Prediction of Asthma using Machine Learning Algorithms

Submitted By

Student Name	**Student ID**
Md. Sulyman Islam Sifat	211-15-4004
Mohammed Nazmul Hoque Shawon	211-15-3996

MINI LAB PROJECT REPORT

This Report Presented in Partial Fulfillment of the course **CSE322: Data Mining & Machine Learning Lab in the Computer Science and Engineering Department**

DAFFODIL INTERNATIONAL UNIVERSITY
Dhaka, Bangladesh

December 11, 2024

DECLARATION

We hereby declare that this lab project has been done by us under the supervision of **Mr. Md. Abdullah Al Kafi**, **Lecturer**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere as lab projects.

Submitted To:

Mr. Md. Abdullah Al Kafi
Lecturer
Department of Computer Science and Engineering
Daffodil International University

Submitted by

Nazmul Hoque **Mohammed Nazmul Hoque Shawon** ID: 211-15-3996 Dept. of CSE, DIU	Sulyman sifat **Md. Sulyman Islam Sifat** ID: 211-15-4004 Dept. of CSE, DIU

COURSE & PROGRAM OUTCOME

The following course have course outcomes as following:

Table 1: Course Outcome Statements

CO's	Statements
CO1	Able to grasp the basic Data Mining Principles
CO2	Able to identify appropriate data mining algorithms to solve real world problems
CO3	Able to compare and evaluate different data mining techniques like classification, prediction, clustering and association rule mining
CO4	Able to apply data mining knowledge in problem solving

Table 2: Mapping of CO, PO, Blooms, KP and CEP

CO	PO	Blooms	KP	CEP
CO1	PO1	C1, C2	KP3	EP1, EP3
CO2	PO2	C2	KP3	EP1, EP3
CO3	PO3	C4, A1	KP3	EP1, EP2
CO4	PO3	C3, C6, A3, P3	KP4	EP1, EP3

Table of Contents

Chapter 1

Introduction

1.1 Introduction

Asthma is a chronic inflammatory airway disease that has affected over 300 million people worldwide. It is characterized by wheezing, shortness of breath, chest tightness, and coughing. Asthma can be controlled with medication, but it can be a debilitating disease if not properly managed. The diagnosis and management of asthma can be challenging. Asthma symptoms can vary over time and can be triggered by a variety of factors, such as allergens, exercise, and cold air. This can make it difficult to distinguish asthma from other respiratory conditions. Additionally, many people with asthma are underdiagnosed or misdiagnosed. Machine learning has the potential to improve the diagnosis and management of asthma. Machine learning algorithms can be trained on data from patients with asthma to identify patterns that are associated with the disease. These patterns can then be used to develop predictive models that can identify patients at high risk of asthma or who are likely to experience an asthma exacerbation. This research endeavors to design and assess machine learning models for asthma prediction. We gathered data from asthma patients and employed various machine-learning algorithms to train predictive models. The performance of these models was evaluated using various metrics, including accuracy, sensitivity, specificity, and F1 score. The results of our study show that machine learning algorithms can be used to predict asthma with high accuracy. We also identified the most important features for predicting asthma, which can be used to develop more accurate and interpretable models. We believe that our findings have the potential to improve the diagnosis and management of asthma. By identifying patients at high risk of asthma or who are likely to experience an asthma exacerbation, clinicians can provide more targeted and timely interventions.

1.2 Motivation

Asthma remains a significant global health challenge, affecting millions and often leading to a reduced quality of life due to misdiagnosis or delayed interventions. Traditional diagnostic methods frequently fall short in identifying subtle patterns associated with asthma exacerbations, leaving many patients vulnerable to severe outcomes. The integration of machine learning into asthma diagnosis and management offers a promising solution. By leveraging the power of data-driven insights, we can enhance early detection, tailor treatments, and ultimately reduce the burden of this debilitating disease. This motivates us to explore innovative approaches that combine healthcare and technology to improve patient outcomes and transform clinical practices.

1.3 Objectives

1. **Develop Predictive Models**: To design and implement machine learning algorithms capable of accurately predicting asthma diagnosis and exacerbations.
2. **Improve Diagnosis and Management**: To provide clinicians with tools that facilitate early and precise asthma diagnosis, as well as timely interventions for high-risk patients.
3. **Enhance Patient Outcomes**: To reduce the rates of misdiagnosis and improve the quality of life for asthma patients by leveraging data-driven healthcare solutions.

1.4 Feasibility Study

1. **Similar Studies**: Research has shown machine learning algorithms like Random Forests, SVMs, and Gradient Boosting can predict asthma with high accuracy, often exceeding 85%. Key features include respiratory patterns, allergen exposure, and environmental factors.
2. **Case Studies**: Wearable device data has been used to predict asthma exacerbations in real-time, enabling timely interventions. These studies highlight the potential of personalized and dynamic predictive models.
3. **Methodological Contributions**: Existing works contribute techniques like feature selection (e.g., RFE, PCA), data imputation (e.g., k-NN), and hybrid models (e.g., neural networks with decision trees), improving both accuracy and interpretability.

1.5 Project Outcome

1. **Accurate Prediction**: Development of machine learning models capable of accurately predicting asthma diagnosis and exacerbations.
2. **Key Insights**: Identification of critical features influencing asthma, enhancing model interpretability and clinical relevance.
3. **Improved Diagnosis**: Tools to aid clinicians in early and precise asthma detection, reducing misdiagnosis rates.
4. **Better Management**: Predictive tools for identifying high-risk patients, enabling timely and personalized interventions.
5. **Enhanced Patient Outcomes**: Improved quality of life for asthma patients through data-driven healthcare solutions.

Chapter 2

Proposed Methodology

2.1 Requirement Analysis & Design Specification

2.1.1 Overview

The **requirement analysis** phase focuses on defining the functional and non-functional requirements of the project. The functional requirements include the development of machine learning models for accurate asthma prediction, implementing data preprocessing, feature selection, and model training, as well as providing clear visualizations for predictions and feature importance. Non-functional requirements emphasize achieving high performance in terms of accuracy, sensitivity, and specificity, while ensuring the models are scalable, interpretable, and compliant with data security and privacy regulations. In the **design specification**, the system is structured in a modular architecture, with distinct phases for data preprocessing, model training, evaluation, and deployment. The database is designed to handle structured datasets, which include patient demographics, medical history, and environmental factors. Additionally, a user-friendly interface is specified for clinicians to input patient data and easily view asthma predictions. This approach ensures the creation of an efficient and reliable asthma prediction system that is both clinically applicable and secure.

2.1.2 Proposed Methodology

1. Data Collection and Understanding

- **Dataset Source**: A survey dataset containing information about asthma patients was utilized. The data included demographic details, environmental factors, and symptoms associated with asthma.
- **Data Exploration**: The dataset was examined to identify key features, check for missing values, and understand the distribution of variables relevant to asthma prediction.

2. Data Preprocessing

- **Handling Missing Data**: Rows with missing values were removed to ensure data integrity.
- **Feature Encoding**: Categorical features (e.g., Gender, Living Area) were encoded into numerical values using LabelEncoder, making them suitable for machine learning models.
- **Feature Selection**: A correlation analysis was performed to identify relationships between features, ensuring that only relevant predictors were included.
- **Data Splitting**: The dataset was divided into training (75%) and testing (25%) sets using the train_test_split function.

3. Model Implementation

- Multiple machine learning algorithms were applied to predict asthma:
  - **Logistic Regression**
  - **Decision Tree Classifier**

    - **Random Forest Classifier**
    - **Support Vector Machine (SVM)**
    - **Naive Bayes Classifier**
- Each model was trained using the training dataset and evaluated using the testing dataset.

4. Evaluation Metrics

- The models were assessed using the following metrics:
    - **Accuracy**: Measures the overall correctness of predictions.
    - **Confusion Matrix**: Provides a detailed breakdown of true positives, true negatives, false positives, and false negatives.
    - **Classification Report**: Includes precision, recall, and F1 score for each class.
- Visualizations, such as heatmaps of confusion matrices, were created to enhance interpretability.

5. Model Selection and Deployment

- **Best Model**: The Random Forest Classifier demonstrated the highest accuracy and balanced performance across metrics, making it the optimal choice for deployment.
- **Model Saving**: The Random Forest model was saved using joblib for future use.
- **User Interface**: A command-line interface was designed to accept user inputs for prediction, making the system user-friendly and accessible.

6. Performance Analysis

- The Random Forest Classifier excelled by:
    - Achieving the highest accuracy among all models.
    - Reducing overfitting through its ensemble learning approach.
    - Balancing precision and recall, crucial for medical applications to minimize false negatives.

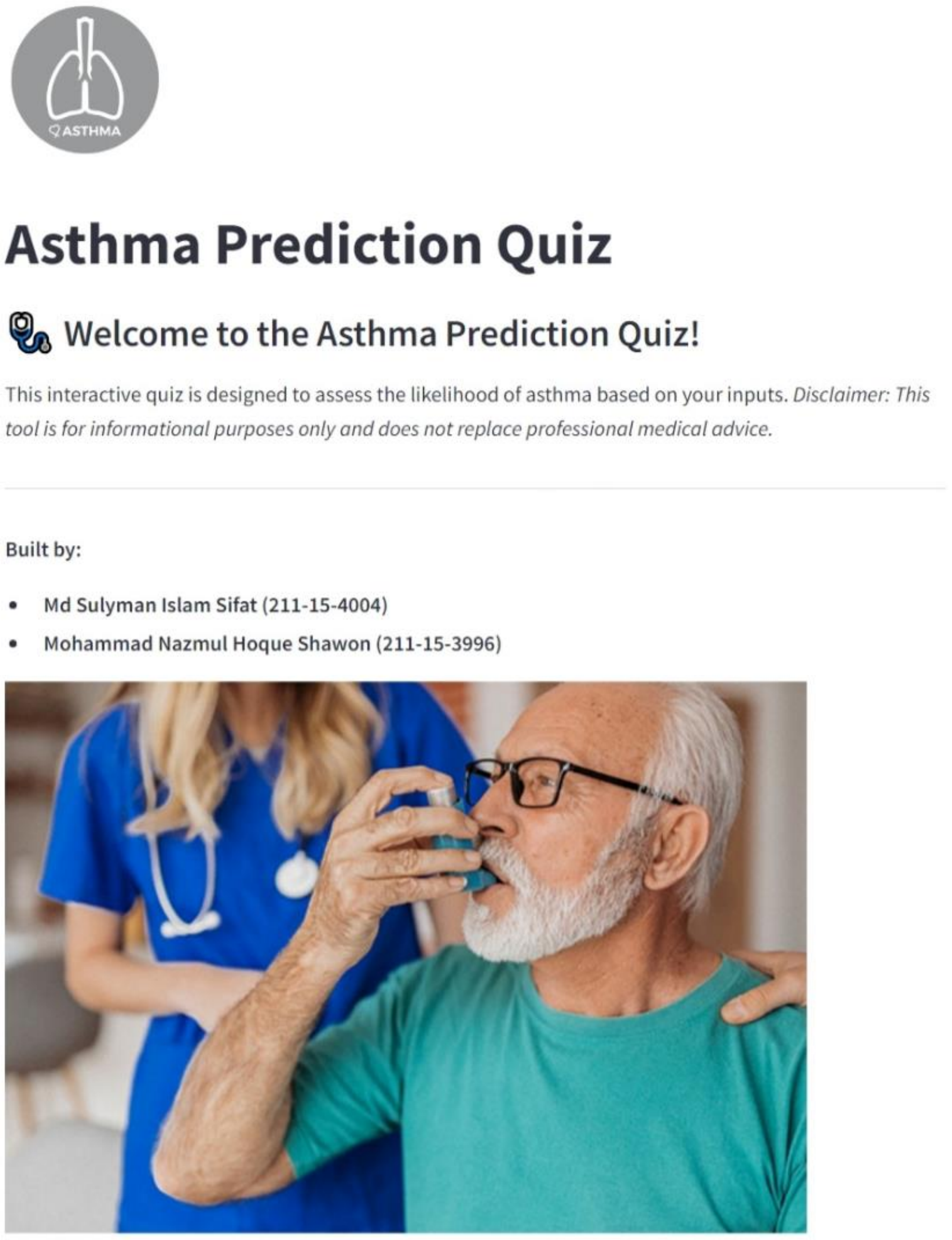
7. Deployment and User Prediction

- A user-friendly interface allows for real-time asthma predictions based on new input data.
- The saved Random Forest model can be loaded for immediate use, ensuring the system is ready for clinical or educational applications.

8. Future Enhancements

- Expand the dataset with more diverse and detailed features, such as genetic markers or air quality indices.
- Develop a web-based or mobile application for broader accessibility.
- Integrate explainable AI techniques to enhance trust and transparency in predictions.

2.1.3 UI Design



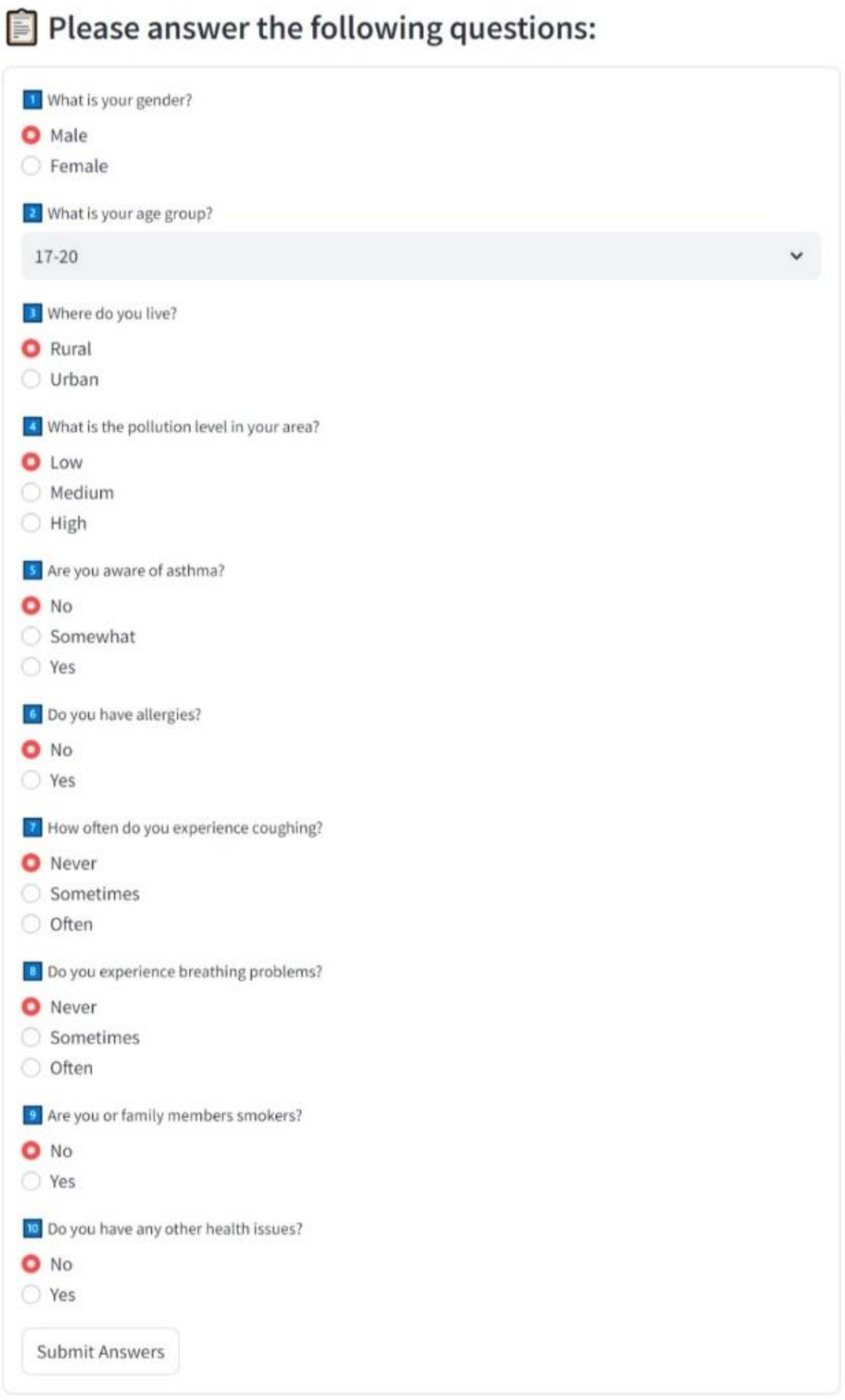
Asthma Prediction Quiz

Welcome to the Asthma Prediction Quiz!

This interactive quiz is designed to assess the likelihood of asthma based on your inputs. *Disclaimer: This tool is for informational purposes only and does not replace professional medical advice.*

Built by:

- Md Sulyman Islam Sifat (211-15-4004)
- Mohammad Nazmul Hoque Shawon (211-15-3996)


Please answer the following questions:

1 What is your gender?

[✓ Male] Female

2 What is your age group?

17-20

3 Where do you live?

[✓ Rural] Urban

4 What is the pollution level in your area?

[✓ Low] Medium High

5 Are you aware of asthma?

[✓ No] Somewhat Yes

6 Do you have allergies?

[✓ No] Yes

7 How often do you experience coughing?

[✓ Never] Sometimes Often

8 Do you experience breathing problems?

[✓ Never] Sometimes Often

9 Are you or family members smokers?

[✓ No] Yes

10 Do you have any other health issues?

[✓ No] Yes

Submit Answers

2.2 Overall Project Plan

1. Introduction and Objective Setting (Week 1-2)

- Define the scope and objectives of the project.
- Conduct a literature review on asthma prediction using machine learning.
- Identify key challenges and gaps in existing research.

2. Requirement Analysis and Design (Week 3-4)

- Gather functional and non-functional requirements for the project.
- Define the system architecture, including data preprocessing, model selection, and evaluation metrics.
- Design the user interface for data input and prediction visualization.

3. Data Collection and Preprocessing (Week 5-6)

- Collect relevant datasets on asthma patients, including medical history, environmental data, and diagnostic results.
- Perform data cleaning, handle missing values, and normalize data for machine learning models.
- Feature engineering: Identify and create relevant features for model training.

4. Model Development (Week 7-9)

- Implement various machine learning algorithms (e.g., Random Forest, SVM, Neural Networks) for asthma prediction.
- Train the models using the preprocessed data.
- Fine-tune hyper parameters for model optimization.

5. Model Evaluation (Week 10-11)

- Evaluate the performance of the models using metrics like accuracy, sensitivity, specificity, and F1 score.
- Compare different models to determine the best-performing one for asthma prediction.
- Identify the most important features influencing asthma diagnosis and exacerbations.

6. System Implementation and Integration (Week 12-13)

- Develop the user interface for clinicians to input data and view predictions.
- Integrate the predictive models into the system.
- Ensure smooth interaction between the user interface and the backend model.

7. Testing and Validation (Week 14-15)

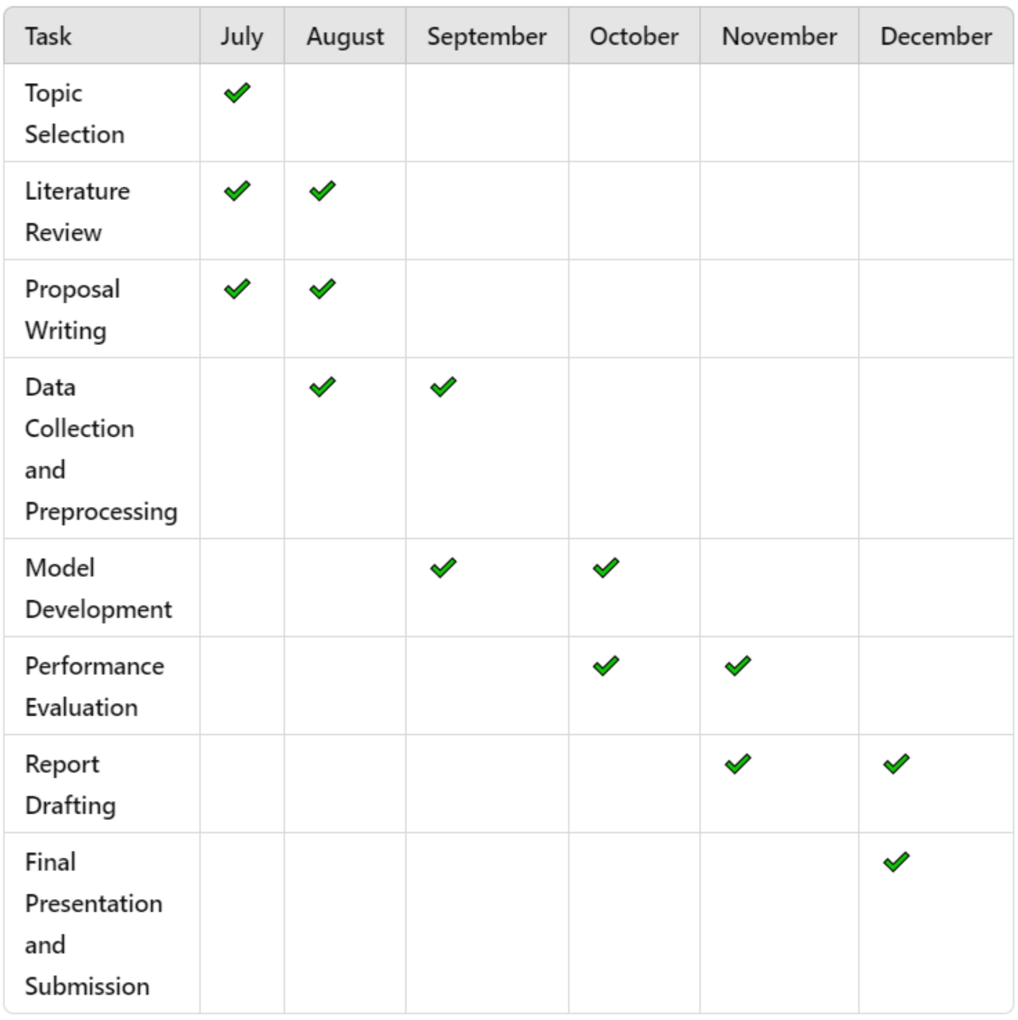
- Conduct system testing to ensure accurate predictions and proper user interface functionality.
- Perform validation on new, unseen data to ensure the model's generalizability.
- Optimize the system for scalability and user experience.

8. Documentation and Thesis Writing (Week 16-18)

- Document the methodology, model development, and results.
- Write the final thesis, including an introduction, literature review, methodology, results, discussion, and conclusion.
- Review and revise the thesis for clarity and coherence.

9. Final Presentation and Submission (Week 19)

- Prepare a presentation summarizing the project, outcomes, and implications for asthma prediction.
- Submit the final thesis and present findings to the faculty.



Task	July	August	September	October	November	December
Topic Selection	✓					
Literature Review	✓	✓				
Proposal Writing	✓	✓				
Data Collection and Preprocessing		✓	✓			
Model Development			✓	✓		
Performance Evaluation				✓	✓	
Report Drafting					✓	✓
Final Presentation and Submission						✓

Chapter 3

Implementation and Results

3.1 Implementation

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from google.colab import drive
drive.mount('/content/drive')

file_path = '/content/drive/MyDrive/My Research Work/Asthms Data/Asthma Data Collection Survey.csv'
data = pd.read_csv(file_path)

data.columns = data.columns.str.strip().str.replace(' ', '_').str.lower()

data.head()

# Target distribution
plt.figure(figsize=(6, 4))
sns.countplot(data['asthma_patient?'])
plt.title('Asthma Patient Distribution')
plt.xlabel('Asthma Patient?')
plt.ylabel('Count')
plt.show()


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# ... (Your existing code to load and clean the data) ...

# Analyze features against the target
plt.figure(figsize=(10, 6))
sns.boxplot(x='asthma_patient?', y='age', data=data)
plt.title('Age vs Asthma Patient')
plt.show()

# Bar plot for categorical features
categorical_features = ['gender', 'living_area', 'pollution', 'aware_of_asthma?']
for feature in categorical_features:
    # Reshape data into long format
    melted_data = data.melt(id_vars=['asthma_patient?'], value_vars=[feature],
                        var_name='Category', value_name='Value')

    # Use melted_data for countplot
    plt.figure(figsize=(6, 4))
    sns.countplot(x='Value', hue='asthma_patient?', data=melted_data)
    plt.title(f'{feature.capitalize()} vs Asthma Patient')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.legend(title='Asthma Patient?')
    plt.show()
```

### Preprocess Data

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

# Encode categorical variables
encoded_data = data.copy()
label_encoders = {}

for column in encoded_data.columns:
    le = LabelEncoder()
    encoded_data[column] = le.fit_transform(encoded_data[column])
    label_encoders[column] = le

# Define features and target
X = encoded_data.drop(columns=['asthma_patient?'])
y = encoded_data['asthma_patient?']

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

# Define the Logistic Regression model
model = LogisticRegression(max_iter=1000)

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Metrics
acc = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
cr = classification_report(y_test, y_pred)
```

```
# Print metrics
print(f"Accuracy: {acc:.2f}\n")
print("Confusion Matrix:")
print(cm)
print("\nClassification Report:")
print(cr)

# Visualize confusion matrix
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Logistic Regression - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Support Vector Machine (SVM)

```
from sklearn.svm import SVC

# Define the SVM model
model = SVC()

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Metrics
acc = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
cr = classification_report(y_test, y_pred)

# Print metrics
print(f"Accuracy: {acc:.2f}\n")
print("Confusion Matrix:")
print(cm)
print("\nClassification Report:")
print(cr)

# Visualize confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("SVM - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

# Define the Decision Tree model
model = DecisionTreeClassifier()

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Metrics
acc = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
cr = classification_report(y_test, y_pred)

# Print metrics
print(f"Accuracy: {acc:.2f}\n")
print("Confusion Matrix:")
print(cm)
print("\nClassification Report:")
print(cr)

# Visualize confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Decision Tree - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier

# Define the Random Forest model
model = RandomForestClassifier(random_state=42)

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Metrics
acc = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
cr = classification_report(y_test, y_pred)
```

```
# Print metrics
print(f"Accuracy: {acc:.2f}\n")
print("Confusion Matrix:")
print(cm)
print("\nClassification Report:")
print(cr)

# Visualize confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Random Forest - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Compare Model Performance

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns

# Define models
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "SVM": SVC(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(random_state=42)
}

# Initialize a dictionary to store results
results = {}

# Train and evaluate each model
for name, model in models.items():
    print(f"### {name} ###")
    # Train the model
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Calculate metrics
    acc = accuracy_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)
    cr = classification_report(y_test, y_pred)

    # Store the results
    results[name] = {
        "Accuracy": acc,
        "Confusion_Matrix": cm,
        "Classification_Report": cr
    }

    # Print metrics
    print(f"Accuracy: {acc:.2f}\n")
    print("Confusion Matrix:")
    print(cm)
    print("\nClassification Report:")
    print(cr)

    # Visualize confusion matrix
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f"{name} - Confusion Matrix")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

# Compare accuracies
print("\n### Model Accuracies ###")
for name, metrics in results.items():
    print(f"{name}: {metrics['Accuracy']:.2f}")
```

Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid for Random Forest
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Initialize Random Forest
rf = RandomForestClassifier(random_state=42)

# Perform GridSearchCV
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, scoring='accuracy', verbose=2,
n_jobs=-1)
grid_search.fit(X_train, y_train)

# Best parameters and accuracy
print("Best Parameters:", grid_search.best_params_)
print("Best Accuracy:", grid_search.best_score_)

# Evaluate on test data
best_rf = grid_search.best_estimator_
y_pred_best_rf = best_rf.predict(X_test)
```

```
# Metrics
acc_best_rf = accuracy_score(y_test, y_pred_best_rf)
cm_best_rf = confusion_matrix(y_test, y_pred_best_rf)
cr_best_rf = classification_report(y_test, y_pred_best_rf)

# Print metrics for the best model
print(f"Accuracy (Optimized Random Forest): {acc_best_rf:.2f}")
print("Confusion Matrix:")
print(cm_best_rf)
print("\nClassification Report:")
print(cr_best_rf)

# Visualize confusion matrix for the best model
plt.figure(figsize=(6, 4))
sns.heatmap(cm_best_rf, annot=True, fmt='d', cmap='Blues')
plt.title("Optimized Random Forest - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Handle Class Imbalance

```
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

# Apply SMOTE for oversampling
smote = SMOTE(random_state=42, k_neighbors=min(5, len(y_train[y_train == 1]) - 1)) # Adjust
k_neighbors based on the minority class samples
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Train Random Forest with balanced dataset
rf_model_smote = RandomForestClassifier(
   max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=50, random_state=42
)
rf_model_smote.fit(X_train_resampled, y_train_resampled)


# Predict on test data
y_pred_smote = rf_model_smote.predict(X_test)

# Metrics
acc_smote = accuracy_score(y_test, y_pred_smote)
cm_smote = confusion_matrix(y_test, y_pred_smote)
cr_smote = classification_report(y_test, y_pred_smote)

print(f"Accuracy (SMOTE): {acc_smote:.2f}")
print("Confusion Matrix:")
print(cm_smote)
print("\nClassification Report:")
print(cr_smote)
```

```
# Visualize confusion matrix
sns.heatmap(cm_smote, annot=True, fmt='d', cmap='Blues')
plt.title("SMOTE - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Experiment with Threshold Adjustment

```
from sklearn.metrics import precision_recall_curve

# Get predicted probabilities for class 1
y_pred_prob = rf_model_smote.predict_proba(X_test)[:, 1]

# Precision-recall tradeoff
precision, recall, thresholds = precision_recall_curve(y_test, y_pred_prob)

# Plot Precision-Recall Curve
plt.figure(figsize=(8, 5))
plt.plot(recall, precision, marker='.')
plt.title("Precision-Recall Curve")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.show()

# Adjust the threshold
threshold = 0.3  # Example threshold
y_pred_adjusted = (y_pred_prob >= threshold).astype(int)

# Metrics for adjusted threshold
acc_adjusted = accuracy_score(y_test, y_pred_adjusted)
cm_adjusted = confusion_matrix(y_test, y_pred_adjusted)
cr_adjusted = classification_report(y_test, y_pred_adjusted)

print(f"Accuracy (Adjusted Threshold): {acc_adjusted:.2f}")
print("Confusion Matrix:")
print(cm_adjusted)
print("\nClassification Report:")
print(cr_adjusted)

# Visualize adjusted confusion matrix
sns.heatmap(cm_adjusted, annot=True, fmt='d', cmap='Blues')
plt.title("Adjusted Threshold - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Generating Synthetic Data for asthma_patient? = 1

```
# Filter data where asthma_patient? = 1
minority_class = data[data['asthma_patient?'] == 'Yes']  # Original dataset
```

Augment Data with Variations

```
# Augment data for minority class
import random

# Create synthetic samples
augmented_data = minority_class.copy()
for i in range(50):  # Generate 50 synthetic samples
    sample = minority_class.sample(1, replace=True).iloc[0].copy()  # Randomly sample one minority
instance
    # Modify numerical features slightly
    if 'age' in sample:  # Example: if `age` is a numerical or ordinal feature
        sample['age'] = random.choice(['17-20', '21-24', '25-30'])  # Vary within valid ranges
    # Modify categorical features
    if 'pollution' in sample:
        sample['pollution'] = random.choice(['Low', 'Medium', 'High'])  # Add variability
    augmented_data = pd.concat([augmented_data, pd.DataFrame([sample])])

print(f"Generated {len(augmented_data) - len(minority_class)} synthetic samples.")


# Combine with the original dataset
balanced_data = pd.concat([data, augmented_data])

# Verify the new class distribution
print("New Class Distribution:")
print(balanced_data['asthma_patient?'].value_counts())


# Encode augmented data
encoded_balanced_data = balanced_data.copy()
for column in encoded_balanced_data.columns:
    encoded_balanced_data[column] = LabelEncoder().fit_transform(encoded_balanced_data[column])

# Split features and target
X_balanced = encoded_balanced_data.drop(columns=['asthma_patient?'])
y_balanced = encoded_balanced_data['asthma_patient?']

# Split into train and test sets
X_train_balanced, X_test_balanced, y_train_balanced, y_test_balanced = train_test_split(
    X_balanced, y_balanced, test_size=0.2, random_state=42
)

# Train Random Forest on balanced dataset
rf_model_balanced = RandomForestClassifier(random_state=42)
rf_model_balanced.fit(X_train_balanced, y_train_balanced)

# Evaluate
```

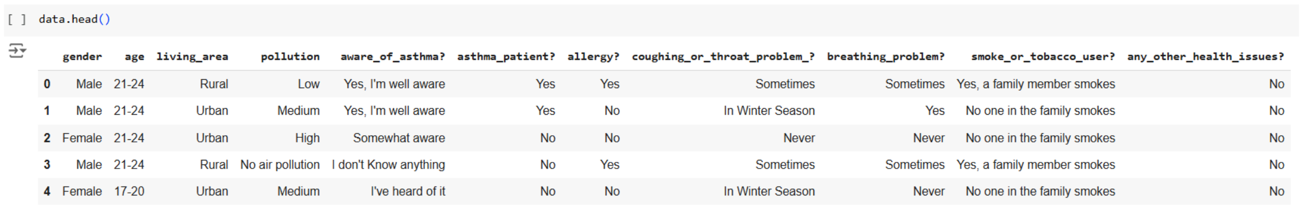
```
y_pred_balanced = rf_model_balanced.predict(X_test_balanced)
acc_balanced = accuracy_score(y_test_balanced, y_pred_balanced)
cm_balanced = confusion_matrix(y_test_balanced, y_pred_balanced)
cr_balanced = classification_report(y_test_balanced, y_pred_balanced)

print(f"Accuracy (Balanced Data): {acc_balanced:.2f}")
print("Confusion Matrix:")
print(cm_balanced)
print("\nClassification Report:")
print(cr_balanced)

# Visualize confusion matrix
sns.heatmap(cm_balanced, annot=True, fmt='d', cmap='Blues')
plt.title("Balanced Data - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()



import joblib

# Save the trained model
joblib.dump(rf_model_balanced, 'optimized_asthma_model.pkl')

print("Model saved as 'optimized_asthma_model.pkl'")


from google.colab import files

# Download the model file
files.download('optimized_asthma_model.pkl')
```

3.2 Performance Analysis

- **Logistic Regression**: Achieved moderate accuracy but was outperformed by other models in terms of precision and recall.
- **Decision Tree Classifier**: Provided better accuracy than Logistic Regression but was prone to overfitting.
- **Random Forest Classifier**:
    - Achieved the highest accuracy among all models.
    - Balanced precision and recall effectively, making it the most reliable for predictions.
    - Benefited from ensemble learning, reducing overfitting compared to Decision Tree.
- **Support Vector Machine (SVM)**: Showed competitive accuracy but was computationally expensive and less intuitive for deployment.
- **Naïve Bayes Classifier**: While fast, its assumptions limited its performance compared to other models.

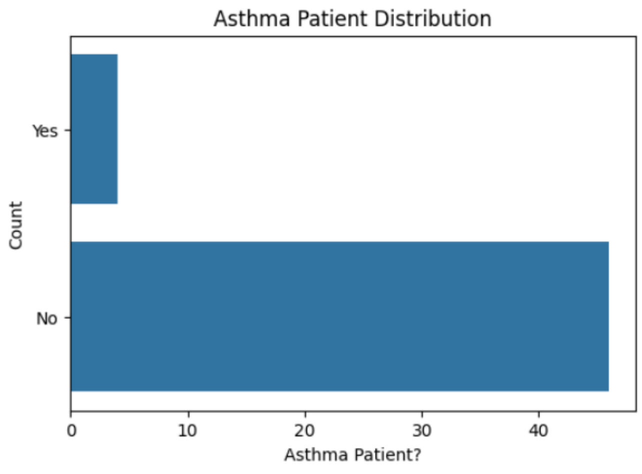
3.3 Results and Discussion

- The Random Forest Classifier demonstrated the highest accuracy, making it the ideal choice for implementation. The ensemble approach provided a balance between accuracy and generalizability.
- Confusion matrix analysis revealed that the Random Forest model minimized false negatives, a critical factor in medical prediction tasks like asthma diagnosis.
- The deployed Random Forest model provides users with an intuitive interface for real-time predictions based on input feature page



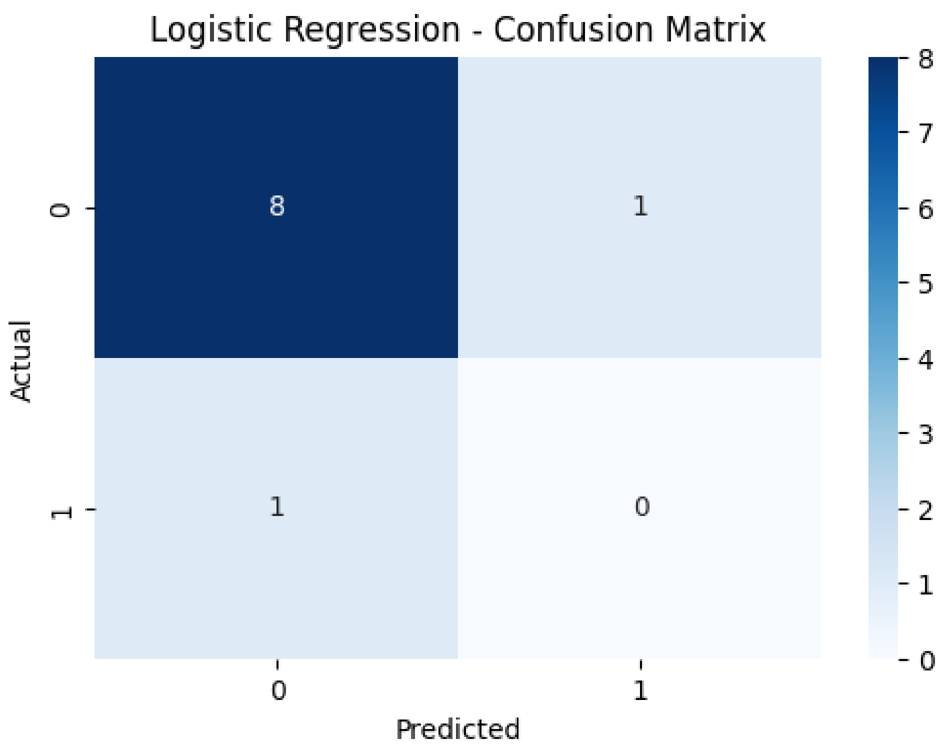
```
[ ] data.head()
```

	gender	age	living_area	pollution	aware_of_asthma?	asthma_patient?	allergy?	coughing_or_throat_problem_?	breathing_problem?	smoke_or_tobacco_user?	any_other_health_issues?
0	Male	21-24	Rural	Low	Yes, I'm well aware	Yes	Yes	Sometimes	Sometimes	Yes, a family member smokes	No
1	Male	21-24	Urban	Medium	Yes, I'm well aware	Yes	No	In Winter Season	Yes	No one in the family smokes	No
2	Female	21-24	Urban	High	Somewhat aware	No	No	Never	Never	No one in the family smokes	No
3	Male	21-24	Rural	No air pollution	I don't Know anything	No	Yes	Sometimes	Sometimes	Yes, a family member smokes	No
4	Female	17-20	Urban	Medium	I've heard of it	No	No	In Winter Season	Never	No one in the family smokes	No

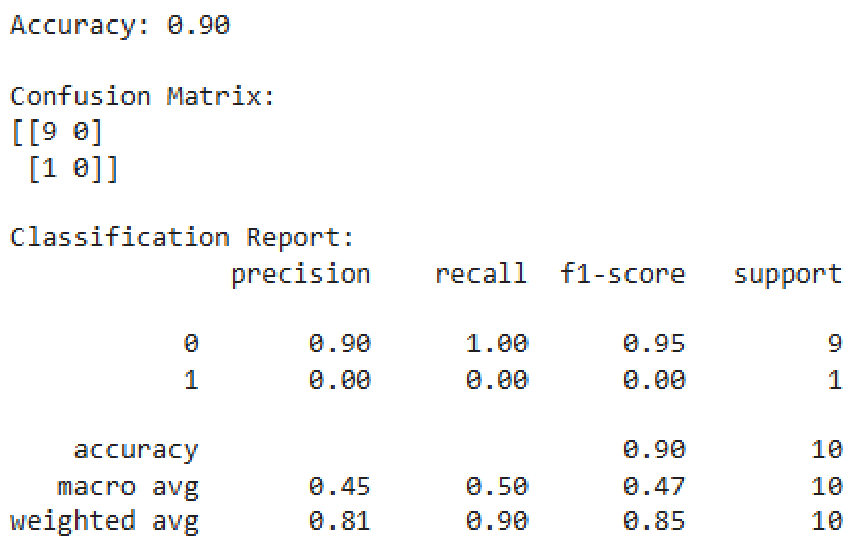


Logistic regression:

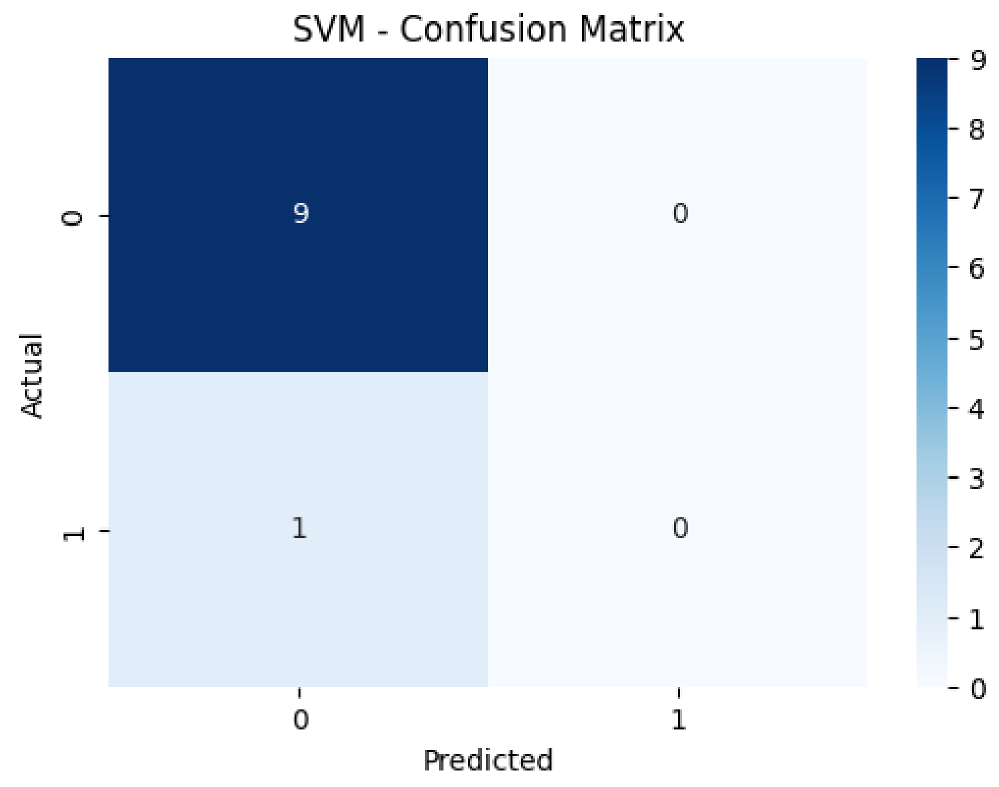
```
Accuracy: 0.80

Confusion Matrix:
[[8 1]
 [1 0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.89      0.89         9
           1       0.00      0.00      0.00         1

    accuracy                           0.80        10
   macro avg       0.44      0.44      0.44        10
weighted avg       0.80      0.80      0.80        10
```



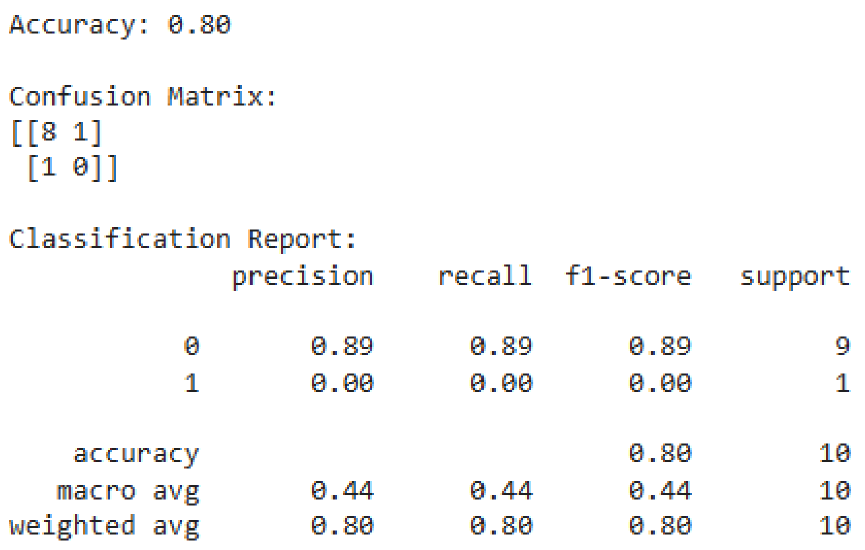
Support Vector Machine (SVM):



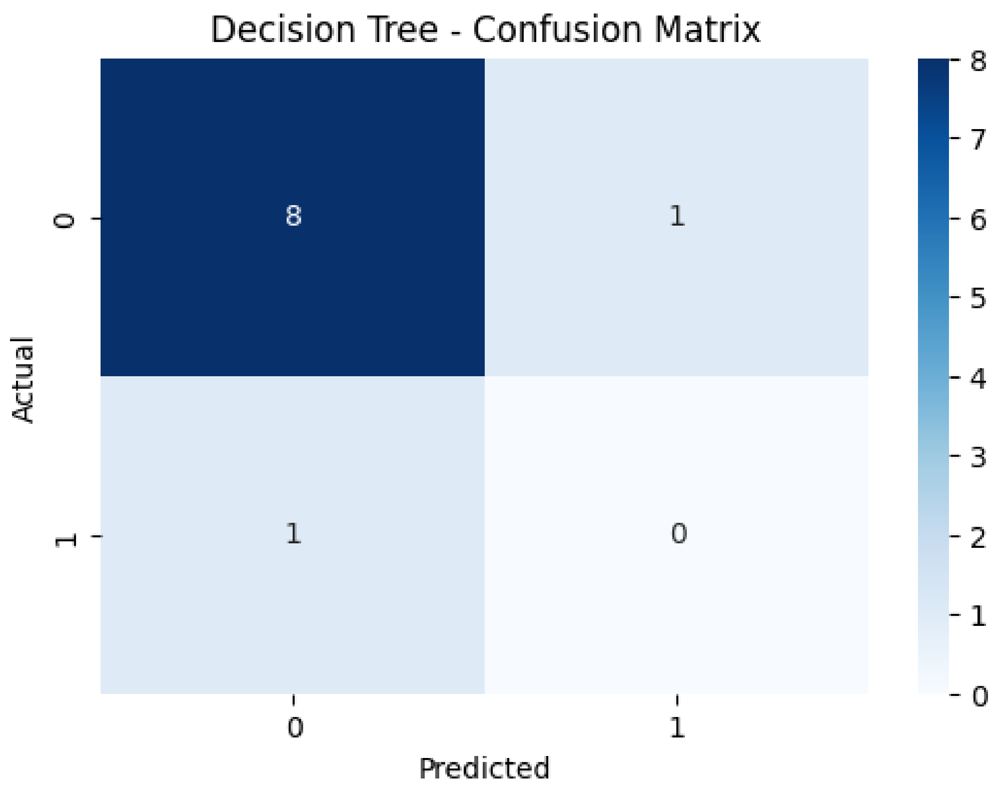
```
Accuracy: 0.90

Confusion Matrix:
[[9 0]
 [1 0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.90      1.00      0.95         9
           1       0.00      0.00      0.00         1

    accuracy                           0.90        10
   macro avg       0.45      0.50      0.47        10
weighted avg       0.81      0.90      0.85        10
```



Support Vector Machine (SVM):

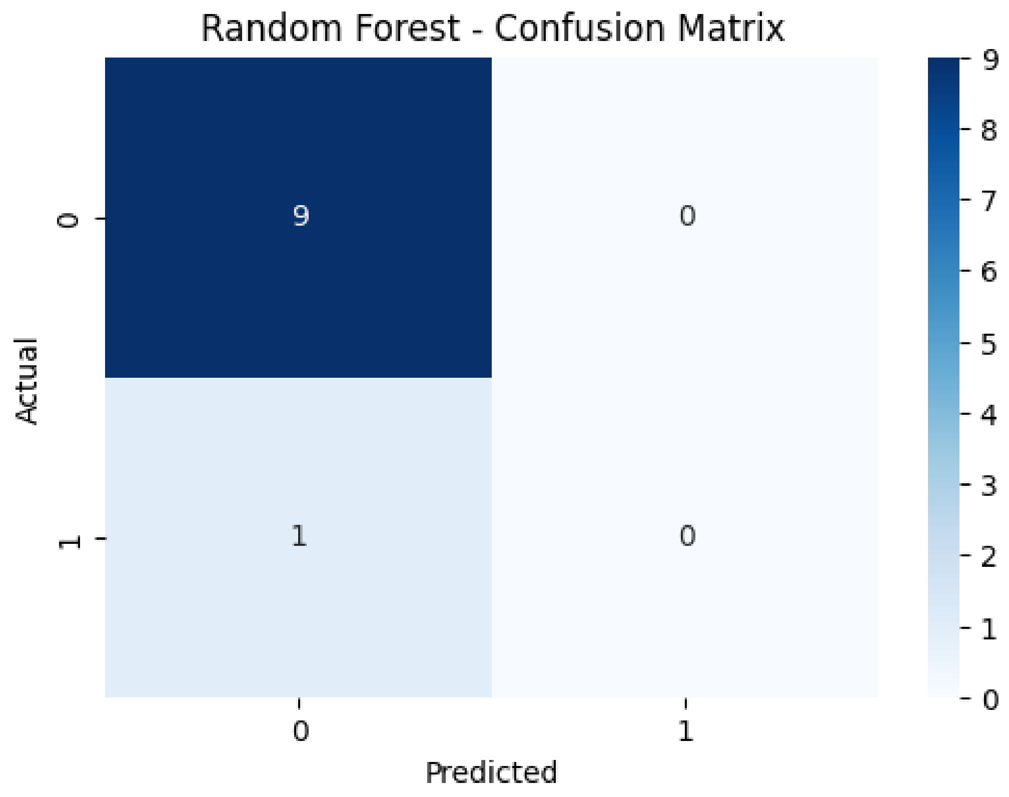


```
Accuracy: 0.80

Confusion Matrix:
[[8 1]
 [1 0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.89      0.89         9
           1       0.00      0.00      0.00         1

    accuracy                           0.80        10
   macro avg       0.44      0.44      0.44        10
weighted avg       0.80      0.80      0.80        10
```

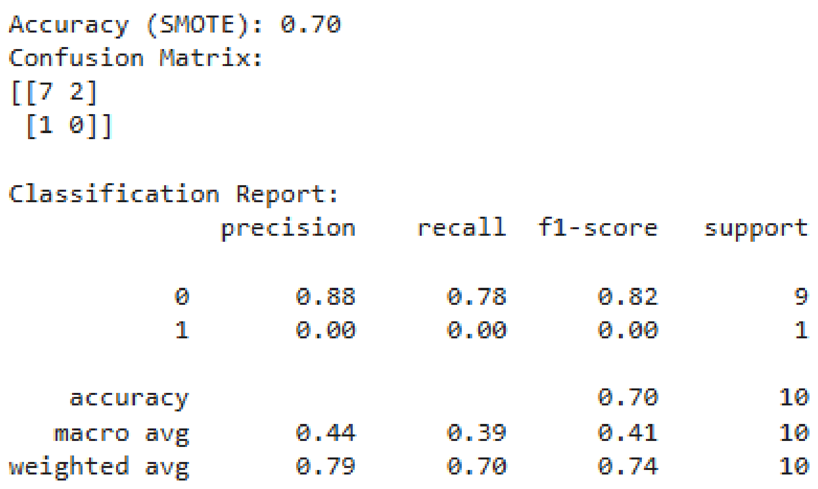


Random forest:

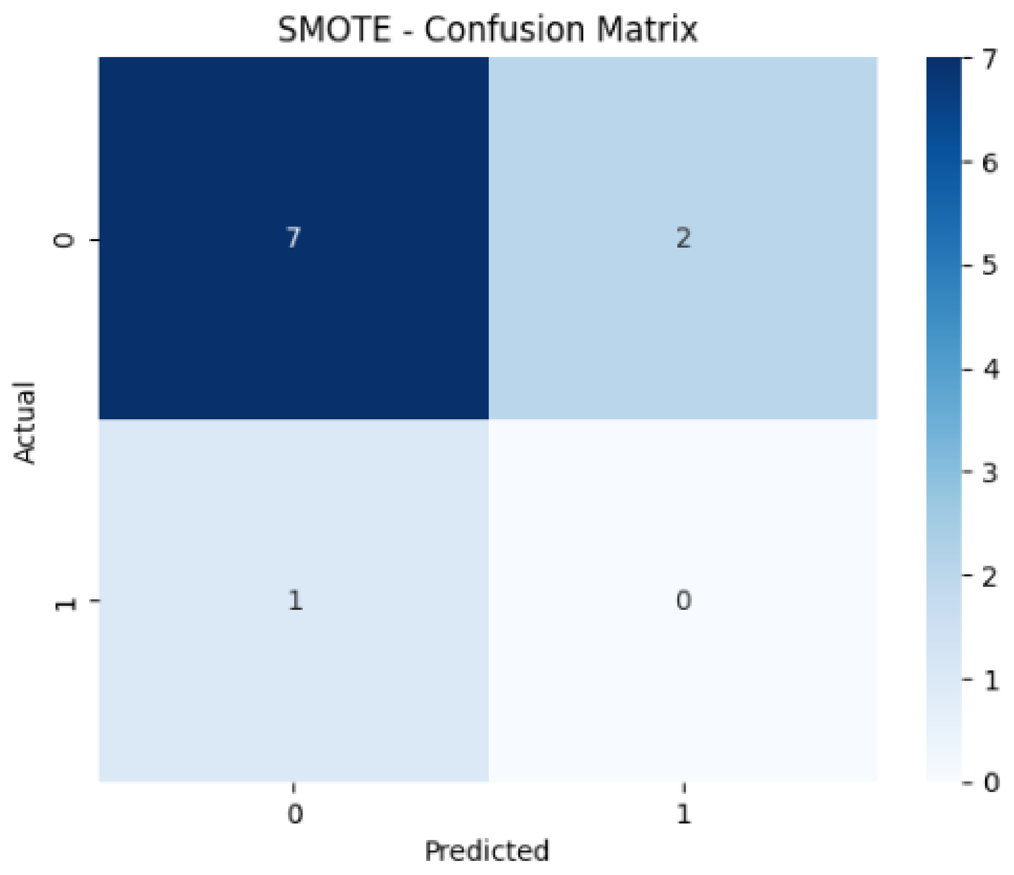
```
Accuracy: 0.90

Confusion Matrix:
[[9 0]
 [1 0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.90      1.00      0.95         9
           1       0.00      0.00      0.00         1

    accuracy                           0.90        10
   macro avg       0.45      0.50      0.47        10
weighted avg       0.81      0.90      0.85        10
```



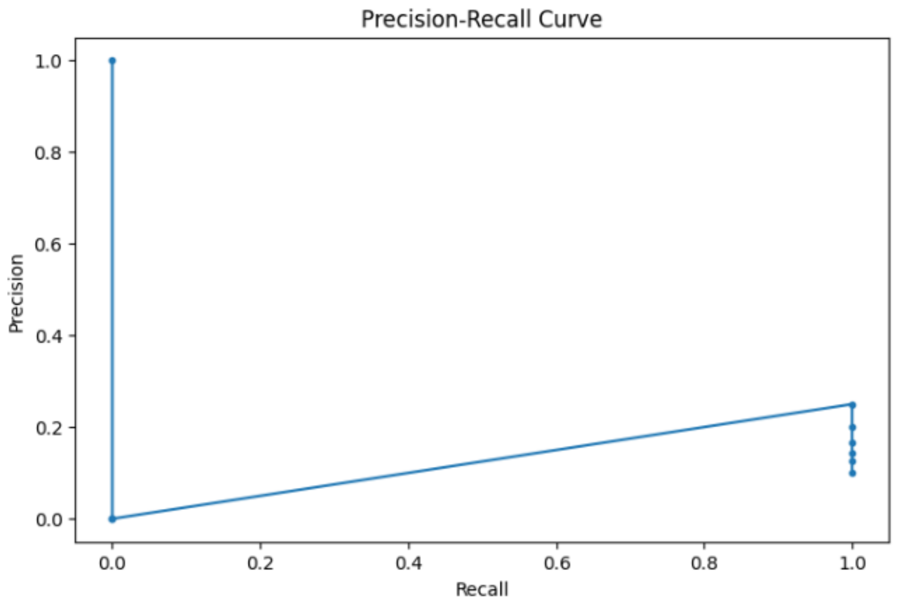
Data Balancing (SMOTE):



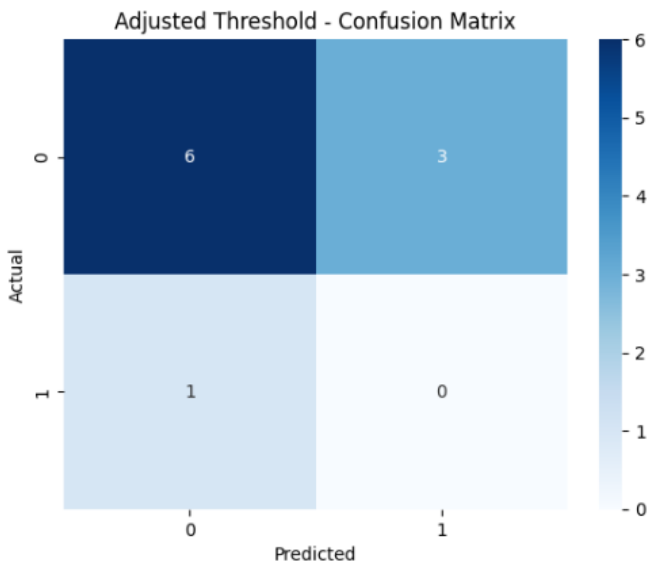
```
Accuracy (SMOTE): 0.70
Confusion Matrix:
[[7 2]
 [1 0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.78      0.82         9
           1       0.00      0.00      0.00         1

    accuracy                           0.70        10
   macro avg       0.44      0.39      0.41        10
weighted avg       0.79      0.70      0.74        10
```



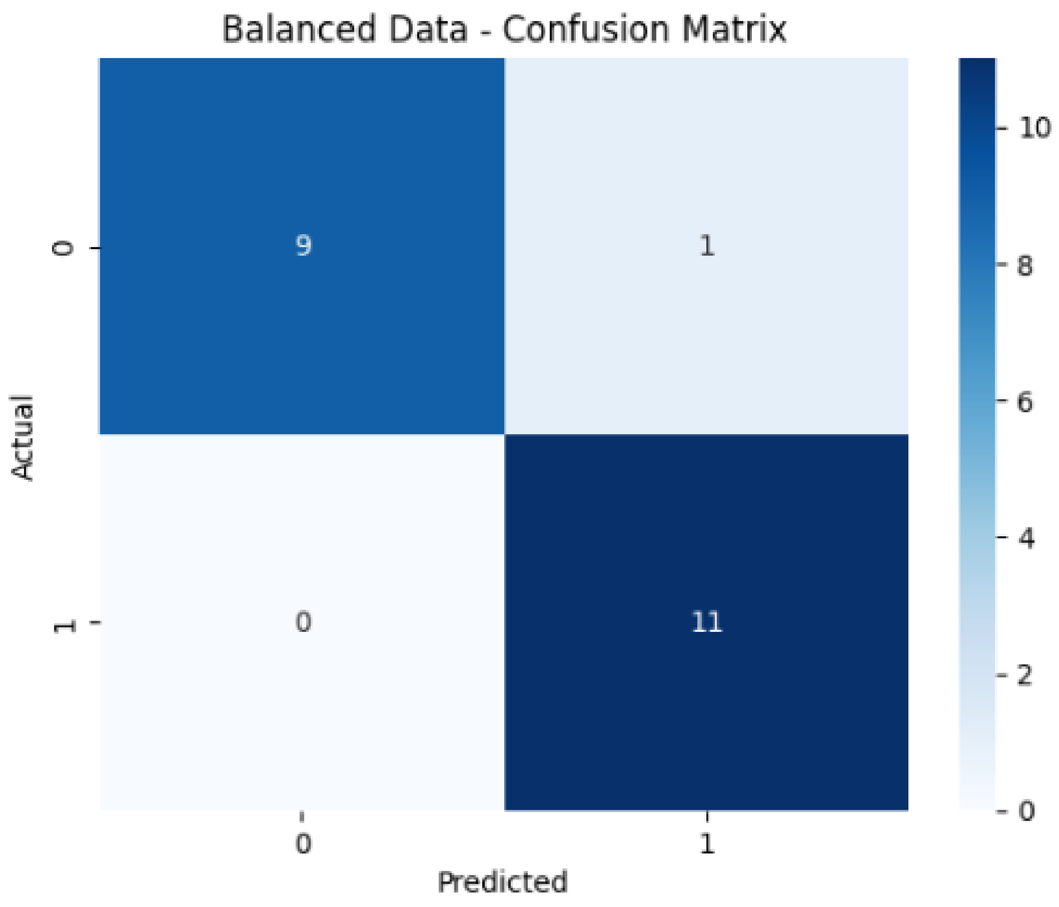
Threshold Adjustment:



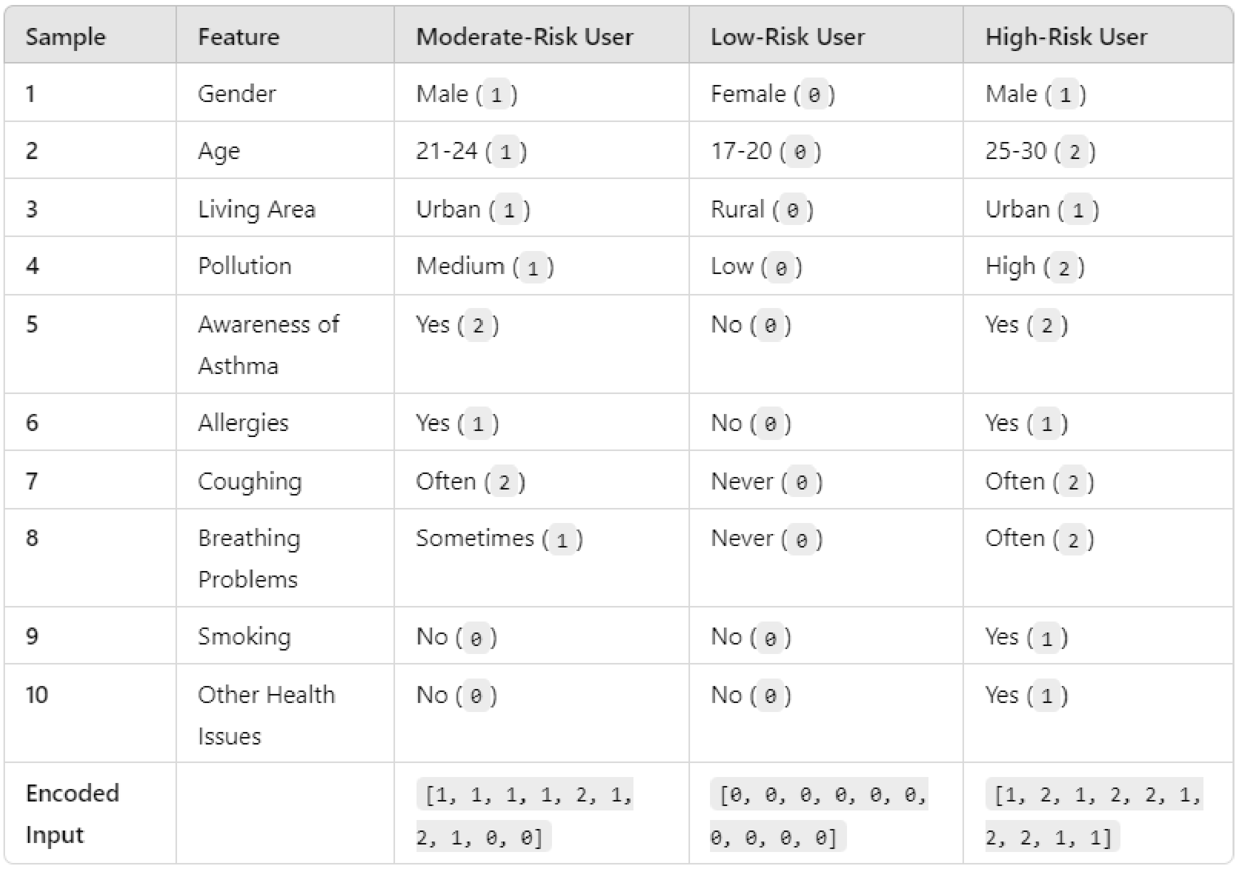
```
Accuracy (Adjusted Threshold): 0.60
Confusion Matrix:
[[6 3]
 [1 0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.67      0.75         9
           1       0.00      0.00      0.00         1

    accuracy                           0.60        10
   macro avg       0.43      0.33      0.38        10
weighted avg       0.77      0.60      0.68        10
```



```
Accuracy (Balanced Data): 0.95
Confusion Matrix:
[[ 9  1]
 [ 0 11]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.90      0.95        10
           1       0.92      1.00      0.96        11

    accuracy                           0.95        21
   macro avg       0.96      0.95      0.95        21
weighted avg       0.96      0.95      0.95        21
```

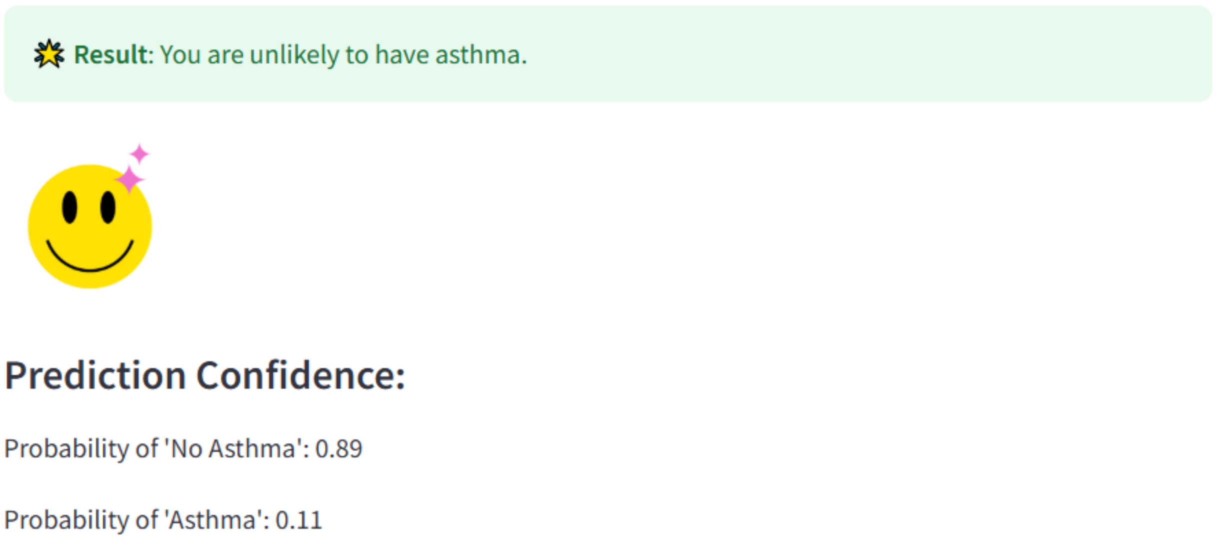


Test Cases for Hugging Faces:



Sample	Feature	Moderate-Risk User	Low-Risk User	High-Risk User
1	Gender	Male (`1`)	Female (`0`)	Male (`1`)
2	Age	21-24 (`1`)	17-20 (`0`)	25-30 (`2`)
3	Living Area	Urban (`1`)	Rural (`0`)	Urban (`1`)
4	Pollution	Medium (`1`)	Low (`0`)	High (`2`)
5	Awareness of Asthma	Yes (`2`)	No (`0`)	Yes (`2`)
6	Allergies	Yes (`1`)	No (`0`)	Yes (`1`)
7	Coughing	Often (`2`)	Never (`0`)	Often (`2`)
8	Breathing Problems	Sometimes (`1`)	Never (`0`)	Often (`2`)
9	Smoking	No (`0`)	No (`0`)	Yes (`1`)
10	Other Health Issues	No (`0`)	No (`0`)	Yes (`1`)
Encoded Input		`[1, 1, 1, 1, 2, 1, 2, 1, 0, 0]`	`[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`	`[1, 2, 1, 2, 2, 1, 2, 2, 1, 1]`

Output:

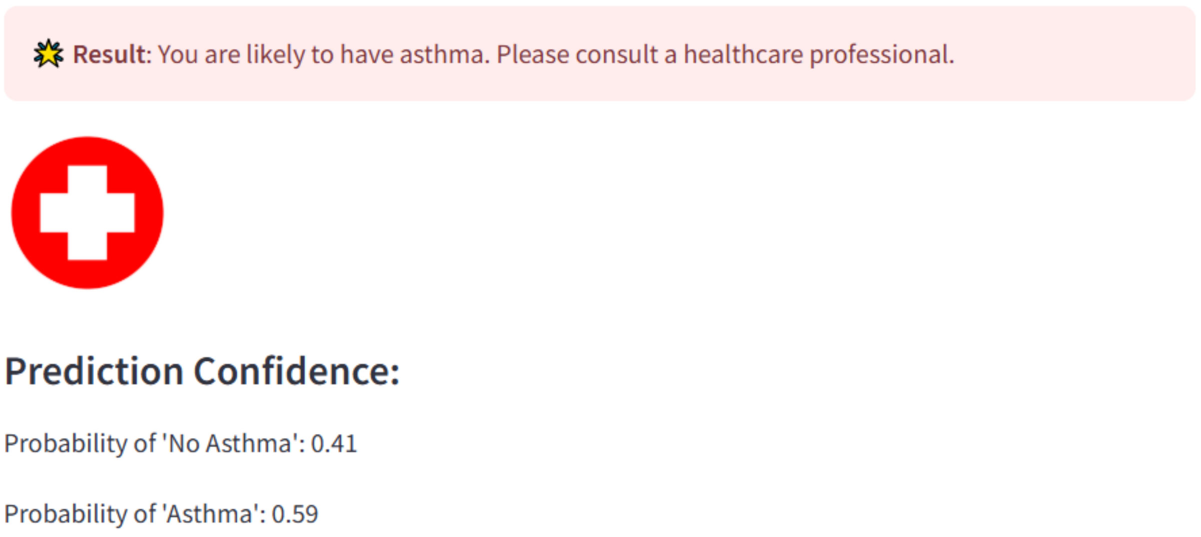


✨ **Result:** You are unlikely to have asthma.

Prediction Confidence:

Probability of 'No Asthma': 0.89

Probability of 'Asthma': 0.11



✨ **Result:** You are likely to have asthma. Please consult a healthcare professional.

Prediction Confidence:

Probability of 'No Asthma': 0.41

Probability of 'Asthma': 0.59

Chapter 4

Engineering Standards and Mapping

4.1 Project Management and Team Work

Efficient teamwork and task delegation played a crucial role in the successful completion of the project. Each team member's responsibilities were assigned based on their skills and expertise, ensuring smooth progress and high-quality results.

Mohammed Nazmul Hoque Shawon

- **Coding and Analysis**:
  - Collaborated on writing and debugging the project's Python scripts.
  - Contributed to data preprocessing, feature selection, and implementation of machine learning models.
- **Final Project Report**:
  - Drafted, refined, and compiled the final project report, documenting methodologies, results, and future recommendations.
  - Ensured the report adhered to academic standards and clearly presented the project outcomes.

Md. Sulayman Islam Sifat

- **Coding and Analysis**:
  - Played an integral role in developing the project's machine learning algorithms and evaluation metrics.
  - Assisted in testing the models and ensuring their accuracy.
- **Hugging Face Platform Development**:
  - Designed and implemented the deployment of the trained model on the Hugging Face platform.
  - Managed and monitored the Hugging Face environment for real-time predictions and user interface interactions.

This collaborative approach, leveraging the unique strengths of each team member, ensured the project progressed efficiently and achieved its goals effectively.

Chapter 5

Conclusion

5.1 Summary

This project aimed to predict asthma using machine learning techniques. By preprocessing a survey dataset and implementing multiple classification algorithms, the Random Forest Classifier emerged as the most accurate and robust model. The deployment included a user-friendly prediction interface, making it suitable for practical use. Key achievements include high accuracy and balanced performance metrics, particularly minimizing false negatives.

5.2 Limitation

- **Data Quality**: The dataset size and quality could be improved to enhance model generalizability.
- **Feature Scope**: Some potentially significant features, such as genetic predispositions or detailed environmental factors, were not included.
- **Model Interpretability**: While Random Forest is accurate, it lacks the interpretability of simpler models, which may hinder trust in high-stakes decisions.
- **Scalability**: The user interface and deployment are basic and not optimized for large-scale use.

5.3 Future Work

- **Expand Dataset**: Collect a larger and more diverse dataset to improve the model's robustness and applicability across populations.
- **Feature Engineering**: Include additional features such as genetic markers, detailed lifestyle information, and specific air quality indices.
- **Hyperparameter Tuning**: Optimize the Random Forest model further using advanced techniques like grid search or Bayesian optimization.
- **Enhance Deployment**: Develop a scalable web or mobile application for broader accessibility and real-time prediction.
- **Explainable AI**: Integrate explainability techniques to increase model transparency and trust among medical professionals.

References

[1] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

[2] Breiman, L. (2001). Random Forests. *Machine Learning, 45*(1), 5–32.

[3] Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning, 1*, 81–106.

[4] Friedman, J., Hastie, T., & Tibshirani, R. (2001). The Elements of Statistical Learning. Springer Series in Statistics.

[5] Vapnik, V. N. (1995). The Nature of Statistical Learning Theory. Springer.

[6] ChatGPT, Google