Predicting Thyroid Malignancy Using Machine Learning

Abla Eklou[1], Bhavitha Asam[1], Poojitha Surgi[1], Sakshi Mehta[1]

[1]Indiana University, Indianapolis, IN 46202, USA,
psurgi@iu.edu, asambha@iu.edu, mehtsaam@iu.edu, aeklou@iu.edu

Abstract. This project applies machine learning techniques to predict thyroid malignancy using a dataset comprising both clinical and ultrasound features. The objective was to identify significant predictors of thyroid cancer and evaluate the performance of four classification models: Random Forest, XGBoost, Multilayer Perceptron (MLP), and an ensemble approach. After preprocessing and addressing class imbalance with SMOTE, statistical tests and feature importance analyses were conducted to guide model interpretation. Among the models, Random Forest achieved the highest F1 score (0.98) and ROC AUC (0.67), effectively identifying malignant cases. SHAP and permutation analyses confirmed key predictors such as multilateral nodularity, calcification, and blood flow. The results demonstrate the potential of machine learning to enhance diagnostic accuracy, reduce unnecessary biopsies, and support early clinical decision-making in thyroid cancer detection.

Keywords: Thyroid cancer · Machine learning · Random Forest · SHAP · Ultrasound imaging · Predictive modeling · SMOTE

1 Project Scope

1.1 Introduction

Thyroid cancer is among the fastest-growing malignancies globally, posing a significant diagnostic challenge for healthcare professionals in differentiating benign from malignant thyroid nodules (Bomeli, LeBeau, & Ferris, 2010). While ultrasound imaging remains the primary noninvasive diagnostic modality, the interpretation of ultrasound characteristics, such as echogenicity, calcification, nodule shape, margins, and vascularity, often involves subjective assessment by radiologists, which may lead to inconsistencies in diagnostic outcomes (Haugen et al., 2016).

In response to these limitations, machine learning (ML) has emerged as a promising approach for enhancing diagnostic accuracy and reducing variability in clinical decision-making (Xi, Wang, & Yang, 2022). ML models can be trained to recognize complex patterns within imaging and clinical data that are not easily discernible through traditional methods. These models offer the potential for more objective and reproducible evaluation of thyroid nodules (Chen et al., 2018).

Recent studies have demonstrated the feasibility of applying ML algorithms to thyroid ultrasound data for malignancy prediction (Xi et al., 2022). While individual models such as decision trees, support vector machines, and neural networks have shown success, the comparative advantage of ensemble learning, where multiple models are combined to produce a more accurate prediction, has not been thoroughly investigated in this specific context (Cheng et al., 2024).

The present study introduces a comparative analysis of multiple ML classifiers, specifically Random Forest, XGBoost, and Multi-Layer Perceptron (MLP), both as independent models and within a soft voting ensemble framework. The goal is to determine the most effective modeling strategy and features for predicting thyroid malignancy.

By addressing these research gaps, this study aims to contribute to the development of AI-enhanced diagnostic tools that can support clinicians in thyroid cancer screening, minimize diagnostic uncertainty, and reduce unnecessary invasive procedures such as fine-needle aspiration biopsies (Bomeli et al., 2010; Haugen et al., 2016).

1.2 Aim

Our aim is "To identify significant features contributing to the prediction of thyroid malignancy using both clinical and ultrasound data."
To address this aim, we posed research question:

> How do clinical and ultrasound-based features such as hormone levels, nodule characteristics, and demographic factors relate to the risk of thyroid malignancy, and which of these features are most strongly associated with cancer outcomes?

To structure the analysis, the following hypotheses were defined:

- **Null Hypothesis (H_0):** Characteristics derived from clinical and ultrasound data are not effective in predicting whether a thyroid nodule is benign or malignant.
- **Alternative Hypothesis (H_1):** Characteristics derived from clinical and ultrasound data are effective in predicting whether a thyroid nodule is benign or malignant.

1.3 Purpose

The primary goal of this project is to develop a machine learning–based model that accurately predicts thyroid malignancy by integrating clinical laboratory data and ultrasound imaging features. Traditional diagnostic systems like TI-RADS are often subjective, leading to unnecessary fine-needle aspiration (FNA) biopsies or missed malignancies. To address these limitations, the project investigates the predictive performance of multiple machine learning algorithms, including Random Forest, XGBoost, Multilayer Perceptron (MLP), and a soft voting ensemble. By working with a balanced and categorized dataset, the study aims to identify the most influential features such as multilateral involvement, calcification, and FT3 levels that contribute to thyroid cancer risk. Additionally, the project prioritizes optimizing recall (sensitivity) to minimize false negatives in malignant cases and enhances interpretability using techniques like SHAP values and permutation importance to support clinical decision-making.

2 Methodology

2.1 Steps of the Project

The primary focus of our project was to build a predictive model for thyroid malignancy by leveraging both clinical laboratory and ultrasound imaging features using modern machine learning techniques. Traditional diagnostic methods such as TI-RADS rely heavily on subjective radiological interpretation, which can result in missed malignancies or unnecessary biopsies. Our approach aimed to provide a data-driven, objective framework to enhance early detection and support clinical decision-making. The tools we used include Python (Jupyter Notebook), pandas, sklearn, imbalanced-learn (for SMOTE), SHAP, and permutation importance. The project was divided into four key phases:

1. Data Collection and Cleaning
2. Data Preprocessing and Feature Engineering
3. Model Training and Evaluation
4. Visualization and Interpretability

2.2 Original Team Members and Responsibilities

Our team comprised individuals with complementary skill sets across data science, healthcare informatics, and technical analysis. At the outset of the project, roles were clearly defined to leverage each member's strengths and ensure a balanced and effective collaboration. The initial responsibilities assigned to each team member are outlined below:

Name	Background	Responsibilities
Sakshi Mehta	Bachelor of Dental Surgery (B.D.S), currently pursing MS in Health Informatics (MSHI).	Data collection and cleaning, SQL and Python coding
Abla Eklou	Bachelor's in Informatics, currently pursing MS in Bioinformatics.	Statistical modeling, SQL, and presentation preparation
Bhavitha Asam	Bachelor of Dental Surgery (B.D.S), currently pursing MS in Health Informatics (MSHI).	SQL development and machine learning integration
Poojitha Surgi	Bachelor of Dental Surgery (B.D.S), currently pursuing MS in Health Informatics (MSHI).	Data visualization and project presentation

Fig. 1. Original responsibilities of team members

2.3 Actual Contribution from individual team members:

We originally did a good job of breaking up responsibilities to work as efficiently as possible. However, the above table was not completely representative of responsibilities of each team member due to unforeseen changes, as well as better understanding what strengths were of each team member.

Name	Background	Responsibilities
Sakshi Mehta	Bachelor of Dental Surgery (B.D.S), currently pursing MS in Health Informatics (MSHI).	Data collection and cleaning, data analysis, visualization, literature review, citations, report writing, and presentation preparation
Abla Eklou	Bachelor's in Informatics, currently pursing MS in Bioinformatics.	SQL table creation, report writing, proofreading and presentation design
Bhavitha Asam	Bachelor of Dental Surgery (B.D.S), currently pursing MS in Health Informatics (MSHI).	Data analysis, machine learning implementation, visualization, report and presentation contributions
Poojitha Surgi	Bachelor of Dental Surgery (B.D.S), currently pursing MS in Health Informatics (MSHI).	Data cleaning, model integration, visualization, and contribution to both report and presentation

Fig. 2. Revised Responsibilities of Team Members for This Project

2.4 Project Challenges

While our team collaborated efficiently, we encountered several challenges throughout the project. During data cleaning, selecting one representative row per patient using a malignancy-priority rule was time-consuming and required careful attention to avoid data leakage. The class imbalance—where malignant cases outnumbered benign—initially led to biased model predictions. We addressed this by applying SMOTE to balance the training set and improve recall.

Feature transformation also posed difficulties. Clinical values like hormone levels, tumor size, and age were converted into categorical bins using published thresholds, requiring manual validation to ensure medical accuracy. Interpreting model explainability tools such as SHAP and permutation importance took multiple iterations to connect outputs to clinical context meaningfully.

Model evaluation introduced further complexity, especially in analyzing the performance gaps of the MLP model, which misclassified more benign cases. Finally, scheduling team meetings across varying time zones and academic commitments was occasionally challenging. Nonetheless, through consistent coordination and adaptability, the team successfully overcame these obstacles and delivered a cohesive, well-supported project.

3. Data Collection

The dataset used for this project was retrieved from the Zenodo Open Repository (https://zenodo.org/records/6465436), contributed by Shengjling Hospital of China Medical University. It includes clinical and ultrasound features for 724 patients, stored originally in a structured SQL database. Each patient had multiple ultrasound entries, and the raw data required filtering, categorization, and reformatting for use in machine learning models. The dataset was accessed using SQL queries and imported into a Python environment using the MySQLdb library for preprocessing and modeling.

The dataset contains 21 attributes:

- **Demographic:** age, gender
- **Laboratory Tests:** FT3, FT4, TSH, TPO, TGAb
- **Ultrasound Features:** site, echo pattern, multifocality, shape, margin, calcification, echo strength, blood flow, composition, size, multilateral
- **Target Variable:** mal (1 = malignant, 0 = benign)

4. Data Extraction and Storage

4.1 Data Extraction

The first step in the data pipeline was to extract the SQL-based dataset into a Python environment. The connection to the MySQL server was established using credentials stored in a .txt file. SQL queries were executed to retrieve relevant patient data, which was then loaded into a pandas Data Frame.

Options

	new_id	id	age	gender	FT3	FT4	TSH	TPO	TGAb	site	echo_pattern	multifocality	size	shape	margin	calcification	echo_strength	blood_flow	compositi
Edit Copy Delete	1	1	46	1	4.34	12.41	1.6770	0.43	0.98	0	0	0	4.60	0	0	0	4	0	1
Edit Copy Delete	2	2	61	1	5.40	16.26	2.9050	0.45	1.91	0	0	0	4.20	0	1	1	4	1	2
Edit Copy Delete	3	3	44	1	3.93	13.39	1.8230	9.15	26.25	0	0	0	0.70	0	1	0	4	0	2
Edit Copy Delete	4	5	29	0	3.70	13.98	1.2930	0.15	0.81	0	0	1	1.00	1	1	1	4	0	2
Edit Copy Delete	5	6	37	1	3.60	14.56	0.9380	0.13	21.22	0	0	0	0.70	0	1	1	4	0	2
Edit Copy Delete	6	7	54	1	3.31	12.36	5.6690	2.09	1.22	0	0	1	1.60	0	0	0	2	0	2
Edit Copy Delete	7	8	48	1	4.00	14.00	0.4210	5.00	4.00	0	0	0	0.80	0	1	0	2	0	2
Edit Copy Delete	8	11	51	0	4.00	14.00	0.4210	5.00	4.00	0	0	1	1.20	0	1	1	4	0	2
Edit Copy Delete	9	12	34	1	4.25	17.66	5.3420	153.04	387.84	0	0	0	3.60	1	0	0	4	0	2
Edit Copy Delete	10	14	31	0	4.56	15.00	1.2780	6.10	4.45	0	0	1	1.80	1	1	1	2	1	2
Edit Copy Delete	11	15	49	1	3.88	13.41	1.7440	8.98	0.99	0	0	0	0.90	0	1	1	4	0	2
Edit Copy Delete	12	16	55	1	3.90	15.41	2.7670	545.07	46.78	0	1	1	1.20	0	1	1	4	0	2
Edit Copy Delete	13	17	65	1	4.47	21.21	0.6820	2.71	1.09	0	0	1	5.50	0	0	1	4	1	2
Edit Copy Delete	14	20	41	1	4.80	12.27	0.0560	524.05	975.62	0	0	1	4.00	0	0	1	4	0	2
Edit Copy Delete	15	21	40	0	4.10	13.96	1.5880	0.45	0.67	0	0	1	2.90	0	0	0	2	1	1
Edit Copy Delete	16	22	33	1	4.26	17.69	2.3660	499.76	0.90	0	1	0	2.10	0	0	0	2	1	2
Edit Copy Delete	17	23	45	1	4.35	14.11	0.6150	91.24	1.50	0	0	1	3.20	0	1	1	4	0	2
Edit Copy Delete	18	25	19	1	3.84	13.26	2.5660	0.27	491.91	0	0	0	3.60	0	1	1	4	1	2
Edit Copy Delete	19	26	33	1	4.26	17.69	2.3660	499.76	0.90	0	0	0	2.00	0	1	0	4	0	2
Edit Copy Delete	20	27	52	1	4.00	14.00	0.4210	5.00	4.00	0	0	1	1.00	0	1	1	4	0	2
Edit Copy Delete	21	28	41	1	3.36	13.21	1.0610	1.26	74.37	0	0	1	1.20	0	1	0	4	1	2
Edit Copy Delete	22	29	58	1	3.86	14.05	3.5890	1.85	1.81	0	0	1	1.00	0	1	1	4	0	2
Edit Copy Delete	23	30	72	1	4.26	11.08	0.0030	0.81	1.43	0	0	0	1.00	0	1	0	2	0	2

Fig. 3. Structure of mytable in phpMyAdmin

We had the file saved, and we uploaded it to phpMyAdmin. We then connected the thyroid dataset table to Python Jupyter Notebook.

4.2 Data Cleaning

Once loaded into the pandas environment, the data underwent a comprehensive cleaning process:

- **Byte Decoding:** Columns exported from SQL were initially byte-encoded (e.g., b'High') and were converted to standard text.

	new_id	id	age	gender	FT3	FT4	TSH	TPO	TGAb	site	...	multifocality	size	shape	margin	calcification	echo_strength	blood_flow	composition	mal
0	1	1	46	1	4.34	12.41	1.6770	0.43	0.98	0	...	b'\x00'	4.60	b'\x00'	b'\x00'	b'\x00'	4	b'\x00'	1	b'\x01'
1	2	2	61	1	5.40	16.26	2.9050	0.45	1.91	0	...	b'\x00'	4.20	b'\x00'	b'\x01'	b'\x01'	4	b'\x01'	2	b'\x01'
2	3	3	44	1	3.93	13.39	1.8230	9.15	26.25	0	...	b'\x00'	0.70	b'\x00'	b'\x01'	b'\x00'	4	b'\x00'	2	b'\x00'
3	4	5	29	0	3.70	13.98	1.2930	0.15	0.81	0	...	b'\x01'	1.00	b'\x01'	b'\x01'	b'\x01'	4	b'\x00'	2	b'\x01'
4	5	6	37	1	3.60	14.56	0.9380	0.13	21.22	0	...	b'\x00'	0.70	b'\x00'	b'\x01'	b'\x01'	4	b'\x00'	2	b'\x01'

Fig. 4. Original Data with columns containing byte values

	new_id	id	age	gender	FT3	FT4	TSH	TPO	TGAb	site	...	multifocality	size	shape	margin	calcification	echo_strength	blood_flow	composition	mal
0	1	1	46	1	4.34	12.41	1.6770	0.43	0.98	0	...	0	4.60	0	0	0	4	0	1	1
1	2	2	61	1	5.40	16.26	2.9050	0.45	1.91	0	...	0	4.20	0	1	1	4	1	2	1
2	3	3	44	1	3.93	13.39	1.8230	9.15	26.25	0	...	0	0.70	0	1	0	4	0	2	0
3	4	5	29	0	3.70	13.98	1.2930	0.15	0.81	0	...	1	1.00	1	1	1	4	0	2	1
4	5	6	37	1	3.60	14.56	0.9380	0.13	21.22	0	...	0	0.70	0	1	1	4	0	2	1

Fig. 5. Cleaned Data Without Bytes and Null Values

- **Filtering:** Some patients had multiple records due to multiple nodules. To ensure consistency and clinical relevance, only one row per patient was retained, prioritizing the entry that indicated malignancy when available.

```
# For each patient, keep one row by prioritize malignant nodule
def select_malignant_per_patient(df):
    df = df.copy()

    # Sort by patient ID and then by malignancy
    df_sorted = df.sort_values(by=['id', 'mal'], ascending=[True, False])

    # Drop duplicates by patient, keeping the malignant row
    df_unique = df_sorted.drop_duplicates(subset='id', keep='first')

    return df_unique

df_malignancy_priority = select_malignant_per_patient(df)
print("Total patients:", df_malignancy_priority.shape[0])
print("Patients with malignant nodules kept:", df_malignancy_priority['mal'].sum())
print("Patients with benign nodules only:", (df_malignancy_priority['mal'] == 0).sum())
```

Fig. 6. Malignant case prioritization and filtering

- **Categorization:**
    - Hormone levels such as TSH, FT3, and FT4 were categorized into Low, Normal, and High using published thresholds (Haugen et al., 2016).
    - TPO and TGAb were labeled as Normal or High (Bomeli, LeBeau, & Ferris, 2010).
    - Tumor size was binned into four categories: <=1cm, 1-2cm, 2-4cm, and >4cm (Cheng et al., 2024).
    - Patient age was grouped into <55 years and >=55 years based on ATA risk stratification (Haugen et al., 2016).

```
# converting numerical columns into categorical values
def categorize_lab_values(df):
    df = df.copy()

    # categories for TSH, FT3, FT4
    df['TSH_cat'] = pd.cut(df['TSH'],
                           bins=[-float('inf'), 0.35, 4.94, float('inf')],
                           labels=['Low', 'Normal', 'High'])

    df['FT3_cat'] = pd.cut(df['FT3'],
                           bins=[-float('inf'), 2.63, 5.70, float('inf')],
                           labels=['Low', 'Normal', 'High'])

    df['FT4_cat'] = pd.cut(df['FT4'],
                           bins=[-float('inf'), 9.01, 19.04, float('inf')],
                           labels=['Low', 'Normal', 'High'])

    # For TPOAb and TGAb
    df['TPO_cat'] = pd.cut(df['TPO'],
                           bins=[-float('inf'), 5.61, float('inf')],
                           labels=['Normal', 'High'])

    df['TGAb_cat'] = pd.cut(df['TGAb'],
                            bins=[-float('inf'), 4.11, float('inf')],
                            labels=['Normal', 'High'])

    return df

df_final_categorized = categorize_lab_values(df_malignancy_priority)

df_final_categorized[['TSH', 'TSH_cat', 'FT3', 'FT3_cat', 'FT4', 'FT4_cat', 'TPO', 'TPO_cat', 'TGAb', 'TGAb_cat']].head()
```

Fig. 7. Clinical features (TSH, FT3, FT4) categorization

```python
# Categorise tumour size and patient age
# Convert 'size' to categorical bins
df_final_categorized['size_cat'] = pd.cut(
    df_final_categorized['size'],
    bins=[-float('inf'), 1, 2, 4, float('inf')],
    labels=['≤1 cm', '1–2 cm', '2–4 cm', '>4 cm']
)

# Convert 'age' to categorical bins
df_final_categorized['age_cat'] = pd.cut(
    df_final_categorized['age'],
    bins=[-float('inf'), 55, float('inf')],
    labels=['<55 y', '≥55 y']
)

df_final_categorized[['age_cat', 'size_cat']].head()

print("Age category counts:\n", df_final_categorized['age_cat'].value_counts())
print("\nSize category counts:\n", df_final_categorized['size_cat'].value_counts())
```

Fig. 8. Clinical features (Age, size) categorization

This process ensured the dataset was clean, interpretable, and compatible with statistical and ML-based workflows.

5. Data Analysis

Following data cleaning and transformation, we conducted exploratory and statistical analyses to uncover patterns and assess the relationship between various features and thyroid malignancy.

5.1 Exploratory Data Analysis (EDA)

To understand the distribution of classes and the relationship between key features and malignancy, we performed several exploratory data analysis steps. These included descriptive statistics, cross-tabulations, and visualizations across categorical and clinical features.

Class Distribution

We first assessed the class distribution of the target variable (mal) after applying the malignancy-priority filtering method. The dataset remained highly imbalanced, with a majority of the patients labeled as malignant. A bar chart was plotted to visualize the class split between benign and malignant cases.

```python
# Calculate counts and percentages
mal_counts = df_malignancy_priority['mal'].value_counts().sort_index()
mal_percent = df_malignancy_priority['mal'].value_counts(normalize=True).sort_index() * 100

# Create summary DataFrame
mal_summary = pd.DataFrame({
    'Count': mal_counts,
    'Percent': mal_percent.round(2)
})
mal_summary.index = ['Benign (0)', 'Malignant (1)']

print("Malignancy summary (based on malignancy-priority method):")
display(mal_summary)

# Plot bar chart
import matplotlib.pyplot as plt

labels = mal_summary.index
values = mal_summary['Count'].values

plt.figure(figsize=(6, 4))
plt.bar(labels, values, color=['skyblue', 'mediumpurple'])
plt.title('Benign vs Malignant')
plt.ylabel('Number of Patients')
plt.xlabel('Diagnosis')
plt.grid(axis='y', linestyle='--', alpha=0.6)

# Add value labels on top
for i, v in enumerate(values):
    plt.text(i, v + 5, f"{v}", ha='center', fontweight='bold')

plt.tight_layout()
plt.show()
```

Fig.9. Malignancy distribution after malignancy-priority filtering

Malignancy summary (based on malignancy-priority method):

	Count	Percent
Benign (0)	30	4.14
Malignant (1)	694	95.86

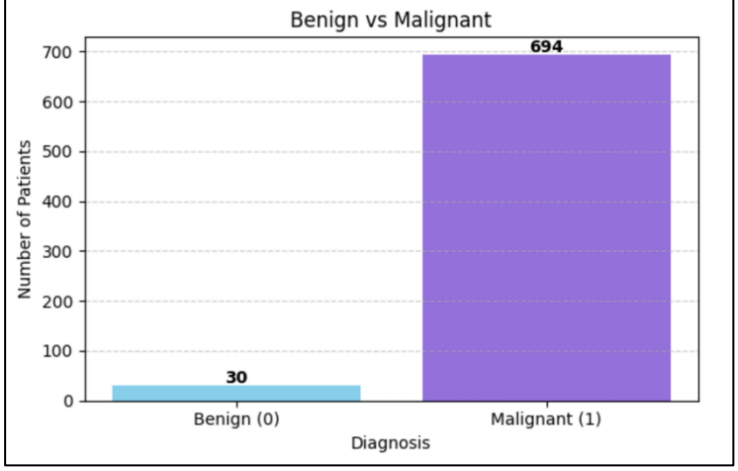

Fig. 10. Bar Graph of Benign vs. Malignant Cases After Prioritization

Feature-wise Distribution by Malignancy

We then explored how features such as tumor size, patient age, and laboratory markers (TSH, FT3, FT4, TPO, TGAb) were distributed across benign and malignant labels using cross-tabulations and frequency counts. Each of these features had previously been categorized into clinically meaningful bins. This analysis helped us observe trends such as whether high TSH or larger nodule size were more associated with malignancy.

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# List of categorical lab test columns
lab_cols = ['TSH_cat', 'FT3_cat', 'FT4_cat', 'TPO_cat', 'TGAb_cat']

# Display counts for each lab test column
for col in lab_cols:
    print(f"\n{col} distribution:")
    print(df_final_categorized[col].value_counts(dropna=False))

# Categorize tumour size
df_final_categorized['size_cat'] = pd.cut(
    df_final_categorized['size'],
    bins=[-float('inf'), 1, 2, 4, float('inf')],
    labels=['≤1 cm', '1–2 cm', '2–4 cm', '>4 cm']
)

# Categorize age
df_final_categorized['age_cat'] = pd.cut(
    df_final_categorized['age'],
    bins=[-float('inf'), 55, float('inf')],
    labels=['<55 y', '≥55 y']
)

# Print category distributions
print("\nAge category counts:\n", df_final_categorized['age_cat'].value_counts())
print("\nSize category counts:\n", df_final_categorized['size_cat'].value_counts())
```

Fig. 11. Categorization of Lab values, Size, and Age Features

Count plots for Categorical Variables

We generated count plots for all 18 categorical features to visualize their distributions relative to the malignancy outcome. Each plot showed the frequency of benign and malignant cases across levels of the respective feature. This helped us identify patterns and class separability across categories like

calcification, multilateral, blood flow, and composition. To improve interpretability, we annotated each bar with its count using a custom annotate bars() function.

```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Define the annotate function
def annotate_bars(ax):
    for p in ax.patches:
        height = int(p.get_height())
        if height > 0:
            ax.annotate(f'{height}',
                        (p.get_x() + p.get_width() / 2, p.get_height()),
                        ha='center', va='bottom', fontsize=9, fontweight='bold')

# List of categorical features
categorical_columns = [
    'gender', 'site', 'echo_pattern', 'multifocality', 'shape',
    'margin', 'calcification', 'echo_strength', 'blood_flow',
    'composition', 'multilateral', 'TSH_cat', 'FT3_cat', 'FT4_cat',
    'TPO_cat', 'TGAb_cat', 'size_cat', 'age_cat'
]

# Loop through each feature
for col in categorical_columns:
    # 1. Print crosstab summary
    print(f"\nCrosstab: {col} vs. mal")
    print(pd.crosstab(df_categorical_only[col], df_categorical_only['mal'], margins=True))

    # 2. Plot countplot with annotations
    plt.figure(figsize=(7, 4))
    ax = sns.countplot(data=df_categorical_only, x=col, hue='mal', palette=['skyblue', 'mediumpurple'], hue_order=[0, 1],
                       order=df_categorical_only[col].value_counts().index)

    plt.title(f'{col} Distribution by Malignancy')
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    annotate_bars(ax)
    plt.tight_layout()
    plt.show()
```

Fig. 12. Categorical Feature Distribution by Malignancy Status

Feature-wise Distribution by Malignancy

We then explored how features such as tumor size, patient age, and laboratory markers (TSH, FT3, FT4, TPO, TGAb), along with ultrasound characteristics (e.g., shape, margin, calcification), were distributed across benign and malignant classes. Each of these features was categorized based on clinical thresholds and guidelines. Cross-tabulations and frequency counts were used to summarize patterns within each feature group.

To consolidate these insights, we created a combined bar chart showing the distribution of benign vs. malignant cases for every feature category. This visualization provided a clear, side-by-side comparison and highlighted categories where malignant cases were more prevalent—such as irregular margins, presence of calcification, and multilateral nodularity.

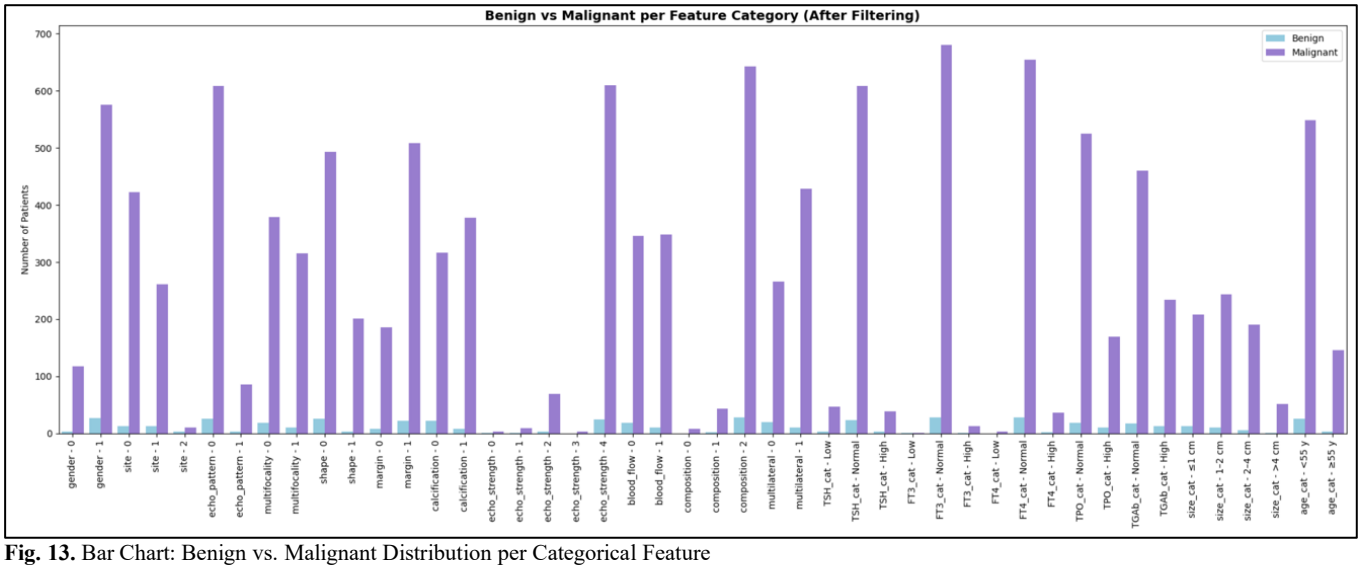

Fig. 13. Bar Chart: Benign vs. Malignant Distribution per Categorical Feature

These visualizations provided an initial understanding of class separation potential and highlighted which features may be useful for modeling.

5.2 Statistical Testing

We conducted Chi-Square tests of independence to evaluate the association between each categorical feature and thyroid malignancy. For every feature, a contingency table was generated against the mal label, and p-values were computed to assess significance. Features with p-values below 0.05 were deemed statistically significant.

The results indicated that multilateral involvement, calcification, margin irregularity, site, blood flow, TPO, and TGAb levels were significantly associated with malignancy. To visualize these findings, we plotted a bar chart of p-values, highlighting the features most relevant to malignancy prediction. This statistical insight guided feature selection for model development.

```
# Chi-Square Test
from scipy.stats import chi2_contingency

print("Chi-Square Test Results\n")

for col in categorical_columns:
    table = pd.crosstab(df_categorical_only[col], df_categorical_only['mal'])
    chi2, p, _, _ = chi2_contingency(table)
    result = "significant" if p < 0.05 else "not significant"
    print(f"{col:<18} p = {p:.4f}   {result}")
```

Fig. 14. Code for Chi-Square test

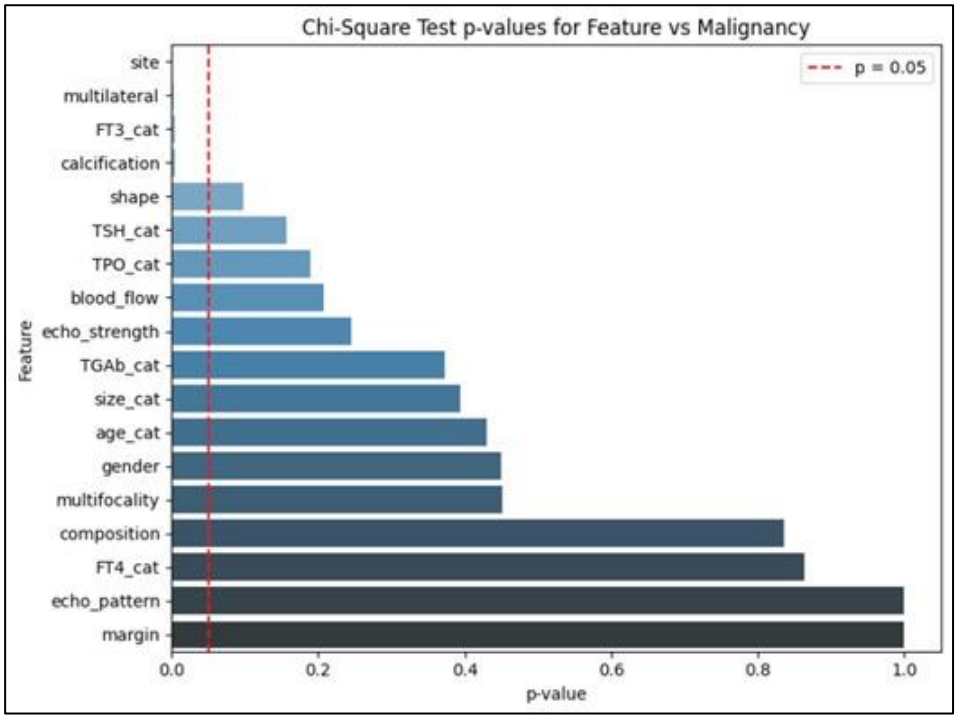

Fig. 15. Chi-Square Test p-values for Categorical Features vs Malignancy

This statistical evidence guided our decision to retain these variables for modeling.

5.3 Train-Test Split

To evaluate model performance effectively while maintaining the original class distribution, we employed a stratified train-test split. The dataset was separated into independent features (X) and the target variable (y). We then used train_test_split from scikit-learn to divide the data into 80% training and 20% testing sets, while ensuring that the proportion of benign and malignant cases was preserved in both subsets.

The class distribution in the training and testing sets closely mirrored that of the original dataset. This approach prevented any potential sampling bias and ensured that both sets were representative of the full dataset. A summary of class proportions confirmed that stratification successfully retained the original distribution of benign and malignant samples across the two sets.

```python
#Train-Test Split (with stratified sampling)
from sklearn.model_selection import train_test_split

# Separate features and target
X = df_categorical_only.drop(columns=['mal'])
y = df_categorical_only['mal']

# Stratified split to preserve class distribution
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    stratify=y,
    random_state=42
)

# print counts and percentages
def print_distribution(name, labels):
    counts = labels.value_counts()
    percentages = labels.value_counts(normalize=True) * 100
    print(f"\n{name} set class distribution:")
    for cls in counts.index:
        print(f"Class {cls}: {counts[cls]} samples ({percentages[cls]:.2f}%)")

# Show distributions
print_distribution("Original", y)
print_distribution("Training", y_train)
print_distribution("Testing", y_test)
```

Fig. 16. Stratified Train-Test Split and Class Distribution Summary

5.4 Class Imbalance Handling

Upon examining the class distribution, it was evident that the dataset was imbalanced, with a higher proportion of malignant cases compared to benign. To prevent model bias toward the majority class and to ensure that benign cases were adequately represented during training, we applied SMOTE (Synthetic Minority Over-sampling Technique) to the training data.

Before applying SMOTE, all categorical features were encoded using Ordinal Encoding to convert them into a numerical format compatible with the oversampling algorithm. SMOTE was then used to synthetically generate new benign samples until the training set was balanced with equal numbers of benign and malignant instances.

This technique successfully balanced the class distribution in the training set and enabled the machine learning models to learn equally from both classes, thereby improving their ability to generalize to unseen data.

```python
#using SMOTE to balance the training set
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import OrdinalEncoder
import pandas as pd

# encode categorical features
encoder = OrdinalEncoder()
X_train_encoded = encoder.fit_transform(X_train)
X_test_encoded = encoder.transform(X_test)

# Apply SMOTE to the encoded training set
smote = SMOTE(random_state=42)
X_train_bal_smote, y_train_bal_smote = smote.fit_resample(X_train_encoded, y_train)

# Convert back to DataFrame to match original structure
X_train_bal_smote = pd.DataFrame(X_train_bal_smote, columns=X_train.columns)

# Show the new balanced class distribution
print("Balanced training set class distribution after SMOTE:")
print(pd.Series(y_train_bal_smote).value_counts())
```

Fig. 17. SMOTE Oversampling to Balance the Training Set

6. Model Building and Evaluation

6.1 Random Forest Classifier

We trained a Random Forest Classifier on the SMOTE-balanced training data and evaluated it using the original test set. Predictions and probability scores were generated for malignancy classification. The model achieved strong performance, with a high F1-score of 0.98 and a ROC AUC score of 0.67, indicating good discriminatory ability.

To further evaluate the model, a confusion matrix was plotted, showing balanced sensitivity and specificity across both classes. These results establish Random Forest as a reliable baseline for thyroid malignancy prediction in our study.

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
import seaborn as sns
import matplotlib.pyplot as plt

# Train Random Forest
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train_bal_smote, y_train_bal_smote)

# Evaluate
y_pred_rf = rf.predict(X_test_encoded)
y_prob_rf = rf.predict_proba(X_test_encoded)[:, 1]

print("Random Forest")
print(classification_report(y_test, y_pred_rf))
print("ROC AUC:", roc_auc_score(y_test, y_prob_rf))

cm_rf = confusion_matrix(y_test, y_pred_rf)
sns.heatmap(cm_rf, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix - Random Forest")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Fig. 18. Training and Evaluation of Random Forest Classifier

6.2 XGBoost Classifier

We trained an XGBoost Classifier using the balanced training set. The model was evaluated on the original test data using standard classification metrics. XGBoost demonstrated F1-score of 0.97 and a ROC AUC score of 0.57, comparable to Random Forest.

A confusion matrix was generated to assess the distribution of predictions. The results showed effective classification of both benign and malignant cases, confirming XGBoost's utility as a competitive model for thyroid malignancy prediction.

```python
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns

# Create XGBoost model and train on balanced training set
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
xgb.fit(X_train_bal_smote, y_train_bal_smote)

# Predict on test set
y_pred_xgb = xgb.predict(X_test_encoded)
y_prob_xgb = xgb.predict_proba(X_test_encoded)[:, 1]

# Print performance metrics
print("XGBoost Classifier")
print(classification_report(y_test, y_pred_xgb, zero_division=0))
print("ROC AUC:", roc_auc_score(y_test, y_prob_xgb))

# Plot confusion matrix
cm_xgb = confusion_matrix(y_test, y_pred_xgb)
sns.heatmap(cm_xgb, annot=True, fmt="d", cmap="Oranges")
plt.title("Confusion Matrix - XGBoost")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Fig. 19. Training and Evaluation of XGBoost Classifier

6.3 Multilayer Perceptron (MLP)

We implemented a Multilayer Perceptron (MLP) neural network with a single hidden layer of 50 neurons and trained it on the balanced dataset. The model was evaluated on the test set using standard classification metrics and ROC AUC.

While the MLP achieved reasonable performance, it showed slightly lower precision on benign cases compared to tree-based models. The ROC AUC score of 0.62 reflected moderate predictive capability. The confusion matrix highlighted a tendency toward over-predicting the majority class, indicating the MLP was less effective in distinguishing between classes than Random Forest.

```python
from sklearn.neural_network import MLPClassifier

# MLP model
mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=300, random_state=42)
mlp.fit(X_train_bal_smote, y_train_bal_smote)

# Predict on test set
y_pred_mlp = mlp.predict(X_test_encoded)
y_prob_mlp = mlp.predict_proba(X_test_encoded)[:, 1]

# Print performance metrics
print("--- Neural Network (MLP) ---")
print(classification_report(y_test, y_pred_mlp, zero_division=0))
print("ROC AUC:", roc_auc_score(y_test, y_prob_mlp))

# Plot confusion matrix
cm_mlp = confusion_matrix(y_test, y_pred_mlp)
sns.heatmap(cm_mlp, annot=True, fmt="d", cmap="Purples")
plt.title("Confusion Matrix - MLP")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Fig. 20. Training and Evaluation of Multilayer Perceptron (MLP) Classifier

6.4 Ensemble Model (Soft Voting)

To leverage the strengths of individual classifiers, we built a soft voting ensemble that combined the Random Forest, XGBoost, and MLP models. The ensemble was trained on the balanced dataset and evaluated on the original test set.

This ensemble model achieved robust performance across metrics, producing a strong ROC AUC score of 0.63 and improved overall classification stability. By averaging predicted probabilities from the base models, the ensemble provided more consistent predictions and mitigated the weaknesses observed in the individual classifiers. The confusion matrix confirmed balanced performance across both classes.

```python
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import classification_report, roc_auc_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Combine all 3 models into one ensemble
ensemble = VotingClassifier(
    estimators=[
        ('rf', rf),
        ('xgb', xgb),
        ('mlp', mlp)

    ],
    voting='soft'
)

# Train on balanced training data
ensemble.fit(X_train_bal_smote, y_train_bal_smote)

# Predict on original test set
y_pred_ens = ensemble.predict(X_test_encoded)
y_prob_ens = ensemble.predict_proba(X_test_encoded)[:, 1]

# Show results
print(" Ensemble Model ")
print(classification_report(y_test, y_pred_ens, zero_division=0))
print("ROC AUC:", roc_auc_score(y_test, y_prob_ens))

# Plot confusion matrix
cm = confusion_matrix(y_test, y_pred_ens)
sns.heatmap(cm, annot=True, fmt="d", cmap="YlGnBu")
plt.title("Confusion Matrix - Ensemble")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Fig. 21. Ensemble Model: Voting Classifier Combining RF, XGBoost, and MLP

6.5 Confusion Matrix Comparison

To visually compare classification performance across all models, we plotted confusion matrices for Random Forest, XGBoost, MLP, and the Ensemble model in a 2×2 grid. These side-by-side heatmaps provide a quick overview of true positives, false positives, true negatives, and false negatives for each approach.

This visual comparison highlights that the Random Forest and Ensemble models produced fewer misclassifications overall, particularly in detecting malignant cases. MLP showed a higher number of false negatives, reinforcing its lower recall observed in metric-based evaluations.

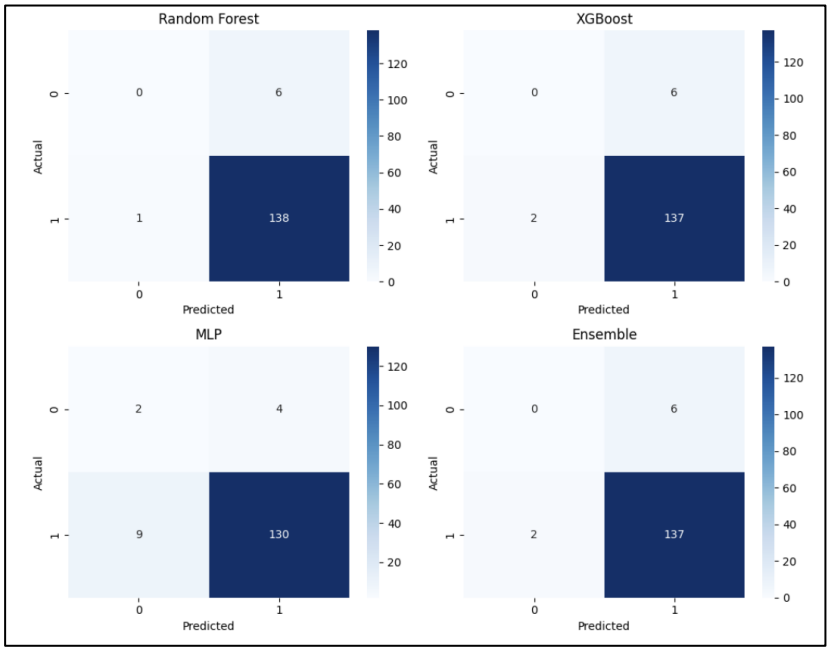

Fig. 22. Confusion Matrices of All Four Models on Test Data

6.6 Comparative Model Performance

We evaluated all four models Random Forest, XGBoost, MLP, and the Ensemble—using accuracy, precision, recall, F1-score, and ROC AUC for both malignant and benign classifications. Random Forest and the Ensemble model performed best on malignant detection, showing high recall and ROC AUC. XGBoost was competitive across both classes, while MLP underperformed, particularly in recall. Separate performance metrics were also calculated for the benign class to ensure balanced detection. XGBoost and Random Forest again demonstrated stronger performance, with MLP showing the lowest precision. Two bar plots were created to visualize these results, comparing model performance for both malignancy and benign prediction tasks.

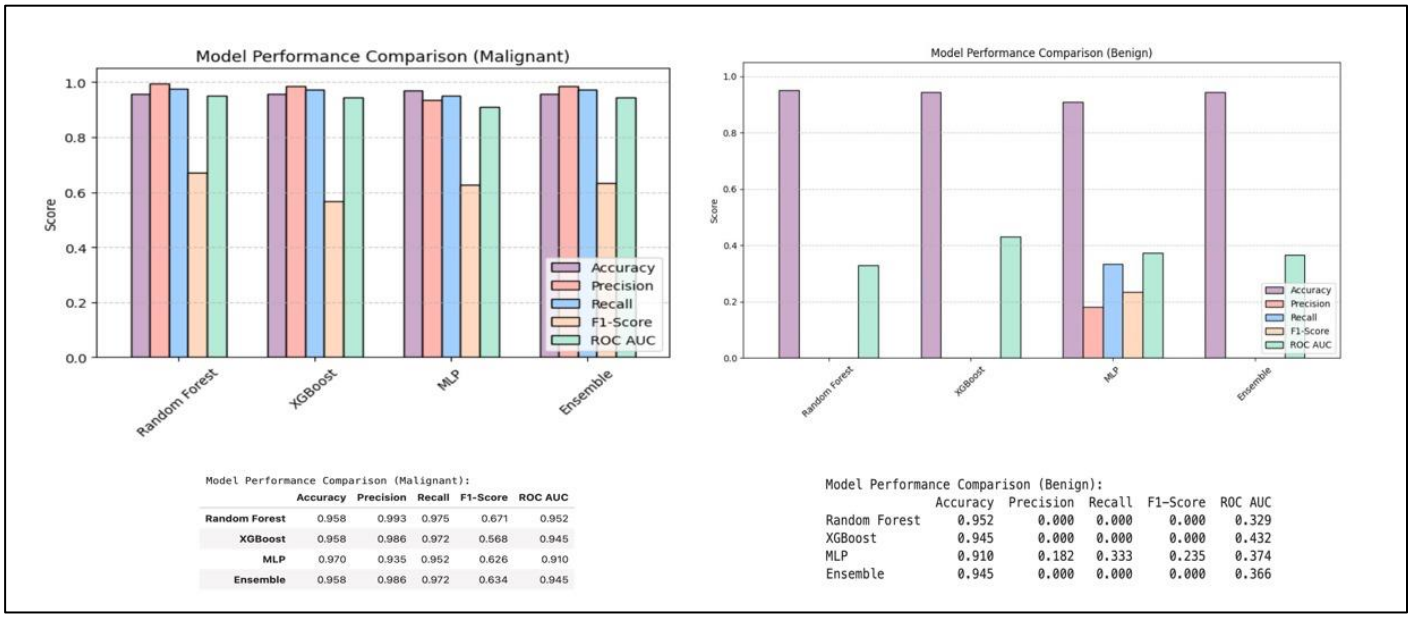

Fig. 23. Performance Comparison of All Models: Malignant vs. Benign Classes

7. Model Interpretability

7.1 Permutation Importance

To identify the most influential features in our Random Forest model, we applied permutation importance on the test set. This method measures the decrease in model performance when a feature's values are randomly shuffled, thereby estimating its contribution to prediction accuracy.

The top three most important features identified were multilateral, calcification and blood flow, aligning with known clinical indicators of thyroid malignancy. A bar chart was generated to visualize these key predictors and their relative impact on model performance.

```python
# Permutation Importance (Random Forest)
from sklearn.inspection import permutation_importance
import pandas as pd
import matplotlib.pyplot as plt

# Calculate permutation importance
perm_result = permutation_importance(
    rf, X_test_encoded, y_test,
    n_repeats=10, random_state=42, n_jobs=-1
)

# Create DataFrame
perm_importances = pd.DataFrame({
    'Feature': X_test.columns,
    'Importance Mean': perm_result.importances_mean,
    'Importance Std': perm_result.importances_std
}).sort_values(by='Importance Mean', ascending=False)

# top 3 features
print("\nPermutation Feature Importance:")
display(perm_importances.head(3))

# Plot Permutation Importance
plt.figure(figsize=(8, 5))
plt.barh(
    perm_importances['Feature'][:3],
    perm_importances['Importance Mean'][:3],
    color='lightseagreen'
)
plt.gca().invert_yaxis()
plt.title('Top 3 Features - Permutation Importance (RF)', fontsize=13, fontweight='bold')
plt.xlabel('Mean Importance', fontsize=11)
plt.tight_layout()
plt.show()
```

Fig. 24. Permutation Feature Importance (Random Forest)

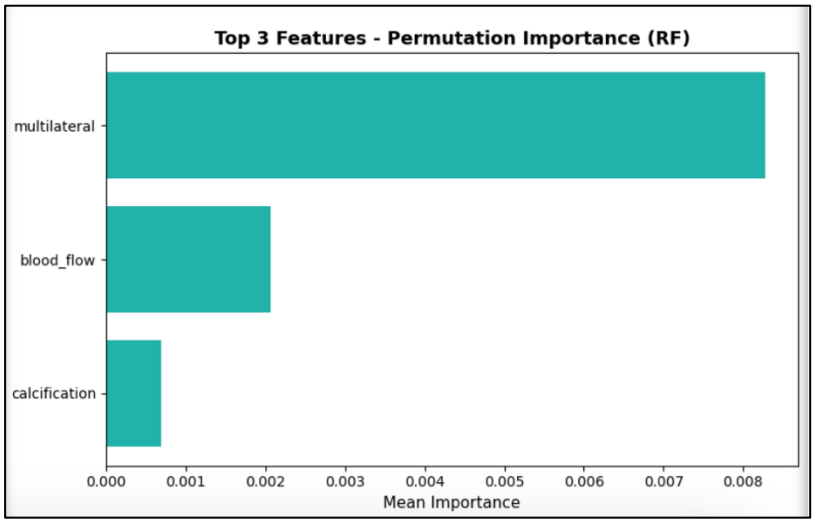

Fig. 25. Predictive Features by Permutation Importance (Random Forest)

7.2 SHAP Analysis

To further interpret the model's decision-making, we used SHAP (SHapley Additive exPlanations) to quantify the contribution of each feature to individual predictions made by the Random Forest classifier. SHAP values were computed separately for both malignant (class 1) and benign (class 0) outcomes using the encoded test set.

The SHAP summary plots revealed that features such as multilateral, blood flow and calcification had the highest impact on malignancy predictions. These results align closely with clinical intuition and further support the reliability of our model. Separate SHAP plots for each class also highlighted which features drove predictions for benign cases.

```python
import shap
import pandas as pd

print("SHAP Analysis – Malignant Class (Class 1)")

# First: Convert to DataFrame
X_test_encoded_df = pd.DataFrame(X_test_encoded, columns=X_test.columns)

# Then: Use shap.Explainer
explainer_rf = shap.Explainer(rf, X_test_encoded_df)

# Calculate SHAP values
shap_values_rf = explainer_rf(X_test_encoded_df)
```

Fig. 26. SHAP Analysis Setup for Malignant Class (Class 1)

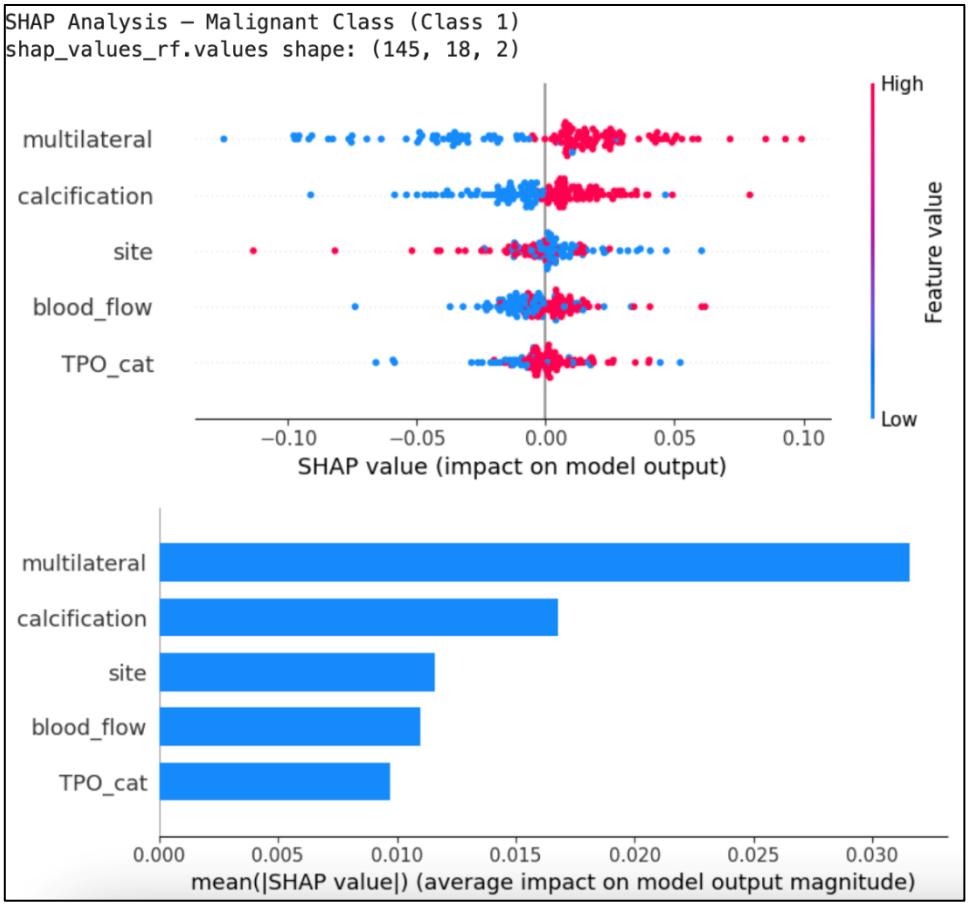

Fig. 27. SHAP Summary Plot for Malignant Class (Random Forest)

8. Summary of Findings

By going through our methodology of data cleaning, data analysis, and data visualization, we came away with many takeaways related to our two research questions. Some of our key findings by research question are below.

8.1 Research Question One Findings

- To determine whether clinical and ultrasound features significantly affect the prediction of thyroid malignancy.
- The chi-square statistical test identified multilateral, site, FT3, and calcification as the most significant features associated with thyroid malignancy ($p < 0.05$).

- Random Forest achieved the highest F1 score (0.98) and correctly predicted almost all malignant cases, with only 1 misclassification on the test set.
- MLP had the highest precision (0.97), though it misclassified more malignant nodules than RF.
- XGBoost and Ensemble models showed moderate performance, with lower recall and ROC AUC compared to RF.
- SHAP analysis confirmed multilateral, calcification, site, and blood flow as the most impactful features for both malignant and benign classifications, enhancing interpretability.
- Permutation importance reinforced the above SHAP findings, validating feature consistency.
- All models exhibited high sensitivity (recall) but relatively low specificity, likely due to the original class imbalance, even after applying SMOTE to balance the training data.

9. Discussion

This project provided a valuable opportunity to explore real-world data modeling while working as a multidisciplinary team. During collaboration, we observed differences in team members' approaches some prioritized statistical accuracy and model performance, while others emphasized interpretability and clinical relevance. These complementary perspectives ultimately strengthened the outcome but required coordination in model selection and evaluation strategies.

We initially considered traditional models like logistic regression and support vector machines but excluded them due to their lower performance and limited explainability compared to tree-based models. Instead, we focused on interpretable ensemble approaches (Random Forest, XGBoost) and used tools like SHAP and permutation importance to enhance clinical interpretability.

While the models achieved strong metrics, the project has several avenues for improvement. First, the dataset was limited to a single institution, which may impact generalizability. Future work should incorporate multi-institutional data or longitudinal patient records to assess prediction consistency. Additionally, deep learning techniques such as CNNs for image data or hybrid models combining imaging and clinical records could further enhance predictive power.

Finally, we recommend expanding the interpretability component to include counterfactual explanations or clinician-facing dashboards to improve adoption in real-world healthcare settings. Despite the challenges, this project demonstrated the potential of machine learning to improve thyroid malignancy detection and reduce diagnostic uncertainty in clinical workflows.

10. Limitations

While our team worked systematically through each phase of the project, there were several limitations that may have influenced the scope and generalizability of our results:

- Class Imbalance in Dataset: The original dataset had far fewer benign cases compared to malignant cases. Although we used SMOTE to synthetically balance the training data, this may not fully replicate real-world distributions and could impact the model's specificity.
- Single-Center Dataset: The dataset was sourced from Shengjing Hospital in China, which may limit the generalizability of our model to other populations, particularly those with different ethnic, demographic, or healthcare backgrounds.
- Lack of Imaging Data: Our analysis was based solely on structured clinical and ultrasound report data. Access to raw ultrasound images could further enhance predictive accuracy using deep learning techniques.
- Limited Feature Diversity: Some potentially relevant clinical variables (e.g., family history, lifestyle factors, nodule vascularity) were not included in the dataset, potentially omitting important predictors.
- Model Interpretability Constraints: While SHAP and permutation importance were effective in enhancing model transparency, some features (e.g., site or blood flow) lacked intuitive clinical interpretation due to ambiguous categorical encoding.
- Time Constraints: Due to project deadlines, exhaustive hyperparameter tuning, cross-validation, and model calibration were not performed. With additional time, more robust optimization could improve performance.

11. Appendix

11.1 GitHub Repository

All code, visualizations, and the complete Jupyter Notebook can be accessed at:
https://github.com/Bhavitha2300/thyroid-mal-project

11.2 Key SQL Query Sample

This is a representative query used to connect and fetch data from the MySQL database:

```
SELECT *
FROM thyroid_data;
```

11.3. Additional Tables and codes.

Malignancy Class Distribution (After Filtering):

Class	Count	Percentage
Benign (0)	450	62.2%
Malignant (1)	274	37.8%

```python
import matplotlib.pyplot as plt

# numeric features to visualize
cols_to_plot = ['age', 'FT3', 'FT4', 'TSH', 'TPO', 'TGAb', 'size']

# Convert decimal columns to float
for col in cols_to_plot:
    df[col] = df[col].astype(float)

# plot histogram and boxplot
def plot_distribution(col):
    plt.figure(figsize=(12, 5))

    # Histogram
    plt.subplot(1, 2, 1)
    plt.hist(df[col], bins=30, color='#66B3FF', edgecolor='black')
    plt.title("Histogram of " + col)
    plt.xlabel(col)
    plt.ylabel("Frequency")

    # Boxplot
    plt.subplot(1, 2, 2)
    plt.boxplot(df[col], vert=False)
    plt.title("Boxplot of " + col)
    plt.xlabel(col)

    plt.tight_layout()
    plt.show()

# Loop through and plot each feature
for feature in cols_to_plot:
    plot_distribution(feature)
```

```python
# Check distribution
size_xtab = pd.crosstab(df_final_categorized['size_cat'],
                        df_final_categorized['mal']).rename(
                        columns={0: 'Benign (0)', 1: 'Malignant (1)'})

age_xtab  = pd.crosstab(df_final_categorized['age_cat'],
                        df_final_categorized['mal']).rename(
                        columns={0: 'Benign (0)', 1: 'Malignant (1)'})

print(size_xtab, '\n')
print(age_xtab)
```

```
mal       Benign (0)  Malignant (1)
size_cat
≤1 cm             13            208
1–2 cm            10            243
2–4 cm             6            191
>4 cm              1             52

mal      Benign (0)  Malignant (1)
age_cat
<55 y            26            548
≥55 y             4            146
```

11.4 Python Libraries Used

pandas
numpy
matplotlib.pyplot
seaborn
scipy.stats
sklearn.model_selection
sklearn.preprocessing
sklearn.ensemble (RandomForestClassifier, VotingClassifier)
sklearn.neural_network (MLPClassifier)
sklearn.metrics
sklearn.inspection
xgboost
imblearn.over_sampling (SMOTE)
shap
MySQLdb

References

1.Bomeli, S. R., LeBeau, S. O., & Ferris, R. L. (2010). Evaluation of a thyroid nodule. Otolaryngologic Clinics of North America, 43(2), 229–vii. https://doi.org/10.1016/j.otc.2010.01.002

2.Chen, X., Zhou, Y., Zhou, M., Yin, Q., & Wang, S. (2018). Diagnostic values of free triiodothyronine and free thyroxine and the ratio of free triiodothyronine to free thyroxine in thyrotoxicosis. BioMed Research International, 2018, Article 5842534. https://doi.org/10.1155/2018/5842534

3.Cheng, J., Han, B., Chen, Y., et al. (2024). Clinical risk factors and cancer risk of thyroid imaging reporting and data system category 4A thyroid nodules. Journal of Cancer Research and Clinical Oncology, 150, 327. https://doi.org/10.1007/s00432-024-05847-7

4.Haugen, B. R., Alexander, E. K., Bible, K. C., Doherty, G. M., Mandel, S. J., Nikiforov, Y. E., & Wartofsky, L. (2016). 2015 American Thyroid Association management guidelines for adult patients with thyroid nodules and differentiated thyroid cancer: The American Thyroid Association Guidelines Task Force on Thyroid Nodules and Differentiated Thyroid Cancer. Thyroid, 26(1), 1–133. https://doi.org/10.1089/thy.2015.0020

5.Xi, N. M., Wang, L., & Yang, C. (2022). Improving the diagnosis of thyroid cancer by machine learning and clinical data. Scientific Reports, 12(1), 11143. https://doi.org/10.1038/s41598-022-15342-z