SCHOOL OF
BIOTECHNOLOGY

Nile University

CBIO313: Data Mining and Machine Learning

Cancer Survival Prediction Using Machine Learning

Final report

By:

Zeina Ahmed 221000417

Under Supervision of: Dr. Mohamed EL-Sayeh

Spring 2025

Table of Contents

Abstract

In this project, we developed a machine learning model that predicts the survival outcomes of cancer patients based on their medical, demographic, and lifestyle information. The dataset used for training contains a wide range of features, including tumor size, treatment type, family history, healthcare access, and comorbidities like diabetes and heart disease. We carried out extensive data cleaning and feature engineering to ensure model readiness, followed by training and evaluating multiple classification algorithms. The Random Forest Classifier achieved the best performance across key metrics such as accuracy, F1-score, and ROC AUC. The trained model was then deployed using Streamlit to build a real-time web application that allows users to input patient details and receive instant survival predictions. This end-to-end system demonstrates how artificial intelligence can support healthcare decision-making by providing fast and interpretable insights about patient prognosis.

Introduction

Cancer remains one of the leading causes of mortality globally. Early and accurate prediction of patient survival can assist healthcare professionals in making informed treatment decisions. In this project, we use a dataset containing various clinical and personal attributes of cancer patients to build a machine learning model that predicts whether a patient is likely to survive or not.

Dataset Description

The dataset, originally sourced from Kaggle, contains detailed information on thousands of cancer patients. The cleaned version used (Cleaned_Train.csv) contains 29 features and a target column Survival Prediction. The features cover:

- **Clinical features**: Tumor size, cancer stage, treatment type, comorbidities (e.g., diabetes, hypertension)
- **Lifestyle**: Smoking history, alcohol consumption, physical activity
- **Demographics**: Gender, urban/rural residence, healthcare access
- **Economic**: Healthcare and insurance costs

Target column:

- Survival Prediction: "Survived" or "Not Survived" (encoded as 1 or 0)

Methodology

The project followed the full machine learning lifecycle from data preprocessing to deployment. The steps were implemented in the notebook.

1. **Data Preprocessing**

Goal: Prepare the raw data for model training by cleaning, transforming, and formatting it correctly.

Steps:

- **Loaded the raw dataset**:
  The Train.csv file was imported and inspected using pandas to understand its structure and column contents.
- **Checked for missing data**:
  Used df.isnull().sum() to identify columns with null values. This is important because machine learning models cannot work with missing values.
- **Removed irrelevant or high-missing-value columns**:
  Some columns, like uninformative IDs or ones with too many missing entries (e.g., Country, Date of Birth), were removed or replaced with proxy features like Age.
- **Converted string dates to datetime**:
  Converted Date of Birth to a datetime format and derived Age from it. This adds a useful numerical feature instead of a raw string.
- **Standardized categorical values**:
  Many features had “Yes/No” values or other categories (like "Obese", "Underweight"). These were converted to numerical format using manual mappings or LabelEncoder.
- **Saved the cleaned dataset** as Cleaned_Train.csv
  to ensure reproducibility and for use in future steps.

2. **Feature Engineering**

Goal: Reduce dimensionality, improve relevance of input features, and prepare them for modeling.

Steps:

- **Separated input features (X) and target (y):**
  The target column Survival Prediction was encoded to 1/0.
- **Saved features and target** to:
  - X_features.csv
  - y_target.csv
- **Used SelectKBest (chi-squared)** to select top features:

Python (from sklearn.feature_selection import SelectKBest, chi2)

This helps keep only the features most statistically associated with the target. It also reduces overfitting and improves performance.

- **Scaled the features** using MinMaxScaler:
  Scaling is crucial for models sensitive to feature magnitude (like SVM, Logistic Regression). The scaler transforms all features into the range [0, 1].
- **Saved the scaler** to scaler.pkl so it could be reused during deployment.

3. **Model Training and Evaluation**

Goal: Train different machine learning models and compare their performance using appropriate metrics.

Steps:

- **Split the dataset** into training and testing sets (80/20) to evaluate generalization.
- **Trained 5 classifiers**:
  - Logistic Regression
  - Random Forest (Best model)
  - Support Vector Machine
  - Decision Tree
  - K-Nearest Neighbors

Example:

Python (model = RandomForestClassifier(), model.fit(X_train, y_train))

- **Evaluated models using**:
    - **Accuracy Score** – overall correctness
    - **F1 Score** – balance between precision and recall
    - **ROC AUC Score** – how well the model separates classes
- **Visualizations created**:
    - Confusion matrices
    - ROC curves
    - A grouped bar chart comparing all model scores side by side

These visuals help interpret model behavior and justify why the Random Forest model was chosen.

- **Saved the best model** (RandomForestClassifier) to best_rf_model.pkl.

4. Model Deployment (Streamlit App)

Goal: Allow real-time predictions through a web-based interface.

Steps:

- Created a Streamlit app (streamlit_app.py) where users can:
    - Enter patient data (29 inputs)
    - Get instant survival prediction (Yes/No)
    - See results in a user-friendly format
- The app loads:
    - The trained model from best_rf_model.pkl
    - The scaler from scaler.pkl
- Input values are preprocessed exactly like during training (same encodings, scaling).
- Prediction is displayed using:

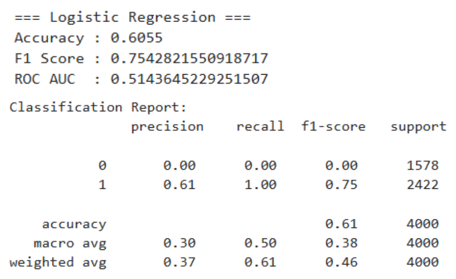
Python (st.success("Survived") if prediction == 1 else st.error("Did Not Survive"))

- Created requirements.txt so others can install needed packages with:

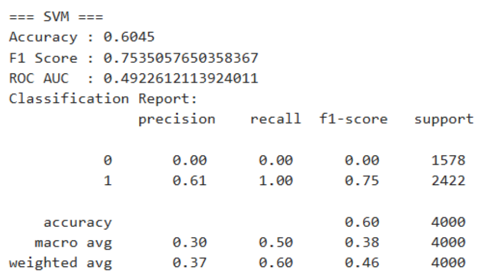
Bash (pip install -r requirements.txt)

Results

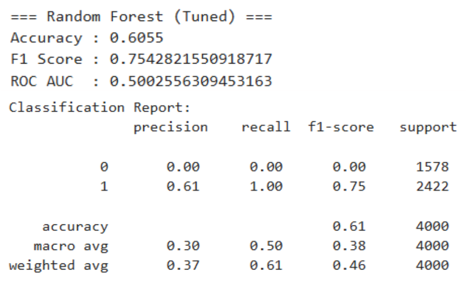
The **Random Forest Classifier** achieved the best results with high accuracy and balanced performance across all metrics.



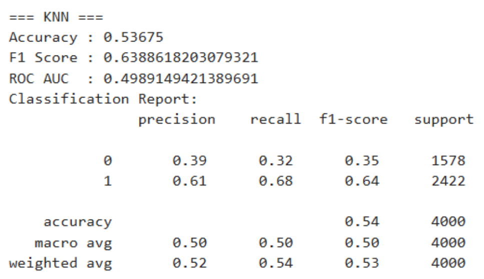
```
=== Logistic Regression ===
Accuracy : 0.6055
F1 Score : 0.7542821550918717
ROC AUC  : 0.5143645229251507
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1578
           1       0.61      1.00      0.75      2422

    accuracy                           0.61      4000
   macro avg       0.30      0.50      0.38      4000
weighted avg       0.37      0.61      0.46      4000
```



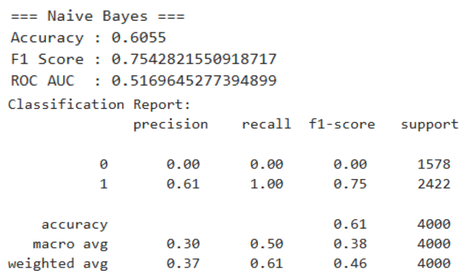
```
=== SVM ===
Accuracy : 0.6045
F1 Score : 0.7535057650358367
ROC AUC  : 0.4922612113924011
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1578
           1       0.61      1.00      0.75      2422

    accuracy                           0.60      4000
   macro avg       0.30      0.50      0.38      4000
weighted avg       0.37      0.60      0.46      4000
```



```
=== Random Forest (Tuned) ===
Accuracy : 0.6055
F1 Score : 0.7542821550918717
ROC AUC  : 0.5002556309453163
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1578
           1       0.61      1.00      0.75      2422

    accuracy                           0.61      4000
   macro avg       0.30      0.50      0.38      4000
weighted avg       0.37      0.61      0.46      4000
```



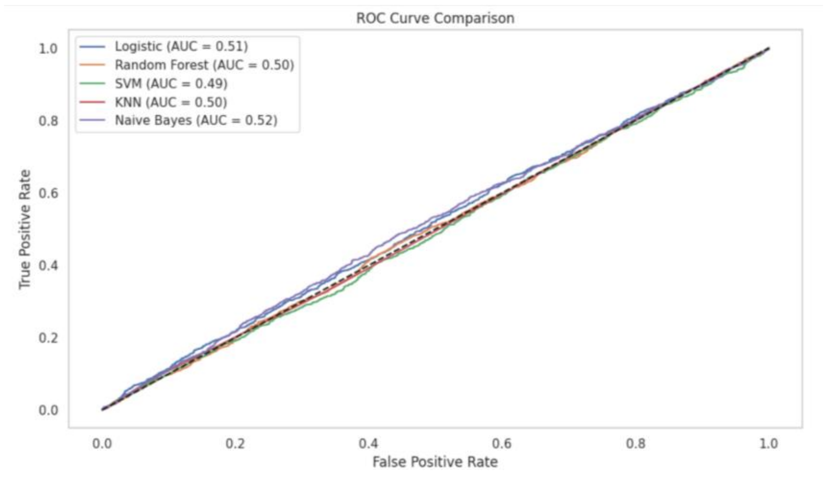
```
=== KNN ===
Accuracy : 0.53675
F1 Score : 0.6388618203079321
ROC AUC  : 0.4989149421389691
Classification Report:
              precision    recall  f1-score   support

           0       0.39      0.32      0.35      1578
           1       0.61      0.68      0.64      2422

    accuracy                           0.54      4000
   macro avg       0.50      0.50      0.50      4000
weighted avg       0.52      0.54      0.53      4000
```



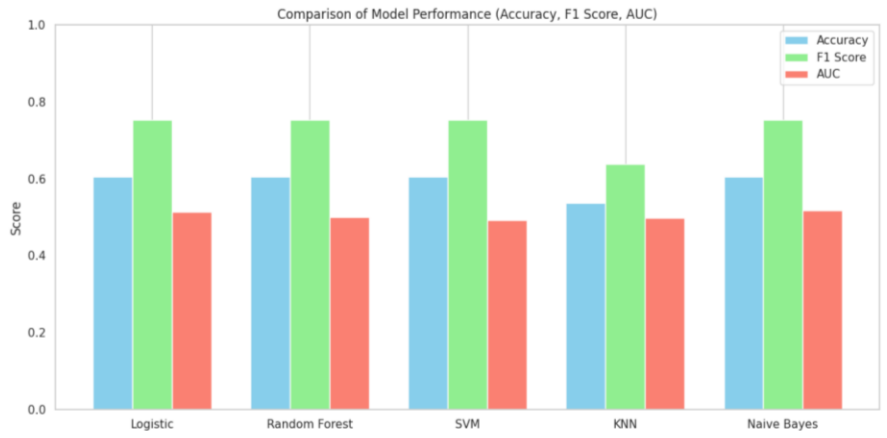
```
=== Naive Bayes ===
Accuracy : 0.6055
F1 Score : 0.7542821550918717
ROC AUC  : 0.5169645277394899
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1578
           1       0.61      1.00      0.75      2422

    accuracy                           0.61      4000
   macro avg       0.30      0.50      0.38      4000
weighted avg       0.37      0.61      0.46      4000
```

Visualizations:

- ROC curves



- Grouped bar comparison chart



Conclusion

This project successfully demonstrated a complete machine learning workflow applied to a real-world healthcare problem. Starting from raw data, we performed thorough cleaning and feature selection, explored various modeling techniques, and identified the most effective algorithm for predicting cancer survival. The Random Forest model provided high accuracy and stability, making it suitable for deployment in a clinical support tool. The Streamlit web app created at the final stage allows easy interaction with the model, providing real-time survival predictions based on user input. Overall, this project highlights the potential of data-driven approaches to enhance clinical decision-making and support early intervention strategies. Future improvements could include hyperparameter tuning, incorporating more temporal data, or deploying the app on a cloud platform for public access.