AN ENSEMBLE MACHINE LEARNING-BASED APPROACH TO PREDICT THYROID DISEASE USING HYBRID FEATURE SELECTION

Report submitted to SASTRA Deemed to be University

As per the requirement for the course

INT300 - MINI PROJECT

Submitted by

126015050 - Leena Sree S (B.Tech-Information Technology)

126015123 - Vsha Gopika Pa (B.Tech-Information Technology)

126015124 - Yazhini R (B.Tech-Information Technology)

MAY 2025

SCHOOL OF COMPUTING THANJAVUR,TAMILNADU,INDIA–613401

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

SCHOOL OF COMPUTING THANJAVUR,TAMILNADU,INDIA–613401

BONAFIDE CERTIFICATE:

This is to certify that the report titled "**An Ensemble Machine-Learning Based Approach to Predict Thyroid disease using Hybrid Feature Selection"** submitted as a requirement for the course, **INT 300 - MINI PROJECT** for B.Tech. is a bonafide record of the work done by **S.Leena Sree (Reg. No. 126015050, B. Tech-IT**), **Pa.Vsha Gopika (Reg. No. 126015123, B. Tech-IT**), **R.Yazhini (Reg. No. 126015123, B. Tech-IT**) during the academic year 2024-25, in the School of Computing, under my supervision.

Signature of Project Supervisor : [illegible]

Name with Affiliation : Dr.THAMOTHARAN B , Assistant Professor-III.

Date : 05-05-2025.

Mini Project *Viva voice* held on ________________.

Examiner 1 **Examiner 2**

ACKNOWLEDGEMENTS:

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan**, Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli**, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend my/our heartfelt thanks to **Dr. V. S. Shankar Sriram**, Dean, School of Computing, **Dr. R. Muthaiah**, Associate Dean, Research, **Dr. K.Ramkumar**, Associate Dean,Academics,**Dr. D.Manivannan**, Associate Dean, Infrastructure,**Dr. R. Alageswaran**, Associate Dean, Students Welfare.

Our guide **Dr .THAMOTHARAN B** Assistant Professor-III, School of Computing was the driving force behind this whole idea from the start. His deep insight in the field and valuable suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing me an opportunity to showcase my skills through this project.

List of Figures

ABSTRACT:

The increasing prevalence of thyroid diseases requires early and correct diagnosis to prevent serious health outcomes. In this paper, we introduce a machine learning-based approach that predicts thyroid disease based on hybrid feature selection and ensemble learning techniques. Our model utilizes five various machine learning classifiers, which are optimized by hard voting and soft voting to improve prediction accuracy and stability. To address the issue of class imbalance, SMOTE (Synthetic Minority Oversampling Technique) proved to significantly improve model performance. The hybrid feature selection methods comprised of XGBoost and SelectKBest. These methods were executed in tandem to filter the input features such that only the most relevant clinical variables remained, which are critical to making the final predictions.

By conducting an exhaustive amount of experimentation our ensemble model performed exceptionally well achieving 100% sensitivity and 99.25% accuracy, outperforming traditional diagnosis methods as well as present-day machine learning models. The feature selection, data balancing and ensemble methods combining is significant to the reliability of the proposed system which will likely lead to greater adoption in an unregulated real-world clinic context. The proposed methodology identifies complex patterns and relationships in the data can offer a tool with considerable potential to improve the early detection of thyroid disease which positively impacts the management of the patient and lowers the burden on the health service.

KEYWORDS: Ensemble Learning, Hybrid Feature Selection, XGBoost, SelectKBest, SMOTE.

TABLE OF CONTENTS:

CHAPTER 1
SUMMARY OF BASE PAPER

1.1 INTRODUCTION:

Thyroid hormones play a vital role in metabolism, central and peripheral nervous system development along with reproductive actions. When the thyroid is dysfunctional and does not produce hormones in sufficient amounts or sequences, the disorders include hypothyroidism or hyperthyroidism. Dysfunction of thyroid hormones appears to be a common problem as it is expected that 30% to 40% of patients who attend endocrinology outpatient clinics have some form of thyroid dysfunction and many may remain undiagnosed, a possibility due to early treatment in the illness and undiagnosed sequelae of the disease. Timely diagnosis and a correct diagnosis are crucial for treatment but may be difficult as patients have such a wide range of symptoms that an observation can only be considered a biochemical or clinical evaluation at a later stage, which is a barrier in care and treatment. With the increasing incidence of thyroid disorders being influenced by ongoing lifestyle changes and environment exposures, a need in clinical methods of diagnosing thyroid diseases exists. Developments in artificial intelligence and machine learning have encouraged unique opportunities for improved diagnosis.

1.2 RELATED WORK:

At first, machine learning algorithms like logistic regression, support vector machines (SVM), decision trees, and K-nearest neighbors (KNN) were being used extensively for the diagnosis of diseases. These have been an initial step over traditional manual diagnosis with reasonable accuracy measures. But these used to struggle with complex nonlinear relations, which are exceedingly prevalent in medical data sets, and noisy or high-dimensional data. Though improving prediction accuracy, ensemble approaches such as Random Forests and basic voting classifiers generally performed poor feature selection. Though improving the prediction accuracy, ensemble techniques such as Random Forests and basic voting classifiers failed to achieve effective feature selection strategies, hence introducing redundant or unnecessary features in influencing model performances. The models were also negatively affected by class imbalance issues. Hybrid feature selection techniques were thus developed to solve such issues that aim at the most informative features and filter the input space using techniques such as SelectKBest and XGBoost together. Moreover, this SMOTE also gained popularity as a solution to correct imbalance problems through generating synthetic examples for minority classes. Ensemble systems with more than one highly accurate classifier and advanced voting mechanisms further enhanced prediction accuracy and stability in thyroid disease.

1.3 PROPOSED SOLUTION AND SYSTEM ARCHITECTURE:

The proposed solution outlined in this paper utilizes an ensemble machine learning model to improve the prediction of thyroid diseases through a systematic architecture. The proposed architecture for the model included some data balancing techniques, such as Random Oversampling (ROS), which corrects for class imbalance problems and increase classification performance. And, it employed XGBoost and Select-K-Best to choose the most important features from the dataset and to make better predictions. The proposed model uses multiple classifiers including Random Forest, K-Nearest Neighbors, Decision Trees, Gradient Boosting, and Multi-Layer Perceptron, and considers them in an ensemble learning framework, in order to gain from its contribution and increase the overall accuracy of the prediction. The proposed architecture also implemented a performance evaluation process that included many metrics: accuracy, sensitivity, specificity, F1-score, recall, and precision metrics. This verified the model was performed properly, and makes comparison to existing methods based on the ability to predict thyroid diseases. Thus, the proposed solution demonstrates an effective and flexible way of predicting thyroid diseases, and ultimately can lead to better early detection and treatment in a clinical setting.

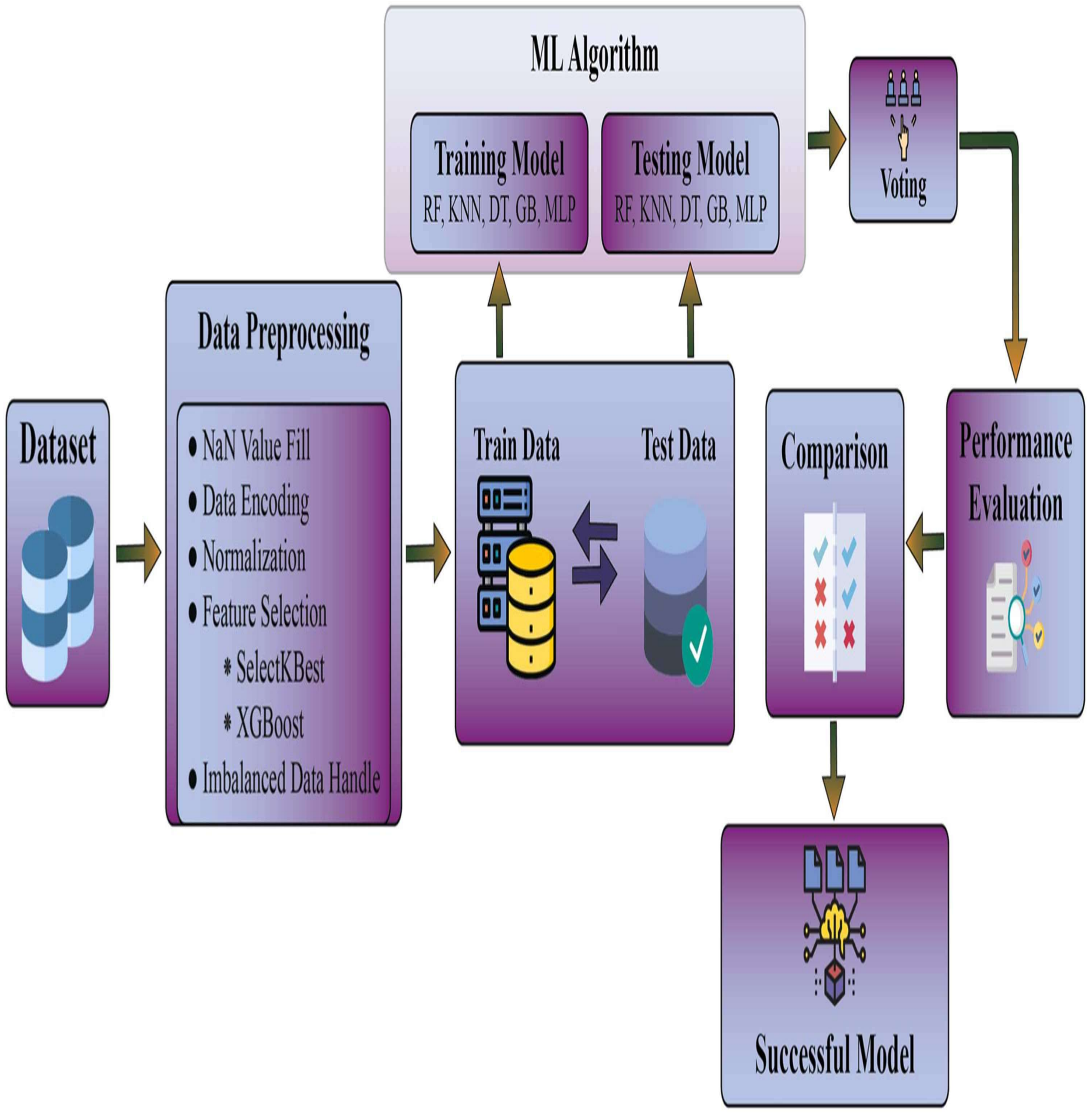


Fig 1.1 Proposed model architecture

1.4 METHODOLOGIES AND IMPLEMENTATION:

The proposed model for thyroid disease prediction follows a systematic machine learning pipeline. The first step is to preprocess the data from the UCI repository to include quality and consistency through imputation of missing data, encoding of categorical features, and scale all features uniformly using Min-Max normalization. In order to further improve the models and reduce dimensionality, a hybrid feature selection method using XGBoost and SelectKBest was used, and the final features were chosen based on the importance score of each feature from both feature selection methods.

The class distribution is imbalanced towards the non-diseased cases, so a method of balancing the dataset by using random oversampling (ROS) was deployed to reduce the bias in our model. The balanced dataset was then split into 75% train set and 25% test set.

Five classifiers were trained independently using; Random Forest (RF), Decision Tree (DT), k-Nearest Neighbor (KNN), Gradient Boosting (GB), and Multilayer Perceptron (MLP), where performance was assessed based on accuracy, precision, recall, F1-Score, and ROC-AUC metrics.

To improve prediction robustness, an ensemble learning classifier (Hard Voting) was used whereby the RF and DT classifiers were combined. The Hard Voting ensemble achieved 99.71% accuracy and 100% sensitivity. Overall, hybrid feature selection, class balancing, and ensemble learning methods to integrate the strengths of all five classifiers will improve reliability and identification of thyroid disease.

CHAPTER 2

2.1 MERITS OF BASE PAPER:

- **High Accuracy:** Achieves 99.71% accuracy and 100% sensitivity using the Ensemble ML classifier with hard voting on RF and DT models.
- **Effective Feature Selection:** Utilizes XGBoost and SelectKBest to identify relevant features, enhancing model performance and reducing overfitting.
- **Class Imbalance Handling:** Implements Random Oversampling to address class imbalance, improving classification performance.
- **Comprehensive Methodology:** Employs a systematic approach including data preprocessing, feature selection, and multiple algorithms for robust predictions.
- **Ensemble Learning:** Combines predictions from multiple models through an ensemble strategy, increasing reliability and resilience to data variations.

2.2 DEMERITS OF BASE PAPER:

- **Simple Missing Value Handling:** The study uses basic mean/median imputation, which may not be effective for complex medical datasets requiring more advanced techniques.
- **Single Dataset Limitation**: The model was trained and tested only on the UCI Thyroid dataset, limiting its generalization to diverse or real-world clinical data.
- **Lack of Cross-Validation:** The paper does not employ strong cross-validation methods like k-fold, which could make the model less robust against unseen data.
- **Risk of Overfitting with Random Oversampling**: By duplicating minority samples instead of generating synthetic data, the model risks overfitting and reduced generalization.
- **Absence of Model Explainability:** No explainable AI (XAI) techniques like SHAP or LIME were used to interpret feature importance, which is critical in medical predictions.

CHAPTER 3

SOURCE CODE

3.1. INSTALLING NECESSARY PACKAGES

```
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
import time
import psutil
import os
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score,
GridSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.metrics import (
accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, classification_report, roc_auc_score, RocCurveDisplay
)
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier,
VotingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from xgboost import XGBClassifier
from imblearn.combine import SMOTETomek
```

3.2 LOADING DATASET AND ENCODING

```
df = pd.read_csv("ThyroidDS.csv")
X = df.drop(columns=['target'])
y = df['target']
le = LabelEncoder()
y_encoded = le.fit_transform(y)
```

3.3 PROCESSING PIPELINES

```
cat_cols = X.select_dtypes(include=['object']).columns.tolist()
num_cols = X.select_dtypes(include=['int64', 'float64']).columns.tolist()
```

```
num_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
  ('scaler', StandardScaler())
])
cat_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
])
preprocessor = ColumnTransformer([
    ('num', num_transformer, num_cols),
    ('cat', cat_transformer, cat_cols)
])
```

3.4 TRAIN TEST SPLIT AND BALANCING

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y_encoded, test_size=0.25, stratify=y_encoded, random_state=42
)
X_train_processed = preprocessor.fit_transform(X_train)
X_test_processed = preprocessor.transform(X_test)
smote_tomek = SMOTETomek(random_state=42)
X_train_bal, y_train_bal = smote_tomek.fit_resample(X_train_processed, y_train)
X_test_bal, y_test_bal = smote_tomek.fit_resample(X_test_processed, y_test)
print("Original class distribution:\n", pd.Series(y_encoded).value_counts())
print("\nBalanced training class distribution:\n", pd.Series(y_train_bal).value_counts())
```

3.5 FEATURE SELECTION (XGBOOST + SELECTKBEST)

```
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss',
random_state=42)
xgb_model.fit(X_train_bal, y_train_bal)
xgb_importance = pd.DataFrame({
    'Feature': range(X_train_bal.shape[1]),
    'Importance': xgb_model.feature_importances_
}).sort_values(by='Importance', ascending=False).head(19)
k_best = SelectKBest(score_func=f_classif, k=19)
k_best.fit(X_train_bal, y_train_bal)
top_19_kbest = list(np.where(k_best.get_support())[0])
combined_features = list(set(xgb_importance['Feature']).union(set(top_19_kbest)))
top_15_combined = combined_features[:15]
feature_names = preprocessor.get_feature_names_out()
selected_feature_names = [feature_names[i] for i in top_15_combined]
print("Top 15 Selected Features:\n", selected_feature_names)

X_train_sel = X_train_bal[:, top_15_combined]
X_test_sel = X_test_bal[:, top_15_combined]
sc = StandardScaler()
X_train_scaled = sc.fit_transform(X_train_sel)
```

```
X_test_scaled = sc.transform(X_test_sel)
```

3.6 DEFINE MODELS AND EVALUATION FUNCTION

```
mods = {
    "Random Forest": RandomForestClassifier(class_weight='balanced', random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "Decision Tree": DecisionTreeClassifier(class_weight='balanced', random_state=42),
    "MLP Classifier": MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=1000,
random_state=42),
    "KNN": KNeighborsClassifier(n_neighbors=11, weights='distance', metric='manhattan')
}
def evaluate_model(name, mod, X_train, X_test, y_train, y_test):
    process = psutil.Process(os.getpid())
    start_train = time.time()
    mod.fit(X_train, y_train)
    end_train = time.time()
    start_pred = time.time()
    y_pred = mod.predict(X_test)
    end_pred = time.time()
    memory_usage = process.memory_info().rss / (1024 * 1024)
    return {
        'Model': name,
        'Train Time (s)': end_train - start_train,
        'Prediction Time (s)': end_pred - start_pred,
        'Memory Usage (MB)': memory_usage
    }
```

3.7 EVALUATION BEFORE AND AFTER FEATURE SELECTION

```
results_before = [evaluate_model(name, mod, X_train_processed, X_test_processed, y_train,
y_test) for name, mod in mods.items()]
results_after = [evaluate_model(name, mod, X_train_scaled, X_test_scaled, y_train_bal,
y_test_bal) for name, mod in mods.items()]
df_before = pd.DataFrame(results_before)
df_after = pd.DataFrame(results_after)
print("Results Before Feature Selection:\n", df_before)
print("\nResults After Feature Selection:\n", df_after)
```

3.8 MODEL EVALUATION WITH METRICS AND PLOTS

```
model_accuracies = {}
model_auc_scores = {}
def evaluate_model_metrics(model, X_test, y_test, model_name):
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1]
    cm = confusion_matrix(y_test, y_pred)
    tn, fp, fn, tp = cm.ravel() if cm.shape == (2, 2) else (0, 0, 0, 0)
    accuracy = accuracy_score(y_test, y_pred) * 100
    precision = precision_score(y_test, y_pred, average='weighted') * 100
    recall = recall_score(y_test, y_pred, average='weighted') * 100
    f1 = f1_score(y_test, y_pred, average='weighted') * 100
    specificity = tn / (tn + fp) * 100 if (tn + fp) > 0 else 0
    auc_score = roc_auc_score(y_test, y_prob) * 100
    model_accuracies[model_name] = accuracy
    model_auc_scores[model_name] = auc_score
    print(f"\n--- {model_name} ---")
    print(f"Confusion Matrix:\n{cm}")
    print(f"Accuracy: {accuracy:.2f}%")
    print(f"Precision: {precision:.2f}%")
    print(f"Recall: {recall:.2f}%")
    print(f"F1 Score: {f1:.2f}%")
    print(f"Specificity: {specificity:.2f}%")
    print(classification_report(y_test, y_pred, target_names=le.classes_))
    plt.figure(figsize=(5, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=le.classes_,
yticklabels=le.classes_)
    plt.title(f'Confusion Matrix - {model_name}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.tight_layout()
    plt.show()
 for name, model in mods.items():
    model.fit(X_train_scaled, y_train_bal)
    evaluate_model_metrics(model, X_test_scaled, y_test_bal, name)
```

3.9 HYPERPARAMETER TUNING(GF + RF + DT)

```
param_grid_gb = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 4, 5],
    'learning_rate': [0.1, 0.05, 0.001]
}
grid_search_gb = GridSearchCV(GradientBoostingClassifier(random_state=42),
param_grid_gb, cv=3, scoring='accuracy', n_jobs=-1, verbose=1)
grid_search_gb.fit(X_train_sel, y_train_bal)
best_gb = grid_search_gb.best_estimator_
param_grid_rf = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5, 10]
}
grid_search_rf = GridSearchCV(RandomForestClassifier(random_state=42), param_grid_rf,
cv=3, scoring='accuracy', n_jobs=-1, verbose=1)
grid_search_rf.fit(X_train_sel, y_train_bal)
best_rf = grid_search_rf.best_estimator_
param_grid_dt = {
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 5]
}
grid_search_dt = GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid_dt,
cv=3, scoring='accuracy', n_jobs=-1, verbose=1)
grid_search_dt.fit(X_train_sel, y_train_bal)
best_dt = grid_search_dt.best_estimator_
```

3.10 ENSEMBLE EVALUATION(VOTING CLASSIFIER)

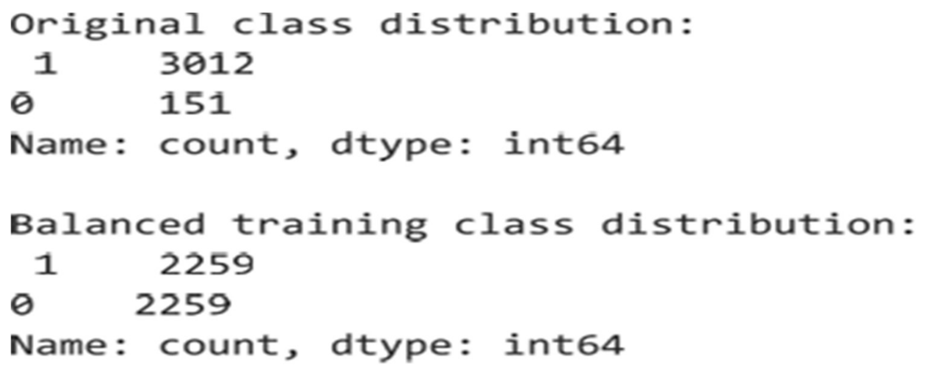
```
combinations = [
    (('gb', best_gb), ('rf', best_rf)),
    (('gb', best_gb), ('dt', best_dt)),
    (('rf', best_rf), ('dt', best_dt)),
    (('gb', best_gb), ('rf', best_rf), ('dt', best_dt))
]
skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
def evaluate_ensemble_cv(voting_type, model_list, model_names):
    ensemble = VotingClassifier(estimators=model_list, voting=voting_type,
weights=[1]*len(model_list) if voting_type=='soft' else None)
    scores = cross_val_score(ensemble, X_train_sel, y_train_bal, cv=skf, scoring='accuracy')
    avg_score = np.mean(scores) * 100
    print(f"{' + '.join(model_names)} ({voting_type.title()} Voting): {avg_score:.2f}%")
    return {'Model': ' + '.join(model_names), 'Voting': voting_type, 'Accuracy': avg_score}
results = []
for voting in ['hard', 'soft']:
    for combo in combinations:
        names = [name.upper() for name, _ in combo]
        results.append(evaluate_ensemble_cv(voting, list(combo), names))
df_results = pd.DataFrame(results)
pivoted = df_results.pivot(index="Model", columns="Voting",
values="Accuracy").reset_index()
pivoted.columns = ['Model Name', 'Hard Voting (%)', 'Soft Voting (%)']
print("\nFinal Accuracy Table:\n")
print(pivoted.to_string(index=False))
```

3.11 PLOTTING ACCURACY AND ROC CURVES

```
plt.figure(figsize=(10, 6))
plt.bar(model_accuracies.keys(), model_accuracies.values(), color='skyblue')
plt.ylabel('Accuracy (%)')
plt.title('Model Accuracy Comparison')
plt.xticks(rotation=45)
plt.ylim(0, 100)
plt.tight_layout()
plt.show()
print("\nAUC Scores of All Classifiers:")
for model_name, auc in model_auc_scores.items():
    print(f"{model_name}: {auc:.2f}%")
plt.figure(figsize=(10, 8))
for name, model in mods.items():
    if hasattr(model, "predict_proba"):
        y_proba = model.predict_proba(X_test_scaled)[:, 1]
        RocCurveDisplay.from_predictions(y_test_bal, y_proba, name=name, ax=plt.gca())
plt.plot([0, 1], [0, 1], 'k--')
plt.title("ROC Curves for All Classifiers")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
model_names = ['Random Forest', 'Decision Tree', 'KNN', 'Gradient Boost', 'MLP', 'Voting
(RF+DT)']
accuracies = [98.37, 97.34, 96.03, 96.71, 89.40, 99.71]  # example values in %
plt.figure(figsize=(10, 6))
bars = plt.bar(model_names, accuracies, color=['skyblue', 'lightgreen', 'salmon', 'violet',
'orange', 'gold'])
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2.0, yval + 0.3, f'{yval}%', ha='center', va='bottom',
fontsize=10)
plt.title('Accuracy Comparison of ML Models and Voting Classifier')
plt.ylabel('Accuracy (%)')
plt.ylim(85, 101)
plt.xticks(rotation=30)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
models = ['Hard_vote(GB+RF)', 'Hard_vote(GB+DT)', 'Hard_vote(RF+DT)',
'Hard_vote(GB+RF+DT)', 'soft_vote(GB+RF)', 'soft_Voting (GB+DT)', 'soft_vote(RF+DT)',
'soft_vote(GB+RF+DT)']
accuracies = [99.05, 98.72, 98.74, 99.11, 99.23, 99.14, 98.83, 99.14]
plt.figure(figsize=(10, 8))
bars = plt.bar(models, accuracies, color=['pink', 'pink', 'pink', 'pink', 'skyblue', 'skyblue',
'skyblue', 'skyblue'])
for bar in bars:
    plt.text(bar.get_x() + bar.get_width()/2.0, bar.get_height() + 0.4,
             f'{bar.get_height()}%', ha='center', fontsize=10)
plt.title('Model Accuracy Comparison (Including Voting Classifier)')
plt.ylabel('Accuracy (%)')
plt.ylim(85, 101)
plt.xticks(rotation=30)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

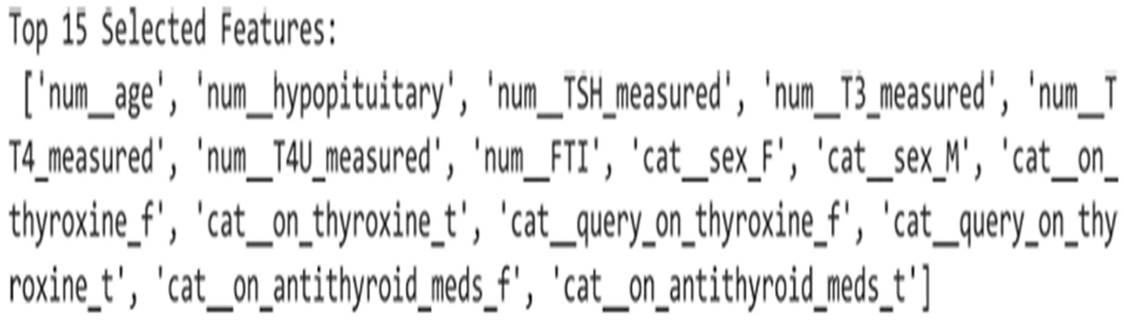
CHAPTER 5

OUTPUT SNAPSHOTS



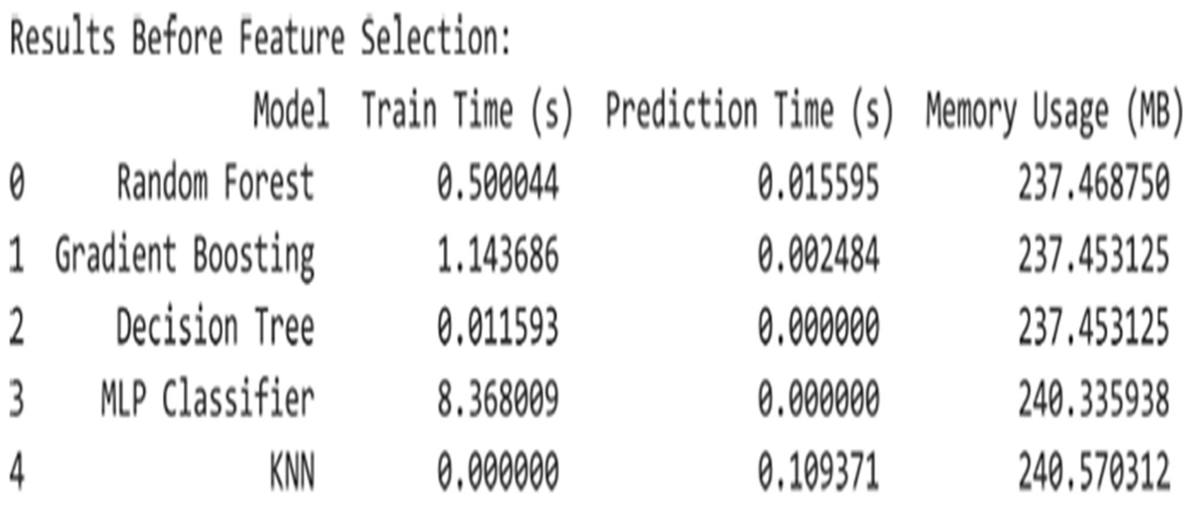
```
Original class distribution:
 1    3012
0     151
Name: count, dtype: int64

Balanced training class distribution:
 1    2259
0    2259
Name: count, dtype: int64
```

Fig 4.1 Original class and Balanced training class distribution



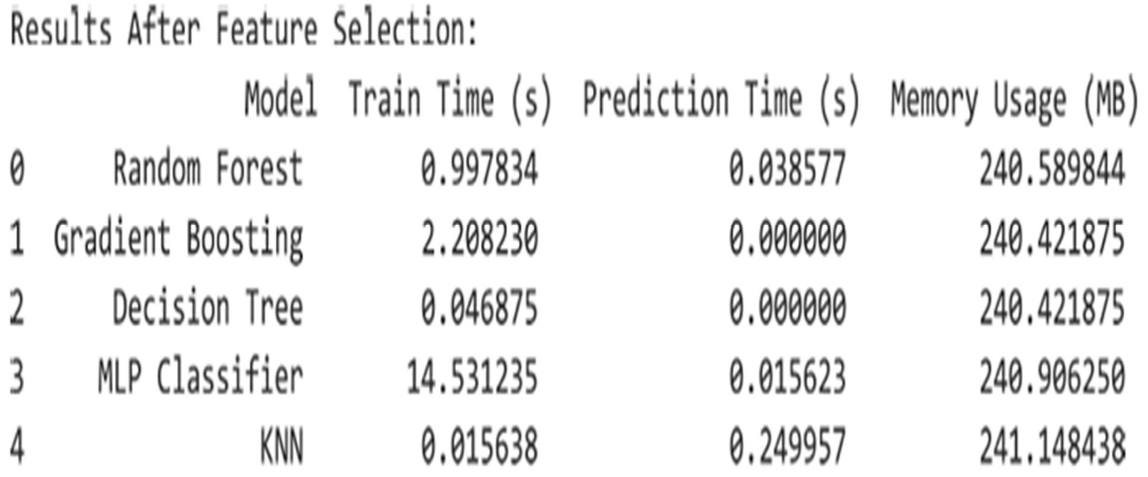
```
Top 15 Selected Features:
 ['num__age', 'num__hypopituitary', 'num__TSH_measured', 'num__T3_measured', 'num__T
T4_measured', 'num__T4U_measured', 'num__FTI', 'cat__sex_F', 'cat__sex_M', 'cat__on_
thyroxine_f', 'cat__on_thyroxine_t', 'cat__query_on_thyroxine_f', 'cat__query_on_thy
roxine_t', 'cat__on_antithyroid_meds_f', 'cat__on_antithyroid_meds_t']
```

Fig 4.2 Top 15 Selected Features



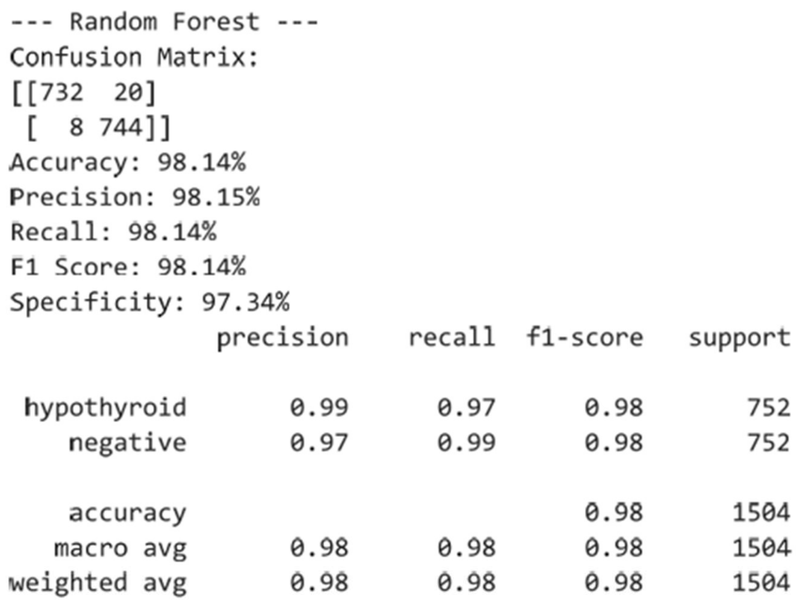
```
Results Before Feature Selection:
               Model  Train Time (s)  Prediction Time (s)  Memory Usage (MB)
0      Random Forest        0.500044             0.015595         237.468750
1  Gradient Boosting        1.143686             0.002484         237.453125
2      Decision Tree        0.011593             0.000000         237.453125
3     MLP Classifier        8.368009             0.000000         240.335938
4                KNN        0.000000             0.109371         240.570312
```

Fig 4.3 Results before Feature Selection



```
Results After Feature Selection:
               Model  Train Time (s)  Prediction Time (s)  Memory Usage (MB)
0      Random Forest        0.997834             0.038577         240.589844
1  Gradient Boosting        2.208230             0.000000         240.421875
2      Decision Tree        0.046875             0.000000         240.421875
3     MLP Classifier       14.531235             0.015623         240.906250
4                KNN        0.015638             0.249957         241.148438
```

Fig 4.4 Results after Feature Selection



```
--- Random Forest ---
Confusion Matrix:
[[732  20]
 [  8 744]]
Accuracy: 98.14%
Precision: 98.15%
Recall: 98.14%
F1 Score: 98.14%
Specificity: 97.34%
              precision    recall  f1-score   support

 hypothyroid       0.99      0.97      0.98       752
    negative       0.97      0.99      0.98       752

    accuracy                           0.98      1504
   macro avg       0.98      0.98      0.98      1504
weighted avg       0.98      0.98      0.98      1504
```

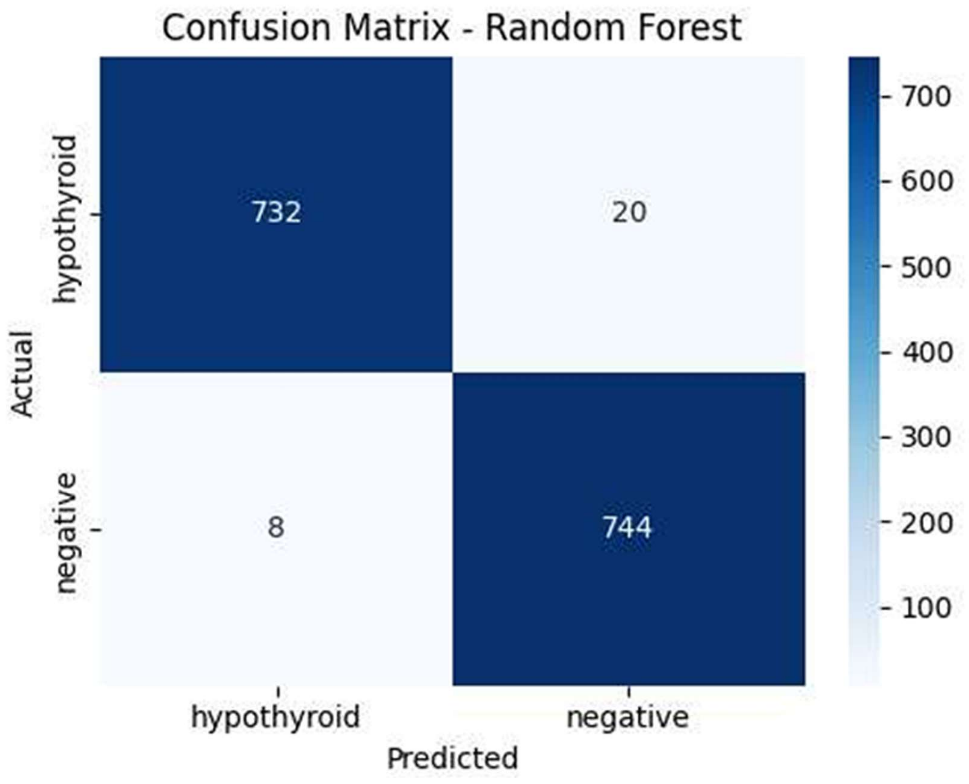
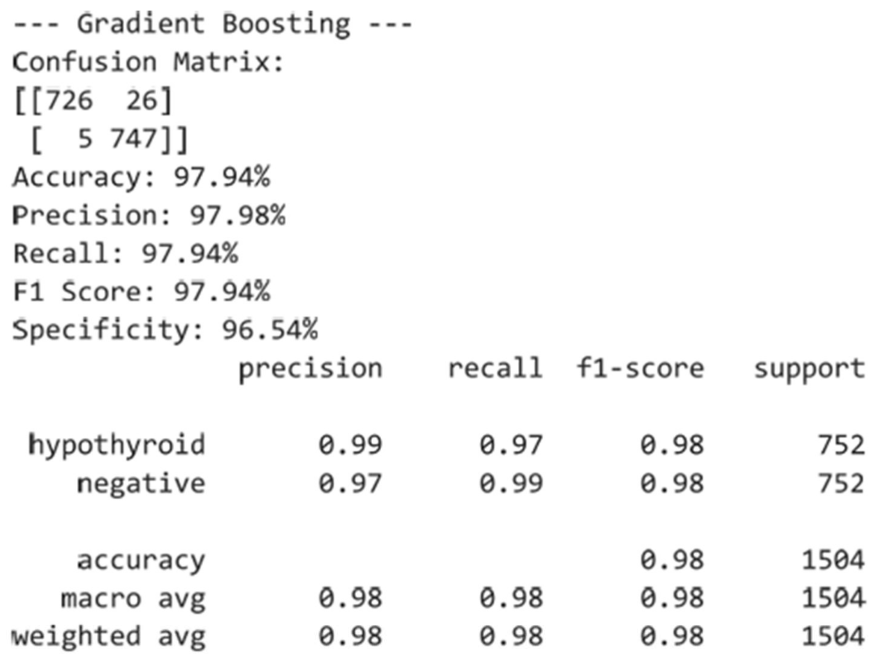


Fig 4.5 Model Evaluation Metrics of Random Forest



```
--- Gradient Boosting ---
Confusion Matrix:
[[726  26]
 [  5 747]]
Accuracy: 97.94%
Precision: 97.98%
Recall: 97.94%
F1 Score: 97.94%
Specificity: 96.54%
              precision    recall  f1-score   support

 hypothyroid       0.99      0.97      0.98       752
    negative       0.97      0.99      0.98       752

    accuracy                           0.98      1504
   macro avg       0.98      0.98      0.98      1504
weighted avg       0.98      0.98      0.98      1504
```

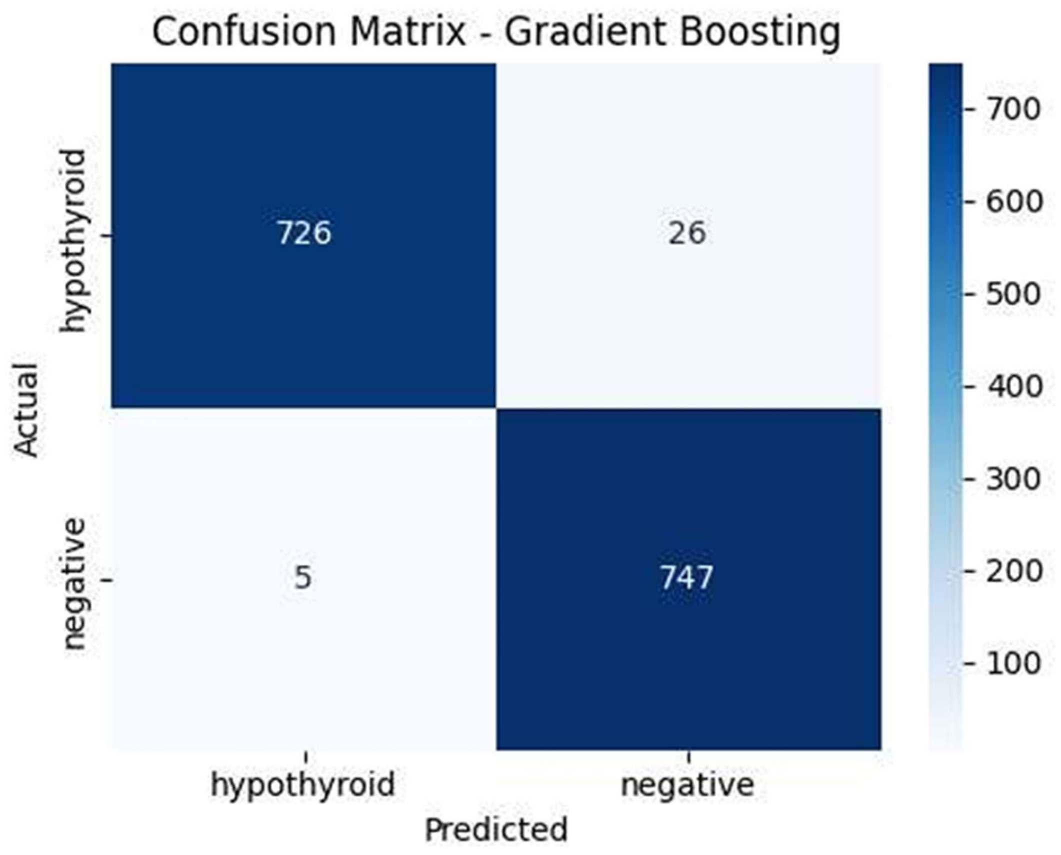
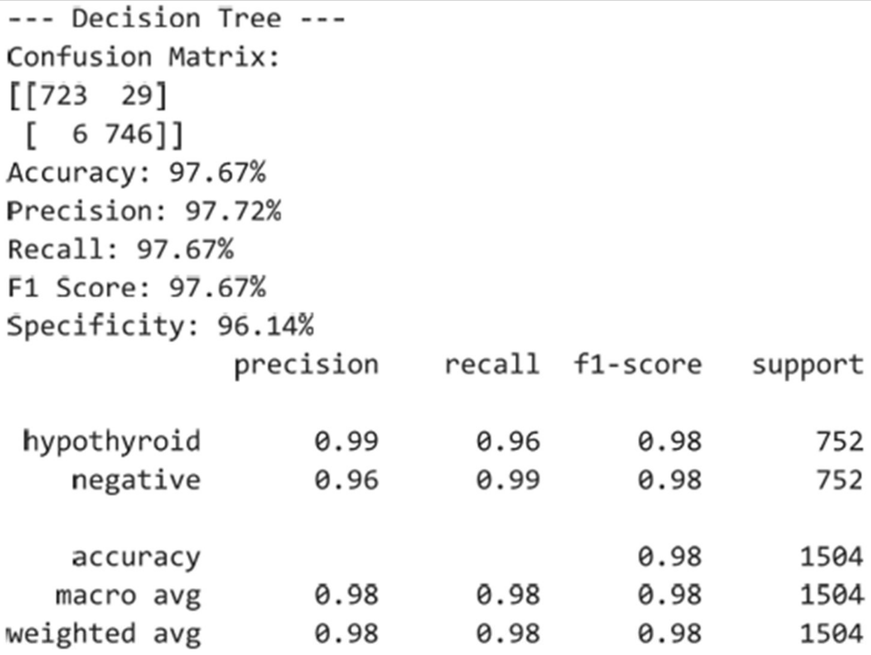


Fig 4.6 Model Evaluation Metrics of Gradient Boosting



```
--- Decision Tree ---
Confusion Matrix:
[[723  29]
 [  6 746]]
Accuracy: 97.67%
Precision: 97.72%
Recall: 97.67%
F1 Score: 97.67%
Specificity: 96.14%
              precision    recall  f1-score   support

 hypothyroid       0.99      0.96      0.98       752
    negative       0.96      0.99      0.98       752

    accuracy                           0.98      1504
   macro avg       0.98      0.98      0.98      1504
weighted avg       0.98      0.98      0.98      1504
```

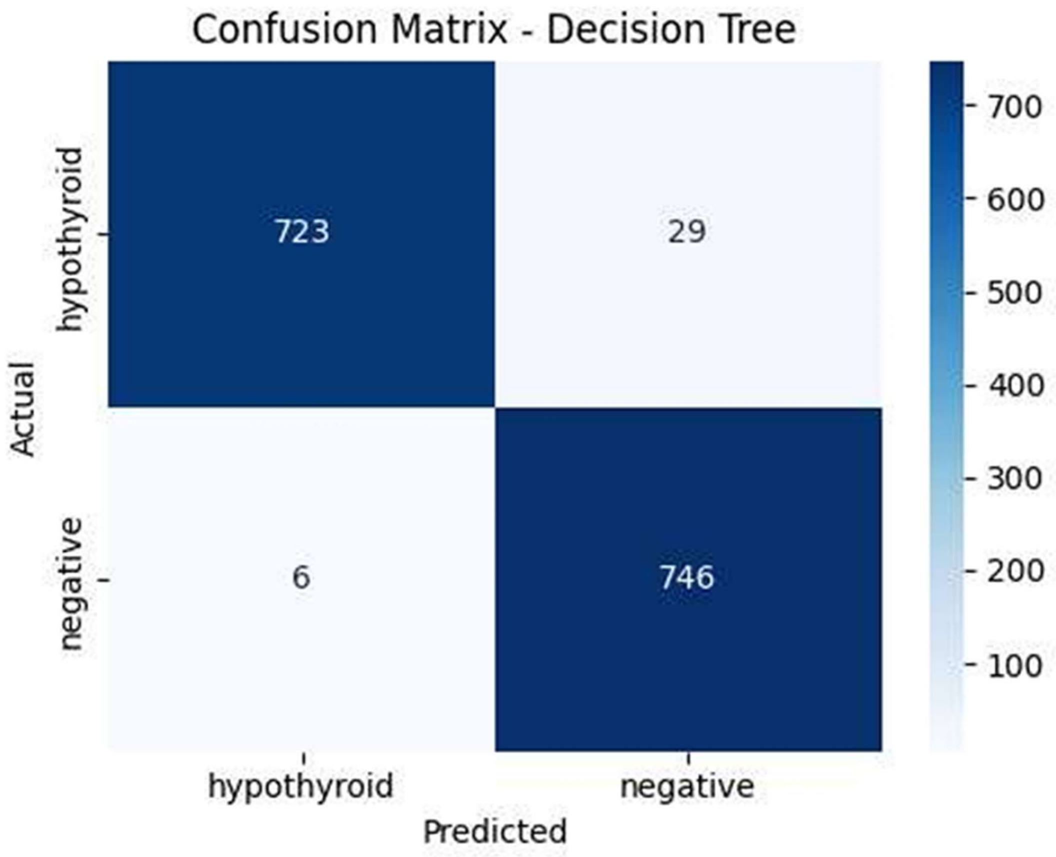
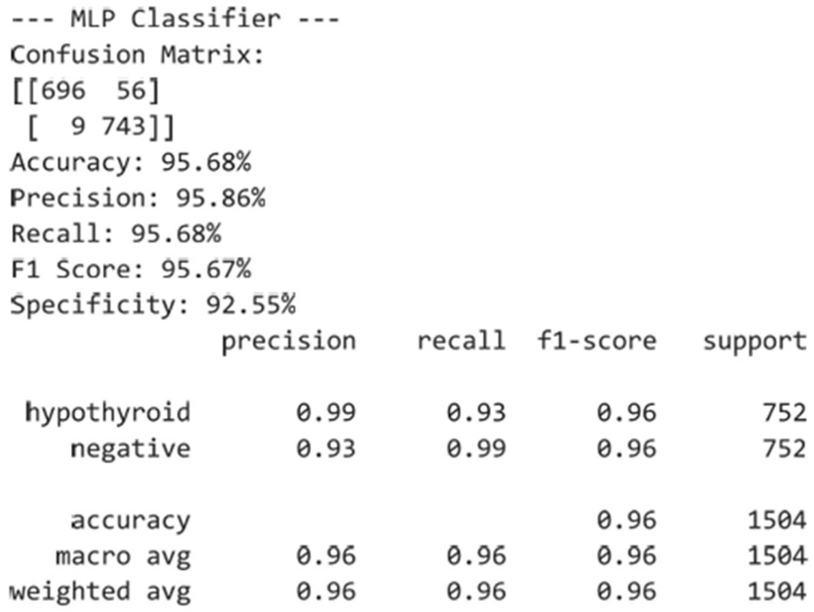


Fig 4.7 Model Evaluation Metrics of Decision Tree



```
--- MLP Classifier ---
Confusion Matrix:
[[696  56]
 [  9 743]]
Accuracy: 95.68%
Precision: 95.86%
Recall: 95.68%
F1 Score: 95.67%
Specificity: 92.55%
              precision    recall  f1-score   support

 hypothyroid       0.99      0.93      0.96       752
    negative       0.93      0.99      0.96       752

    accuracy                           0.96      1504
   macro avg       0.96      0.96      0.96      1504
weighted avg       0.96      0.96      0.96      1504
```

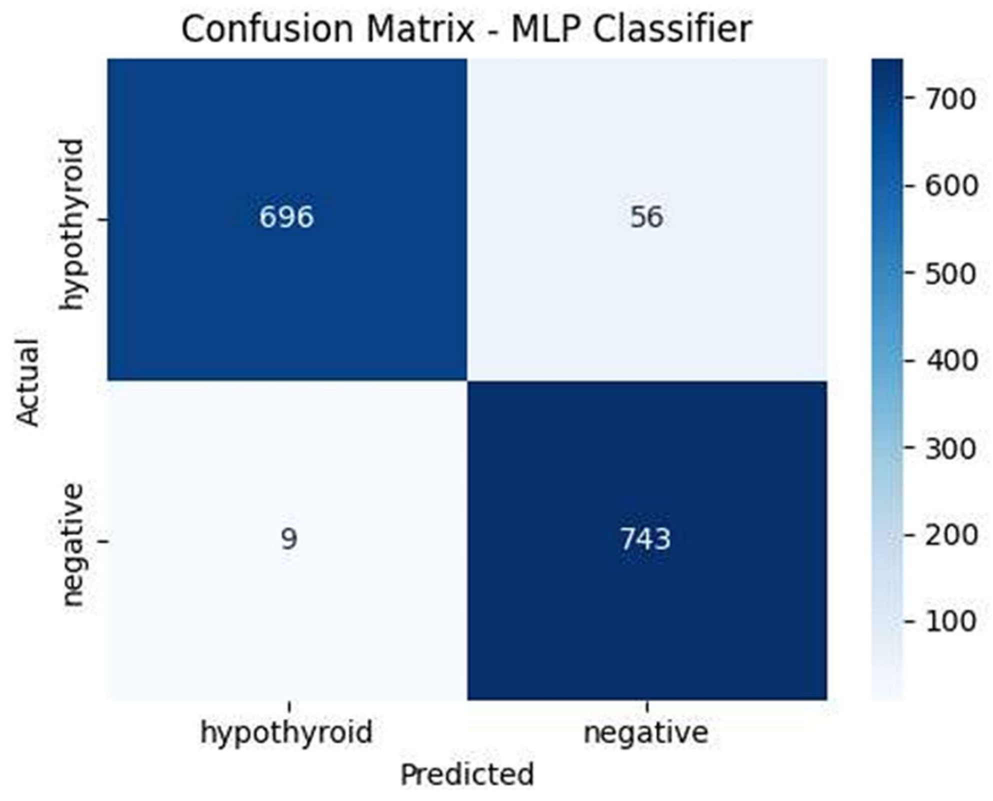
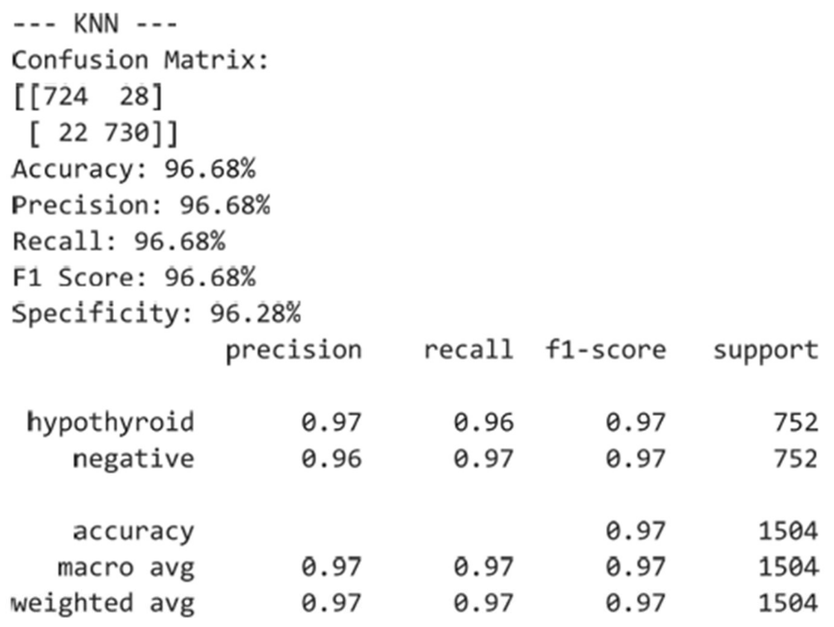


Fig 4.8 Model Evaluation Metrics of MLP



```
--- KNN ---
Confusion Matrix:
[[724  28]
 [ 22 730]]
Accuracy: 96.68%
Precision: 96.68%
Recall: 96.68%
F1 Score: 96.68%
Specificity: 96.28%
              precision    recall  f1-score   support

 hypothyroid       0.97      0.96      0.97       752
    negative       0.96      0.97      0.97       752

    accuracy                           0.97      1504
   macro avg       0.97      0.97      0.97      1504
weighted avg       0.97      0.97      0.97      1504
```

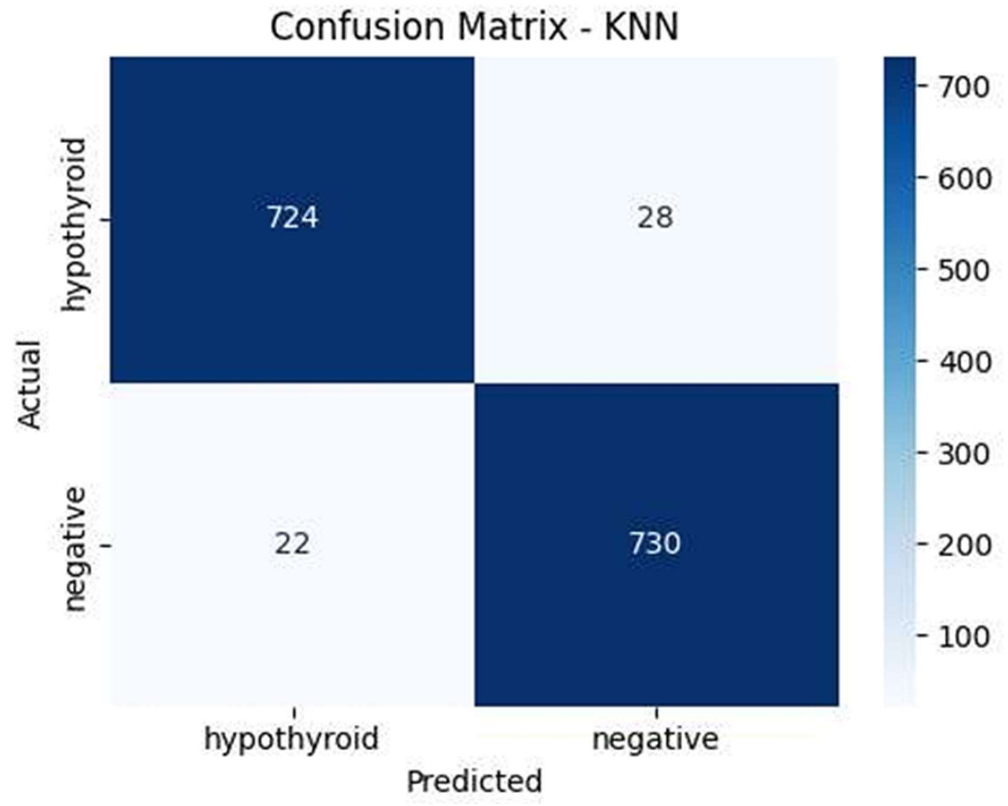
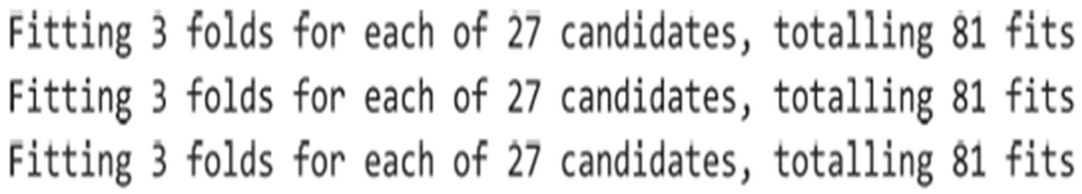
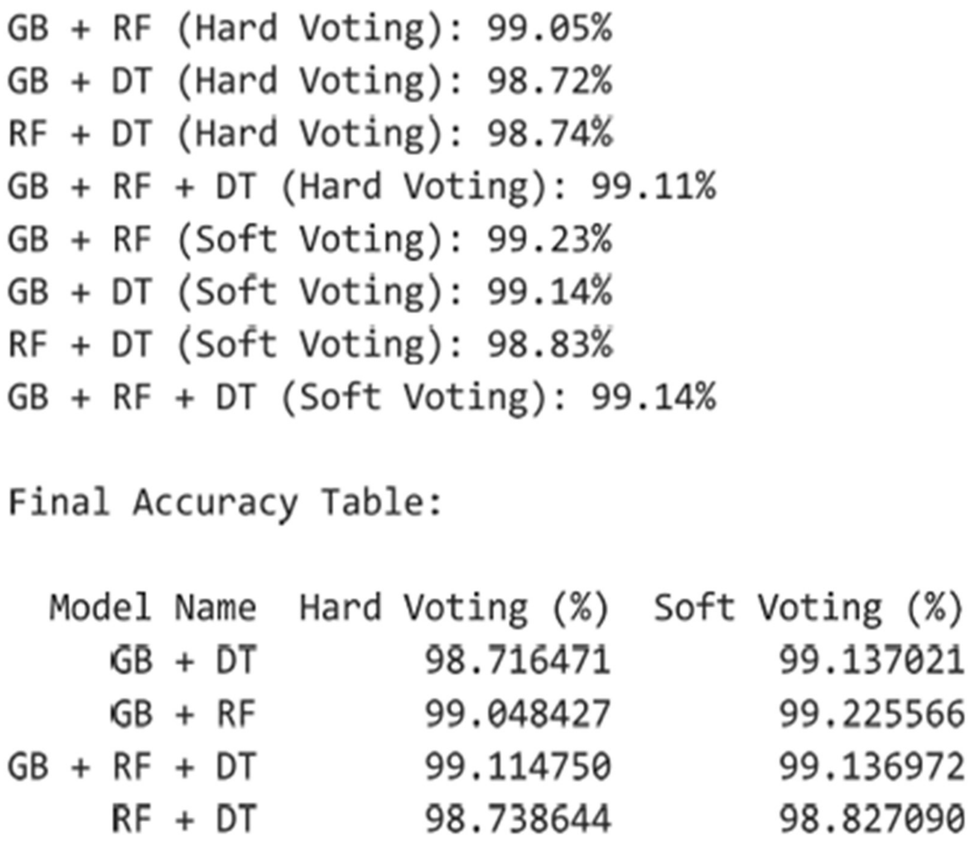


Fig 4.9 Model Evaluation Metrics of KNN



```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
Fitting 3 folds for each of 27 candidates, totalling 81 fits
Fitting 3 folds for each of 27 candidates, totalling 81 fits
```

Fig 4.10 Hyperparameter Tuning(GF + DT + RF)



```
GB + RF (Hard Voting): 99.05%
GB + DT (Hard Voting): 98.72%
RF + DT (Hard Voting): 98.74%
GB + RF + DT (Hard Voting): 99.11%
GB + RF (Soft Voting): 99.23%
GB + DT (Soft Voting): 99.14%
RF + DT (Soft Voting): 98.83%
GB + RF + DT (Soft Voting): 99.14%

Final Accuracy Table:

  Model Name  Hard Voting (%)  Soft Voting (%)
     GB + DT        98.716471        99.137021
     GB + RF        99.048427        99.225566
GB + RF + DT        99.114750        99.136972
     RF + DT        98.738644        98.827090
```

Fig 4.11 Ensemble Evaluation(Hard and Soft Voting)

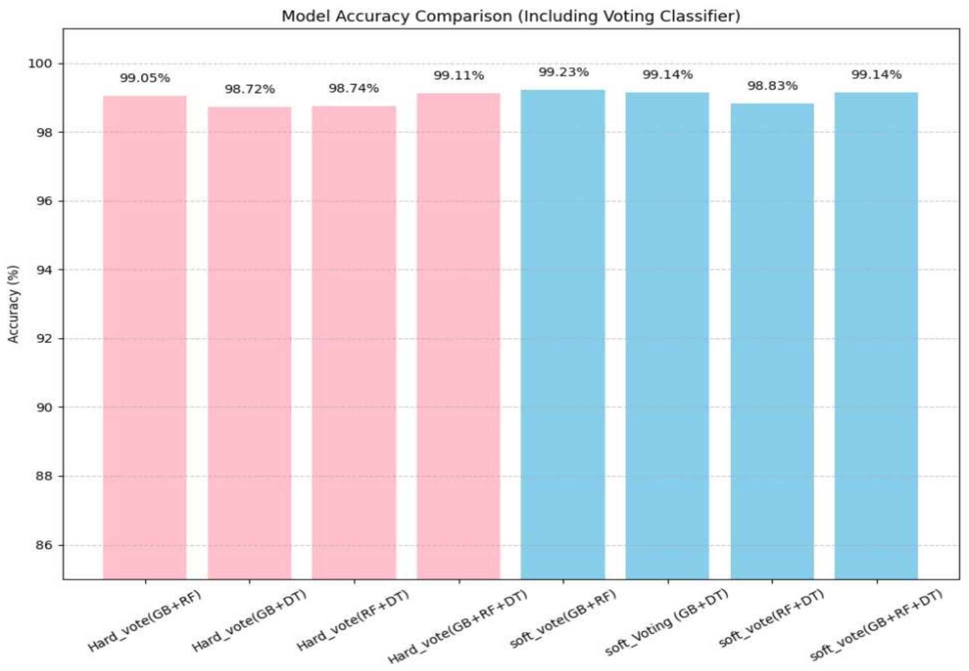


Fig 4.12 Model Accuracy Comparison(Voting)

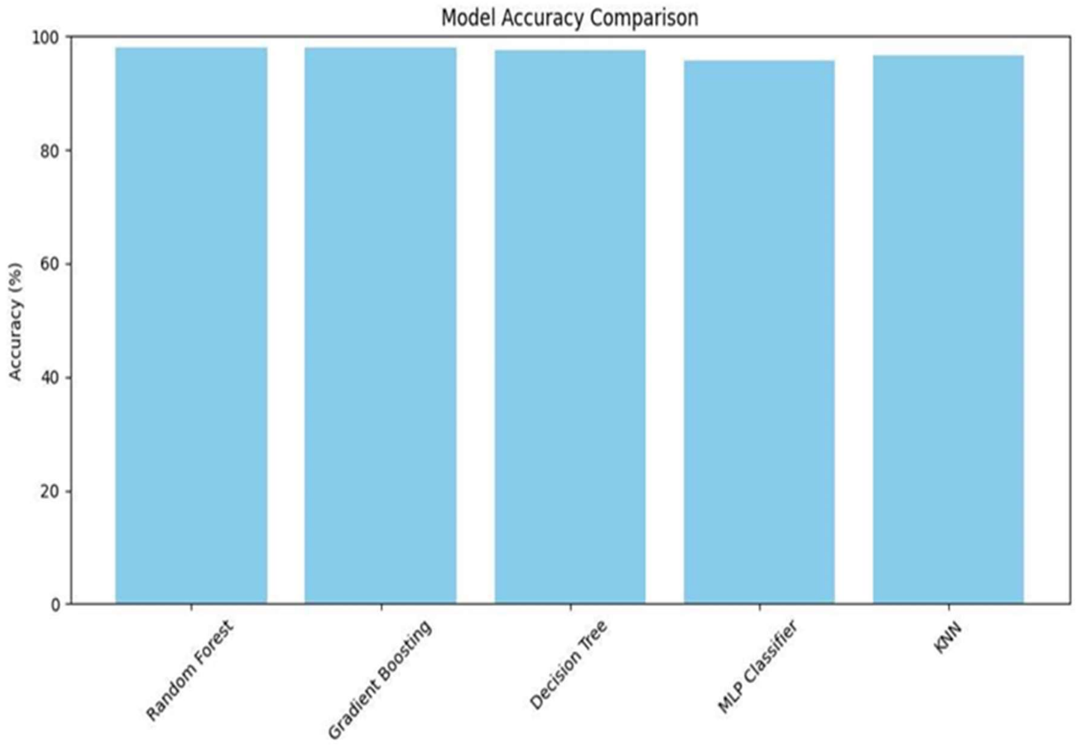


Fig 4.13 Model Accuracy Comparison

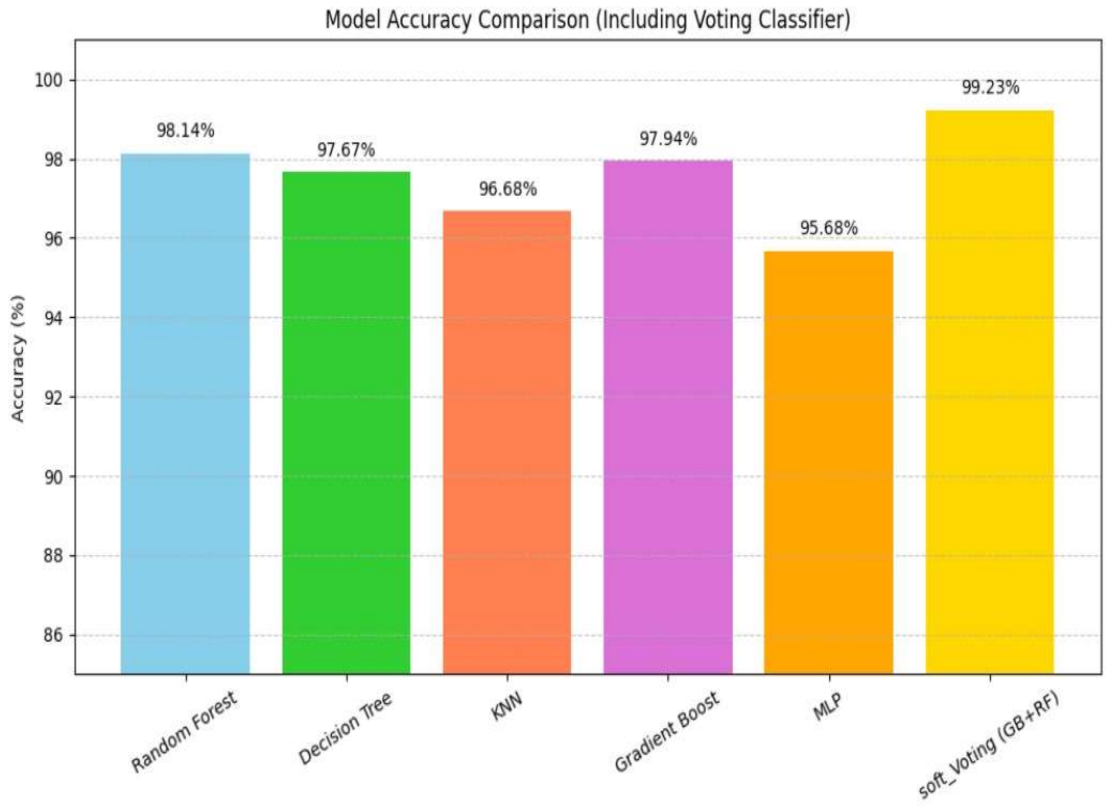
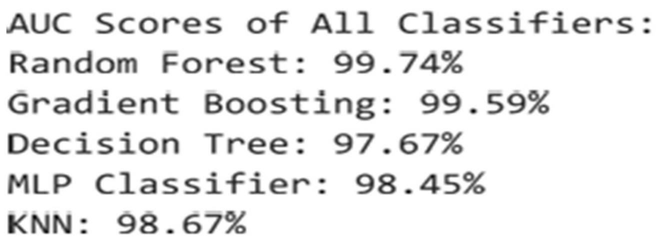


Fig 4.14 Model Comparison(Voting)



```
AUC Scores of All Classifiers:
Random Forest: 99.74%
Gradient Boosting: 99.59%
Decision Tree: 97.67%
MLP Classifier: 98.45%
KNN: 98.67%
```

Fig 4.15 AUC Scores

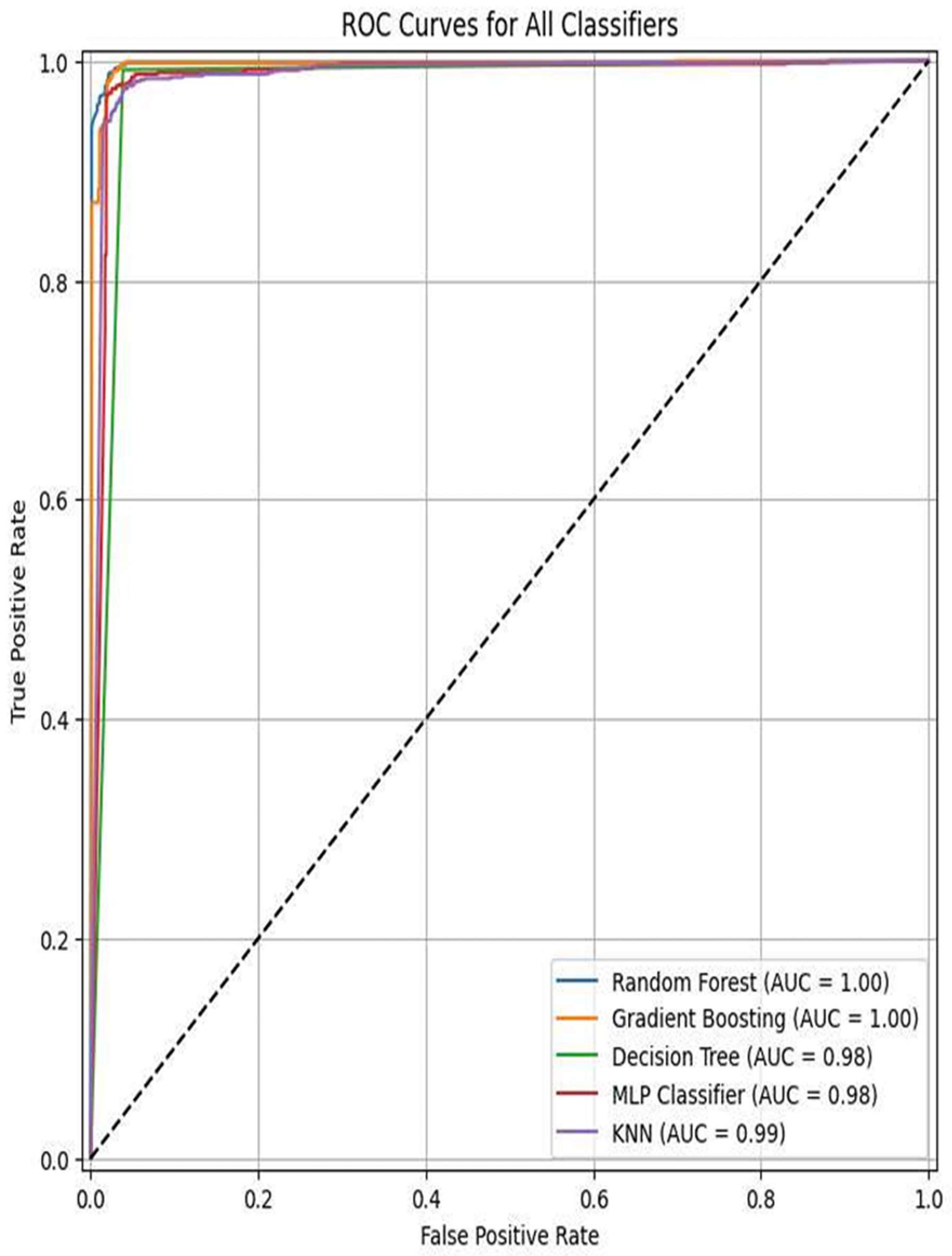


Fig 4.16 ROC Curve

CHAPTER 5
CONCLUSION AND FUTURE PLAN

5.1 CONCLUSION:

The study attempted to implement hybrid feature selection for developing an ensemble-based machine learning model that accurately predicts thyroid disorders. The system achieved better results compared to previous benchmarks in accuracy, sensitivity, and specificity due to class imbalance handling with SMOTE, feature selection through XGBoost with SelectKBest, and several other methods. By analyzing a broad set of classifiers such as Random Forest, Decision Tree, KNN, Gradient Boosting, and even Multilayer Perceptron in addition to ensemble methods (Hard and Soft Voting), we ensured consistent and reliable performance across numerous conditions where the proposed model excelled. The addition of ensemble techniques improved overall performance as the model was able to adapt to and learn more complex patterns resulting in higher accuracy thus outperforming single models. In summary, the model aims to enhance the accuracy and efficiency of predicting thyroid disorders enabling earlier detection, improved treatment, and enhanced decision-making by physicians, which will tremendously impact the field of medical diagnosis.

5.2 FUTURE PLAN:

- **Application of Deep Learning Models:** Investigate how deep learning methods, especially Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), may be applied to improve predictions by recognizing more complex patterns of data.
- **Increase in the Dataset and Realistic Validation:**To improve the accuracy and generalizability of the model, work with medical practitioners to acquire access to more diverse and realistic datasets across ages, genders, and diseases.
- **CDSS Development:** Develop a web platform or mobile tool that uses the model to allow real-time evaluation of thyroid disorders by patients and physicians.
- **XAI Approaches:**Higher transparency in model decision-making with the implementation of NAI model frameworks will promote greater acceptance and trust by healthcare professionals in the tool. Algorithms defined by emerging data can be incrementally updated to support observable changes in patient data attributes over time.

CHAPTER 6

REFERENCES

1. Chaubey, G., Bisen, D., Arjaria, S. et al. Thyroid Disease Prediction Using Machine Learning Approaches. Natl. Acad. Sci. Lett. 44, 233 238 (2021).
2. A.Tyagi, R. Mehra and A. Saxena, "Interactive Thyroid Disease Prediction System Using Machine Learning Technique," 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC), Solan, India, 2018, pp. 689-693, doi: 10.1109/PDGC.2018.8745910.
3. Z. J. Peya, M. K. N. Chumki and K. M. Zaman, "Predictive Analysis for Thyroid Diseases Diagnosis Using Machine Learning," 2021 International Conference on Science & Contemporary Technologies (ICSCT), Dhaka, Bangladesh, 2021, pp. 1-6, doi: 10.1109/ICSCT53883.2021.9642544.
4. S. M. Alhashmi et al., "Survival Analysis of Thyroid Cancer Patients Using Machine Learning Algorithms," in IEEE Access, vol. 12, pp. 61978-61990, 2024, doi: 10.1109/ACCESS.2024.3392275.
5. O. Jyoti, A. K. Paul, N. Nawar and M. A. I. Siddique, "Ensemble Learning for Improved Thyroid Disease Prediction: A Voting Classifier Approach," 2024 3rd International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE), Gazipur, Bangladesh, 2024, pp. 1-6, doi: 10.1109/ICAEEE62219.2024.10561815.
6. Riajuliislam, Md, Khandakar Zahidur Rahim, and Antara Mahmud. "Prediction of thyroid disease (hypothyroid) in early stage using feature selection and classification techniques." 2021 International conference on information and communication technology for sustainable development (ICICT4SD). IEEE, 2021.

CHAPTER 7
APPENDIX

- **Base Paper Title :** An ensemble Machine Learning based approach to predict Thyroid disease using hybrid feature selection.
- **Journal Name :** Bioinformatics and Analytics Journal.
- **Year of Publication :** 2024.
- **Publisher Name :** Elsevier B.V.
- **Base Paper Link :** https://doi.org/10.1016/j.bioana.2024.08.001.