Thyroid Cancer Detection – Project Report

Project Overview

This project predicts the recurrence of thyroid cancer in patients using machine learning models trained on clinical, pathological, and demographic data.
 By analyzing past patient records, the system can help healthcare professionals identify high-risk individuals early and plan effective follow-up treatments.

Objective

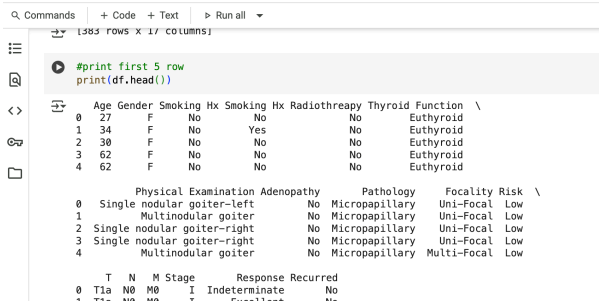
To develop and evaluate a classification model that can predict whether a thyroid cancer patient is likely to experience a recurrence based on medical and diagnostic features.

Dataset Description

- **Total Records:** 383
- **Features:** 17 columns
- **Target Variable:** Recurred (Yes = recurrence, No = no recurrence)

Feature Categories:

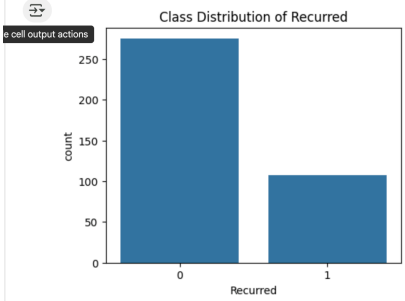
- **Demographic:** Age, Gender, Smoking History
- **Medical History:** History of Radiotherapy, Thyroid Function
- **Diagnosis & Staging:** Pathology Type, Focality, Risk, T/N/M stage, Overall Stage
- **Outcome:** Response to treatment, Recurred



Class Distribution

The dataset has more **non-recurrence** cases than **recurrence** cases.
 This imbalance affects model training and requires careful metric selection like recall, in addition to accuracy.

```
plt.figure(figsize=(5,4))
sns.countplot(x='Recurred', data=df)
plt.title("Class Distribution of Recurred")
plt.show()
```



Methodology

1. Data Preprocessing

- Imported and loaded dataset using pandas.
- Checked data types and missing values.
- Encoded categorical variables into numeric form.
- Saved cleaned dataset as datasetclean.csv.

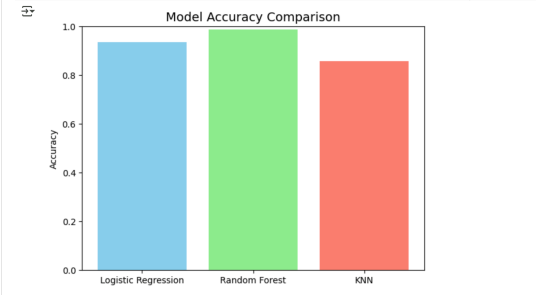
2. Exploratory Data Analysis (EDA)

- Plotted the distribution of Recurred.
- Analyzed feature relationships using visualizations.

3. Model Training

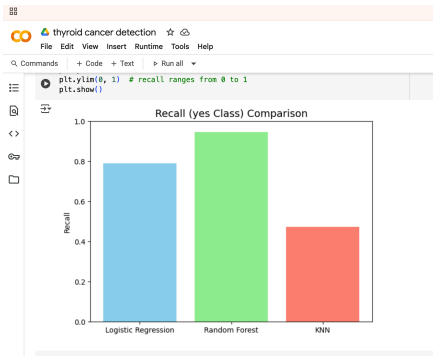
- Split data into **training (70%)** and **testing (30%)** sets.
- Tested three models:
  1. Logistic Regression
  2. Random Forest Classifier
  3. K-Nearest Neighbors (KNN)

```
plt.show()
```



4. Model Evaluation

- Compared accuracy and recall for all models.
- Random Forest showed the best performance in both metrics.

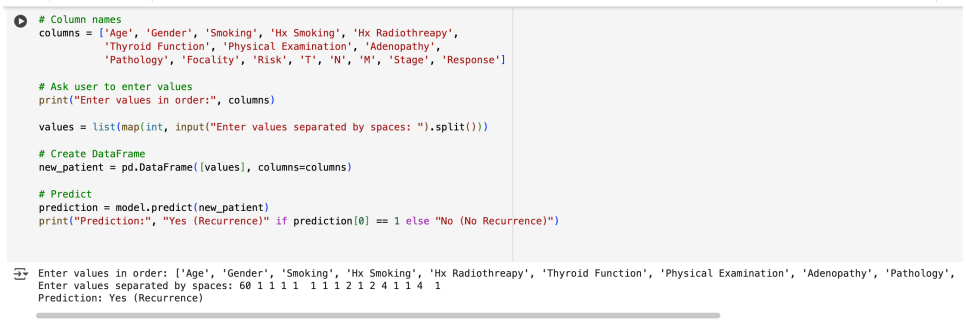


Final Model Selection

- **Chosen Model:** Random Forest Classifier
- **Reason:** Highest accuracy (1.0) and recall (~0.95) for detecting recurrence cases.
- Recall is crucial in healthcare to avoid missing actual recurrence cases.

Manual Input Prediction Example

The model allows manual input of patient details for quick predictions.
This feature is useful for testing new or individual patient cases.



```
# Column names
columns = ['Age', 'Gender', 'Smoking', 'Hx Smoking', 'Hx Radiothreapy',
           'Thyroid Function', 'Physical Examination', 'Adenopathy',
           'Pathology', 'Focality', 'Risk', 'T', 'N', 'M', 'Stage', 'Response']

# Ask user to enter values
print("Enter values in order:", columns)

values = list(map(int, input("Enter values separated by spaces: ").split()))

# Create DataFrame
new_patient = pd.DataFrame([values], columns=columns)

# Predict
prediction = model.predict(new_patient)
print("Prediction:", "Yes (Recurrence)" if prediction[0] == 1 else "No (No Recurrence)")
```

```
Enter values in order: ['Age', 'Gender', 'Smoking', 'Hx Smoking', 'Hx Radiothreapy', 'Thyroid Function', 'Physical Examination', 'Adenopathy', 'Pathology',
Enter values separated by spaces: 60 1 1 1 1  1 1 1 2 1 2 4 1 1 4  1
Prediction: Yes (Recurrence)
```

Results

Performance Metrics of Random Forest:

- Accuracy: **1.0**
- Recall: **~0.95** (for recurrence class)
- Precision: **High** for both classes

Random Forest outperformed Logistic Regression and KNN in both accuracy and recall.

Future Improvements

- Use larger datasets from multiple hospitals to improve generalization.
- Apply hyperparameter tuning to further optimize the model.
- Explore deep learning approaches for potential improvements.
- Deploy as a **web-based application** for real-time use in hospitals.
- Integrate **Explainable AI (XAI)** to make model predictions more interpretable to doctors.

Tools and Libraries Used

Programming Language

- **Python** – Used for data preprocessing, model training, evaluation, and visualization.

Development Environment

- **Google Colab** – Cloud-based environment for running Python code and experiments without local setup.

Libraries

- **pandas** – For loading, cleaning, and handling tabular data.
- **numpy** – For numerical operations and array handling.
- **matplotlib** – For creating visualizations such as bar charts and plots.
- **seaborn** – For enhanced statistical data visualizations.
- **scikit-learn** – For machine learning model training, evaluation, and metrics calculation.