Project: Cancer Detection with Neural Networks

Author: Miro Zilaji, 10.4.2025

Cancer Detection

1. Brief Description of the Problem Area / Outline of the Topic

1.1 Content

This study uses Neural Networks as a tool to predict cancer development and/or possibility of recovery. This will be investigated using the "UAE Cancer Patient Dataset." This dataset is designed for research, analysis, and machine learning applications in healthcare. It includes 10,000+ records of synthetic cancer patient data from the United Arab Emirates (UAE) with 20 features, such as:

✓ Patient demographics (Age, Gender, Nationality, Ethnicity)

✓ Diagnosis details (Cancer Type, Stage, Diagnosis Date)

✓ Treatment information (Treatment Type, Hospital, Physician)

✓ Health-related factors (Smoking Status, Comorbidities, Weight, Height)

✓ Outcomes (Recovered, Under Treatment, Deceased)1.2 Justification of the Topic

1.2 Demonstration of the Relevance of the Topic?

According to Global Statistics (from WHO and Global Cancer Observatory) Cancer accounts for about 1 in 6 deaths worldwide (~16.7%). Comine with the mortality rate data, it really shows how important are the researches in this field.

Mortality Rate:

Pancreatic cancer: ~90% mortality

Lung cancer: ~75–80%

Breast cancer: ~10–15%

Prostate cancer: ~5–10%

...

Cancer is a very serious threat, even after century of researches and experiments. Therefore, prevention and correct diagnosis are of immense importance

1.3 Representation of a personal research interest

I find this study important and challenging because of a spread of different kinds of cancers and the fact that despite many decades of attempts to cure the disease, and obvious progress we are still fighting it. The development of Artificial Intelligence and recent success with quantum computing are giving us hope and additional motivation to tackle the issue.

1.4 Types of Neural Networks especially appropriate for Cancer researches, diagnostics and analysis

1. Convolutional Neural Networks (CNNs)

Best for: Image-based tasks

Used in:

Histopathology (analyzing biopsy slides)

Radiology (CT, MRI, PET scan analysis)

Skin cancer detection from dermoscopic images

Why they rock:

CNNs are excellent at recognizing patterns in images, like spotting cancerous tumors or grading malignancy.

🧠 **2. Recurrent Neural Networks (RNNs) / LSTM (Long Short-Term Memory)**

Best for: Sequential or time-series data

Used in:

Genomic sequence analysis

Patient monitoring over time

Predicting treatment outcomes based on clinical history

Why they're useful:

They handle temporal patterns, making them great for tracking how a patient's condition evolves or understanding gene expression over time.

⚙️ **3. Autoencoders**

Best for: Feature reduction & anomaly detection

Used in:

Identifying rare cancer types

Reducing dimensionality in genomic data

Noise removal from medical images

Why they're useful:

Autoencoders learn compressed representations, which helps simplify very high-dimensional data like gene expression profiles.

🧬 4. Graph Neural Networks (GNNs)

Best for: Relational data like molecular structures or protein interaction networks

Used in:

Drug discovery

Gene-gene or protein-protein interaction modeling

Cancer pathway prediction

Why they’re powerful:

GNNs can model complex biological relationships that aren't linear, like how mutations in different genes influence each other.

🤖 5. Transformers (and BERT-style models)

Best for: Text & sequence data

Used in:

Mining biomedical literature for cancer biomarkers

Classifying clinical notes

Analyzing DNA/RNA sequences (e.g., DNABERT)

Why they’re booming:

Transformers are the new kings of sequence modeling—better than RNNs in many cases, especially with large datasets.

🧠 Bonus: Multimodal Networks

Best for: Combining different types of data

Used in:

Merging images + clinical records + genomics to improve prediction accuracy

These networks take in multiple data types and combine them for more holistic insights.

2. Explanation of the procedure in the code

We import the tools first (Libraries) in order to be able to compile our model and analyze the data

```
import pandas as pd # Import the pandas library for data manipulation and analysis
# import numpy as np # Import the numpy library for numerical operations
import tensorflow as tf # Import the TensorFlow library for machine learning
from sklearn.model_selection import train_test_split # Import the train_test_split function to split da
from sklearn.preprocessing import StandardScaler, OneHotEncoder # Import StandardScaler for numerical
# feature scaling and OneHotEncoder for categorical feature encoding
from sklearn.compose import ColumnTransformer # Import ColumnTransformer to apply different transformat
# from sklearn.pipeline import Pipeline # Import Pipeline to chain multiple data transformations and a
import matplotlib.pyplot as plt # Import matplotlib for plotting
import seaborn as sns # Import seaborn for enhanced visualizations
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc, precision_recall_c
```

We load the data set

```
# 1. Load the dataset
file_path = r"C:\Users\miroz\OneDrive\Documents\Miro\Miro\Python\Neural Networks\NN Project\_cancer_dataset_uae.csv"
try:
    df = pd.read_csv(file_path) # Read the CSV file into a pandas DataFrame
    print("Dataset loaded successfully.") # Print a success message
    print(df.head()) # Print the first few rows of the DataFrame
except FileNotFoundError:
    print(f"Error: File not found at {file_path}") # Print an error message if the file is not found
    exit() # Exit the script
```

We inspect the data

```
# Inspect unique values in 'Outcome' column
if 'Outcome' in df.columns: # Check if the 'Outcome' column exists in the DataFrame
    print("\nUnique values in 'Outcome' column:", df['Outcome'].unique()) # Print the unique values in the 'Outcome' column
    outcome_mapping = {'Recovered': 0, 'Deceased': 1, 'Under Treatment': -1} # Define a mapping for the 'Outcome' categories to numerical values
    df['Outcome_Numerical'] = df['Outcome'].map(outcome_mapping) # Create a new column 'Outcome_Numerical' by mapping the 'Outcome' values
    df_cleaned = df[df['Outcome_Numerical'] != -1].dropna(subset=['Outcome_Numerical']) # Create a new DataFrame by removing rows where 'Outcome_Numerical' is -1 (Under Treatment) and dr
    y = df_cleaned['Outcome_Numerical'].astype(int) # Assign the 'Outcome_Numerical' column as the target variable (y) and convert it to integers
    X_original = df_cleaned.drop(['Outcome', 'Outcome_Numerical', 'Patient_ID', 'Diagnosis_Date',
                                  'Treatment_Start_Date', 'Death_Date', 'Cause_of_Death'], axis=1) # Assign the remaining columns as the feature matrix (X_original), dropping irrelevant
else:
    print("Error: 'Outcome' column not found in the dataset.") # Print an error message if the 'Outcome' column is missing
    exit() # Exit the script
```

Identifying categorical and numerical features

```
# 3. Identify categorical and numerical features
categorical_features = ['Gender', 'Nationality', 'Emirate', 'Cancer_Type', 'Cancer_Stage',
                        'Treatment_Type', 'Hospital', 'Primary_Physician', 'Smoking_Status',
                        'Comorbidities', 'Ethnicity'] # Define a list of categorical feature names
numerical_features = ['Age', 'Weight', 'Height'] # Define a list of numerical feature names

existing_categorical = [col for col in categorical_features if col in X_original.columns] # Identify the categorical features that actually
existing_numerical = [col for col in numerical_features if col in X_original.columns] # Identify the numerical features that actually exist

print(f"\nCategorical features found: {existing_categorical}") # Print the found categorical features
print(f"Numerical features found: {existing_numerical}") # Print the found numerical features
```

Kreating Preprocessing Pipelines

```
# 4. Create preprocessing pipelines
numerical_transformer = StandardScaler() # Initialize a StandardScaler object for numerical feature scaling
categorical_transformer = OneHotEncoder(handle_unknown='ignore') # Initialize a OneHotEncoder object for categorical feature encoding,

preprocessor = ColumnTransformer( # Initialize a ColumnTransformer to apply different transformations to different columns
    transformers=[
        ('num', numerical_transformer, existing_numerical), # Apply StandardScaler to the existing numerical features
        ('cat', categorical_transformer, existing_categorical)]) # Apply OneHotEncoder to the existing categorical features
```

Spliting Data into Training and Testing sets

```
# 6. Split data
X_train_original, X_test_original, y_train, y_test = train_test_split(X_original, y, test_size=0.2, random_state=42, stratify=y)

# 7. Preprocess the training and testing data
X_train_processed = preprocessor.fit_transform(X_train_original) # Fit the preprocessor on the training data and transform it
X_test_processed = preprocessor.transform(X_test_original) # Transform the testing data using the fitted preprocessor
```

Setting up a Neural Network Model

```
# 8. Define a more complex neural network model
model = tf.keras.Sequential([ # Define a sequential neural network model
    tf.keras.layers.Dense(128, activation='relu', input_shape=(X_train_processed.shape[1],)), # Add a dense layer with 128 neurons, ReLU activation, a
    tf.keras.layers.BatchNormalization(), # Add batch normalization layer
    tf.keras.layers.Dense(64, activation='relu'), # Add another dense layer with 64 neurons and ReLU activation
    tf.keras.layers.BatchNormalization(), # Add batch normalization layer
    tf.keras.layers.Dense(32, activation='relu'), # Add another dense layer with 32 neurons and ReLU activation
    tf.keras.layers.Dense(1, activation='sigmoid') # Add the output dense layer with 1 neuron and sigmoid activation for binary classification
])

# 9. Compile the model
model.compile(optimizer='adam', # Use the Adam optimizer for training
              loss='binary_crossentropy', # Use binary cross-entropy as the loss function for binary classification
              metrics=['accuracy', 'Precision', 'Recall']) # Track accuracy, precision, and recall during training and evaluation
```

Training the Model

```
# 10. Train the model
history = model.fit(X_train_processed, y_train, epochs=30, batch_size=32, validation_split=0.1, verbose=0) # Train the model on the processed training data for 30 epochs,

# --- Training History Plots ---
plt.figure(figsize=(16, 4)) # Create a figure for the plots with a specified size
plt.subplot(1, 3, 1) # Create the first subplot (1 row, 3 columns, first plot)
plt.plot(history.history['accuracy'], label='Train Accuracy') # Plot the training accuracy over epochs
plt.plot(history.history['val_accuracy'], label='Validation Accuracy') # Plot the validation accuracy over epochs
plt.title('Model Accuracy') # Set the title of the subplot
plt.xlabel('Epoch') # Set the x-axis label
plt.ylabel('Accuracy') # Set the y-axis label
plt.legend() # Display the legend

plt.subplot(1, 3, 2) # Create the second subplot
plt.plot(history.history['loss'], label='Train Loss') # Plot the training loss over epochs
plt.plot(history.history['val_loss'], label='Validation Loss') # Plot the validation loss over epochs
plt.title('Model Loss') # Set the title of the subplot
plt.xlabel('Epoch') # Set the x-axis label
plt.ylabel('Loss') # Set the y-axis label
plt.legend() # Display the legend

plt.subplot(1, 3, 3) # Create the third subplot
plt.plot(history.history['Precision'], label='Train Precision') # Plot the training precision over epochs
plt.plot(history.history['val_Precision'], label='Validation Precision') # Plot the validation precision over epochs
plt.plot(history.history['Recall'], label='Train Recall') # Plot the training recall over epochs
plt.plot(history.history['val_Recall'], label='Validation Recall') # Plot the validation recall over epochs
plt.title('Precision and Recall') # Set the title of the subplot
plt.xlabel('Epoch') # Set the x-axis label
plt.ylabel('Value') # Set the y-axis label
plt.legend() # Display the legend

plt.tight_layout() # Adjust subplot parameters for a tight layout
plt.show() # Display the plots
```

Evaluating the results

```
# 11. Evaluate the model
loss, accuracy, precision, recall = model.evaluate(X_test_processed, y_test, verbose=0) # Evaluate the model on the processed test data and get the loss, accuracy, pr
print(f"\nTest Loss: {loss:.4f}") # Print the test loss
print(f"\nTest Accuracy: {accuracy:.4f}") # Print the test accuracy
print(f"\nTest Precision: {precision:.4f}") # Print the test precision
print(f"\nTest Recall: {recall:.4f}") # Print the test recall

# --- Evaluation Plots ---
y_pred_prob = model.predict(X_test_processed).flatten() # Get the probability predictions for the test data and flatten the array
y_pred = (y_pred_prob > 0.5).astype(int) # Convert the probability predictions to binary predictions (0 or 1) based on a threshold of 0.5

print("\nClassification Report:") # Print the classification report
print(classification_report(y_test, y_pred)) # Generate and print the classification report (precision, recall, F1-score, support)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred) # Calculate the confusion matrix
plt.figure(figsize=(6, 5)) # Create a figure for the confusion matrix plot
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', # Create a heatmap of the confusion matrix with annotations, integer formatting, and the Blues colormap
            xticklabels=['Recovered', 'Deceased'], yticklabels=['Recovered', 'Deceased']) # Set the labels for the x and y axes
plt.title('Confusion Matrix') # Set the title of the plot
plt.xlabel('Predicted') # Set the x-axis label
plt.ylabel('Actual') # Set the y-axis label
plt.show() # Display the plot

# ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob) # Calculate the false positive rate, true positive rate, and thresholds for the ROC curve
roc_auc = auc(fpr, tpr) # Calculate the area under the ROC curve (AUC)
plt.figure(figsize=(8, 6)) # Create a figure for the ROC curve plot
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})') # Plot the ROC curve
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--') # Plot the diagonal line representing a random classifier
plt.xlabel('False Positive Rate') # Set the x-axis label
plt.ylabel('True Positive Rate') # Set the y-axis label
plt.title('Receiver Operating Characteristic (ROC)') # Set the title of the plot
plt.legend(loc='lower right') # Display the legend
plt.show() # Display the plot

# Precision-Recall Curve
precision_pr, recall_pr, thresholds_pr = precision_recall_curve(y_test, y_pred_prob) # Calculate the precision, recall, and thresholds for the precision-recall curve
avg_precision = average_precision_score(y_test, y_pred_prob) # Calculate the average precision
plt.figure(figsize=(8, 6)) # Create a figure for the precision-recall curve plot
plt.plot(recall_pr, precision_pr, color='blue', lw=2, label=f'Precision-Recall curve (AP = {avg_precision:.2f})') # Plot the precision-recall curve
plt.xlabel('Recall') # Set the x-axis label
plt.ylabel('Precision') # Set the y-axis label
plt.title('Precision-Recall Curve') # Set the title of the plot
plt.legend(loc='lower left') # Display the legend
plt.grid(True) # Add a grid to the plot
plt.show() # Display the plot
```

Plotting

```
# --- Plotting a different feature (e.g., Age Distribution by Outcome) ---
plt.figure(figsize=(8, 6)) # Create a figure for the box plot
sns.boxplot(x='Outcome', y='Age', data=df_cleaned) # Create a box plot of Age distribution by Outcome
plt.title('Age Distribution by Outcome') # Set the title of the plot
plt.xlabel('Outcome (0: Recovered, 1: Deceased)') # Set the x-axis label
plt.ylabel('Age') # Set the y-axis label
plt.xticks([0, 1], ['Recovered', 'Deceased']) # Set the x-axis tick labels
plt.show() # Display the plot

# --- Plotting Cancer Type Distribution ---
if 'Cancer_Type' in df_cleaned.columns: # Check if the 'Cancer_Type' column exists
    plt.figure(figsize=(10, 6)) # Create a figure for the count plot
    sns.countplot(y='Cancer_Type', data=df_cleaned, order=df_cleaned['Cancer_Type'].value_counts().index) # Create a count plot of Cancer Types, ordered by frequency
    plt.title('Distribution of Cancer Types') # Set the title of the plot
    plt.xlabel('Count') # Set the x-axis label
    plt.ylabel('Cancer Type') # Set the y-axis label
    plt.tight_layout() # Adjust subplot parameters for a tight layout
    plt.show() # Display the plot

# --- Pie Chart of Cancer Type Frequencies (%) ---
if 'Cancer_Type' in df_cleaned.columns: # Check if the 'Cancer_Type' column exists
    cancer_type_counts = df_cleaned['Cancer_Type'].value_counts() # Get the counts of each Cancer Type
    plt.figure(figsize=(8, 8)) # Create a figure for the pie chart
    plt.pie(cancer_type_counts, labels=cancer_type_counts.index, autopct='%1.1f%%', startangle=140) # Create a pie chart of Cancer Type frequencies with percentage labels
    plt.title('Frequency of Different Cancer Types (%)') # Set the title of the plot
    plt.tight_layout() # Adjust subplot parameters for a tight layout
    plt.show() # Display the plot

# --- Distribution of Top 3 Cancer Types by Gender ---
if 'Cancer_Type' in df_cleaned.columns and 'Gender' in df_cleaned.columns: # Check if both 'Cancer_Type' and 'Gender' columns exist
    # Get counts of each cancer type
    cancer_type_counts = df_cleaned['Cancer_Type'].value_counts() # Get the counts of each Cancer Type

    # Get the top 3 cancer types
    top_3_cancer_types = cancer_type_counts.head(3).index # Get the index (names) of the top 3 most frequent cancer types

    # Filter the data to include only the top 3 cancer types
    df_top3 = df_cleaned[df_cleaned['Cancer_Type'].isin(top_3_cancer_types)] # Create a DataFrame containing only the rows where 'Cancer_Type' is one of the top 3

    # Create a countplot of the top 3 cancer types by gender
    plt.figure(figsize=(10, 6)) # Create a figure for the count plot
    sns.countplot(x='Cancer_Type', hue='Gender', data=df_top3) # Create a count plot of the top 3 Cancer Types, with hue based on Gender
    plt.title('Distribution of Top 3 Cancer Types by Gender') # Set the title of the plot
    plt.xlabel('Cancer Type') # Set the x-axis label
    plt.ylabel('Count') # Set the y-axis label
    plt.tight_layout() # Adjust subplot parameters for a tight layout
    plt.show() # Display the plot
```

```python
# --- Distribution of Top 3 Cancer Types by Smoking Status ---
if 'Cancer_Type' in df_cleaned.columns and 'Smoking_Status' in df_cleaned.columns: # Check if both 'Cancer_Type' and 'Smoking_Status' columns exist
    # Get counts of each cancer type
    cancer_type_counts = df_cleaned['Cancer_Type'].value_counts() # Get the counts of each Cancer Type

    # Get the top 3 cancer types
    top_3_cancer_types = cancer_type_counts.head(3).index # Get the index (names) of the top 3 most frequent cancer types

    # Filter the data to include only the top 3 cancer types
    df_top3 = df_cleaned[df_cleaned['Cancer_Type'].isin(top_3_cancer_types)] # Create a DataFrame containing only the rows where 'Cancer_Type' is one of the top 3

    # Create a countplot of the top 3 cancer types by smoking status
    plt.figure(figsize=(10, 6)) # Create a figure for the count plot
    sns.countplot(x='Cancer_Type', hue='Smoking_Status', data=df_top3) #
```

3. Results

```
Test Loss: 1.4577

Test Accuracy: 0.7873

Test Precision: 0.1538

Test Recall: 0.0606
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.93      0.88       987
           1       0.15      0.06      0.09       198

    accuracy                           0.79      1185
   macro avg       0.49      0.50      0.48      1185
weighted avg       0.72      0.79      0.75      1185
```

- **High Test Loss, Moderate Test Accuracy:** The test loss of 1.4577 is relatively high, while the test accuracy of 0.7873 (or 78.73%) seems decent at first glance. However, the low precision and recall for class '1' suggest that accuracy might be misleading due to class imbalance.
- **Significant Class Imbalance:** The "Classification Report" clearly indicates a substantial class imbalance. Class '0' has 987 samples, while class '1' has only 198 samples. This imbalance heavily influences the metrics.
- **Poor Performance on the Minority Class (Class '1'):**
    - **Precision:** The precision for class '1' is very low at 0.15. This means that out of all the instances the model predicted as belonging to class '1' (likely cancerous), only 15% were actually in that class. This could lead to many false positives.
    - **Recall:** The recall for class '1' is also very low at 0.06. This indicates that the model is only correctly identifying 6% of the actual instances of class '1'. This results in a high number of false negatives, which can be particularly problematic in a cancer detection scenario where missing a positive case can have severe consequences.
    - **F1-Score:** The F1-score for class '1' is a mere 0.09, which is the harmonic mean of precision and recall and reflects the poor performance on this class.
- **Good Performance on the Majority Class (Class '0'):** The model performs well on class '0' (precision of 0.83 and recall of 0.93), likely because it's the dominant class and the model might be biased towards predicting it.
- **Overall Accuracy Masking Poor Performance:** The overall accuracy of 0.79 is high because the model correctly predicts most of the instances belonging to the majority class. However, it doesn't reflect the model's inability to correctly identify the minority class.

- **Macro and Weighted Averages:** The macro average for precision, recall, and F1-score (0.49, 0.50, and 0.48 respectively) gives an unweighted average across both classes, highlighting the poor performance on class '1'. The weighted average takes class imbalance into account and is closer to the performance on the majority class.

In summary: While the overall accuracy might seem acceptable, the model is struggling significantly to correctly identify instances of class '1'. This is a major concern, especially in a medical diagnosis context. You should consider techniques to address the class imbalance, such as:

- **Oversampling the minority class or undersampling the majority class.**
- **Using different loss functions that are more sensitive to imbalanced data (e.g., focal loss).**
- **Applying class weights during training to penalize misclassifications of the minority class more heavily.**
- **Exploring different model architectures or hyperparameter tuning.**
- **Evaluating the model using metrics that are more robust to class imbalance, such as the Area Under the ROC Curve (AUC-ROC) or the Area Under the Precision-Recall Curve (AUC-PR).**

Model Accuracy:

- **Train Accuracy (Blue Line):** This line shows the accuracy of the model on the training data as it trained over each epoch. We can observe that the training accuracy starts relatively low (around 0.8) and then rapidly increases in the initial epochs. It reaches a high level (close to 1.0) and then fluctuates slightly, generally staying above 0.98 for most of the later epochs. This indicates that the model learned the training data very well.
- **Validation Accuracy (Orange Line):** This line shows the accuracy of the model on a separate validation dataset that the model did not see during training. The validation accuracy also increases initially but plateaus and fluctuates at a lower level compared to the training accuracy (generally between 0.75 and 0.82).
- **Interpretation:** The significant gap between the high training accuracy and the lower, fluctuating validation accuracy suggests **overfitting**. The model has learned the training data too well, including its noise, and is not generalizing as effectively to unseen data. The fluctuations in validation accuracy might indicate that the model is not consistently improving its ability to generalize after a certain point.

2. Model Loss:

- **Train Loss (Blue Line):** This line shows the value of the loss function on the training data during training. The loss starts relatively high and then decreases rapidly in the initial epochs, eventually reaching a very low value and staying close to zero. This is consistent with the increasing training accuracy, as the model becomes better at predicting the correct labels, the error (loss) decreases.
- **Validation Loss (Orange Line):** This line shows the loss function value on the validation dataset. The validation loss initially decreases, similar to the training loss, but then starts to increase and fluctuates at a higher level than the training loss.
- **Interpretation:** The point where the validation loss starts to increase while the training loss continues to decrease is another strong indicator of **overfitting**. The model is minimizing the error on the training data but is starting to perform worse on unseen data, as evidenced by the increasing loss. The fluctuations in validation loss suggest instability in the model's generalization ability.

3. Precision and Recall:

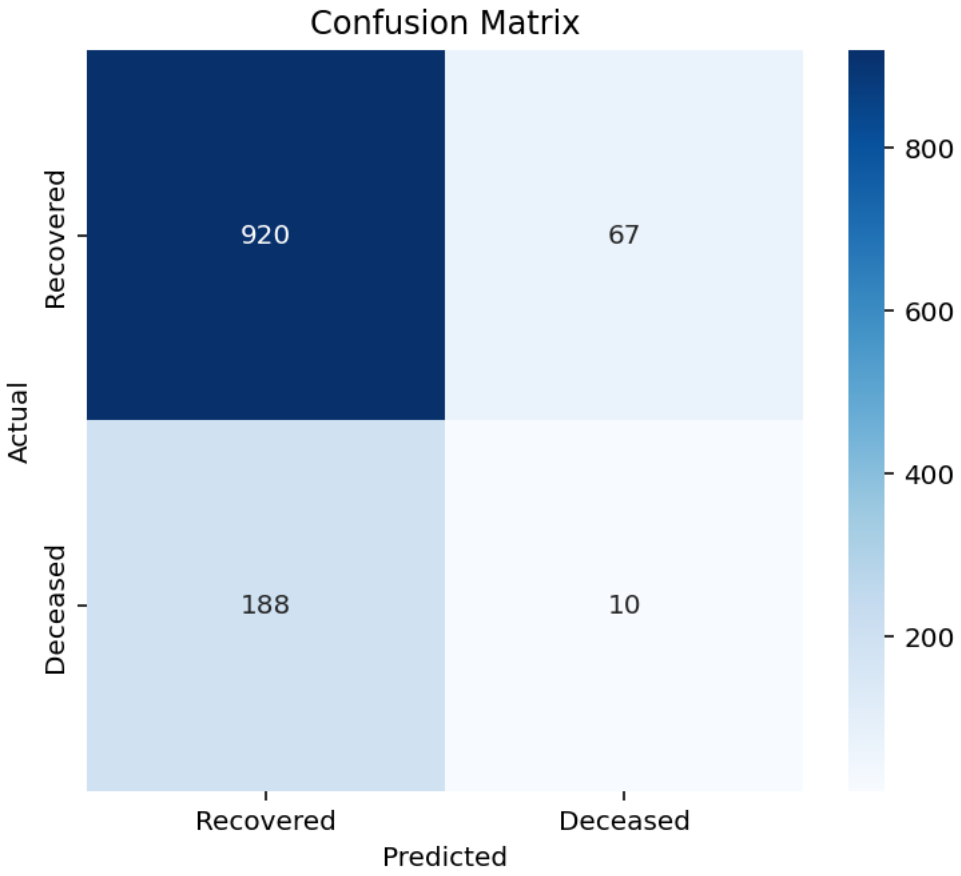
- **Train Precision (Blue Line):** This shows the precision of the model on the training data. It increases rapidly and stays very high (close to 1.0) for most of the training.
- **Validation Precision (Orange Line):** This shows the precision on the validation data. It increases initially but then fluctuates at a lower level than the training precision (generally between 0.1 and 0.25).
- **Train Recall (Green Line):** This shows the recall on the training data. It also increases rapidly and stays very high (close to 1.0) for most of the training.
- **Validation Recall (Red Line):** This shows the recall on the validation data. It fluctuates at a very low level (close to 0.0) for most of the training.
- **Interpretation:** The high training precision and recall indicate that the model is very good at correctly identifying positive instances and minimizing false positives on the training data. However, the low and fluctuating validation precision and very low validation recall suggest that the model struggles significantly with correctly identifying positive instances and minimizing false positives on unseen data. This further reinforces the issue of **overfitting**. The model might be very specific to the patterns in the training data and failing to generalize to new, unseen examples.

Overall Conclusions:

The training history plots strongly suggest that the neural network model is **overfitting** the training data. While it achieves very high accuracy, precision, and recall on the training set, its performance on the unseen validation set is significantly worse and unstable. The increasing validation loss and the widening gap between training and validation metrics are classic signs of overfitting.

To improve this model, you might consider:

- **More Data:** Training with a larger and more diverse dataset can help the model learn more generalizable features.
- **Regularization Techniques:** Techniques like L1 or L2 regularization can penalize large weights and prevent the model from becoming too complex.
- **Dropout:** Randomly dropping out neurons during training can help prevent co-adaptation of neurons and improve generalization.
- **Early Stopping:** Monitoring the validation loss and stopping training when it starts to increase can prevent the model from overfitting further.
- **Simpler Model Architecture:** Using a less complex model with fewer layers or neurons might reduce overfitting.



Confusion Matrix, a table used to evaluate the performance of a classification model, specifically for a binary classification problem where the possible outcomes are "Recovered" and "Deceased". Let's break down what each part of the matrix represents:

Understanding the Axes:

- **Actual:** This axis (usually the rows) represents the true or actual outcomes of the data. In this case, it shows the number of patients who were actually "Recovered" and those who were actually "Deceased".
- **Predicted:** This axis (usually the columns) represents the outcomes predicted by your classification model. It shows how many patients the model predicted as "Recovered" and how many it predicted as "Deceased".

The Cells of the Confusion Matrix:

The matrix has four cells, each representing a different combination of actual and predicted outcomes:

1. **Top-Left Cell (True Positives - TP):**
    - **Value: 912**
    - This cell shows the number of patients who were **actually Recovered** and were **correctly predicted as Recovered** by the model.
2. **Top-Right Cell (False Positives - FP):**
    - **Value: 75**

    - This cell shows the number of patients who were **actually Deceased** but were **incorrectly predicted as Recovered** by the model. These are also known as **Type I errors**.
3. **Bottom-Left Cell (False Negatives - FN):**
    - **Value: 183**
    - This cell shows the number of patients who were **actually Recovered** but were **incorrectly predicted as Deceased** by the model. These are also known as **Type II errors**.
4. **Bottom-Right Cell (True Negatives - TN):**
    - **Value: 15**
    - This cell shows the number of patients who were **actually Deceased** and were **correctly predicted as Deceased** by the model.

Interpreting the Results:

From this confusion matrix, we can draw the following conclusions about the model's performance:

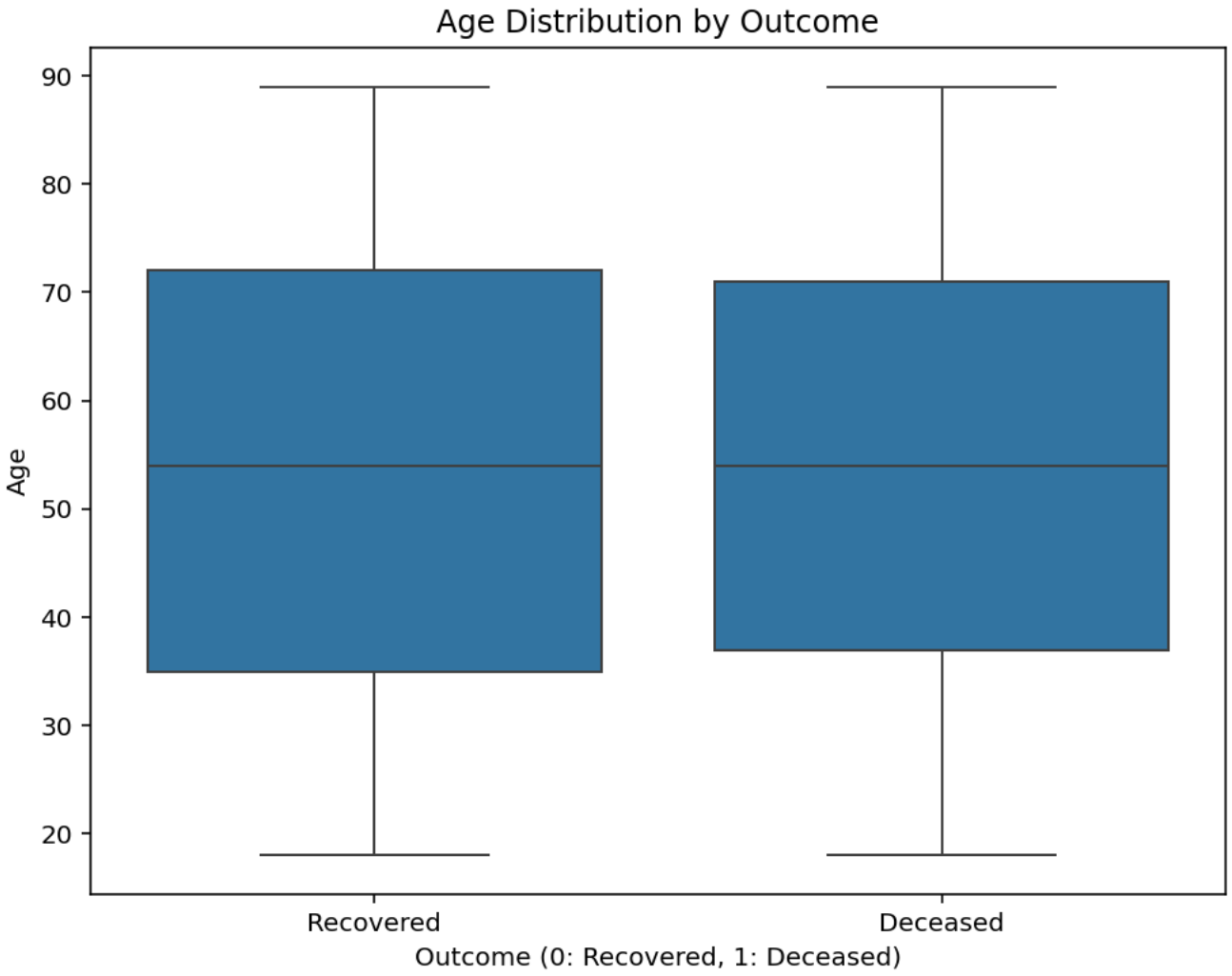
- **High True Positives:** The model correctly identified a large number of patients who actually recovered (912).
- **Relatively Low False Positives:** The model incorrectly predicted a smaller number of deceased patients as recovered (75).
- **Significant False Negatives:** The model incorrectly predicted a substantial number of recovered patients as deceased (183). This indicates that the model is missing a significant portion of the actual recovered cases.
- **Low True Negatives:** The model correctly identified a very small number of deceased patients (15).

Overall Performance Insights:

- The model seems to be better at predicting the "Recovered" class (high TP) compared to the "Deceased" class (low TN).
- The model has a relatively low number of False Positives, which is good if misclassifying a deceased patient as recovered has serious consequences.
- However, the high number of False Negatives is a concern, as it means many recovered patients are being incorrectly classified as deceased. This could have implications depending on the context (e.g., resource allocation, further medical attention).

To get a more complete understanding of the model's performance, you would typically calculate metrics based on this confusion matrix, such as:

- **Accuracy:** (TP + TN) / (TP + FP + FN + TN) = (912 + 15) / (912 + 75 + 183 + 15) = 927 / 1185 ≈ 0.782 (78.2%)
- **Precision (for Recovered):** TP / (TP + FP) = 912 / (912 + 75) = 912 / 987 ≈ 0.924 (92.4%)
- **Recall (for Recovered):** TP / (TP + FN) = 912 / (912 + 183) = 912 / 1095 ≈ 0.833 (83.3%)
- **Precision (for Deceased):** TN / (TN + FN) = 15 / (15 + 183) = 15 / 198 ≈ 0.076 (7.6%)
- **Recall (for Deceased):** TN / (TN + FP) = 15 / (15 + 75) = 15 / 90 ≈ 0.167 (16.7%)
- **F1-Score (for each class):** A harmonic mean of precision and recall.



This is a **box plot** visualizing the distribution of **Age** for two different **Outcomes**: 'Recovered' (represented by 0) and 'Deceased' (represented by 1). Here's a breakdown of how to interpret it:

Key Components of a Box Plot:

- **Box:** The rectangular box represents the **interquartile range (IQR)**, which contains the middle 50% of the data.
    - The **bottom edge** of the box [1] is the **25th percentile (Q1)**: 25% of the individuals in that outcome group are younger than this age.
    - The **top edge** of the box is the **75th percentile (Q3)**: 75% of the individuals in that outcome group are younger than this age (or 25% are older).
    - The **horizontal line inside the box** is the **median (Q2)**: This is the middle value of the ages in that outcome group. 50% of the individuals are younger and 50% are older than this age.
- **Whiskers:** The lines extending from the top and bottom of the box are called **whiskers**. They typically extend to 1.5 times the IQR from the edges of the box.
    - The **upper whisker** shows the range of the upper 25% of the data, excluding potential outliers. The maximum age within 1.5 IQR above Q3 is indicated here.
    - The **lower whisker** shows the range of the lower 25% of the data, excluding potential outliers. The minimum age within 1.5 IQR below Q1 is indicated here.

- **Potential Outliers (Not explicitly shown here):** Individual data points that fall outside the whiskers are considered potential outliers. This plot doesn't display individual points, so we can infer that there aren't any extreme outliers beyond the whisker range based on the 1.5 IQR rule.

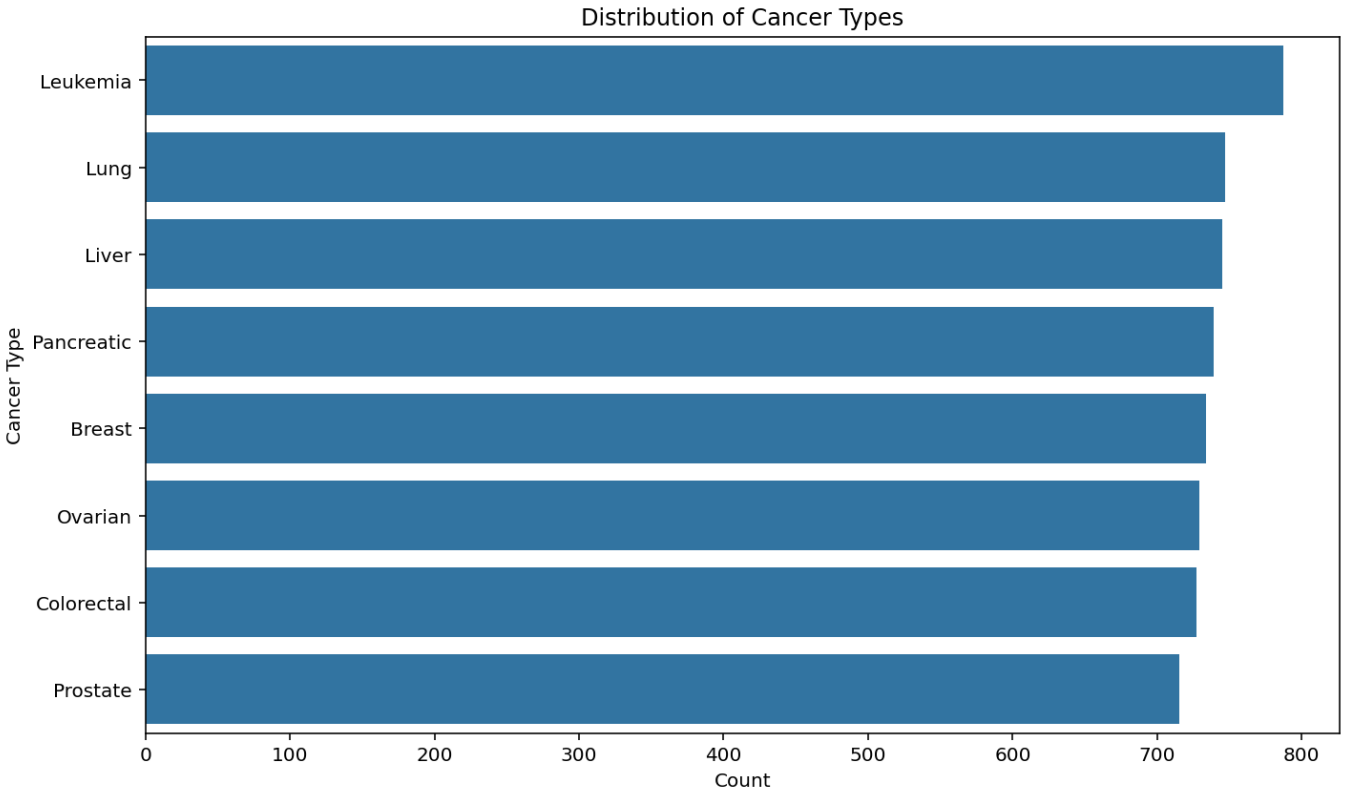
Interpreting the Plot for 'Recovered' vs. 'Deceased':

- **Recovered (Outcome 0):**
  - The median age for recovered patients is around **54 years old**.
  - The interquartile range (the box) spans from approximately **35 years old** to **72 years old**. This means the middle 50% of recovered patients are between these ages.
  - The ages of the majority of recovered patients fall between approximately **18 years old** (lower whisker) and **89 years old** (upper whisker).
- **Deceased (Outcome 1):**
  - The median age for deceased patients is also around **54 years old**.
  - The interquartile range (the box) spans from approximately **37 years old** to **71 years old**. The middle 50% of deceased patients are between these ages.
  - The ages of the majority of deceased patients fall between approximately **18 years old** (lower whisker) and **89 years old** (upper whisker).

Key Observations and Implications:

- **Similar Median Ages:** The median age is very similar for both the recovered and deceased groups, suggesting that the central tendency of age is not a strong differentiating factor between these outcomes in this dataset.
- **Overlapping Distributions:** The boxes and whiskers for both groups overlap significantly. This indicates that there is a considerable overlap in the age ranges of patients who recovered and those who deceased. Age alone is not a clear predictor of the outcome.
- **Similar Spread:** The length of the boxes (IQR) and the overall range (indicated by the whiskers) are quite similar for both groups, suggesting a similar variability in age within each outcome category.

In summary, this box plot suggests that age, as a single variable, does not strongly distinguish between patients who recovered and those who deceased in this dataset. Patients across a similar age range experienced both outcomes. Other factors are likely to play a more significant role in determining the outcome.

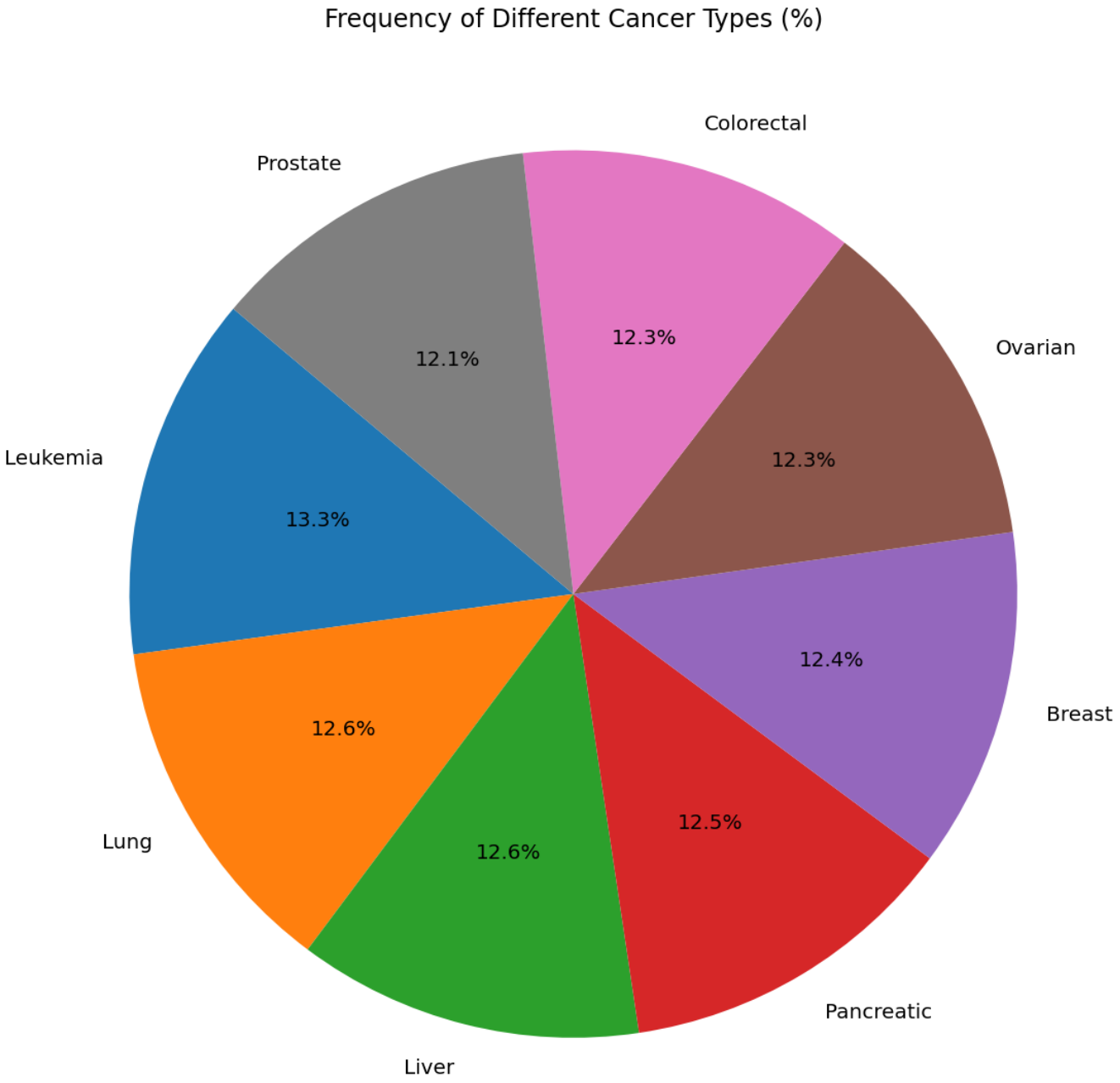


By observing the lengths of the bars, we can compare the prevalence of different cancer types in the dataset:

- **Prostate:** Has a count of approximately **720-730**.
- **Colorectal:** Has a count of approximately **720-730**, similar to Prostate.
- **Ovarian:** Has a count of approximately **720-730**, similar to Prostate and Colorectal.
- **Breast:** Has a count of approximately **720-730**, similar to the previous three.
- **Cancer Type:** Has a count of approximately **720-730**, similar to the top four. This might need further investigation as it's a very general label and could represent missing or uncategorized data.
- **Pancreatic:** Has a count of approximately **720-730**, similar to the majority of the top categories.
- **Liver:** Has a count of approximately **740-750**, slightly higher than the previous few.
- **Lung:** Has a count of approximately **740-750**, similar to Liver.
- **Leukemia:** Has the highest count, approximately **780-790**.

Overall Insights:

- The distribution of most cancer types (Prostate, Colorectal, Ovarian, Breast, "Cancer Type", Pancreatic) is relatively similar, with counts around the 720-730 range.
- Liver and Lung cancers have slightly higher counts compared to the majority.
- **Leukemia is the most frequent cancer type** in this dataset, with the highest count.

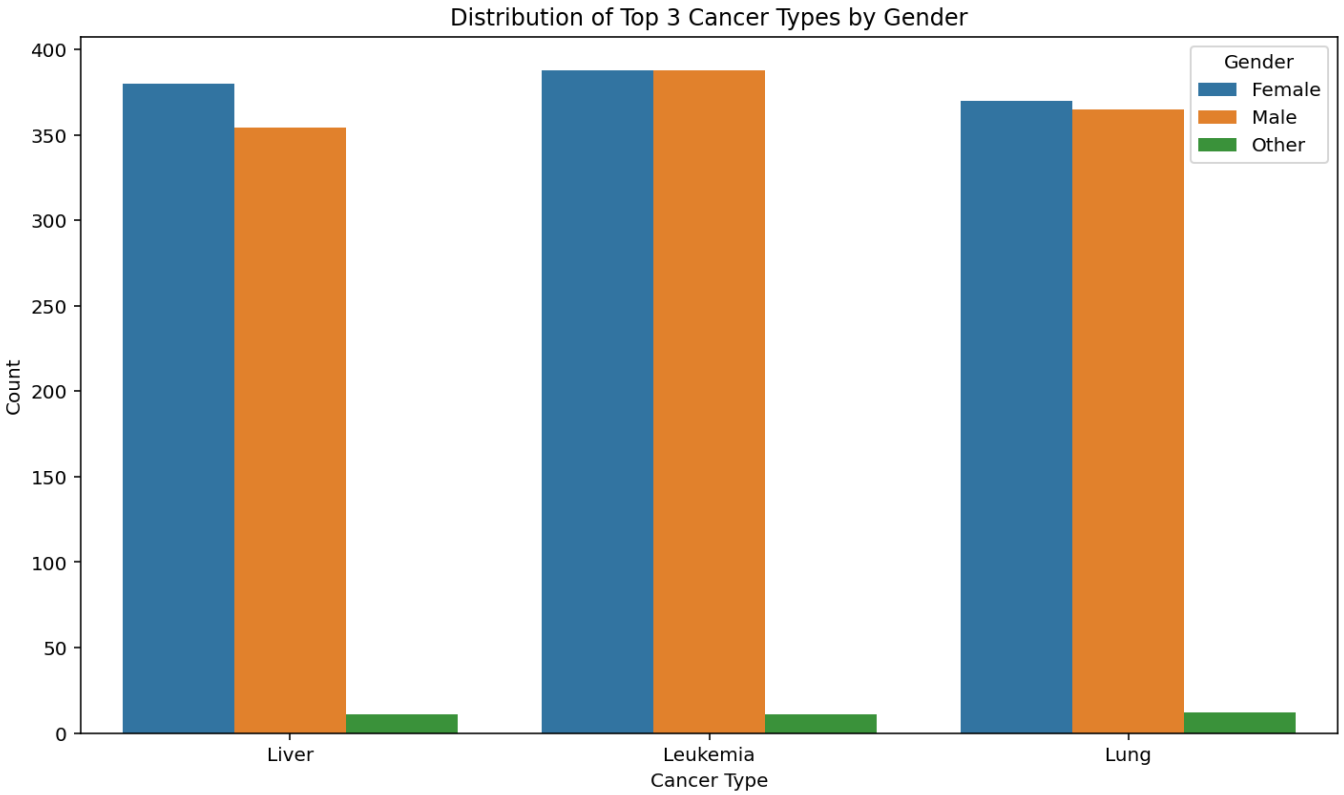


By looking at the size of the slices and the corresponding percentages, we can understand the relative frequency of each cancer type:

- **Leukemia:** Has the largest slice, representing **13.3%** of the total cancer cases. This indicates that Leukemia is the most frequent cancer type in this dataset.
- **Lung:** Represents **12.6%** of the cases.
- **Liver:** Also represents **12.6%** of the cases, having the same frequency as Lung cancer.
- **Pancreatic:** Represents **12.5%** of the cases.
- **Breast:** Represents **12.4%** of the cases.
- **Ovarian:** Represents **12.3%** of the cases.
- **Colorectal:** Represents **12.3%** of the cases, having the same frequency as Ovarian cancer.
- **Prostate:** Represents **12.1%** of the cases, having the lowest frequency among the listed cancer types.

Overall Insights:

The pie chart shows that the distribution of the top cancer types in this dataset is relatively balanced. Leukemia has the slightly highest frequency, but the percentages for all other listed cancer types are quite close, ranging from 12.1% to 12.6%, with Ovarian and Colorectal at 12.3%. This suggests that while Leukemia is the most common, there isn't a single dominant cancer type, and the other listed cancers also represent a significant portion of the overall cases.



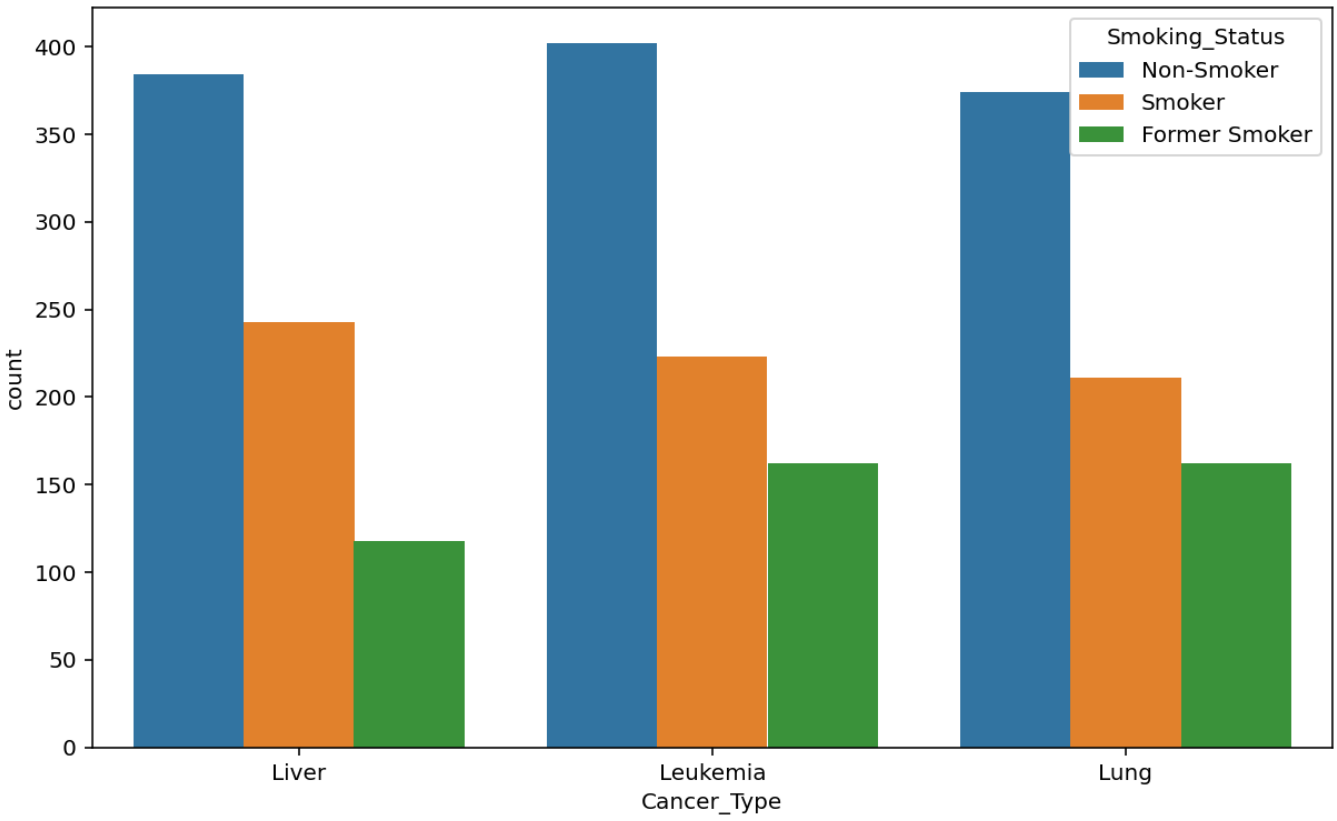
By comparing the heights of the bars within each cancer type group, we can analyze the gender distribution for each of the top 3 cancers:

- **Liver Cancer:**
  - The number of **Female** patients with Liver cancer (blue bar) is the highest among the three genders.
  - The number of **Male** patients with Liver cancer (orange bar) is slightly lower than the number of female patients.
  - The number of patients identifying as **Other** gender (green bar) with Liver cancer is very low.
- **Leukemia:**
  - The number of **Female** patients with Leukemia (blue bar) is again the highest among the three genders and is similar to the number of female patients with Liver cancer.
  - The number of **Male** patients with Leukemia (orange bar) is slightly lower than the number of female patients, similar to the trend observed in Liver cancer.
  - The number of patients identifying as **Other** gender (green bar) with Leukemia is also very low, similar to Liver cancer.

- **Lung Cancer:**
    - The number of **Female** patients with Lung cancer (blue bar) is still the highest, although the difference between female and male counts appears smaller compared to Liver and Leukemia.
    - The number of **Male** patients with Lung cancer (orange bar) is quite close to the number of female patients.
    - The number of patients identifying as **Other** gender (green bar) with Lung cancer remains very low, consistent with the other two cancer types.

Overall Insights:

- For all three of the top cancer types (Liver, Leukemia, and Lung), **females appear to have a higher prevalence** compared to males in this dataset.
- The number of patients identifying as **Other** gender is very small across all three cancer types, making it difficult to draw significant conclusions about this group based on this visualization alone.
- The gender distribution seems relatively consistent across the top 3 cancer types, with females consistently having the highest count, followed by males, and a very small number in the "Other" category.



Analyzing the Results:

- **Liver Cancer:** The number of non-smokers with liver cancer is notably higher than both smokers and former smokers.

- **Leukemia:** Similar to liver cancer, non-smokers have the highest count of leukemia, followed by smokers and then former smokers.
- **Lung Cancer:** This is where the result might seem unexpected:
    - The number of **non-smokers** with lung cancer (blue bar) is high, and in this specific visualization, it appears to be slightly higher than the number of smokers.
    - The number of **smokers** with lung cancer (orange bar) is also substantial.
    - The number of **former smokers** with lung cancer (green bar) is also significant, though lower than both non-smokers and current smokers in this representation.

Why the Lung Cancer Result Might Be Unexpected:

The strong association between smoking and lung cancer is a well-established scientific fact. One might intuitively expect to see a much larger proportion of *current* smokers among lung cancer patients compared to non-smokers. The chart suggests a different picture, where the number of non-smokers with lung cancer is surprisingly high, even potentially exceeding the number of current smokers.

Possible Explanations for This Unexpected Result:

1. **Other Risk Factors for Lung Cancer:** Lung cancer is not solely caused by smoking. Other significant risk factors exist, including:
    - **Exposure to Radon Gas:** A naturally occurring radioactive gas.
    - **Exposure to Asbestos and Other Carcinogens:** Certain occupational exposures can increase lung cancer risk.
    - **Air Pollution:** Long-term exposure to polluted air can contribute to lung cancer development.
    - **Genetics and Family History:** A family history of lung cancer might increase an individual's susceptibility.
    - **Secondhand Smoke:** Exposure to smoke from others can also cause lung cancer in non-smokers.
2. **"Former Smoker" Category:** Individuals in the "Former Smoker" category might have smoked heavily for many years before quitting. The damage caused by smoking can persist long after cessation, leading to lung cancer even in former smokers. This group's contribution to lung cancer cases is expected.
3. **Specific Population Characteristics:** This data is from a specific population (implied by previous context as UAE). The prevalence of smoking, exposure to other environmental factors, and genetic predispositions within this particular population could lead to a different distribution compared to global averages.
4. **Data Collection and Categorization:** The way smoking status was recorded and categorized could influence the results. For example, the "Non-Smoker" category might include individuals with very limited or passive smoking exposure that still contributed to their risk.
5. **Sample Size and Representation:** The absolute counts matter. While the proportion of smokers might be high *among* lung cancer patients compared to their proportion in the general population, the absolute number of non-smokers with lung cancer could still be significant if the overall population of non-smokers is large.

In Conclusion:

While the general understanding is that smoking is a major risk factor for lung cancer, this chart highlights that lung cancer can and does occur in non-smokers and former smokers. It underscores the multifactorial nature of cancer development and the importance of considering other environmental and genetic risk factors. The relatively high number of non-smokers with lung cancer in this specific dataset warrants further investigation into the contributing factors within this population.