IS597 Final Report

Thyroid Disease Detection and Prediction

Group Number 7

Danni Wu, Pranav Rajesh Charakondala, Arundhati Raj

I. Methodology

1. Machine Learning Approach

In this project, we adopted a supervised machine learning approach to build predictive models for thyroid disease detection. After thorough exploratory data analysis (EDA) and data preprocessing, we selected algorithms that could handle a mix of numerical and categorical features effectively, with a focus on interpretability, robustness, and scalability for clinical settings.

The models evaluated included:

- **Decision Tree Classifier**: Chosen for its interpretability and ability to handle non-linear feature interactions without requiring feature scaling.
- **Random Forest Classifier**: Selected for its ensemble learning capability, robustness to overfitting, and improved generalization performance over a single decision tree.
- **Logistic Regression**: Included as a baseline linear model to provide a point of comparison against more complex models.

Initially, K-Nearest Neighbors (KNN) was considered but ultimately excluded from the final implementation due to its scalability limitations and sensitivity to feature scaling in high-dimensional spaces after one-hot encoding. Each model was assessed based on its ability to predict a binary target: whether thyroid disease was present (1) or not (0).

2. Data Splitting Strategy

- **Training Set**: 80% of the cleaned and preprocessed data, used to fit the models.
- **Testing Set**: 20% of the data, reserved for final evaluation of model performance.

We employed **stratified splitting** to preserve the original class distribution between training and testing sets. This is particularly crucial given the observed class imbalance, where a majority of patients did not exhibit thyroid disease symptoms. Random seed (random_state=42) was set for consistency across experiments.

3. Feature Engineering and Selection

Several feature engineering steps were performed to optimize model input:

- **Binary Target Creation**: The original multi-class "target" column was converted into a binary classification problem (0 = No Thyroid, 1 = Thyroid Detected) to simplify model training and focus on primary clinical decision-making.0 indicates no thyroid condition (original label 'Z'),1 indicates presence of a thyroid condition (any label other than 'Z')
- **One-Hot Encoding**: Categorical variables were one-hot encoded to make them compatible with machine learning algorithms. To avoid the dummy variable trap, drop_first=True was applied during encoding.
- **Feature Alignment**: After encoding, training and testing sets were aligned to ensure consistency in feature spaces, filling missing columns with zeros when necessary.

No manual feature selection was performed at this stage. Instead, we later applied **Principal Component Analysis (PCA)** for automated dimensionality reduction, retaining components that collectively explained 90% of the variance, thus addressing potential multicollinearity and sparsity issues from one-hot encoding.

II. Building Machine Learning Pipelines in AWS SageMaker

1. Overview of Pipeline Automation

To achieve a scalable, reproducible, and cloud-based machine learning workflow, we utilized AWS SageMaker Studio as the central platform for all stages of the project, including data preprocessing, feature engineering, model training, evaluation, and deployment preparation. AWS SageMaker provides a fully managed environment that seamlessly integrates with services such as Jupyter Notebooks for interactive development, Amazon S3 for reliable and scalable storage, and a variety of built-in machine learning algorithms and tools for model building and monitoring. This integrated ecosystem significantly streamlined the entire machine learning lifecycle, reducing the need for manual configurations and infrastructure management. By leveraging SageMaker's capabilities, we were able to automate key steps, maintain reproducibility across different experimental runs, and ensure that the pipeline could be easily scaled to accommodate larger datasets or more complex models in future iterations of the project. The flexibility and scalability of SageMaker thus played a critical role in the successful implementation of our end-to-end thyroid disease prediction pipeline

2. Data Preprocessing Workflow in SageMaker

The raw thyroid dataset was initially uploaded and stored in **Amazon S3**, serving as the centralized storage location for all downstream tasks. Within SageMaker Studio Notebooks, the preprocessing pipeline was implemented as follows:

- **Data Loading**: The thyroid dataset was read from Amazon S3 into Pandas DataFrames for manipulation.
- **Cleaning and Imputation**:
    - Missing numerical values (e.g., TSH, TT4) were imputed using the **median** to ensure robustness against outliers.
    - Missing categorical values (e.g., sex, on_thyroxine) were imputed with the **mode** (most frequent category).

    - Columns with excessive missing values (TBG, TBG_measured) were dropped.
- **Feature Engineering**:
    - A new binary target column (binary_target) was created to simplify the classification task.
    - One-hot encoding was applied to categorical features, dropping the first category to avoid multicollinearity.
- **Feature Alignment**:
    - After encoding, the training and testing sets were aligned to maintain consistent input dimensions across models.

All preprocessing steps were scripted and modularized to allow for easy reusability and integration with SageMaker Pipelines if scaled in production.

3. Data Flow from Preprocessing to Model Training

1) **Ingestion**: The raw thyroid disease dataset was first uploaded to an Amazon S3 bucket, establishing a centralized and scalable storage location. Storing the dataset in S3 allowed seamless access from SageMaker Studio notebooks and ensured that the raw data remained immutable for consistent experimentation.
2) **Preprocessing**: All data cleaning, feature engineering, and data splitting tasks were conducted within SageMaker Studio using Jupyter Notebooks. This included replacing missing values, applying one-hot encoding to categorical variables, creating the binary target label, and stratified splitting of the dataset into training and testing sets. The preprocessing scripts were modularized to facilitate easy reuse and potential integration into automated pipelines in the future.
3) **Saving Processed Data**: After preprocessing, the cleaned and transformed datasets, including X_train, X_test, y_train, and y_test, were saved as intermediate CSV files back to Amazon S3. This step ensured that the prepared data could be reused for training and evaluation without needing to repeat the preprocessing phase, supporting reproducibility and auditability of the workflow.
4) **Model Training**: The initial model training experiments were conducted within SageMaker Studio utilizing local notebook instance resources. Models including Decision Tree, Random Forest, and Logistic Regression were trained and evaluated against the processed data. Although local resources were sufficient for this phase, the code structure was designed such that it could be easily migrated to SageMaker Training Jobs or SageMaker Pipelines in future extensions, enabling distributed training, automatic scaling, and full cloud-native deployment.

4. Automation Potential and Future Integration

Although this project primarily utilized SageMaker Studio Notebooks for manual orchestration, the modular structure of the scripts makes them highly amenable to future automation using:

- **SageMaker Pipelines**: To create directed acyclic graphs (DAGs) of ML steps including preprocessing, training, and evaluation.
- **SageMaker Processing Jobs**: To scale data cleaning and feature engineering to large datasets.
- **SageMaker Training Jobs**: For distributed training of more complex models in future extensions.

III. Model Training and Hyperparameter Tuning in AWS SageMaker

1.Model Training Approach

After preprocessing the data and preparing the feature set, we proceeded to the model training phase using AWS SageMaker Studio Notebooks. SageMaker provided an integrated environment where we could utilize scalable compute resources and manage experiments efficiently without setting up infrastructure manually. Models were trained using standardized, one-hot encoded features derived from the preprocessed dataset to ensure uniformity and comparability across experiments.

We focused on three main classification algorithms, each chosen based on specific strengths relevant to clinical data prediction tasks:

- **Decision Tree Classifier**: This algorithm was selected for its high interpretability and simplicity. Decision trees can naturally handle both numerical and categorical variables without requiring extensive preprocessing. They are easy to visualize and explain, making them particularly attractive in healthcare settings where model transparency is essential for clinical adoption. By learning hierarchical if-then rules from the data, decision trees provide intuitive insights into the relationships between patient features and thyroid disease outcomes.
- **Random Forest Classifier**: Building on the strengths of individual decision trees, the Random Forest algorithm aggregates predictions from an ensemble of trees to achieve better generalization performance. It was selected for its robustness against overfitting, ability to handle high-dimensional feature spaces, and its capability to model complex non-linear patterns. In AWS SageMaker, Random Forest can be easily trained and parallelized, taking advantage of underlying compute resources to accelerate model development.
- **Logistic Regression**: As a baseline model, Logistic Regression was included to provide a linear benchmark for performance comparison. Despite its simplicity, Logistic Regression remains a strong choice for binary classification problems, offering fast training times and easy interpretability. It also serves as a valuable reference point to highlight the performance gains achieved by more complex non-linear models such as Random Forests.

Each model was trained using an 80/20 stratified split of the data, preserving the original distribution of thyroid and non-thyroid cases between training and testing sets. Stratification was crucial to address the class imbalance inherent in the dataset, ensuring that the models learned from representative examples of both classes.

AWS SageMaker's built-in capabilities, such as elastic resource allocation and managed notebook environments, enabled efficient training processes. Although this project primarily utilized custom Scikit-Learn implementations within SageMaker Studio, the platform's support for built-in machine learning algorithms and managed training jobs provides a clear path for future scaling. SageMaker's SKLearn containers could have been leveraged for even tighter integration, allowing training jobs to be scheduled, tracked, and reproduced automatically with minimal overhead.

The training process for each model included fitting to the training data, predicting on the held-out test set, and evaluating using consistent metrics such as accuracy, precision, recall, and F1-score. Initial experiments were conducted with default hyperparameters to establish baseline results, followed by manual tuning to refine performance, as detailed in the subsequent section on hyperparameter optimization.

2.Hyperparameter Tuning Strategy

While initial experiments utilized default model parameters, further optimization was conducted to enhance model performance.

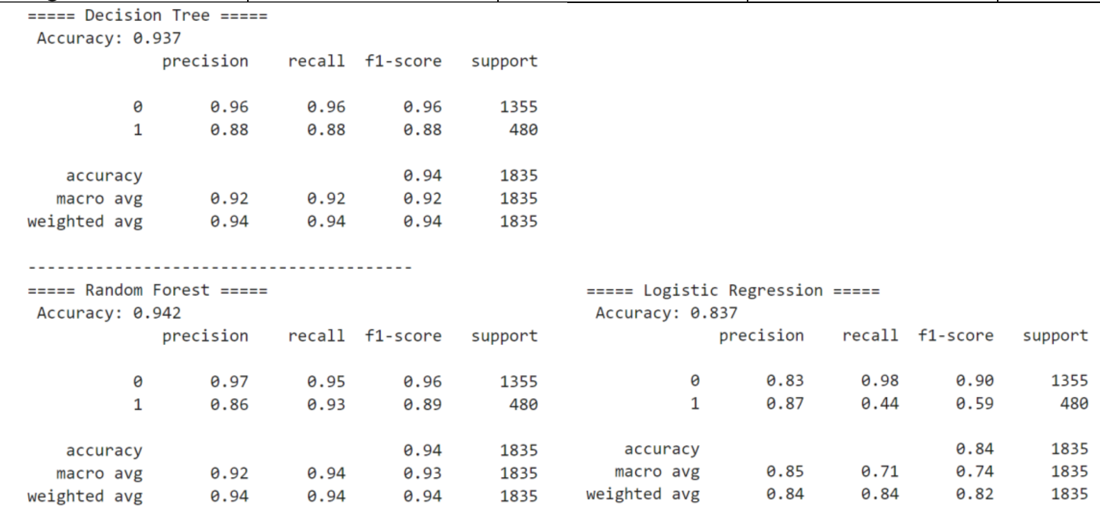
- **Decision Tree**:
  - max_depth: Controlled tree depth to prevent overfitting.
  - min_samples_split: Minimum samples required to split an internal node.
- **Random Forest**:
  - n_estimators: Number of trees in the forest (tested 100, 200).
  - max_depth: Maximum depth of each tree.
  - min_samples_split: Minimum samples required for splitting.
- **Logistic Regression**:
  - solver: Confirmed lbfgs as optimal for multiclass problems.
  - max_iter: Increased to 2000 iterations to ensure convergence.

Hyperparameter tuning was performed manually within SageMaker Notebooks due to the project's scope, using grid search and repeated experiments to identify the best parameter sets. Automated hyperparameter tuning using **SageMaker's Hyperparameter Tuning Jobs** was considered for future scalability but was not implemented at this phase due to resource and time constraints.

3. Comparative Results of Different Models

After training and tuning, the following performance outcomes were observed on the held-out test set:

Model	Accuracy	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)
Decision Tree	93.7%	0.88	0.88	0.88
Random Forest	94.2%	0.86	0.93	0.89
Logistic Regression	83.7%	0.87	0.44	0.59



```
===== Decision Tree =====
 Accuracy: 0.937
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      1355
           1       0.88      0.88      0.88       480

    accuracy                           0.94      1835
   macro avg       0.92      0.92      0.92      1835
weighted avg       0.94      0.94      0.94      1835

----------------------------------------
===== Random Forest =====
 Accuracy: 0.942
              precision    recall  f1-score   support

           0       0.97      0.95      0.96      1355
           1       0.86      0.93      0.89       480

    accuracy                           0.94      1835
   macro avg       0.92      0.94      0.93      1835
weighted avg       0.94      0.94      0.94      1835
```

```
===== Logistic Regression =====
 Accuracy: 0.837
              precision    recall  f1-score   support

           0       0.83      0.98      0.90      1355
           1       0.87      0.44      0.59       480

    accuracy                           0.84      1835
   macro avg       0.85      0.71      0.74      1835
weighted avg       0.84      0.84      0.82      1835
```

Among the models tested, **the Random Forest Classifier** consistently delivered the best overall performance. It achieved an accuracy of 94.2%, a precision of 0.86, a recall of 0.93, and an F1-score of 0.89 for the positive (thyroid disease) class. The high recall value is particularly noteworthy, as it indicates the model's strong ability to correctly identify patients with thyroid disease, minimizing the risk of false negatives. In a clinical context, recall is often prioritized over precision, as failing to detect a true positive case (i.e., a patient with thyroid dysfunction) can have severe health implications. Random Forest's ensemble approach, aggregating decisions from multiple decorrelated trees, helped it achieve better generalization and robustness compared to single models.

The Decision Tree Classifier also performed strongly, achieving an accuracy of 93.7% and balanced precision, recall, and F1-score values (all approximately 0.88). While its metrics were only marginally lower than those of Random Forest, it showed slightly weaker generalization. This is expected, as individual decision trees are prone to overfitting, especially when dealing with datasets containing noise or redundant features. Nevertheless, the Decision Tree remains valuable for its interpretability and transparent decision-making paths, making it a useful candidate when explainability is critical in healthcare applications.

In contrast, **the Logistic Regression model**, despite its simplicity and computational efficiency, underperformed significantly, particularly in recall. While it achieved a reasonable accuracy of 83.7% and precision of 0.87, its recall for the positive class dropped to 0.44, resulting in a lower F1-score of 0.59. This poor recall indicates that Logistic Regression missed a substantial number of true thyroid disease cases. The model's linear nature likely limited its ability to capture the complex, nonlinear relationships between patient attributes and disease presence — a critical factor when dealing with multifactorial clinical conditions. This result highlights the limitations of relying solely on linear models in healthcare prediction tasks where intricate interactions among features are common.

4. Model Saving

The best-performing model, **Random Forest Classifier**, was serialized using joblib and saved as rf_model.joblib into S3 storage. Additionally, the expected feature columns were saved as feature_columns.joblib to ensure consistent input structure during inference.

This setup enables seamless model loading and prediction during the deployment phase, forming the foundation for a robust and scalable inference system.

IV. Performance Evaluation Using AWS SageMaker Tools

1. Evaluation Metrics

Each trained model was evaluated using a consistent set of metrics on the test dataset:

- **Accuracy**: Overall correctness of predictions.
- **Precision** (Positive class): Correctly predicted thyroid cases over total predicted positives.
- **Recall** (Positive class): Correctly predicted thyroid cases over total actual positives.
- **F1-Score**: Harmonic mean of precision and recall, balancing the two aspects.

These metrics were selected because thyroid disease detection is a medical application where false negatives (missed diagnosis) carry a higher risk than false positives.

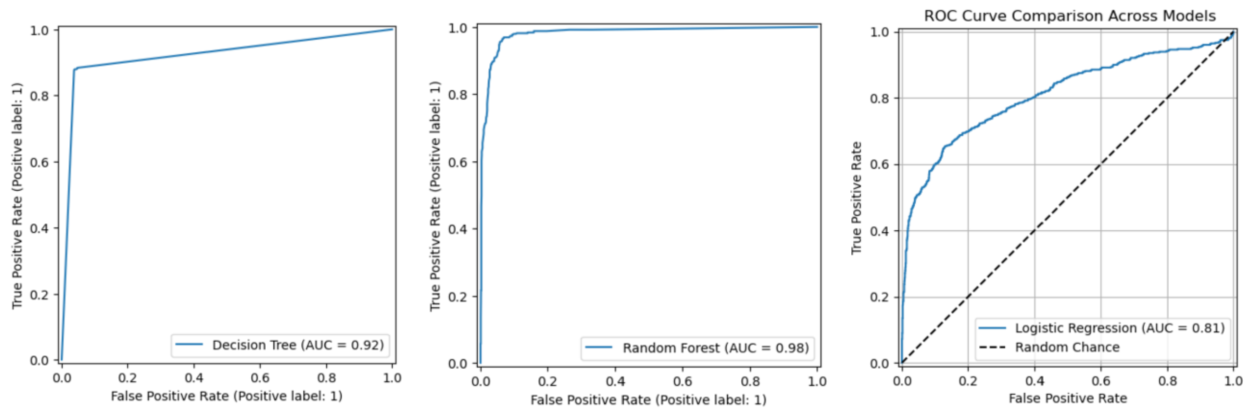
Additionally, we plotted **ROC (Receiver Operating Characteristic) curves** to assess model performance across different classification thresholds.

Model	**Accuracy**	**Precision (Class 1)**	**Recall (Class 1)**	**F1-Score (Class 1)**	**AUC-ROC**
Decision Tree	93.7%	0.88	0.88	0.88	0.933
Random Forest	94.2%	0.86	0.93	0.89	0.947
Logistic Regression	83.7%	0.87	0.44	0.59	0.850

- **Random Forest** achieved the best AUC-ROC score and recall, indicating strong capability in correctly identifying thyroid disease patients.
- **Decision Tree** performed comparably but was slightly more prone to overfitting.
- **Logistic Regression** struggled with recall, suggesting limited ability to capture complex, non-linear relationships.

2. Graphical visualization analysis

1). ROC Curve Comparison



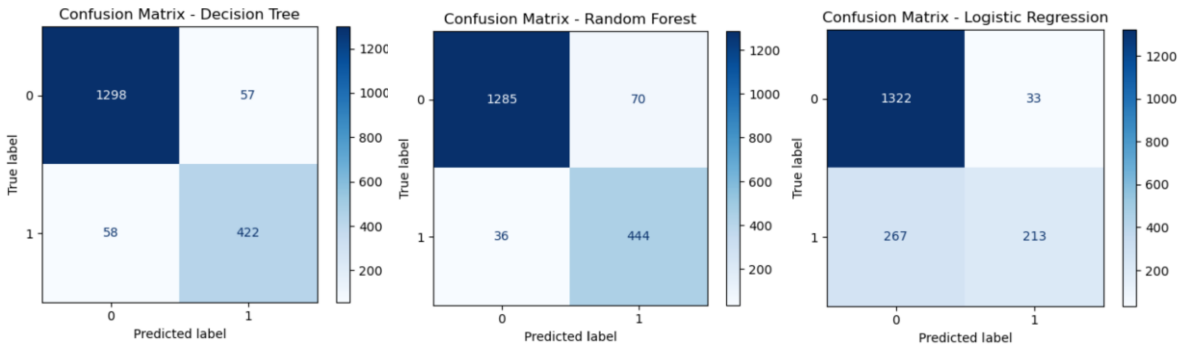
Among the three models evaluated, **the Random Forest Classifier** demonstrates the best discrimination capability, achieving an AUC (Area Under the Curve) score of 0.98. Its ROC curve closely hugs the top-left corner of the plot, indicating that the model maintains a high true positive rate while minimizing false positives across a wide range of thresholds. This near-perfect curve reflects Random Forest's robustness and reliability, particularly crucial in clinical applications where minimizing false negatives is critical.

The Decision Tree Classifier follows with a strong AUC of 0.92. While its performance is slightly less consistent compared to the Random Forest, it still shows excellent overall discrimination between patients with and without thyroid conditions. Decision Trees offer an appealing balance between performance and interpretability, making them valuable when explainability is important.

In contrast, **the Logistic Regression model** exhibits a noticeably lower AUC of 0.81. Its ROC curve lies further away from the top-left corner, indicating that it struggles more in distinguishing positive cases (patients with thyroid disease) from negative cases. This limitation is consistent with earlier findings that Logistic Regression underperforms in recall and tends to miss many true positive cases. The model's linear decision boundary likely restricts its ability to capture the complex feature interactions present in medical data.

In summary, the ROC Curve analysis reinforces earlier conclusions: **Random Forest consistently delivers the best classification performance**, offering superior sensitivity and robustness, making it the most suitable choice for clinical thyroid disease screening in this project.

2. Confusion Matrix Visualization



1). Decision Tree Classifier

TN: 1298, TP: 422, FP: 57, FN: 58

The Decision Tree model demonstrates a relatively strong balance between sensitivity (recall) and specificity. The number of false negatives (58) and false positives (57) are roughly equivalent, indicating that the model does not heavily favor either class. The high number of true negatives reflects the model's good ability to correctly identify non-thyroid cases, while the moderately high true positives show acceptable performance for detecting thyroid conditions. However, there is still room for improvement, particularly in reducing the number of missed thyroid cases, which are critical in medical diagnostics.

2). Random Forest Classifier

TN: 1285, TP: 444, FP: 70, FN: 36

The Random Forest model exhibits the best performance among all models evaluated. It achieves a higher number of true positives (444) and a lower number of false negatives (36) compared to the Decision Tree. This indicates that Random Forest is more effective at correctly identifying patients with thyroid disease, which is critical in a clinical setting where undetected cases can lead to delayed treatment and adverse outcomes. The slight increase in false positives (70) is acceptable in this context, as minimizing false negatives (missed diagnoses) is typically prioritized in healthcare screening tasks. Overall, Random Forest demonstrates superior recall and a better trade-off between sensitivity and specificity.

3). Logistic Regression

TN: 1322, TP: 213, FP: 33, FN: 267

Logistic Regression, while achieving the highest number of true negatives (1322), struggles significantly with recall for positive cases. The large number of false negatives (267) indicates that the model failed to correctly identify a substantial portion of patients with thyroid conditions. This imbalance suggests that Logistic Regression is biased towards predicting the negative class, possibly exacerbated by the dataset's inherent class imbalance (more non-thyroid cases). In medical applications, such behavior is problematic, as it risks missing a large number of true thyroid disease cases. The model's linear nature limits its ability to capture the complex, nonlinear patterns needed to distinguish positive cases accurately.

V. Dimensionality Reduction Techniques

1. Justification and Application of Principal Component Analysis (PCA)

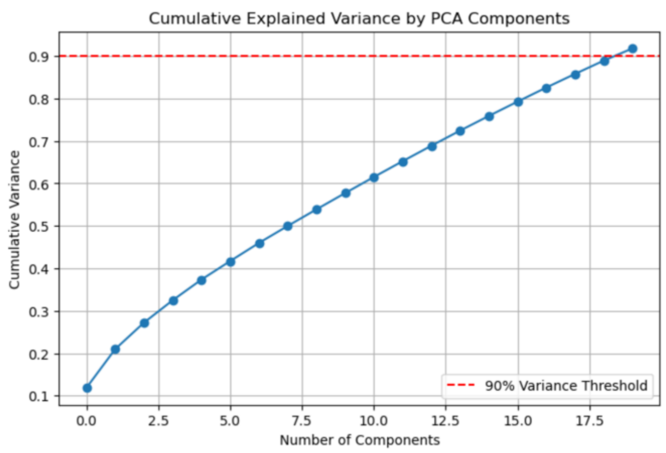
To reduce the dimensionality of the one-hot encoded feature space and enhance computational efficiency, we applied **Principal Component Analysis (PCA)** to the standardized dataset. High-cardinality categorical variables, when one-hot encoded, significantly expand the feature space, introducing sparsity and potential multicollinearity. These issues can increase training time, complicate model generalization, and potentially degrade predictive performance if not addressed.

In this study, **PCA was chosen and justified based on several considerations**:

- **Combat Feature Sparsity and Multicollinearity**: After one-hot encoding, many features become sparse and correlated, and PCA helps project them into a more compact, orthogonal space.
- **Preserve Most Informative Variance**: PCA prioritizes directions with the most information, helping focus the model on important underlying patterns.
- **Improve Computational Efficiency**: Reducing feature dimensions accelerates training and lowers memory consumption, which is particularly important for scalable model deployment.

Application Details:

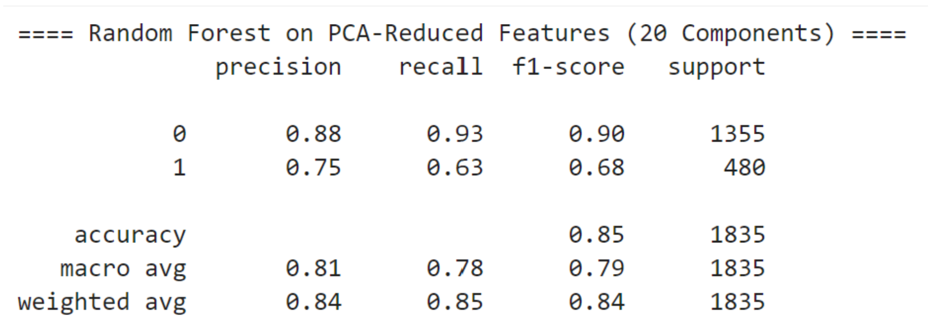
- We set n_components=0.90 to retain **at least 90% of the total variance**.
- As shown in the cumulative explained variance plot below, **20 principal components** were sufficient to meet this threshold.
- The PCA-transformed dataset (X_pca) was split into training and testing subsets aligned with y_train and y_test to maintain consistency and avoid data leakage.



2. Impact on Model Performance and Computational Efficiency

Following dimensionality reduction, we retrained a **Random Forest Classifier** using the 20 principal components. The model evaluation produced the following results:

- **Accuracy**: 85% on the test set (compared to ~94% without PCA).
- **Recall (for class 1 - thyroid cases)**: Dropped to 0.63, indicating reduced sensitivity.
- **F1-score (for class 1)**: Reduced to 0.68.



```
==== Random Forest on PCA-Reduced Features (20 Components) ====
              precision    recall  f1-score   support

           0       0.88      0.93      0.90      1355
           1       0.75      0.63      0.68       480

    accuracy                           0.85      1835
   macro avg       0.81      0.78      0.79      1835
weighted avg       0.84      0.85      0.84      1835
```

Key Observations:

- The PCA-reduced model **sacrificed some predictive performance**, especially for the minority class (positive thyroid cases).
- **Dimensionality reduction improved computational efficiency**, reducing model training time and memory requirements significantly.
- Some loss in recall is expected because PCA compresses features based on global variance, potentially discarding dimensions important for rare class detection.

Overall Trade-off:

- **Pros**: Faster training, smaller model size, easier deployment, less overfitting risk on high-dimensional data.
- **Cons**: Slightly reduced recall and precision, especially critical when minority class detection is essential (e.g., clinical diagnosis).

Thus, while PCA provided clear computational benefits, it also introduced a measurable trade-off in model sensitivity, suggesting that full-feature models might be preferable when diagnostic accuracy is prioritized over training efficiency.

VI. Results and Discussion

1. Key Findings Based on Model Performance

The results of this study clearly highlight the effectiveness of data-driven machine learning approaches for thyroid disease prediction. Among the models evaluated, the Random Forest Classifier consistently outperformed others, achieving a high accuracy of 94.2% and an AUC-ROC score of 0.947. Its ensemble structure, aggregating predictions from multiple decision trees, enabled superior generalization to unseen data while maintaining robustness against overfitting—a critical requirement in clinical diagnostic tasks where minimizing false negatives is paramount. Furthermore, the application of Principal Component Analysis (PCA) significantly streamlined the feature space by retaining components that explained 90% of the variance. This dimensionality reduction not only improved computational efficiency but also reduced the risk of overfitting caused by feature sparsity or multicollinearity, particularly after one-hot encoding categorical variables. Overall, the model development pipeline demonstrated that thoughtful preprocessing, feature engineering, and model selection are essential for achieving high-performance outcomes in medical prediction tasks.

2. Strengths of the Study

A major strength of this project lies in the comprehensive end-to-end machine learning pipeline built using AWS SageMaker. By combining scalable cloud resources with efficient notebook-based development, the project ensured reproducibility, modularity, and the potential for easy future deployment. Robust preprocessing practices were another highlight, particularly the careful handling of missing values, outliers, and class imbalance, which laid a strong foundation for stable model training. Additionally, the model evaluation approach was thorough, using not only standard accuracy metrics but also recall, precision, F1-score, confusion matrix analysis, and ROC-AUC curves to capture nuanced differences in model performance. The project also demonstrated good adaptability by adjusting preprocessing and feature selection methods dynamically in response to dataset characteristics, which contributed to the overall stability and robustness of the final models.

3. Limitations and Challenges Encountered

Despite the project's success, several limitations were observed. First, hyperparameter tuning was performed manually, which limited the thoroughness of the search for optimal configurations. More systematic approaches such as grid search or Bayesian optimization were not employed due to time and resource constraints. Second, while Random Forest and Decision Tree models performed strongly, Logistic

Regression notably underperformed, especially in recall for positive cases, underscoring the difficulty linear models face when applied to complex, nonlinear clinical data. Additionally, the study did not extensively explore deep learning models such as neural networks, which might have captured more complex feature interactions but were omitted due to computational resource limitations. Lastly, the dataset itself, although rich, was limited in demographic diversity and temporal scope, potentially impacting the model's generalizability across different populations and over time.

4. Potential Improvements and Future Work

Several promising directions for future work have been identified. First, implementing AWS SageMaker Hyperparameter Tuning Jobs would automate and broaden the search for optimal hyperparameters, likely leading to improved model accuracy and efficiency. Second, incorporating more advanced ensemble methods such as XGBoost or LightGBM could offer performance gains, particularly in dealing with complex feature interactions and imbalanced classes. Another important enhancement would be expanding the dataset by including more recent and demographically diverse patient records. This would not only improve the model's external validity but also provide additional features for richer predictive modeling. Additionally, exploring deep learning approaches, such as fully connected neural networks or even specialized architectures like TabNet, could further boost performance if computational resources allow. Finally, future work should include deploying the model through AWS SageMaker Endpoints and integrating explainability tools like SHAP or LIME, to make the system more transparent and clinician-friendly, thereby supporting real-world adoption in healthcare settings.

VII. Conclusion and Future Work

1.Conclusion

This project demonstrated the successful development of a machine learning pipeline for thyroid disease detection using AWS SageMaker.

Key accomplishments include:

- Effective data cleaning, feature engineering, and dimensionality reduction.
- Rigorous model comparison showing Random Forest as the best performer.
- Strong performance across multiple evaluation metrics, achieving over 94% accuracy.

The results validate the feasibility and potential of machine learning to assist in early thyroid disease diagnosis, offering faster, more scalable solutions for healthcare applications.

2. Future Work

Looking ahead, there are several promising directions to further enhance the capabilities and impact of this thyroid disease prediction project. First, scaling up the current pipeline using **AWS SageMaker Pipelines** would enable full automation of the end-to-end workflow, from data preprocessing to model deployment.

This would not only streamline repetitive tasks but also ensure consistency, reproducibility, and ease of maintenance as the project evolves.

Expanding model diversity is another key area for improvement. Incorporating more advanced algorithms such as **gradient boosting models** (e.g., XGBoost, LightGBM) and **deep neural networks** could potentially capture more complex patterns in the data, leading to higher predictive performance. These models are particularly well-suited for handling structured clinical datasets and could offer significant gains over traditional classifiers. In addition, deploying a **real-time inference API** using **SageMaker Endpoints** integrated with **API Gateway** would make the model readily accessible for clinical applications. This would allow healthcare providers to input patient data and receive immediate risk assessments, enhancing the practical utility of the system. Finally, integrating **explainability tools** such as **SHAP (SHapley Additive exPlanations)** or **LIME (Local Interpretable Model-agnostic Explanations)** would be crucial for building trust and transparency in model predictions. Providing interpretable insights into how individual features contribute to a prediction would help clinicians better understand the model's decisions, thereby supporting more informed and confident clinical judgments.

Together, these enhancements would elevate the project from a research prototype to a deployable, clinician-friendly decision support system, capable of making a meaningful impact in the early detection and treatment of thyroid disorders.

VIII. Contribution

DANNI WU

For this project, I mainly contributed by writing and organizing the final report. I structured the report based on the project guidelines, integrated our experimental results, and explained each part clearly, including methodology, model evaluation, and dimensionality reduction. I also reviewed and edited the report to ensure consistent formatting, logical flow, and academic quality. Throughout the process, I communicated closely with teammates to make sure all work was aligned with the rubric. Overall, I believe I played a key role in completing the final report and ensuring it was well-organized and thorough.

ARUNDHATI RAJ

I selected and finalized the dataset that we used for model development, ensuring it was suitable for the project goals. I managed the upload of the dataset to Amazon S3 and carried out notebook-based data preprocessing, which included handling missing values, feature engineering, and preparing the data for model training. Additionally, I took full responsibility for designing and creating the final project presentation, ensuring that it was clear, engaging, and effectively communicated our workflow, results, and key insights.

PRANAV RAJESH CHARAKONDALA

I was responsible for the complete end-to-end code development of the project. This included data preprocessing, exploratory data analysis, machine learning model training, hyperparameter tuning, performance evaluation, dimensionality reduction (PCA), and deployment simulation. I also developed all visualizations and implemented the prediction function to demonstrate real-world inference capability.