Lung Cancer Survival Prediction Model

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from xgboost import XGBClassifier
```

In [21]:
```
# Load the processed dataset
df = pd.read_csv("processed_dataset_med.csv")
```

In [22]:
```
# Function to preprocess non-numeric columns
def preprocess_data(df):
    for col in df.select_dtypes(include=["object"]).columns:
        try:
            # Convert date columns to numerical timestamps
            df[col] = pd.to_datetime(df[col]).astype(int) / 10**9
        except Exception:
            # Convert categorical text columns to numerical categories
            df[col] = df[col].astype("category").cat.codes
    return df

# Apply preprocessing
df = preprocess_data(df)
```

In [23]:
```
# Define features (X) and target (y)
X = df.drop(columns=["survived"])  # Assuming 'survived' is the target column
y = df["survived"]
```

In [24]:
```
# Split data into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [25]:
```
# Scaling the features for better performance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [26]:
```
# Train Logistic Regression model
log_model = LogisticRegression()
log_model.fit(X_train, y_train)
```

Out[26]:
```
▾ LogisticRegression
LogisticRegression()
```

In [27]:
```
# Predict and evaluate Logistic Regression
y_pred_log = log_model.predict(X_test)
print("Logistic Regression Model:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_log):.2f}")
print("Classification Report:\n", classification_report(y_test, y_pred_log))
```

```
Logistic Regression Model:
Accuracy: 0.78
Classification Report:
               precision    recall  f1-score   support

           0       0.78      1.00      0.88    138694
           1       0.00      0.00      0.00     39306

    accuracy                           0.78    178000
   macro avg       0.39      0.50      0.44    178000
weighted avg       0.61      0.78      0.68    178000
```

```
c:\Users\amanv\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\met
rics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\amanv\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\met
rics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\amanv\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\met
rics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [28]:

```
# Train XGBoost model
xgb_model = XGBClassifier(n_estimators=100, learning_rate=0.1, max_depth=6, random_
xgb_model.fit(X_train, y_train)
```

Out[28]:

▾ XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds
=None,
              enable_categorical=False, eval_metric=None, feature_types
=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=
None,
```

In [29]:

```
# Predict and evaluate XGBoost
y_pred_xgb = xgb_model.predict(X_test)
print("\nXGBoost Model:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_xgb):.2f}")
print("Classification Report:\n", classification_report(y_test, y_pred_xgb))
```

```
XGBoost Model:
Accuracy: 0.78
Classification Report:
               precision    recall  f1-score   support

           0       0.78      1.00      0.88    138694
           1       0.00      0.00      0.00     39306

    accuracy                           0.78    178000
   macro avg       0.39      0.50      0.44    178000
weighted avg       0.61      0.78      0.68    178000
```

```
c:\Users\amanv\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\met
rics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\amanv\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\met
rics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\amanv\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\met
rics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```