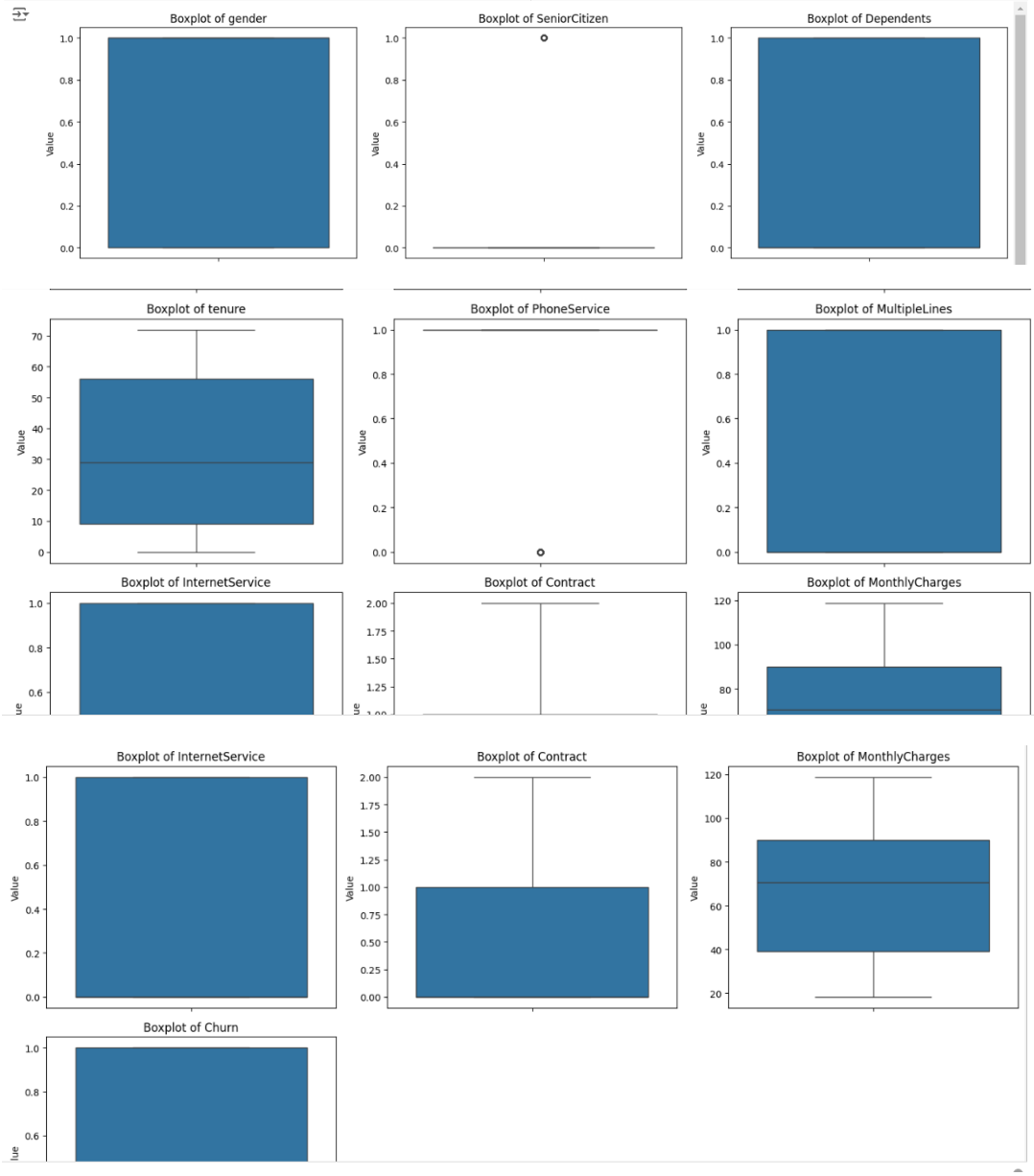
Training Progress and Model Performance Evaluation

```
# Remove any empty subplots (if the number of features is not a multiple of num_cols)
for i in range(num_features, len(axes)):
    fig.delaxes(axes[i])

# Improve the layout
plt.tight_layout()
plt.show()
```



Demonstration:

Importing Libraries: The code starts by importing essential libraries: **Pandas** for data handling, **NumPy** for numerical operations, and **Matplotlib** for visualizations.

Loading the Dataset: It loads a CSV file named "processed_data.csv" from Google Drive into a Pandas DataFrame called dataset, making it ready for analysis.

Defining Outlier Removal Function: A function called remove_outliers_iqr is created to identify and remove outliers in a specified column using the Interquartile Range (IQR) method, which helps maintain data integrity.

Cleaning the Dataset: The function is applied to the 'MonthlyCharges' column, resulting in a new DataFrame, data_cleaned, which excludes outlier entries.

Outputting the Shape of the Cleaned Dataset: The dimensions of the cleaned dataset are printed, providing insight into how many entries remain after removing outliers.

Setting Up the Visualization: A figure for the upcoming plots is created with a specified size to enhance visual clarity.

Creating Boxplot Before Outlier Removal: A boxplot is generated for the 'MonthlyCharges' column from the original dataset. It visually represents the distribution of charges, highlighting potential outliers.

Summary

This code effectively handles data preparation by removing outliers and visualizing the 'MonthlyCharges' distribution, ensuring that the subsequent analysis is based on clean, reliable data.

```
inputs = layers.Input(shape = (X.shape[1],))

x = layers.Dense(32, activation='relu')(inputs)
x = layers.Dense(16, activation='relu')(x)
x = layers.Dense(8, activation='relu')(x)
outputs = layers.Dense(units = 1, activation='sigmoid')(x)

model = tf.keras.Model(inputs, outputs, name = 'ANN_Model')


model.summary()
```

As mentioned from the ANN architecture, this is our code snippet to create a complete ANN Model

After that, we started training the model with model.fit() function. Before that, we use a regularisation technique called **Early stopping**.

```
#stop overfitting
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience= 5, restore_best_weights=True)
history = model.fit(X_train, Y_train, validation_split=0.2, epochs=30, batch_size=64, verbose = 1, callbacks=[early_stopping])
```

```
Epoch 1/30
61/61 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - accuracy: 0.7401 - loss: 0.6074 - val_accuracy: 0.7222 - val_loss: 0.4964
Epoch 2/30
61/61 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7566 - loss: 0.4673 - val_accuracy: 0.7829 - val_loss: 0.4431
Epoch 3/30
61/61 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8016 - loss: 0.4286 - val_accuracy: 0.7819 - val_loss: 0.4341
```

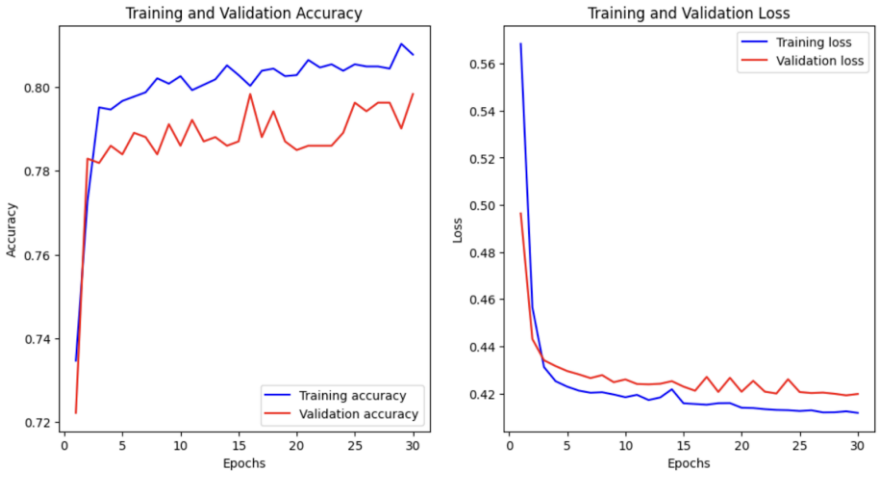
This method is used to prevent **overfitting** (which will be explained later on the learning curve visualization). Early stoping method monitor the model's performance on a validation set during training and halt the process once there's a degrade in model's performance.

Main useful of early stopping:

- **Prevent overfitting**: Once the model learns irrelevant patterns in the training data, which negatively impacts its ability to generalise. It will stop the training and record the best parameters, it helps find a balance between underfitting and overfitting
- **Save computational resources**: Once the training stop early with no progression, lead to unnecessary computations.
- **Improve generalisation**: Result of utilising early stopping generally give the model results before better on unseen data.

Then we run model.fit() to training the model and store all necessary values into a 'history' variable, parameters explain:

- ***X_train, Y_train***: the input and output from our training dataset. Contains the corresponding data points and labels for the model needs to predict
- ***Validation_split = 0.2***: split another 20% from the training data to be used as validation data. The model won't directly train on this 20% but will evaluate its performance on each epoch
- ***Epochs = 30***: number of epochs, or another saying complete passes through the entire training data. The model will go through the data 30 times for training
- ***Batch_size = 64***: Determines how many samples will be processed at once for weights (parameters) are updated (backpropagation)
- ***Callbacks***: apply the early stopping method

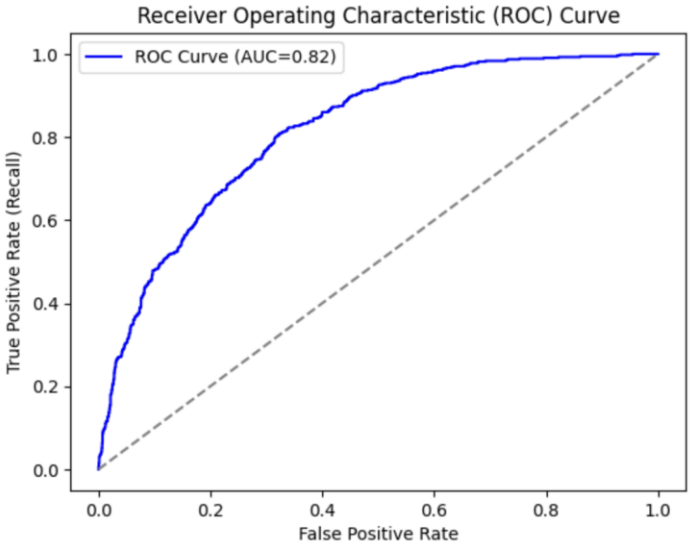


This visualization called **a learning curve**, showing how an ANN model's performance improves through each iterations (epochs). It will show if our model is underfitting, overfitting or properly learning from the data

As we can see from the blue line (training accuracy and loss) keeps increasing due to weights updated. And gradually increase from epochs 10 forward.

However with red line (testing accuracy and loss) starting increasing slightly and almost stop from epochs 5 moving forward.

→ This concluded that our model began to overfitting as our training data increase whiles the testing not.



"The ROC curve shows how well our model predicts customer churn. On the **x-axis**, we have the **False Positive Rate (FPR)**, which tells us how often the model incorrectly predicts that a customer will leave when they actually won't. On the **y-axis**, we have the **True Positive Rate (TPR)**, which shows how many actual churners the model correctly identifies.

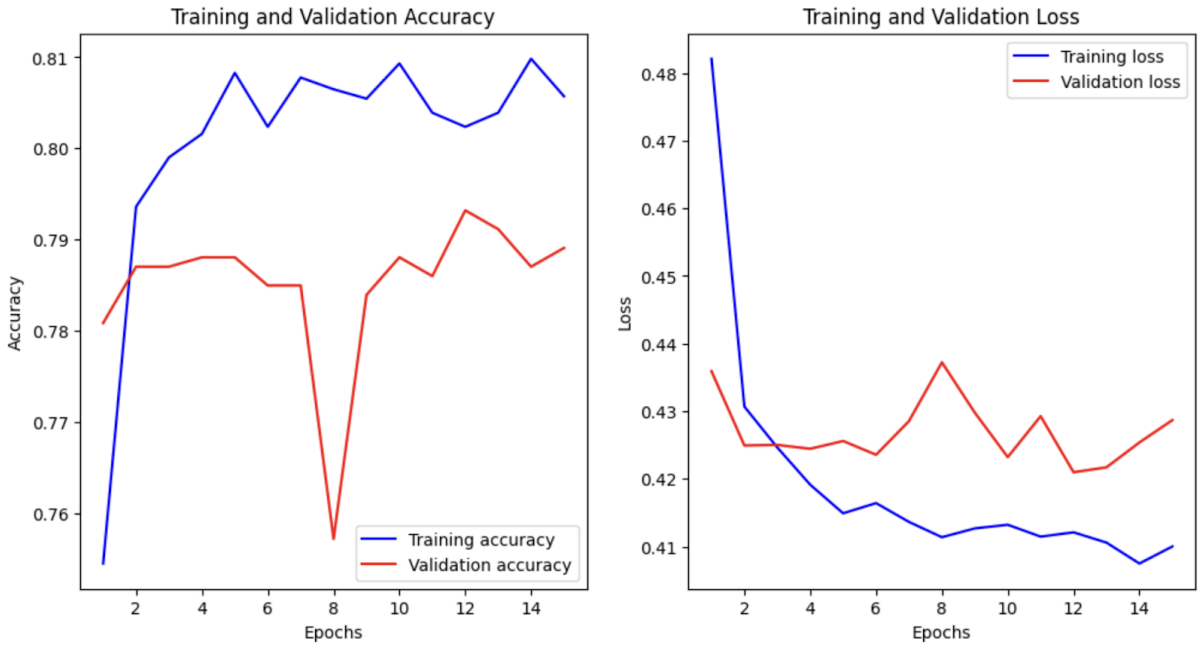
The curve rises well above the diagonal line, which represents guessing randomly. This means our model is good at telling apart customers who will leave from those who will stay. The **Area Under the Curve (AUC)** is **0.82**, which means the model has an 82% chance of correctly distinguishing between a customer who will churn and one who won't.

The ROC curve helps us choose the right threshold for making decisions. For example, if it's more important to catch as many churners as possible, we might lower the threshold to make sure we identify more customers at risk of leaving, even if it means we might flag a few incorrectly.

In summary, this curve shows that our model is effective in predicting which customers are likely to leave, helping us target the right customers for retention efforts."

Key Findings Customer Churn project

After the training step successfully with high accuracy up to 80%. We tried hyper parameter tunning with adding more layers to create more complex ANN model. However, with the visualisation below, we learned a key thing.



→ Model tends to not converge better than the previous one. Even more worse as we can see the validation accuracy and loss are fluctuated everywhere while training curve is smoothly increased in accuracy and decreased in loss

Therefore, we choose our foundation model with 3 layers from the previous training.

Metrics and output performance

```
y_pred_prob = model.predict(X_test)
y_pred = (y_pred_prob >= 0.5).astype(int)

accuracy = accuracy_score(Y_test, y_pred)
conf_matrix = confusion_matrix(Y_test, y_pred)
class_report = classification_report(Y_test, y_pred)
roc_auc = roc_auc_score(Y_test, y_pred_prob)

print(f'Accuracy: {accuracy}')
print(f'Confusion Matrx:\n{conf_matrix}')
print(f'Classification Report:\n{class_report}')
print(f'ROC-AUC Score: {roc_auc}')
```

```
66/66 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Accuracy: 0.7862632084534101
Confusion Matrx:
[[1374  165]
 [ 280  263]]
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.89      0.86      1539
           1       0.61      0.48      0.54       543

    accuracy                           0.79      2082
   macro avg       0.72      0.69      0.70      2082
weighted avg       0.77      0.79      0.78      2082

ROC-AUC Score: 0.8154286883568651
```

- After the training, we used the model to perform analysis on the prediction. Let's look at the **precision** metrics.

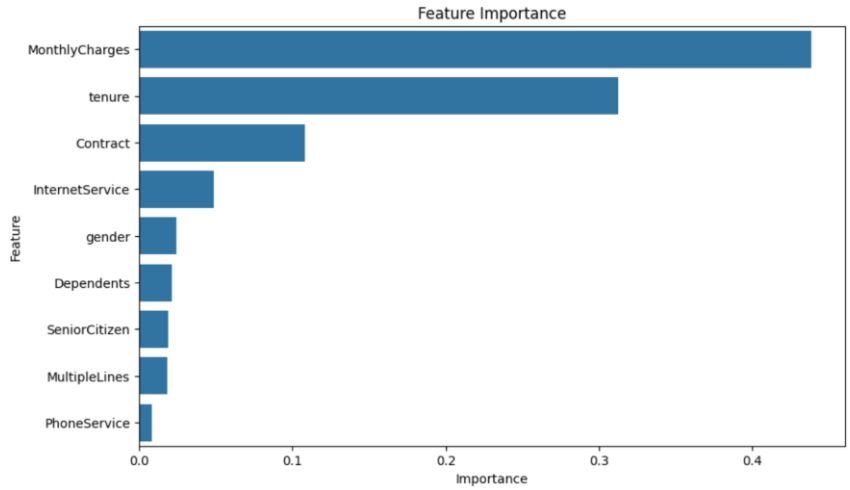
- **Precision score of 0 (83%)**: 83% of the instances predicted as 0 were correctly classified. The model is doing a good job in avoiding false positives when predicting class 0.
- **Precision score of 1 (61%)**: 61% of the instances predicted as 1 were correct. It could suggest that class 1 is harder to distinguish, or that the class distribution is imbalanced, leading to more confusion.

Same concept applied with recall and f1-score

→ Therefore, there is an unbalanced data in this project and we either have 2 options:

- Collect more data of 1 label
- Adding more weights update for the class 1.

Factors contributing to churn and retention



1. Monthly Charges:

- Higher monthly charges are correlated with increased churn rates. This suggests that customers may be sensitive to pricing, especially when monthly costs are higher.

2. Tenure:

- Longer-tenured customers are less likely to churn, indicating that as customers remain longer, they develop loyalty and are more satisfied with the service.

3. Contract Length:

- Customers on longer contracts (One year or Two year) show a significantly lower likelihood of churning, suggesting that contract commitment reduces churn rates.

4.. Internet Service Type:

- Customers using fiber optic internet show a higher likelihood of churning. This could be due to perceived quality or cost issues compared to DSL users.

5. Senior Citizen Status:

- Senior citizens are more likely to churn, possibly due to lower technology usage or budget constraints.

6. Dependents:

- Customers with dependents are less likely to churn, indicating that households with dependents might prioritize stability and longer relationships with service providers.

Recommendations for Targeted Retention Strategies:

1. Review and Adjust Pricing Strategies:

- Consider offering discounted pricing or customized packages for customers with higher monthly charges, especially those nearing the end of their contracts.

2. Retention Programs for New Customers:

- Since tenure is negatively correlated with churn, focus on retention efforts for new customers within their first few months to ensure they remain long-term clients.

3. Promote Long-Term Contracts:

- Encourage customers to switch to longer-term contracts through promotions, loyalty programs, or added incentives. This would reduce churn by increasing contract duration.

4. Improve Fiber Optic Service Quality:

- Since fibre optic users have a higher churn rate, investigate and address any service quality or pricing concerns for these customers to improve satisfaction and reduce churn.

5. Target Senior Citizens with Tailored Offers:

- Develop targeted offers for senior citizens that could include simplified packages, discounts, or customer service aimed at their needs to improve retention.

How the influencing factors were calculated.

1. Loading and Preparing the Data

The customer churn dataset is first imported using pandas. Since many features in the dataset are categorical, such as "gender," "Dependents," and "InternetService," it is necessary to convert them into numerical values. Label Encoding is applied to transform these categories into a format that a machine learning model can interpret. The target variable, "Churn," is also encoded, where 0 represents non-churn and 1 represents churn.

2. Splitting the Data into Training and Testing Sets

Next, the dataset is split into independent features (X) and the target variable (y):

- **X** includes independent variables such as "Contract" and "Phone Service."
- **y** holds the target variable, "Churn."

The data is then divided into training and testing sets, with 70% of the data is kept for model training and 30% set aside for testing. This makes sure that the model's performance can be properly evaluated on unseen data and the accuracy is maintained.

3. Training the Random Forest Classifier

A **Random Forest Classifier** is chosen for modeling. This ensemble method constructs multiple decision trees and combines their outputs to enhance prediction accuracy. It is particularly suitable for datasets that include a mix of categorical and numerical variables, such as this one.

The model is trained using the training data (X_train and y_train), followed by predictions on the test set (X_test). Additionally, class probabilities are generated using the predict_proba method.

4. Evaluating the Model

Feature Importance

The model provides information on feature importance, highlighting which variables are most influential in predicting customer churn. A bar plot can be used to visualize the importance of each feature, with more significant features such as "Contract" and "InternetService" ranking higher.

Confusion Matrix

A **confusion matrix** is generated to evaluate the model's classification performance. The matrix categorizes the results as:

- **True Positives (TP)**: Correctly predicted churn cases.
- **True Negatives (TN)**: Correctly predicted non-churn cases.
- **False Positives (FP)**: Instances where churn was predicted, but did not occur.
- **False Negatives (FN)**: Instances where churn occurred, but was not predicted.

This breakdown offers a clear view of where the model performs well and where errors are being made.

Accuracy

The model's accuracy is then computed, providing the percentage of correct predictions. For example, an accuracy of 82% indicates that 82% of the predictions made by the model are correct and it correctly identifies the categorical values.

5. Visualizations

Key visualizations include:

- A **Feature Importance Plot** showing the most significant features in predicting churn.
- A **Confusion Matrix** offering a visual representation of the correct and incorrect predictions.

Limitations and Proposed Solutions

Limitations:

1. **Data Quality**: If the dataset contains missing or inaccurate values, it can lead to unreliable predictions and skewed results.
2. **Overfitting**: The complex architecture of the neural network may result in overfitting, meaning it performs well on training data but poorly on unseen data.

3. **Interpretability**: Neural networks can act as "black boxes," making it difficult to understand how decisions are made, which is essential for stakeholders.
4. **Feature Selection**: Relying solely on selected features might miss important variables that influence customer churn, limiting the model's effectiveness.
5. **Generalization**: The model may not perform well across different customer segments or under varying conditions, particularly if the training data lacks diversity.

Proposed Solutions:

1. **Enhance Data Quality**: Implement thorough data cleaning and validation processes to ensure the dataset is accurate and complete.
2. **Use Regularization**: Incorporate dropout layers and early stopping to reduce overfitting and improve the model’s ability to generalize to new data.
3. **Improve Interpretability**: Use techniques like LIME or SHAP to provide insights into model predictions, making it easier for stakeholders to understand the results.
4. **Feature Engineering**: Conduct exploratory data analysis (EDA) to identify additional relevant features and ensure the model captures all important variables.
5. **Cross-Validation**: Utilize k-fold cross-validation to assess the model's robustness and ensure consistent performance across different subsets of data.

By addressing these limitations with targeted solutions, we can improve our model's performance and reliability in predicting customer churn, ultimately enhancing our retention strategies.