**Stage 3 Deliverables: Predictive Modeling**

# Defined Architecture of ANN Model

As a data analyst tackling customer churn in the telecommunications industry, I’m building an Artificial Neural Network (ANN) model to predict churn.

**Step 1: Import Libraries**

I start by importing TensorFlow for the neural network and Scikit-learn for data processing.

**Step 2: Prepare the Data**

I split the data into training and testing sets (80-20 split) and standardize the features using StandardScaler to ensure consistency across the input features.

*import tensorflow as tf*

*from sklearn.model\_selection import train\_test\_split*

*from sklearn.preprocessing import StandardScaler*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)*

*scaler = StandardScaler()*

*X\_train\_scaled = scaler.fit\_transform(X\_train)*

*X\_test\_scaled = scaler.transform(X\_test)*

**Step 3: Define the ANN Architecture**

I define the ANN with three hidden layers (128, 64, and 32 neurons) using ReLU activation, and include dropout layers to prevent overfitting. The output layer uses a sigmoid activation for binary classification.

*model = tf.keras.models.Sequential([*

*tf.keras.layers.Dense(128, input\_shape=(X\_train.shape[1],), activation='relu'),*

*tf.keras.layers.Dropout(0.3),*

*tf.keras.layers.Dense(64, activation='relu'),*

*tf.keras.layers.Dropout(0.3),*

*tf.keras.layers.Dense(32, activation='relu'),*

*tf.keras.layers.Dropout(0.2),*

*tf.keras.layers.Dense(1, activation='sigmoid')*

])

**Step 4: Compile the Model**

I compile the model using the Adam optimizer and binary crossentropy as the loss function, with accuracy as the metric.

*model.compile(optimizer='adam',*

*loss='binary\_crossentropy',*

*metrics=['accuracy'])*

**Step 5: Review the Model**

Finally, I review the model summary to ensure everything is configured correctly. This setup should provide a strong foundation for predicting customer churn.

This streamlined process helps me focus on creating an effective ANN model to predict and reduce customer churn in the telecommunications sector.

Trained ANN Model on Provided Dataset

I'll train the ANN model using the Adam optimizer, known for its robust performance in neural networks. The model will be trained for 50 epochs with a batch size of 32.

# Train the model

*history = model.fit(X\_train\_scaled, y\_train,*

*epochs=50,*

*batch\_size=32,*

*validation\_data=(X\_test\_scaled, y\_test),*

*verbose=1)*

**Key Details:**

* **Epochs**: 50 epochs allow the model to learn effectively.
* **Batch Size**: A batch size of 32 balances training speed and stability.
* **Validation Data**: Helps monitor generalization to unseen data.

This setup ensures the model converges well and generalizes effectively, making it a strong tool for predicting customer churn.

Predicted Customer Churn and Evaluated Model Performance

**Step 1: Evaluate the Model’s Accuracy**

First, I’ll evaluate the model’s performance on the test set by calculating the accuracy, which indicates how well the model predicts customer churn compared to the ground truth.

# Calculate the accuracy of the model

*loss, accuracy = model.evaluate(X\_test\_scaled, y\_test)*

*print(f"Test Accuracy: {accuracy:.4f}")*

**Step 2: Make Predictions**

Next, I’ll use the trained model to predict churn probabilities for the test data. These probabilities are then converted into binary labels (0 or 1) based on a threshold of 0.5.

# Predict using the scaled test data

*y\_pred = model.predict(X\_test\_scaled)*

# Convert probabilities to binary labels

*y\_pred\_binary = (y\_pred > 0.5).astype(int)*

**Step 3: Analyze Predictive Performance**

To assess the robustness and accuracy of the model’s predictions, I’ll generate a classification report that includes precision, recall, and F1-score. Additionally, I’ll create a confusion matrix to visualize the model’s performance in terms of true positives, true negatives, false positives, and false negatives.

# Analyze model predictions with classification report

*print("\nClassification Report:")*

*print(classification\_report(y\_test, y\_pred\_binary))*

# Confusion matrix

*print("\nConfusion Matrix:")*

*print(confusion\_matrix(y\_test, y\_pred\_binary))*

# Summary

* **Accuracy**: Measures how often the model correctly predicts churn.
* **Classification Report**: Provides detailed metrics (precision, recall, F1-score) for assessing prediction quality.
* **Confusion Matrix**: Visualizes the distribution of correct and incorrect predictions.

This process allows me to evaluate the model’s predictive performance, ensuring it accurately and reliably identifies customers at risk of churn.

