with the highest recall for churned customers, meaning it was best at catching the customers most likely to leave, but it had a lower accuracy overall. Logistic Regression showed a strong bias towards predicting the customers who would stay, which could be less helpful if we want to focus on preventing churn.

The ANN stands out as the most effective model for the organization. It strikes a balance between identifying customers who might churn and those who will stay, making it a reliable choice for the bank's efforts to keep customers. This model could be the backbone of a system that helps the bank understand its customers better and tailor its services to keep them satisfied and loyal.

For future work, we could look at improving these models even further. More complex neural network architectures, such as those with more layers or different types of layers, could capture subtler patterns in customer behavior. We could also try ensemble methods, where we combine predictions from multiple models to get even better results.

Another area to explore is gathering more data—especially on why customers decide to leave. Understanding the reasons behind churn could help us add new features to our models, making them even more insightful. We could also collect more demographic data or details on customer interactions with the bank's services.

Lastly, it would be worth looking into how different models perform over time and with different segments of customers. This could help the bank tailor its retention efforts to different groups, ensuring that each customer gets the most relevant and effective service possible.