Thyroid_Cancer_Recurrence

April 18, 2025

0.1 THYROID CANCER RECURRENCE | Predictive and Analytic Model

1 Thyroid Cancer Recurrence Prediction

This project analyzes clinical data from thyroid cancer patients to predict recurrence after radioactive iodine (RAI) therapy. It explores patterns, answers key clinical questions, and trains machine learning models for predictive purposes.

1.1 About the Dataset

This dataset focuses on thyroid cancer recurrence after Radioactive Iodine (RAI) therapy. It contains 383 patient records with 13 key attributes, including age, gender, cancer staging, pathology type, risk classification, treatment response, and recurrence status. The data is valuable for predicting cancer recurrence, understanding risk factors, and evaluating treatment outcomes.

1.1.1 Dataset Overview

- Total Rows: 383
- Total Columns: 13
- No Missing Values

1.1.2 Column Descriptions

- **Age**: Age of the patient (in years)
- **Gender**: Patient's gender (Male or Female)
- **Hx Radiotherapy**: History of prior radiotherapy (Yes or No)
- **Adenopathy**: Presence of lymph node involvement (Yes or No)
- **Pathology**: Type of thyroid cancer (e.g., Micropapillary)
- **Focality**: Tumor focality (Uni-Focal or Multi-Focal)
- **Risk**: Cancer risk classification (Low, Intermediate, High)

- **T**: Tumor classification (T1, T2, etc.)
- **N**: Lymph node classification (N0, N1, etc.)
- **M**: Metastasis classification (M0, M1, etc.)
- **Stage**: Cancer staging (Stage I, II, III, IV)
- **Response**: Treatment response (Excellent, Indeterminate, etc.)
- **Recurred**: Whether cancer recurred (Yes or No)

1.2 Key Questions to Explore

1 Are thyroid cancer recurrences more common in men or women?
2 How does age affect recurrence risk?
3 Can we predict recurrence based on tumor staging and pathology?
4 What is the relationship between treatment response and recurrence?

1.2.1 IMPORT LIBRARIES

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from xgboost import XGBClassifier
```

1.2.2 READ DATAFRAME

[2]:
```
# READ THE DATAFRAME
df = pd.read_csv('filtered_thyroid_data.csv')
df
```

[2]:

	Age	Gender	Hx Radiothreapy	Adenopathy	Pathology	Focality	Risk	\
0	27	F	No	No	Micropapillary	Uni-Focal	Low	
1	34	F	No	No	Micropapillary	Uni-Focal	Low	
2	30	F	No	No	Micropapillary	Uni-Focal	Low	
3	62	F	No	No	Micropapillary	Uni-Focal	Low	
4	62	F	No	No	Micropapillary	Multi-Focal	Low	
..	…	…	…	…	…	…	…	…

```
 378   72      M             Yes      Right       Papillary    Uni-Focal  High
 379   81      M             Yes  Extensive       Papillary  Multi-Focal  High
 380   72      M              No  Bilateral       Papillary  Multi-Focal  High
 381   61      M             Yes  Extensive    Hurthel cell  Multi-Focal  High
 382   67      M              No  Bilateral       Papillary  Multi-Focal  High

       T    N   M Stage               Response Recurred
0    T1a   N0  M0     I           Indeterminate       No
1    T1a   N0  M0     I               Excellent       No
2    T1a   N0  M0     I               Excellent       No
3    T1a   N0  M0     I               Excellent       No
4    T1a   N0  M0     I               Excellent       No
..   …  …  ..   …                     …      …
378  T4b  N1b  M1   IVB  Biochemical Incomplete      Yes
379  T4b  N1b  M1   IVB   Structural Incomplete      Yes
380  T4b  N1b  M1   IVB   Structural Incomplete      Yes
381  T4b  N1b  M0   IVA   Structural Incomplete      Yes
382  T4b  N1b  M0   IVA   Structural Incomplete      Yes

[383 rows x 13 columns]
```

1.2.3 EXPLORATORY DATA ANALYSIS

[3]:
```
# DATASET INFO
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 383 entries, 0 to 382
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Age              383 non-null    int64
 1   Gender           383 non-null    object
 2   Hx Radiothreapy  383 non-null    object
 3   Adenopathy       383 non-null    object
 4   Pathology        383 non-null    object
 5   Focality         383 non-null    object
 6   Risk             383 non-null    object
 7   T                383 non-null    object
 8   N                383 non-null    object
 9   M                383 non-null    object
 10  Stage            383 non-null    object
 11  Response         383 non-null    object
 12  Recurred         383 non-null    object
dtypes: int64(1), object(12)
memory usage: 39.0+ KB
```

[4]:
```
# UNIQUE VALUES OF CATEGORICAL FEATURES
df.select_dtypes(include='object').nunique().sort_values()
```

[4]:
```
Gender             2
Hx Radiothreapy    2
Focality           2
M                  2
Recurred           2
Risk               3
N                  3
Pathology          4
Response           4
Stage              5
Adenopathy         6
T                  7
dtype: int64
```

[5]:
```
# UNIQUE VALUES OF NUMERICAL FEATURE
df['Age'].nunique()
```

[5]:
```
65
```

[6]:
```
# DEFINE CATEGORICAL FEATURES
categorical_features = df.select_dtypes(include=['object', 'category']).columns.
 ↪tolist()
categorical_features
```

[6]:
```
['Gender',
 'Hx Radiothreapy',
 'Adenopathy',
 'Pathology',
 'Focality',
 'Risk',
 'T',
 'N',
 'M',
 'Stage',
 'Response',
 'Recurred']
```

[7]:
```
# DEFINE NUMERICAL FEATURES
numerical_features = df.select_dtypes(include='int64')
numerical_features
```

[7]:
```
    Age
0    27
1    34
2    30
```

```
3     62
4     62
..    …
378   72
379   81
380   72
381   61
382   67

[383 rows x 1 columns]
```

[8]:
```python
# DISTRIBUTION OF CATEGORICAL FEATURES
for col in categorical_features:
    order = df[col].value_counts().index.tolist()
    fig = px.histogram(df, x=col, color=col,
                       title=f'DISTRIBUTION OF {col}',
                       category_orders={col: order})
    fig.show()
```

[9]:
```python
# DISTRIBUTION OF NUMERICAL FEATURES
for col in numerical_features:
    fig = px.histogram(df, x=col,
                       title=f'DISTRIBUTION OF {col}',
                       histfunc='count',
                       color_discrete_sequence=['skyblue'],
                       nbins=None,
                       )
    fig.update_layout(bargap=0.1)
    fig.update_traces(xbins=dict(
        start=df[col].min(),
        end=df[col].max(),
        size=1
    ))
    fig.show()
```

[10]:
```python
# RISK LEVEL vs AGE
# NUMBER OF PATIENTS vs RISK LEVEL
fig = px.box(df,
             x='Risk',
             y='Age',
             color='Risk',
             points='all',
             title='RISK LEVEL BY AGE')
fig.show()

risk_counts = df['Risk'].value_counts().reset_index()
risk_counts.columns = ['Risk Level', 'Count']
```

```
fig = px.bar(risk_counts,
             x='Risk Level',
             y='Count',
             text='Count',
             color='Risk Level',
             title='NUMBER OF PATIENTS BY RISK LEVEL')

fig.update_traces(textposition='outside')
fig.show()
```

[11]:
```
# RECURRENCE vs STAGE
stage_order = ['I', 'II', 'III', 'IVA', 'IVB']
df['Stage'] = pd.Categorical(df['Stage'], categories=stage_order, ordered=True)

prop_df = df.groupby('Stage')['Recurred'].value_counts(normalize=True).
 ↪rename('Proportion').reset_index()

fig = px.bar(prop_df,
             x='Stage',
             y='Proportion',
             color='Recurred',
             barmode='stack',
             text=prop_df['Proportion'].round(2),
             title='RECURRENCE RATIO BY STAGE')
fig.update_traces(textposition='inside')
fig.show()


fig = px.histogram(df,
                   x='Stage',
                   color='Recurred',
                   barmode='group',
                   text_auto=True,
                   title='RECURRENCE COUNT BY STAGE')
fig.show()
```

```
C:\Users\infoa\AppData\Local\Temp\ipykernel_21124\1999730532.py:5:
FutureWarning:

The default of observed=False is deprecated and will be changed to True in a
future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
```

[12]:
```
# RECURRENCE vs PATHOLOGY
```

```
prop_df = df.groupby('Pathology')['Recurred'].value_counts(normalize=True).
 ↪rename('Proportion').reset_index()
fig = px.bar(prop_df,
             x='Pathology',
             y='Proportion',
             color='Recurred',
             barmode='stack',
             text=prop_df['Proportion'].round(2),
             title='RECURRENCE RATIO BY PATHOLOGY')
fig.update_traces(textposition='inside')
fig.show()

fig = px.histogram(df,
                   x='Pathology',
                   color='Recurred',
                   barmode='group',
                   text_auto=True,
                   title='RECURRENCE COUNT BY PATHOLOGY')
fig.show()
```

[13]:
```
# RECURRENCE vs RESPONSE
prop_df = df.groupby('Response')['Recurred'].value_counts(normalize=True).
 ↪rename('Proportion').reset_index()
fig = px.bar(prop_df,
             x='Response',
             y='Proportion',
             color='Recurred',
             barmode='stack',
             text=prop_df['Proportion'].round(2),
             title='RECURRENCE RATIO BY RESPONSE')
fig.update_traces(textposition='inside')
fig.show()

fig = px.histogram(df,
                   x='Response',
                   color='Recurred',
                   barmode='group',
                   text_auto=True,
                   title='RECURRENCE COUNT BY RESPONSE')
fig.show()
```

1.2.4 FEATURE ENGINEERING

[14]:
```
# LABEL ENCODING
le = LabelEncoder()
for col in df.select_dtypes(include=['object', 'category']).columns:
    df[col] = le.fit_transform(df[col])
```

[15]:
```
df.head()
```

[15]:

	Age	Gender	Hx Radiothreapy	Adenopathy	Pathology	Focality	Risk	T	N	\
0	27	0	0	3	2	1	2	0	0	
1	34	0	0	3	2	1	2	0	0	
2	30	0	0	3	2	1	2	0	0	
3	62	0	0	3	2	1	2	0	0	
4	62	0	0	3	2	0	2	0	0	

	M	Stage	Response	Recurred
0	0	0	2	0
1	0	0	1	0
2	0	0	1	0
3	0	0	1	0
4	0	0	1	0

[16]:
```
correlation_matrix = df.corr()
```

[17]:
```
# CORRELATION MATRIX
fig = px.imshow(
    correlation_matrix,
    text_auto=".2f",
    color_continuous_scale="RdBu",
    title="CORRELATION MATRIX"
)

fig.update_layout(
    width=1000,
    height=800,
    margin=dict(l=50, r=50, t=50, b=50)
)

fig.show()
```

[18]:
```
# CORRELATION VALUES
corr = df.corr(numeric_only=True)
print(corr['Recurred'].sort_values(ascending=False))
```

```
Recurred            1.000000
Response            0.708957
N                   0.632323
T                   0.556201
Stage               0.449137
M                   0.354360
Gender              0.328189
Age                 0.258897
Hx Radiothreapy     0.174407
Pathology           0.003272
```

```
Adenopathy        -0.182530
Focality          -0.383776
Risk              -0.733376
Name: Recurred, dtype: float64
```

[19]:
```
# DEFINE FEATURES AND TARGET
X = df.drop(['Recurred', 'Risk', 'Response'], axis=1)
y = df['Recurred']
```

[20]:
```
# FEATURE SCALING
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[21]:
```
# SPLIT DATASET
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,␣
 ↪stratify=y, random_state=42)
```

1.2.5 APPLICATION OF MACHINE LEARNING MODELS

[22]:
```
# LOGISTIC REGRESSION MODEL
lr = LogisticRegression()
lr.fit(X_train, y_train)
predict_lr = lr.predict(X_test)
print("Logistic Regression Report:\n", classification_report(y_test,␣
 ↪predict_lr))
```

```
Logistic Regression Report:
              precision    recall  f1-score   support

           0       0.96      0.91      0.93        55
           1       0.80      0.91      0.85        22

    accuracy                           0.91        77
   macro avg       0.88      0.91      0.89        77
weighted avg       0.92      0.91      0.91        77
```

[23]:
```
# RANDOM FOREST CLASSIFIER MODEL
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
predict_rf = rf.predict(X_test)
print("Random Forest Report:\n", classification_report(y_test, predict_rf))
```

```
Random Forest Report:
              precision    recall  f1-score   support

           0       0.91      0.89      0.90        55
           1       0.74      0.77      0.76        22
```

```
    accuracy                           0.86        77
   macro avg       0.82      0.83      0.83        77
weighted avg       0.86      0.86      0.86        77
```

[24]:
```
# XGBOOST MODEL
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
xgb.fit(X_train, y_train)
predict_xgb = xgb.predict(X_test)
print("XGBoost Report:\n", classification_report(y_test, predict_xgb))
```

```
XGBoost Report:
               precision    recall  f1-score   support

           0       0.91      0.89      0.90        55
           1       0.74      0.77      0.76        22

    accuracy                           0.86        77
   macro avg       0.82      0.83      0.83        77
weighted avg       0.86      0.86      0.86        77


C:\Users\infoa\AppData\Roaming\Python\Python312\site-
packages\xgboost\training.py:183: UserWarning:

[22:21:58] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.
```

1.2.6 MODEL EVALUATION

[25]:
```
# METRICS BY MODEL
reports = []

report_lr = classification_report(y_test, predict_lr, output_dict=True)
acc_lr = accuracy_score(y_test, predict_lr)
reports.append({'Model': 'Logistic Regression',
                'Precision': report_lr['macro avg']['precision'],
                'Recall': report_lr['macro avg']['recall'],
                'F1-Score': report_lr['macro avg']['f1-score'],
                'Accuracy': acc_lr})

report_rf = classification_report(y_test, predict_rf, output_dict=True)
acc_rf = accuracy_score(y_test, predict_rf)
reports.append({'Model': 'Random Forest',
                'Precision': report_rf['macro avg']['precision'],
                'Recall': report_rf['macro avg']['recall'],
```

```
                'F1-Score': report_rf['macro avg']['f1-score'],
                'Accuracy': acc_rf})

report_xgb = classification_report(y_test, predict_xgb, output_dict=True)
acc_xgb = accuracy_score(y_test, predict_xgb)
reports.append({'Model': 'XGBoost',
                'Precision': report_xgb['macro avg']['precision'],
                'Recall': report_xgb['macro avg']['recall'],
                'F1-Score': report_xgb['macro avg']['f1-score'],
                'Accuracy': acc_xgb})

df_metrics = pd.DataFrame(reports)

df_melted = df_metrics.melt(id_vars='Model', var_name='Metric',␣
 ↪value_name='Score')

fig = px.bar(df_melted, x='Model', y='Score', color='Metric', barmode='group',
             text_auto='.2f', title='Model Comparison: Accuracy, Precision,␣
 ↪Recall & F1-score (Macro Avg)')
fig.update_layout(yaxis=dict(range=[0.8, 1.05]))
fig.show()
```

1.2.7 FINAL ANALYSIS, INSIGHTS AND CONCLUSION

1.3 Final Analysis and Key Findings

1.3.1 1. Are thyroid cancer recurrences more common in men or women?

While most patients in the dataset are women, **men show a higher proportion of recurrence**. This suggests that male patients may be at increased risk and deserve closer monitoring.

1.3.2 2. How does age affect recurrence risk?

There is a **moderate positive correlation (0.26)** between age and recurrence. Older patients tend to experience recurrence more frequently, although age alone is not the most dominant factor.

1.3.3 3. Can recurrence be predicted based on tumor staging and pathology?

Yes. Variables such as **Stage, T, N, M, and Pathology** all show relevant patterns: - Recurrence rates increase with higher cancer stages (Stage IV). - Some pathology types like **Papillary** and **Hurthel cell** are more frequently associated with recurrence. - These patterns were confirmed by predictive models using these features.

1.3.4 4. What is the relationship between treatment response and recurrence?

A very clear one: patients with **"Structural Incomplete"** or **"Biochemical Incomplete"** responses have significantly higher recurrence rates, while those with an **"Excellent"** response rarely relapse. `Response` shows the **strongest correlation (0.71)** with recurrence.

1.4 Key Insights

- `Risk` classification shows the **strongest negative correlation (-0.73)** with recurrence. Higher-risk patients are much more likely to relapse.
- **Logistic Regression** delivered the best balance between precision and recall.
- **XGBoost** performed more balanced across both classes (recurred / not recurred), while **Random Forest** performed slightly better for non-recurrences.

1.5 Overall Conclusion

The results confirm that it is possible to accurately predict thyroid cancer recurrence using clinical data. Key variables like **tumor stage, treatment response, and risk** allow for early identification of high-risk patients. These models could support clinical decision-making and personalized follow-up strategies.

Author: Adrian Zambrana · April 2025

[ ]: