The German University in Cairo

Faculty of Management Technology

Business Informatics

Artificial Intelligence CSEN933 - Project 2

Submitted by: " Team Zeina "

Malak Moataz 52-2902

Reem Khaled 52-2002

Shahd Islam 52-0723

Sherifa Khaled 52-3677

Zeina Badra 52-6623

Professor: Dr Mohamed Karam

T.A: Ahmed Amgad

Date: 8/12/2023

Today, businesses frequently use the term "customer churn" to refer to the phenomenon of customers ceasing to purchase an organization's goods or services. Large-scale research is done on the subject to determine the likelihood of client churn. The examination of customer churn is important as it sheds light on the consequences that consumer behavior has on both finances and reputation. For instance, acquiring new clients comes at a significantly higher expense than keeping hold of current ones. Retaining a leading market position in a cutthroat market and enhancing customer satisfaction by addressing the underlying reasons of customer churn are two further advantages of customer churn analysis.

Predicting customer churn poses several challenges. One big issue is that the data we use is often imbalanced, meaning there are fewer examples of customers actually leaving. Sorting through the data noise to pick out the important features is tricky, and we also have to take into account how customer behavior changes over time. Another challenge is figuring out what's causing customers to leave versus what's just happening at the same time. People are different, so understanding the various types of customers and dealing with different definitions of "churn" in different industries is crucial. When it comes to the models we use, it's not enough for them to just make predictions; we need to be able to understand why they make the predictions they do. This is especially important when dealing with complex models. We also need to make sure everyone is on the same page with what "churn" actually means. Lastly, customer journeys are always changing, and our models need to keep up. We have to account for all the different ways customers interact with us, and deploying and integrating these models into our existing systems has to be seamless for them to be effective.

The Communications and Telecom industry is quite invested in understanding customer churn. We decided to use a telecom dataset from Kaggle called "Telco Customer Churn" to predict churn. We picked this dataset because we're already interested in customer churn in the telecom sector. It's a decently large dataset with 7043 rows, offering various features to work with. It's a popular dataset, ranking high in search results for "churn" with almost 2 million views. One of the big advantages is that the dataset is clean, with few missing values and redundant data, making it convenient and useful for our analysis.

Neural networks, inspired by the human brain, are a type of machine learning model. They consist of interconnected nodes or "neurons," organized into layers including an input layer, one or more hidden layers, and an output layer. Each connection between neurons has a weight that adjusts during training to learn patterns in the data. Activation functions introduce non-linearity to computations. Through backpropagation, the network refines weights based on prediction errors, improving its ability to make accurate predictions on new data. Neural networks excel in applications like image recognition, speech processing, natural language processing, and predictive modeling due to their adaptability and pattern-learning capabilities.

Our chosen algorithm (methodology) is a Sequential Neural Network (SNN). Our preference for SNN in predicting customer churn is grounded in its compatibility with the dataset's characteristics and the inherent nature of the prediction task. SNNs are notably well-suited for tasks involving sequential or tabular data with fixed-size input vectors, where the order of input elements is less crucial. In the context of customer churn prediction, the temporal aspect of data, such as the tenure of a customer, aligns seamlessly with the sequential nature that SNNs handle adeptly.

Furthermore, the inclusion of the "Contract" attribute strengthens the case for SNNs. The dynamics of the telecommunications industry, where companies provide incentives and promotions for long-term customers, offer bundled services as rewards for loyalty, and ensure price stability for extended contracts, highlight the temporal dependencies associated with customer behavior. Analyzing the "Contract" attribute in the context of churn prediction involves understanding sequential patterns tied to the terms of customer contracts.

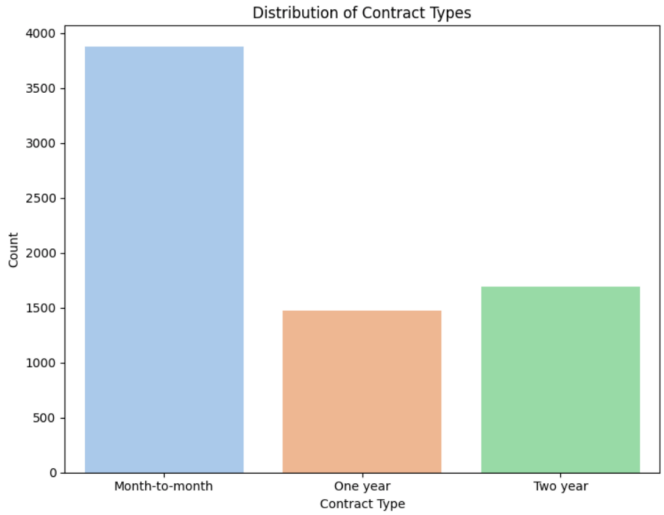
In contrast, Convolutional Neural Networks (CNNs) excel in spatial pattern recognition, making them more suitable for image-related tasks. Recurrent Neural Networks (RNNs) are adept at handling sequential or time-series data, emphasizing the importance of input order. Long Short-Term Memory Networks (LSTMs), a specialized form of RNNs, are effective in capturing long-term dependencies within sequential data.

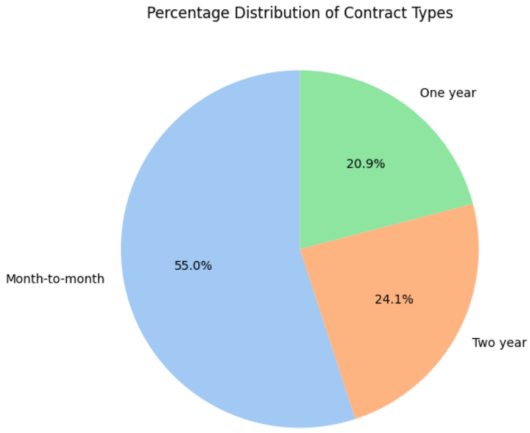
Because customer churn prediction requires understanding patterns over time, considering both the duration of customer association (Tenure) and contract specifics, Sequential Neural Networks (SNNs) are a suitable choice. SNNs' structure, with interconnected layers learning patterns over time, aligns well with the dynamic nature of customer behavior and the temporal dynamics in churn prediction.

The dataset included a total of 21 attributes, featuring a conclusive "Churn" column with values of "yes" and "no". Following a thorough examination of attributes such as "gender," "paperless billing," "payment method," and "Total Charges," we formulated a hypothesis that the most probable predictors of churn were "Tenure" (the duration of the customer's association with the company in months) and "Contract" (the customer's contract term – Month-to-month, One year, Two years). Our assessment revealed that all other attributes showed no discernible correlation with the dependent variable "Churn." To explore potential relationships with the "Churn" variable, we created distinct diagrams using 'matplotlib.pyplot' in Google Colab. The "Contract" attribute drew our attention due to certain telecom industry dynamics. Telecom companies often provide attractive incentives and promotions for long-term customers, and they frequently offer bundled services to customers who remain loyal for extended periods, aiming to enhance customer retention. Additionally, long-term contracts can provide price stability, shielding customers from inflationary impacts. These industry insights led us to suspect a significant correlation between churn and the customer's contract. To ensure no potentially valuable variable was overlooked, both independent variables were individually tested against churn.

Contract:

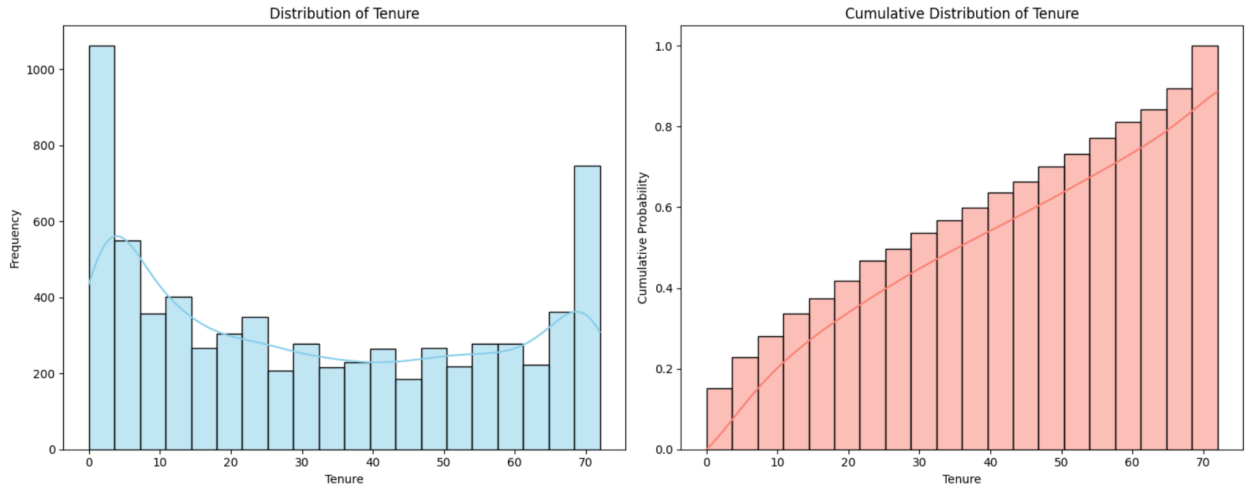
Digging in deeper to each separate independent variable, we'll start with 'Contract'. From the below graphs, we can see that the 'Month-Month' contract type has the most amount of subscribers which is approximately about 3900 (as seen in the bar chart below), coming in second is the contract type of 'Two- year' which is approx.1600 and the final contract type with the least subscription count is 'One year'. This can also be represented by the pie chart below.





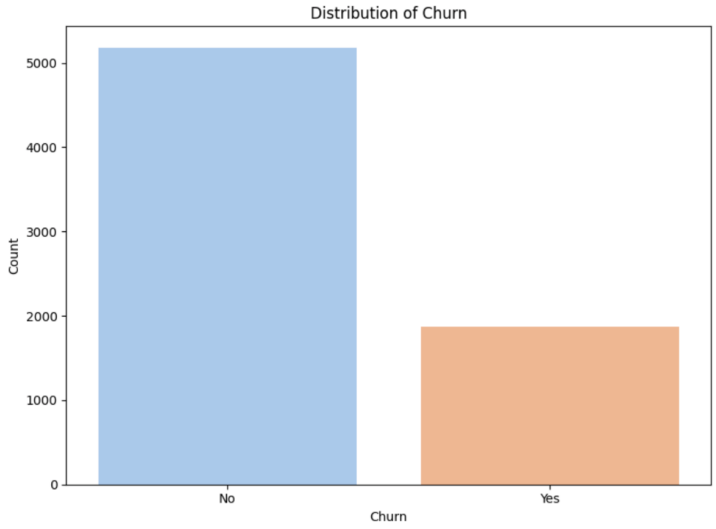
Tenure:

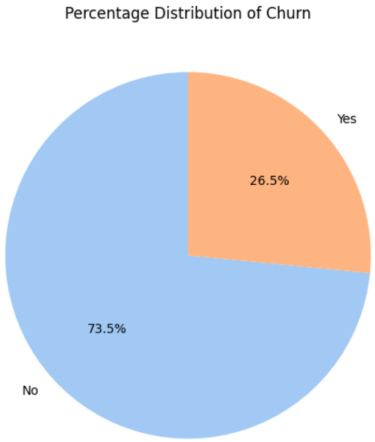
The histogram graph generated below illustrates the distribution of the 'tenure' attribute in the dataset. The left subplot displays a regular histogram, providing a visual representation of how frequently different tenure values occur. The x-axis represents the 'Tenure' values, and the y-axis represents the frequency or count of instances with each tenure value. This histogram allows you to observe the distribution of customer tenure in the dataset, showing which tenure periods are more common or less common. The graph shows a high distribution of customers with high and low tenure. The right subplot displays a cumulative distribution function, demonstrating the cumulative probability of observing a tenure value up to a certain point. This allows for insights into the overall distribution and the proportion of customers with tenures below specific thresholds. Together, these subplots offer a comprehensive view of the 'tenure' attribute's distribution and cumulative characteristics in the dataset.



Churn:

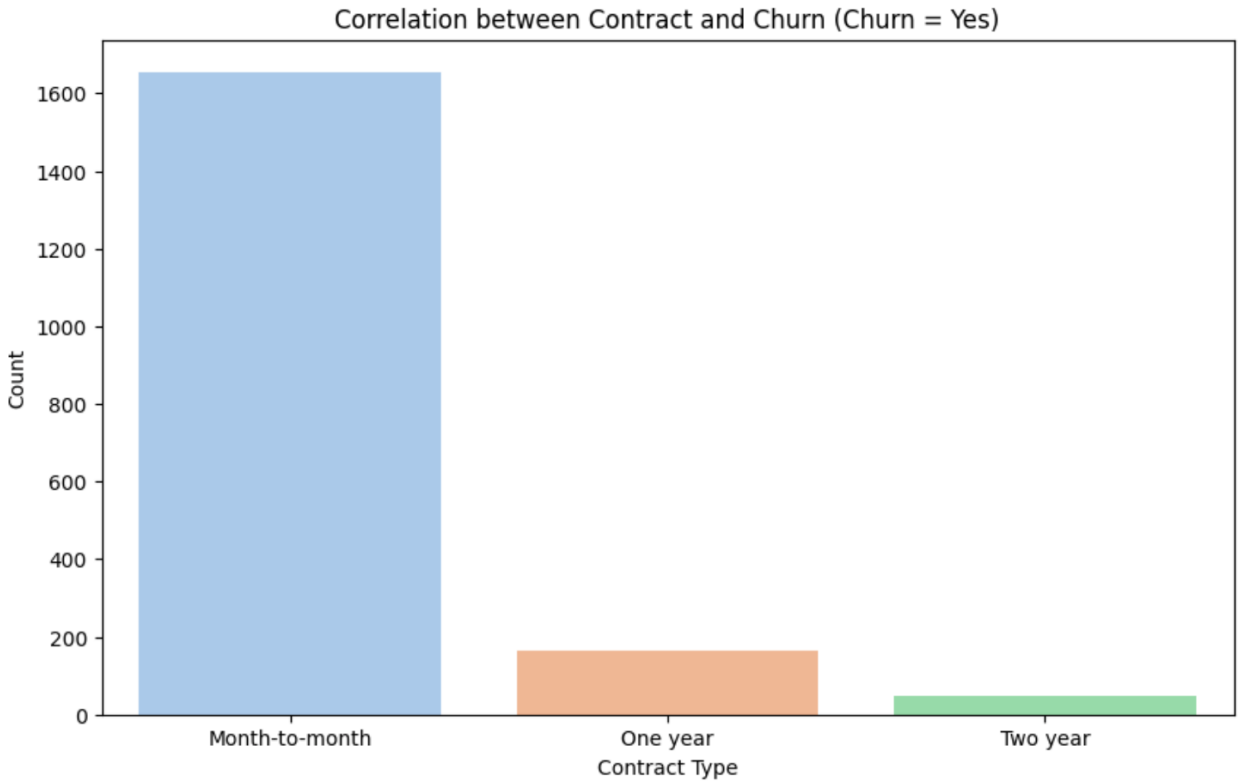
Moving on to the dependent variable Churn, we represented this variable using a bar chart and pie chart. The graphs show that the distribution of customers who have churned versus the customers who have not are not exactly equal. There are more instances of customers who have not churned than customers who did, and so it is recommended for future research to include a higher number of customers who have churned in order to make the training algorithm more accurate, so the testing algorithm could be more reliable. Almost 75% (>5000) of the customers did not turn in this dataset.





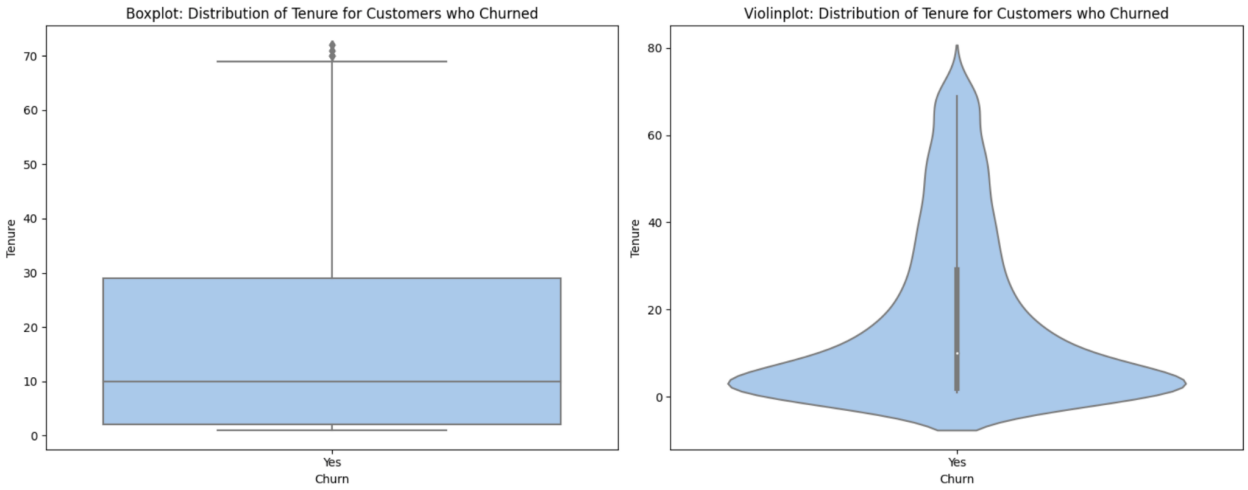
Contract VS Churn:

The following graph is a bar chart, showing the relationship between contract and churn (='Yes'). It shows that the contract type 'Month-to-month' has the highest churn rate with more than 1600 customers who have churned, while the contract type 'Two year' has the lowest churn rate with less than 200 records of customers churned. So we can conclude that customers who churn typically opt for a 'Month-to-month' contract type, which proves a high correlation between the independent variable 'Contract' and the dependent variable 'Churn'.



Tenure VS Churn:

The following graphs are a boxplot and violin plot, which we used to represent the correlation between customers who have churned and tenure. As we can see from the below boxplot and violin plot, most customers churn within the first 30 months, with an average tenure of 10 months. The bold line in the violin plot represents the range of the central tendency and the dot in the middle shows the average, similar to the box plot. To conclude, tenure has a high correlation with the dependent variable churn.



These two variables, tenure and contract, will be used in the following Neural Network classification algorithm to predict churn.

Our code in Google Colab prepares and implements a neural network model for binary classification of customer churn using relevant features from a dataset. It begins by extracting the 'tenure' and 'Contract' columns as input features ('X') and the 'Churn' column as the target variable ('y'). The ordinal categorical feature 'Contract' is label encoded using scikit-learn's LabelEncoder. The target variable 'Churn' is then mapped to binary labels ('Yes' mapped to 1, 'No' mapped to 0). The dataset is split into training and testing sets using the train_test_split function. A sequential neural network model is constructed using Keras with two dense hidden layers and a sigmoid activation function in the output layer for binary classification. The model is compiled with the Adam optimizer and binary cross-entropy loss function. After training the model on the training set, its performance is evaluated on the test set, and the accuracy is printed as the final result.

The neural network model designed for binary classification of customer churn exhibits a three-layer architecture, comprising two dense hidden layers with 64 and 32 neurons, respectively, and a final dense layer with a sigmoid activation function. We will monitor Loss and Accuracy specifically for this model over the 10 epochs employed, an epoch being one complete pass through the entire training dataset.

Loss quantifies the difference between the predicted values of the model and the actual values in the training data. During the training of a classification neural network model, the goal is to minimize this loss. In the below figure we can observe the Loss per epoch which means that the loss is being calculated and monitored at the end of each epoch during the training process. The loss per epoch is a measure of how well the model is learning from the data over the course of training. The results of each epoch for one of our runs produced the below numbers. Ideally, as the model learns and the training progresses, the loss should decrease and our model does in fact show a decrease in loss from 56.47% to 49.08%. The accuracy can also be seen increasing with every epoch from 73.91% to 75.22%. This indicates that the model is learning and improving. The final results of this run gave a 46.85% loss and 75.94% accuracy.

```
Epoch 1/10
<ipython-input-18-005774d3a910>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X['Contract'] = le.fit_transform(X['Contract'])
141/141 [==============================] - 1s 4ms/step - loss: 0.5647 - accuracy: 0.7391 - val_loss: 0.4678 - val_accuracy: 0.7613
Epoch 2/10
141/141 [==============================] - 0s 2ms/step - loss: 0.5017 - accuracy: 0.7422 - val_loss: 0.4577 - val_accuracy: 0.7693
Epoch 3/10
141/141 [==============================] - 0s 3ms/step - loss: 0.4951 - accuracy: 0.7431 - val_loss: 0.4554 - val_accuracy: 0.7666
Epoch 4/10
141/141 [==============================] - 0s 2ms/step - loss: 0.4909 - accuracy: 0.7324 - val_loss: 0.4532 - val_accuracy: 0.7613
Epoch 5/10
141/141 [==============================] - 0s 2ms/step - loss: 0.4831 - accuracy: 0.7499 - val_loss: 0.4765 - val_accuracy: 0.7613
Epoch 6/10
141/141 [==============================] - 0s 3ms/step - loss: 0.4950 - accuracy: 0.7435 - val_loss: 0.4617 - val_accuracy: 0.7684
Epoch 7/10
141/141 [==============================] - 0s 3ms/step - loss: 0.4874 - accuracy: 0.7502 - val_loss: 0.4542 - val_accuracy: 0.7693
Epoch 8/10
141/141 [==============================] - 0s 3ms/step - loss: 0.4919 - accuracy: 0.7437 - val_loss: 0.4546 - val_accuracy: 0.7666
Epoch 9/10
141/141 [==============================] - 0s 2ms/step - loss: 0.4960 - accuracy: 0.7402 - val_loss: 0.4552 - val_accuracy: 0.7666
Epoch 10/10
141/141 [==============================] - 0s 3ms/step - loss: 0.4908 - accuracy: 0.7522 - val_loss: 0.4610 - val_accuracy: 0.7693
45/45 [==============================] - 0s 2ms/step - loss: 0.4685 - accuracy: 0.7594
Test Accuracy: 0.7594038248062134
```

Furthermore, some additional analyses were conducted on 5 instances of the data set for testing, shown in the following figure. One of the runs that we executed showed that the model, with a total of 2305 trainable parameters, achieves a test accuracy of 75.94%. Example predictions versus actual labels highlight instances of correct and incorrect predictions. The confusion matrix indicates 954 true negatives, 82 false positives, 257 false negatives, and 116 true positives. The classification report offers precision, recall, and F1-score metrics for both classes, revealing that while precision is reasonably high, recall for class 1 (churn) is comparatively lower. The overall accuracy is 76%, accompanied by a weighted average F1-score of 0.73. These results provide a comprehensive evaluation of the model's performance, guiding further analysis and potential enhancements to improve its predictive capabilities.

```
 Layer (type)                Output Shape              Param #
=================================================================
 dense_15 (Dense)            (None, 64)                192

 dense_16 (Dense)            (None, 32)                2080

 dense_17 (Dense)            (None, 1)                 33

=================================================================
Total params: 2305 (9.00 KB)
Trainable params: 2305 (9.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
45/45 [==============================] - 0s 3ms/step - loss: 0.4617 - accuracy: 0.7594
Test Accuracy: 0.7594
45/45 [==============================] - 0s 2ms/step

Example Predictions vs Actual Labels:
Example 1: Predicted=[1], Actual=1
Example 2: Predicted=[0], Actual=0
Example 3: Predicted=[0], Actual=0
Example 4: Predicted=[1], Actual=1
Example 5: Predicted=[0], Actual=0

Confusion Matrix:
[[954  82]
 [257 116]]

Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.92      0.85      1036
           1       0.59      0.31      0.41       373

    accuracy                           0.76      1409
   macro avg       0.69      0.62      0.63      1409
weighted avg       0.73      0.76      0.73      1409
```

To conclude our project research results, The model demonstrates reasonable accuracy but shows some challenges in correctly identifying instances of class 1 (churn), as reflected in the lower recall for class 1. The confusion matrix and classification report offer a detailed understanding of the model's strengths and weaknesses, providing valuable insights for further analysis or model improvement. Overall, we recommend this model.

Dataset Link: https://www.kaggle.com/datasets/blastchar/telco-customer-churn/data