**IRIS FLOWER SPECIES PREDICTION**

**ABSTRACT**

The classification of iris flower species is a foundational problem in machine learning, often used for educational and experimental purposes due to its simplicity and structured dataset. In this project, we develop a web-based application that predicts the species of an iris flower based on its morphological features—sepal length, sepal width, petal length, and petal width. The prediction is carried out using two supervised machine learning algorithms: Random Forest and Logistic Regression.

We trained both models using the well-known Iris dataset and evaluated their performance based on accuracy, confusion matrix, and ROC curves. The application is built using Flask for the backend and provides a user-friendly interface for both authenticated users and new visitors. After logging in or registering, users can input flower measurements and receive real-time species predictions.

The results show that both algorithms perform well, with Random Forest achieving slightly higher accuracy. This system demonstrates the practical use of machine learning models in web applications and can serve as a stepping stone for more complex classification tasks in the future**.**

**INTRODUCTION**

**Key characteristics:**

* **User Authentication System**Provides secure login and registration functionality using SQLite and Flask sessions.
* **Machine Learning Integration**

Implements Random Forest and Logistic Regression models trained on the Iris dataset using scikit-learn.

* **Web-based Interface** User-friendly web app developed using Flask and HTML/CSS (Bootstrap), allowing users to interact with the ML models.
* **Real-time Prediction** Accepts user inputs for sepal and petal dimensions and provides instant prediction of the iris flower species.
* **Model Performance Evaluation** Displays key metrics such as accuracy, confusion matrix, and classification report for both models.
* **Data Preprocessing**Checks for missing values and duplicates before training the models.
* **Interactive Visualization**Shows accuracy comparison and ROC curves to visualize and compare model performance.

**Work flow:**

* **User Authentication**
  + Register: Users sign up with full name, email, and password.
  + Login: Users log in with their credentials.
* **Access Dashboard**
  + After login, users are redirected to the dashboard.
  + Dashboard provides access to the prediction system.
* **Data Preprocessing**
  + Load iris.csv dataset.
  + Check for missing and duplicate values.
  + Split dataset into features (X) and target (y).
  + Perform train-test split (e.g., 80-20).
* **Model Training**
  + Train Random Forest and Logistic Regression models.
  + Evaluate using accuracy, classification report, confusion matrix, and ROC curves.
* **Model Comparison**
  + Display comparison using bar chart (accuracy), heatmaps (confusion matrix), and ROC curves.
* **Species Prediction**
  + User inputs sepal and petal dimensions.
  + Select model for prediction.
  + Display predicted Iris species.
* **Logout**
  + User logs out and session is cleared.

**EXISTING SYSTEM**

The existing systems for Iris flower species classification are primarily based on traditional machine learning techniques using the well-known Iris dataset. These systems typically involve pre-written scripts or simple models that demonstrate basic classification algorithms. They are widely used in academic settings to teach the fundamentals of supervised learning. Some key characteristics of existing systems include:

* **Limited Algorithm Use**: Most systems use basic models such as K-Nearest Neighbors, Decision Trees, or Logistic Regression, often without optimization or ensemble techniques.
* **Minimal User Interaction**: Existing systems are usually run in programming environments like Jupyter Notebook or Python scripts, offering little to no user interface for interaction.
* **Lack of Deployment**: These models often remain in offline environments, without web-based interfaces or deployment on platforms for real-time predictions.
* **No Data Visualization or Insights**: Many basic systems lack visualizations to understand data patterns or to present model performance metrics effectively.
* **No Authentication or Personalization**: These systems do not include user login, session management, or customization features, which limits practical, real-world application.

In summary, while existing systems serve well for educational and demonstration purposes, they often lack interactivity, scalability, and user-focused design needed for deployment in real-world applications.

**Disadvantages of Random Forest**

* Complexity: It builds multiple trees, making it computationally heavier than simpler models.
* Less interpretable: Harder to interpret compared to Logistic Regression or Decision Trees.
* Overfitting risk: Can overfit if the number of trees or depth is not well-tuned.
* Slower prediction: Prediction time can be slower for large datasets compared to simple models.

**Disadvantages of Logistic Regression**

* Assumes linearity: Performs poorly if the relationship between features and output is not linear.
* Limited flexibility: Cannot handle complex patterns or interactions like tree-based models.
* Sensitive to outliers: Outliers can affect model performance significantly.
* Not ideal for non-linearly separable data: Struggles when classes overlap in feature space.

**PROPOSED SYSTEM**

The proposed *Iris Flower Species Prediction System* is designed to address the shortcomings of existing systems by providing an interactive, accurate, and user-friendly platform for predicting flower species. Unlike traditional implementations, this system incorporates advanced machine learning algorithms such as Logistic Regression and Random Forest to improve prediction accuracy. It features a web-based interface developed using Flask, allowing users to input flower measurements and receive real-time species predictions without needing programming knowledge. The system also includes user authentication through login and registration, enabling secure access and personalized usage. Additionally, it offers visual representations of the dataset and model performance through graphs and charts, helping users better understand the results. By integrating machine learning, visualization, and web technologies, the proposed system provides a complete and scalable solution suitable for both educational and real-world applications.

**Advantages of Random Forest**

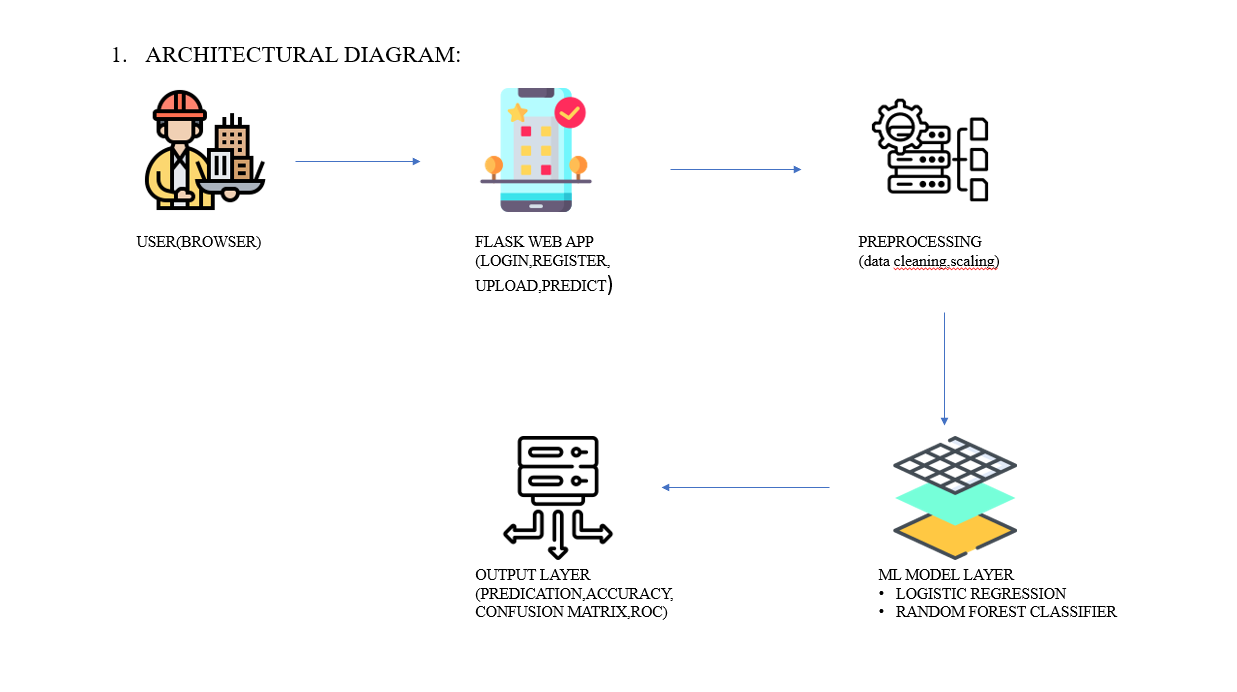
* High Accuracy: Often provides strong performance due to ensemble learning.
* Handles Non-Linearity: Can model complex relationships between features.
* Robust to Overfitting: Less likely to overfit compared to individual decision trees.
* Handles Missing Data: Can still perform well even with missing or incomplete data.
* Feature Importance: Helps in identifying the most influential features.

**Advantages of Logistic Regression**

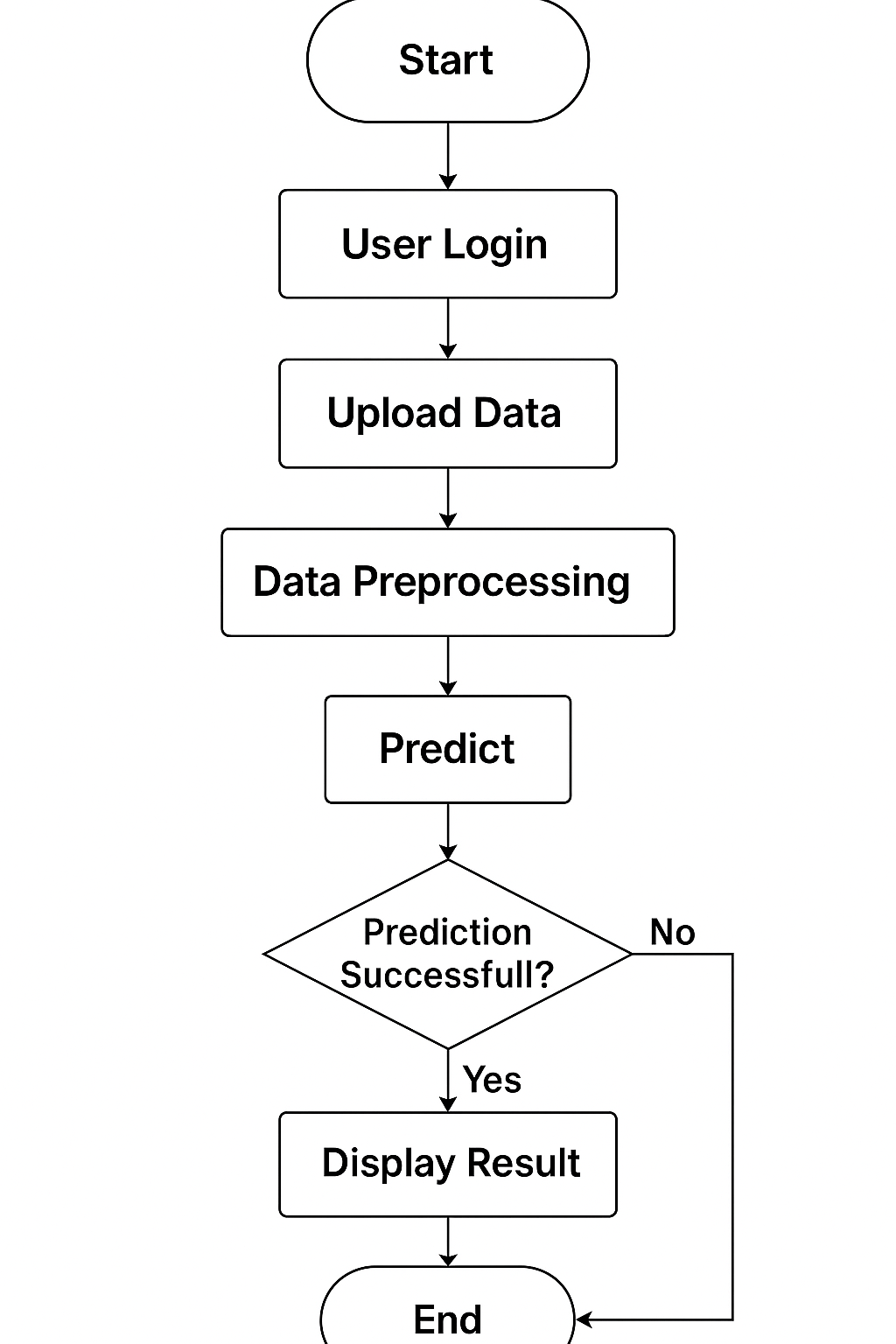
* Simple and Fast: Easy to implement and computationally efficient.
* Interpretability: Coefficients can be interpreted directly to understand feature impact.
* Works Well for Linearly Separable Data: Performs well when classes are linearly separable.
* Probabilistic Output: Provides class probabilities, which is useful in many applications.
* Low Overfitting Risk: Less prone to overfitting when regularization is used.

**UML DIAGRAMS**

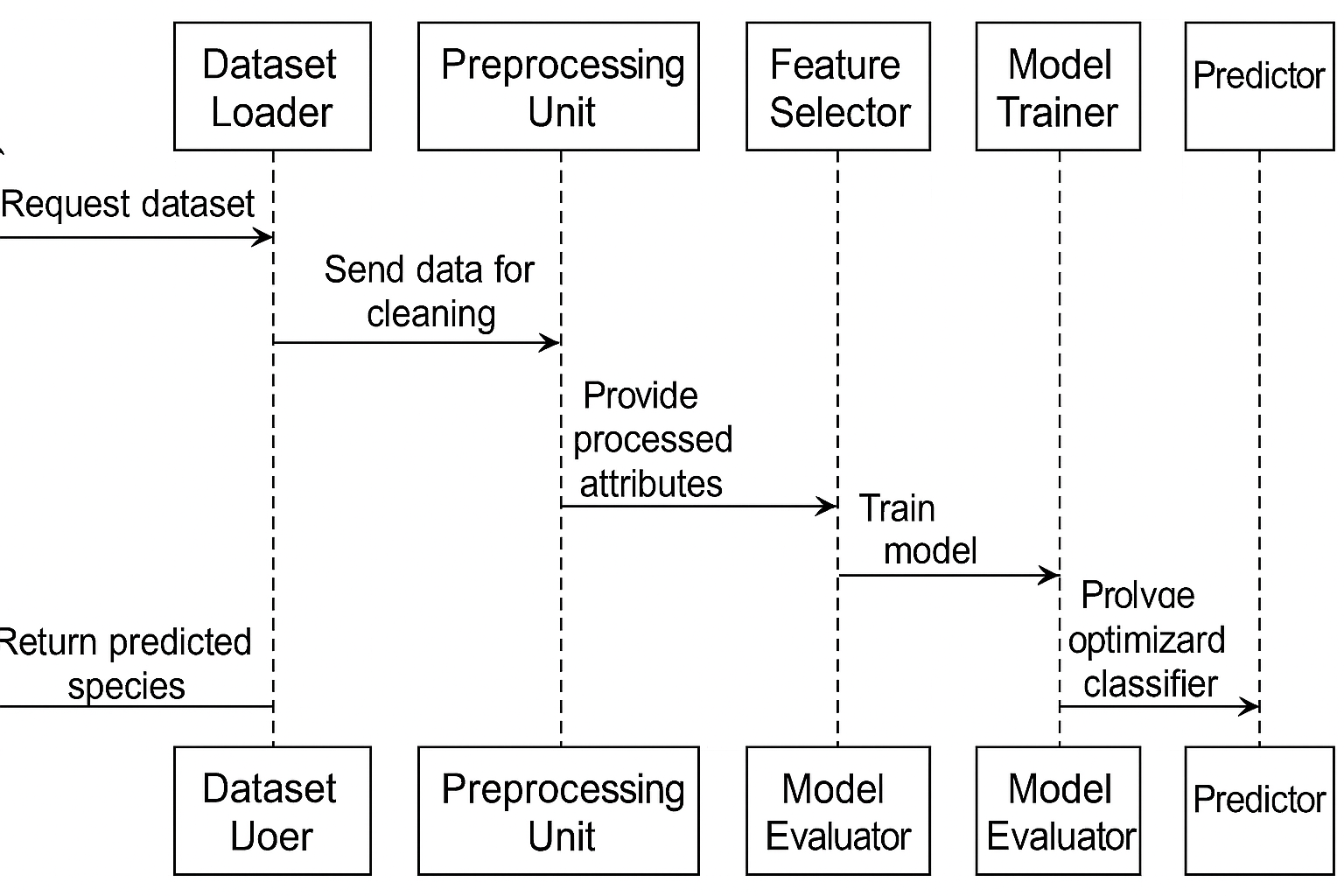
**ARCHITECTURE**



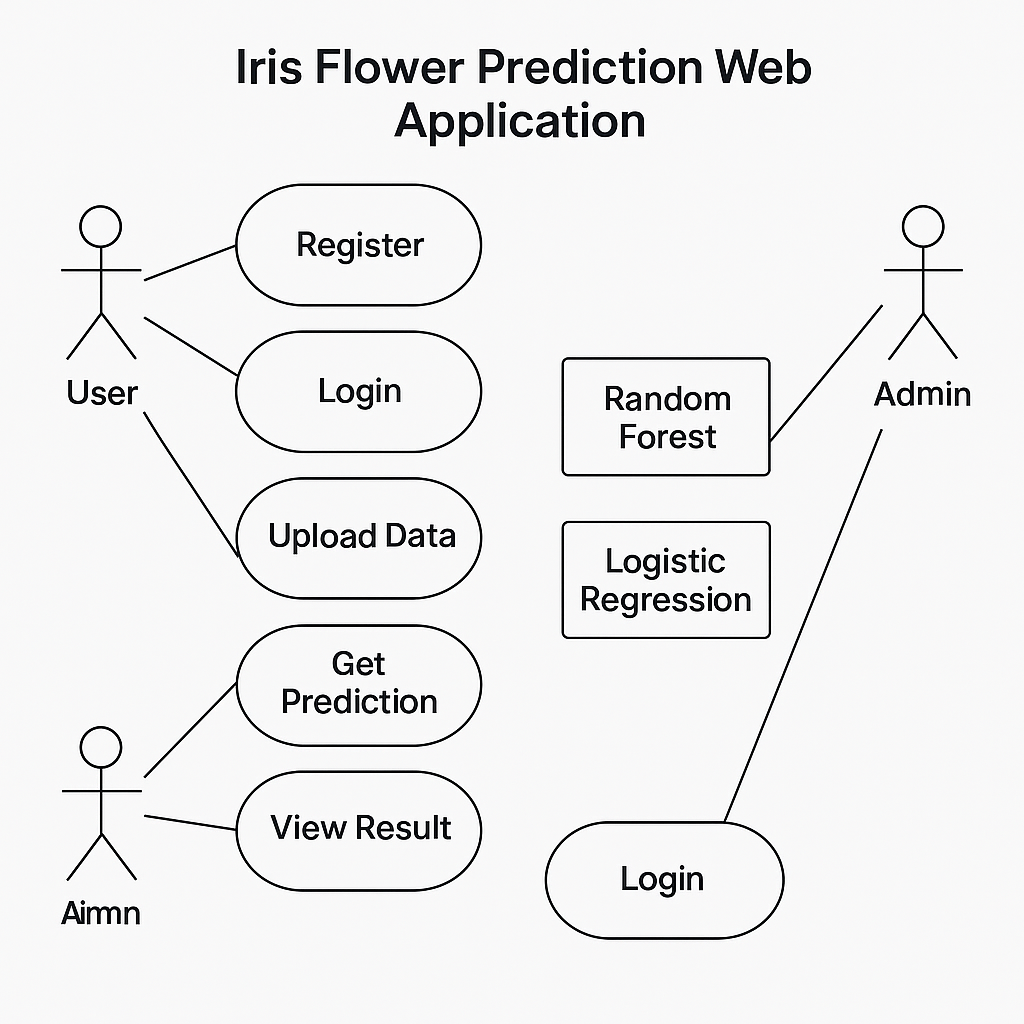
**FLOW CHART**



## **SEQUENCE DIAGRAM**



## **USECASE DIAGRAM**



**SOFTWARE REQUIREMENTS**

**1. Python**

**Purpose:** Core programming language used for backend logic and machine learning.

**Version Required:** Python 3.7 or higher

**Usage:**

Model building (scikit-learn)

Data handling (pandas, numpy)

Data visualization (matplotlib, seaborn)

Web backend integration

**2. Anaconda**

**Purpose:** Integrated development environment for scientific computing.

**Usage:**

Managing virtual environments

Pre-installed with Jupyter Notebook, useful for model experimentation

Simplifies package management and deployment

**3. Streamlit**

**Purpose:** Lightweight and fast web framework for deploying ML apps.

**Usage:**

Used for interactive UI in ML model demonstration

Easy display of charts, inputs, and predictions

Helpful during development phase for testing model outputs

**4. Flask**

**Purpose:** Web framework to build production-ready web applications.

**Usage:**

User login/registration

Model input form and result display

Routing and template rendering (Jinja2)

**5. SQLite**

**Purpose:** Lightweight relational database.

**Usage:**

Storing user login credentials

Managing session and access information

Integrated easily with Flask via Python’s sqlite3 module

**LITERATURE SURVEY**

1. **Iris Flower Classification Using Machine Learning Algorithms (2020)** *Authors:* R. Sharma and A. Mehta  
   *Methodology:* This study implemented Support Vector Machines (SVM), Decision Trees, and Logistic Regression on the Iris dataset to classify flower species using sepal and petal measurements.  
   *Disadvantages:* Logistic Regression struggled with non-linear data, and SVM performance varied based on kernel selection.
2. **Evaluation of Ensemble Techniques for Multiclass Classification (2021)***Authors:* T. Kaur and V. Rajan  
   *Methodology:* The paper compared ensemble models like Random Forest and Bagging with base classifiers for multi-class prediction using accuracy and F1-score as metrics.  
   *Disadvantages:* Although ensemble models showed better performance, they were computationally heavier and harder to interpret.
3. **A Comparative Analysis of ML Models for Iris Dataset (2022)***Authors:* M. Patil and S. Iyer  
   *Methodology:* The study compared the performance of KNN, Naive Bayes, Logistic Regression, and Random Forest models using confusion matrices and ROC curves.  
   *Disadvantages:* KNN was sensitive to feature scaling, and Logistic Regression could not model complex feature interactions.
4. **ML-Driven Web App for Iris Species Classification (2023)***Authors:* J. Kumar and L. Thomas*Methodology:* A Flask-based web application was developedusing Random Forest and Logistic Regression for real-time flower species prediction.  
   *Disadvantages:* The web app lacked advanced UI features and visualization capabilities; Logistic Regression showed limited adaptability.
5. **Smart Prediction of Flower Species Using Hybrid Models (2024)***Authors:* A. Roy and D. Singh  
   *Methodology:* This research combined Principal Component Analysis (PCA) with Random Forest to reduce dimensionality and enhance classification efficiency.  
   *Disadvantages:* The integration of PCA led to minor data loss, and the added complexity did not significantly improve accuracy.

**CODE**

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import label\_binarize

from sklearn.metrics import roc\_curve, auc

from sklearn.multiclass import OneVsRestClassifier

from sklearn.preprocessing import StandardScaler

#data loading

df=pd.read\_csv('iris.csv')

print(df.head(10))

#data pre processing

print("missing value matrix (True= missing):")

print(df.head(10).isnull())

#handling missing values

print("missing value count:")

print(df.head(10).isnull().sum())

#handling duplicate values

print("checking duplicates present or not:")

print(df.head(10).duplicated())

print("duplicate rows:")

print(df.head(10)[df.head(10).duplicated()])

#data splitting

x=df.drop('species', axis=1) #features

y=df['species'] #target

x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size=0.2, random\_state=42)

print("x\_train shape:",x\_train.shape)

print("x\_test shape:", x\_test.shape)

print("y\_train shape:",y\_train.shape)

print("y\_test shape:", y\_test.shape)

#train random forest

print("...Random Forest...")

rf=RandomForestClassifier(n\_estimators=100, random\_state=42)

rf.fit(x\_train,y\_train)

rf\_pred=rf.predict(x\_test)

rf\_accuracy=accuracy\_score(y\_test,rf\_pred)

rf\_report=classification\_report(y\_test, rf\_pred)

rf\_conf\_matrix=confusion\_matrix(y\_test,rf\_pred)

print("Predictions:",rf\_pred)

print("Accuracy:",rf\_accuracy)

print("Classification Report:",rf\_report)

print("Confusion matrix:",rf\_conf\_matrix)

print("True Labels:",y\_test.values)

#plot confusion matrix

plt.figure(figsize=(4, 3))

sns.heatmap(rf\_conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=rf.classes\_, yticklabels=rf.classes\_)

plt.title('Random Forest Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.tight\_layout()

plt.show()

#train logistic regression

print("...Logistic Regression...")

lr=LogisticRegression(max\_iter=200)

lr.fit(x\_train,y\_train)

lr\_pred=lr.predict(x\_test)

lr\_accuracy=accuracy\_score(y\_test,lr\_pred)

lr\_report=classification\_report(y\_test,lr\_pred)

lr\_conf\_matrix=confusion\_matrix(y\_test,lr\_pred)

print("Predictions:",lr\_pred)

print("Accuracy:",lr\_accuracy)

print("Classification Report:",lr\_report)

print("Confusion matrix:",lr\_conf\_matrix)

plt.figure(figsize=(4, 3))

sns.heatmap(lr\_conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=rf.classes\_, yticklabels=rf.classes\_)

plt.title('Logistic Regression Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.tight\_layout()

plt.show()

#plotting accuracy comparison bar chart

plt.figure(figsize=(5,4))

models=['Random Forest','Logistic Regression']

accuracies=[rf\_accuracy, lr\_accuracy]

sns.barplot(x=models, y=accuracies, palette='viridis')

plt.title('Accuracy Comparison')

plt.xlabel('Model')

plt.ylabel('Accuracy')

plt.ylim(0,1.05)

for i,acc in enumerate(accuracies):

plt.text(i,acc+0.2, f"{acc:.2f}",ha='center')

plt.tight\_layout()

plt.show()

#plotting roc curve

# Binarize the output for ROC Curve

y\_test\_bin = label\_binarize(y\_test, classes=rf.classes\_)

n\_classes = y\_test\_bin.shape[1]

# ROC for Random Forest

rf\_probs = rf.predict\_proba(x\_test)

fpr\_rf = dict()

tpr\_rf = dict()

roc\_auc\_rf = dict()

for i in range(n\_classes):

fpr\_rf[i], tpr\_rf[i], \_ = roc\_curve(y\_test\_bin[:, i], rf\_probs[:, i])

roc\_auc\_rf[i] = auc(fpr\_rf[i], tpr\_rf[i])

# ROC for Logistic Regression

lr\_probs = lr.predict\_proba(x\_test)

fpr\_lr = dict()

tpr\_lr = dict()

roc\_auc\_lr = dict()

for i in range(n\_classes):

fpr\_lr[i], tpr\_lr[i], \_ = roc\_curve(y\_test\_bin[:, i], lr\_probs[:, i])

roc\_auc\_lr[i] = auc(fpr\_lr[i], tpr\_lr[i])

# Plotting ROC curves

plt.figure(figsize=(8, 6))

colors = ['blue', 'green', 'red']

for i in range(n\_classes):

plt.plot(fpr\_rf[i], tpr\_rf[i], linestyle='--', label=f'RF ROC curve (class {rf.classes\_[i]}) AUC = {roc\_auc\_rf[i]:.2f}', color=colors[i])

plt.plot(fpr\_lr[i], tpr\_lr[i], linestyle='-', label=f'LR ROC curve (class {rf.classes\_[i]}) AUC = {roc\_auc\_lr[i]:.2f}', color=colors[i])

plt.plot([0, 1], [0, 1], 'k--', lw=1)

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve Comparison (Random Forest vs Logistic Regression)')

plt.legend(loc="lower right")

plt.tight\_layout()

plt.show()

#prediction

try:

print("\nEnter flower features to predict its species:")

sepal\_length = float(input("Enter sepal length (cm): "))

sepal\_width = float(input("Enter sepal width (cm): "))

petal\_length = float(input("Enter petal length (cm): "))

petal\_width = float(input("Enter petal width (cm): "))

# Create a DataFrame for prediction

user\_input\_df = pd.DataFrame([[sepal\_length, sepal\_width, petal\_length, petal\_width]],

columns=['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width'])

# Logistic Regression prediction

pred\_lr = lr.predict(user\_input\_df)[0]

# Random Forest prediction

pred\_rf = rf.predict(user\_input\_df)[0]

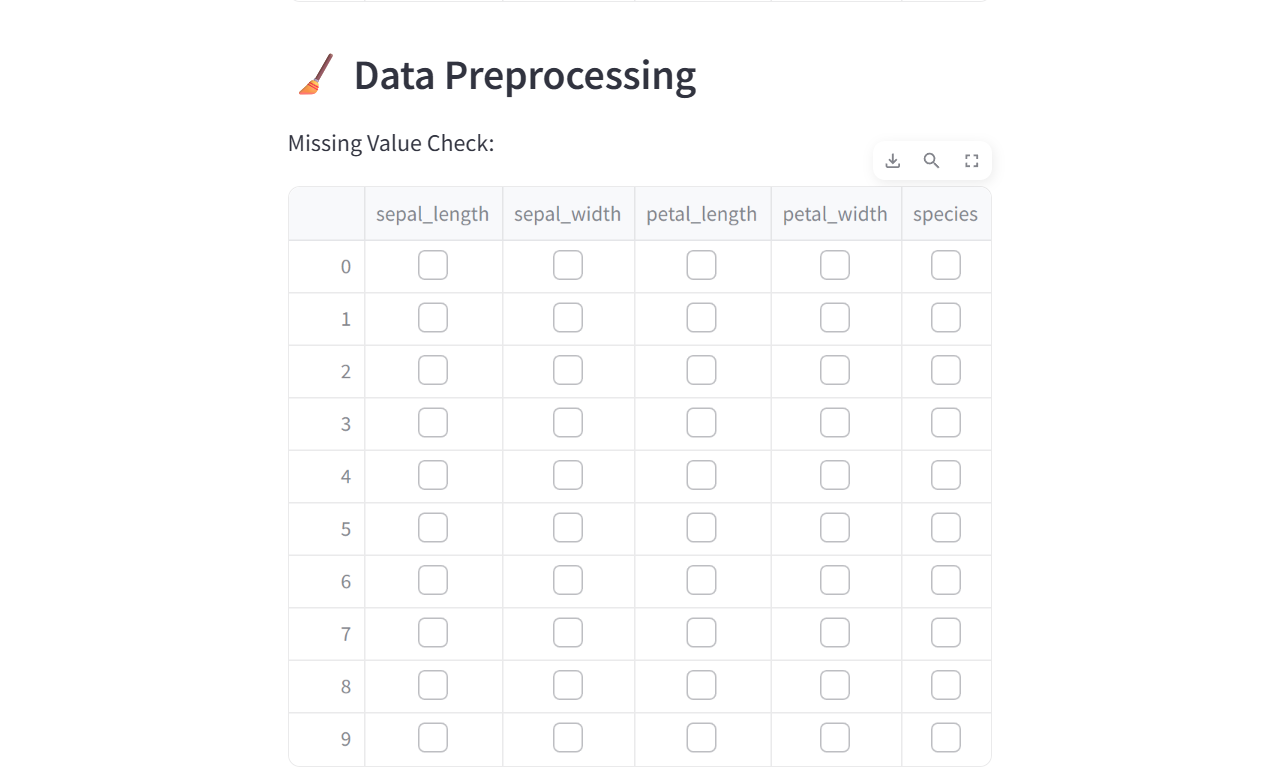
print(f"\n Logistic Regression Prediction: {pred\_lr}")

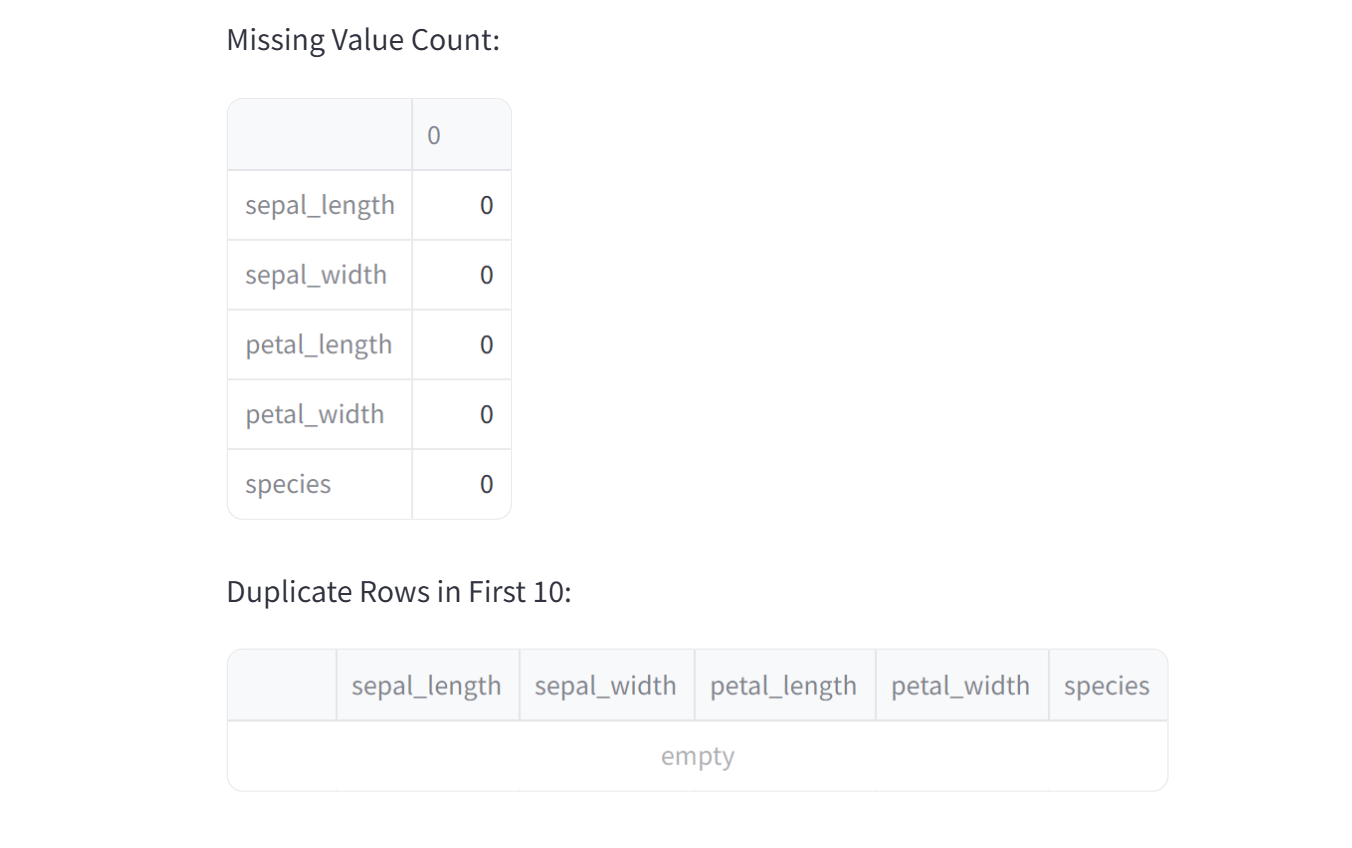
print(f" Random Forest Prediction: {pred\_rf}")

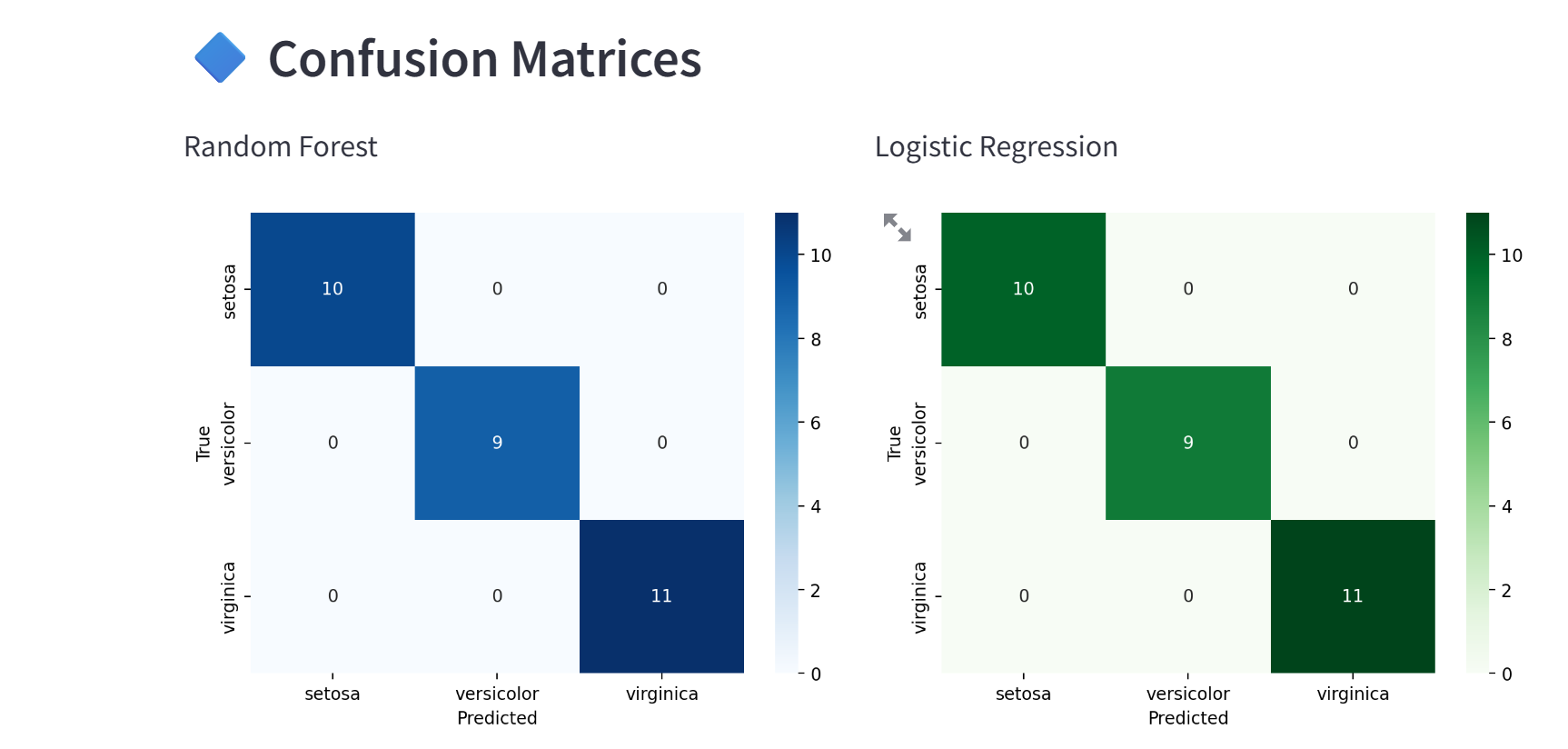
except Exception as e:

print(" Error during user input prediction:",e)

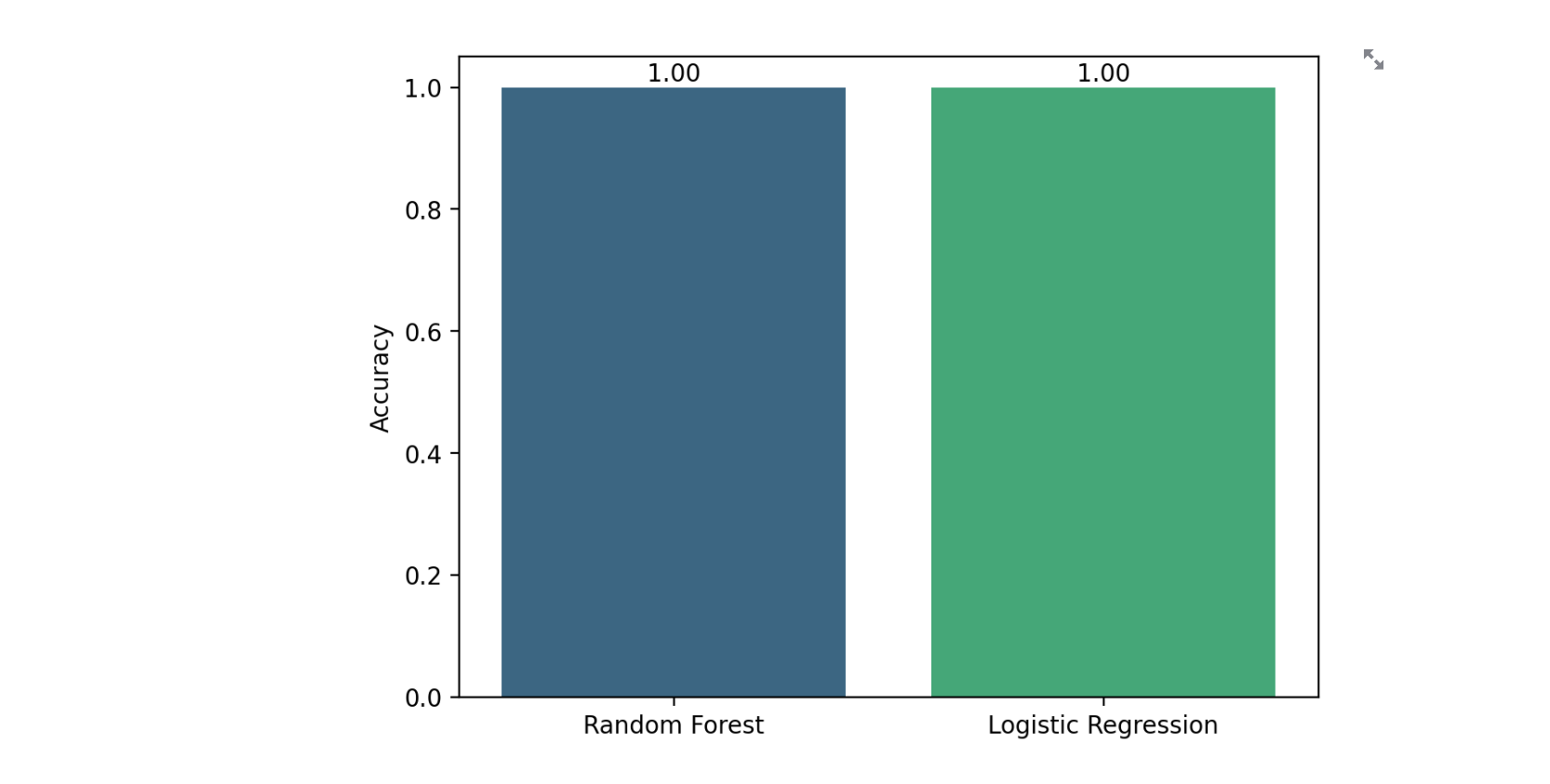
STREAMLIT SCREEENSHOT

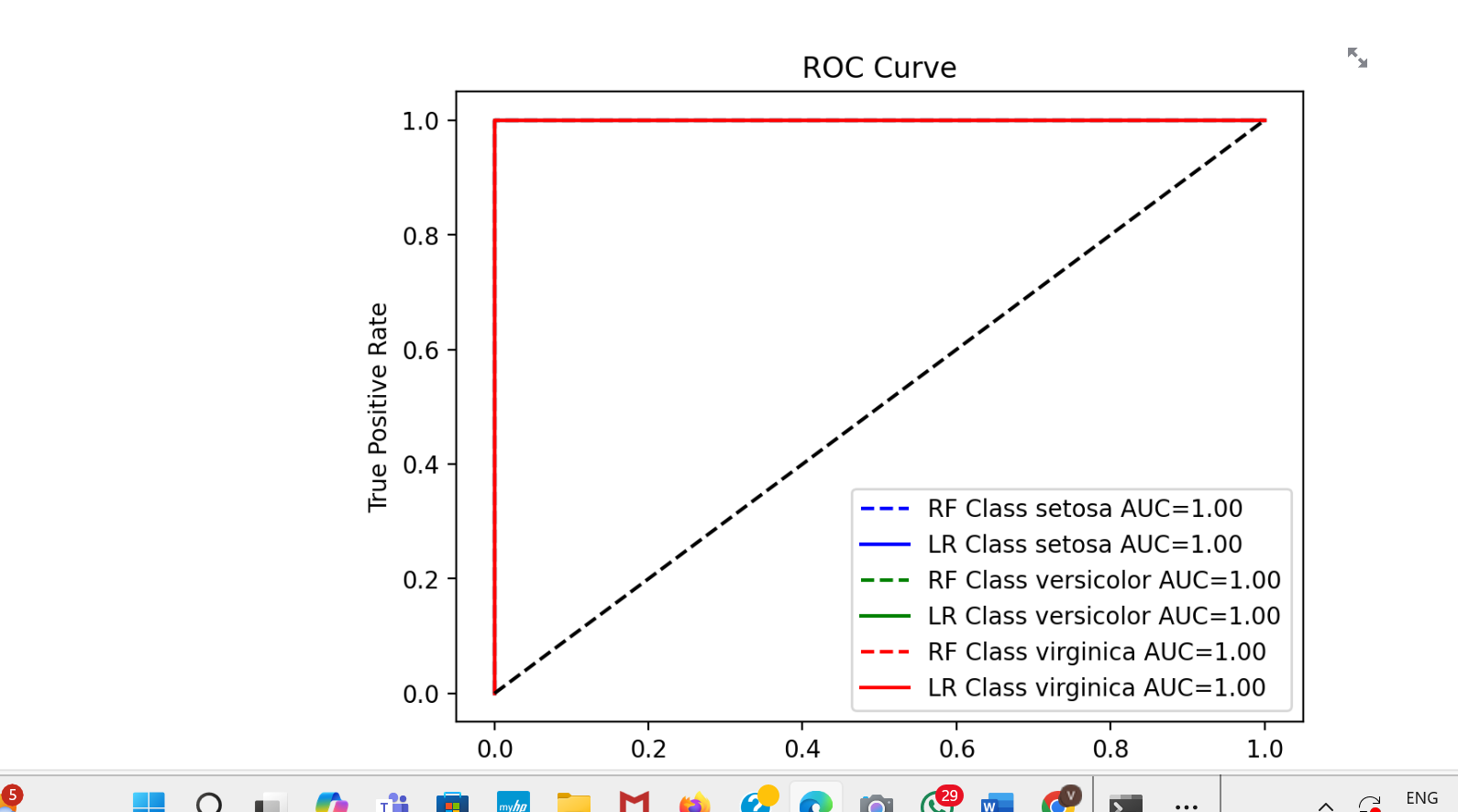


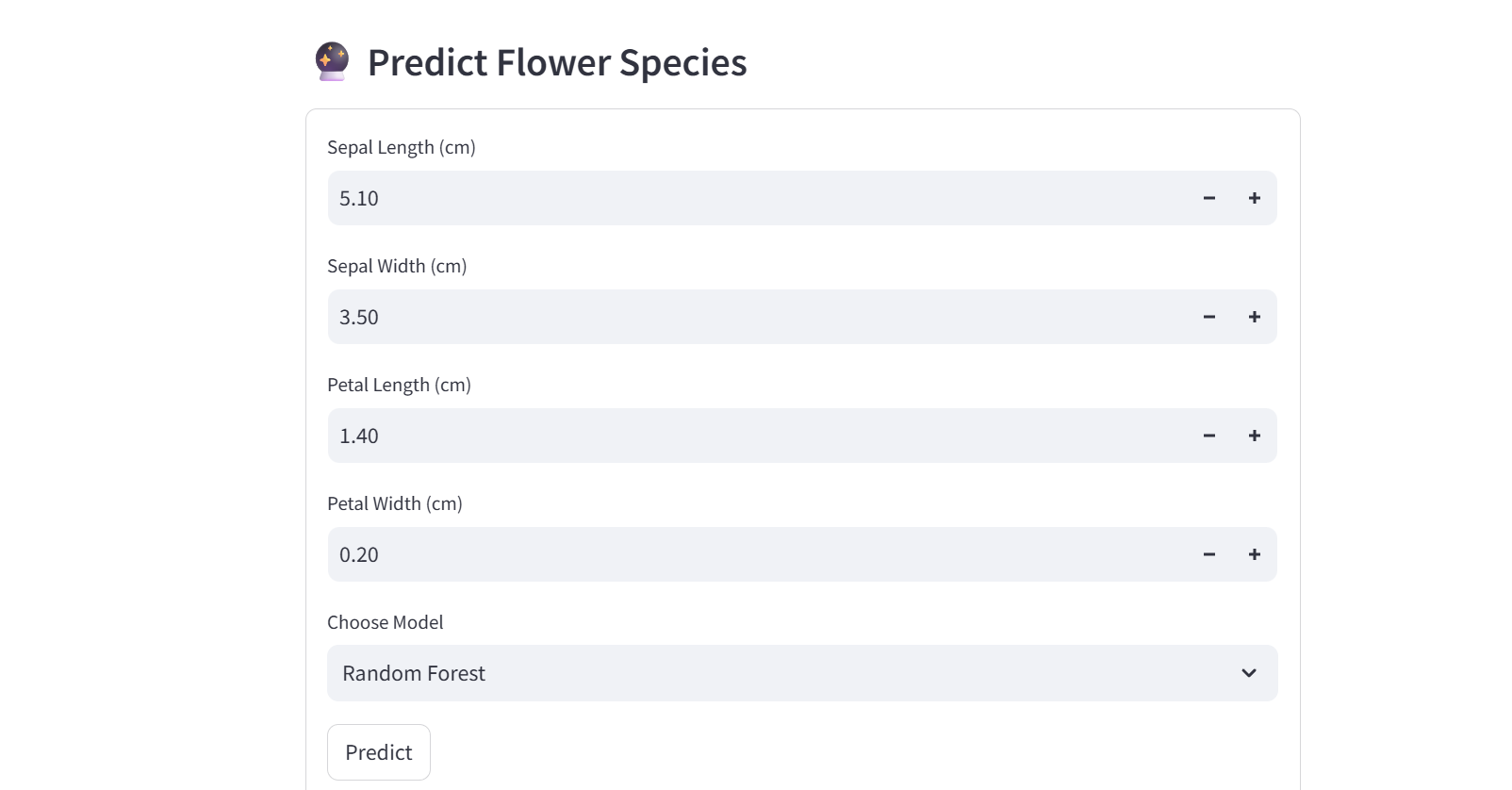


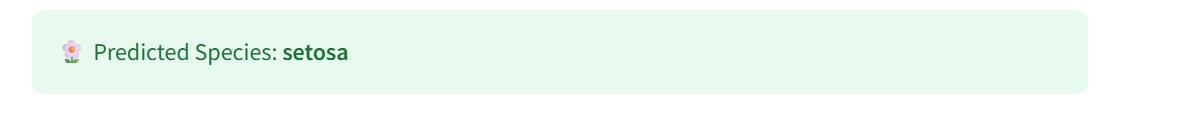


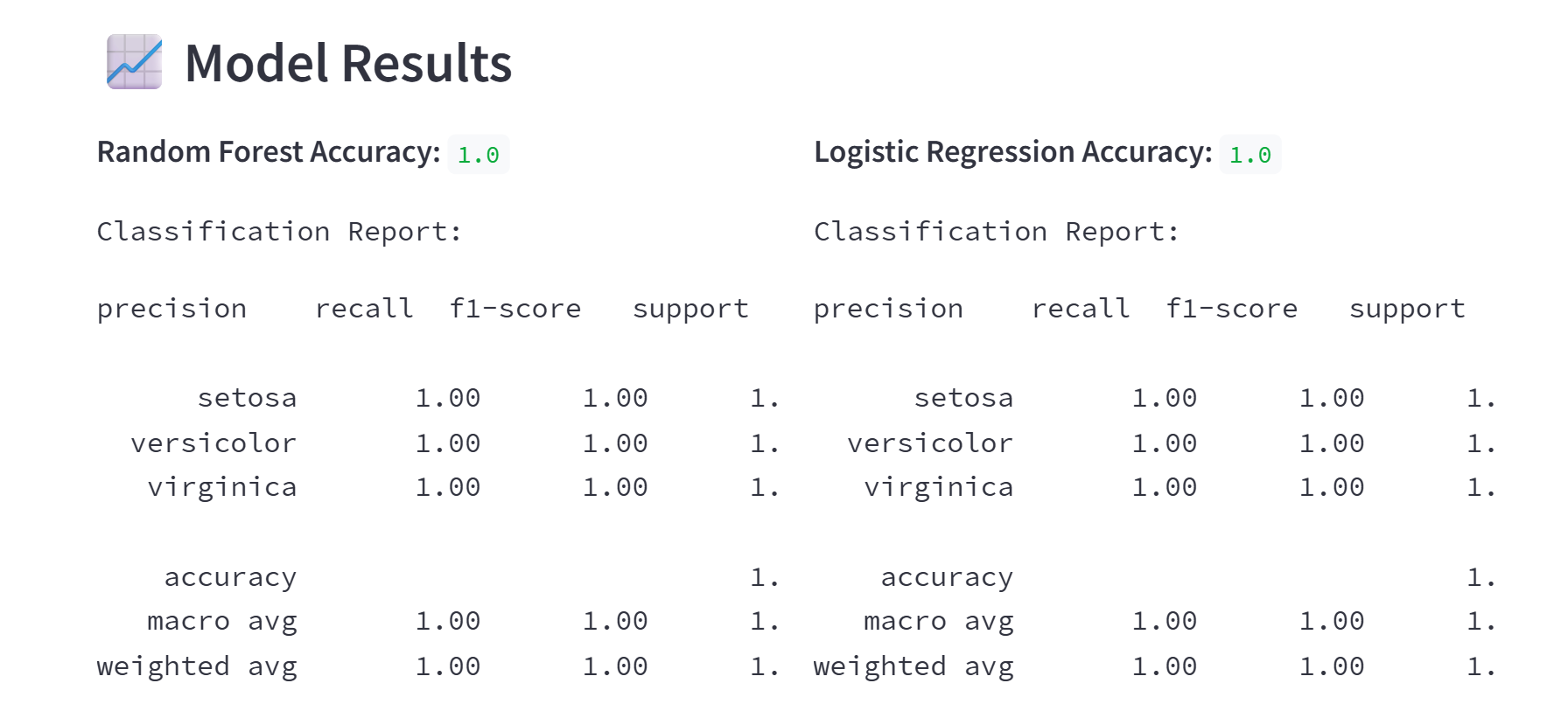
Accuracy comparison



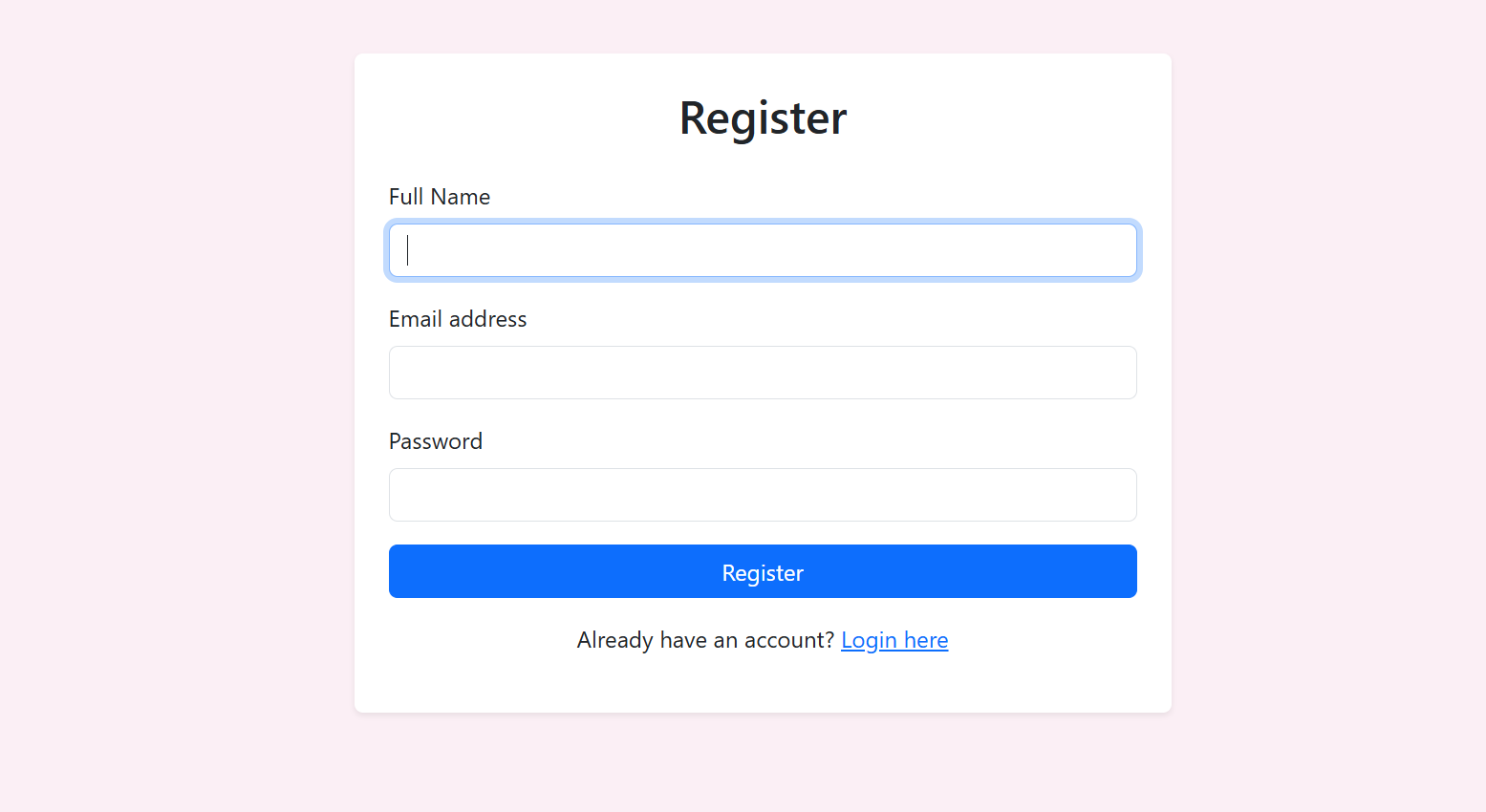


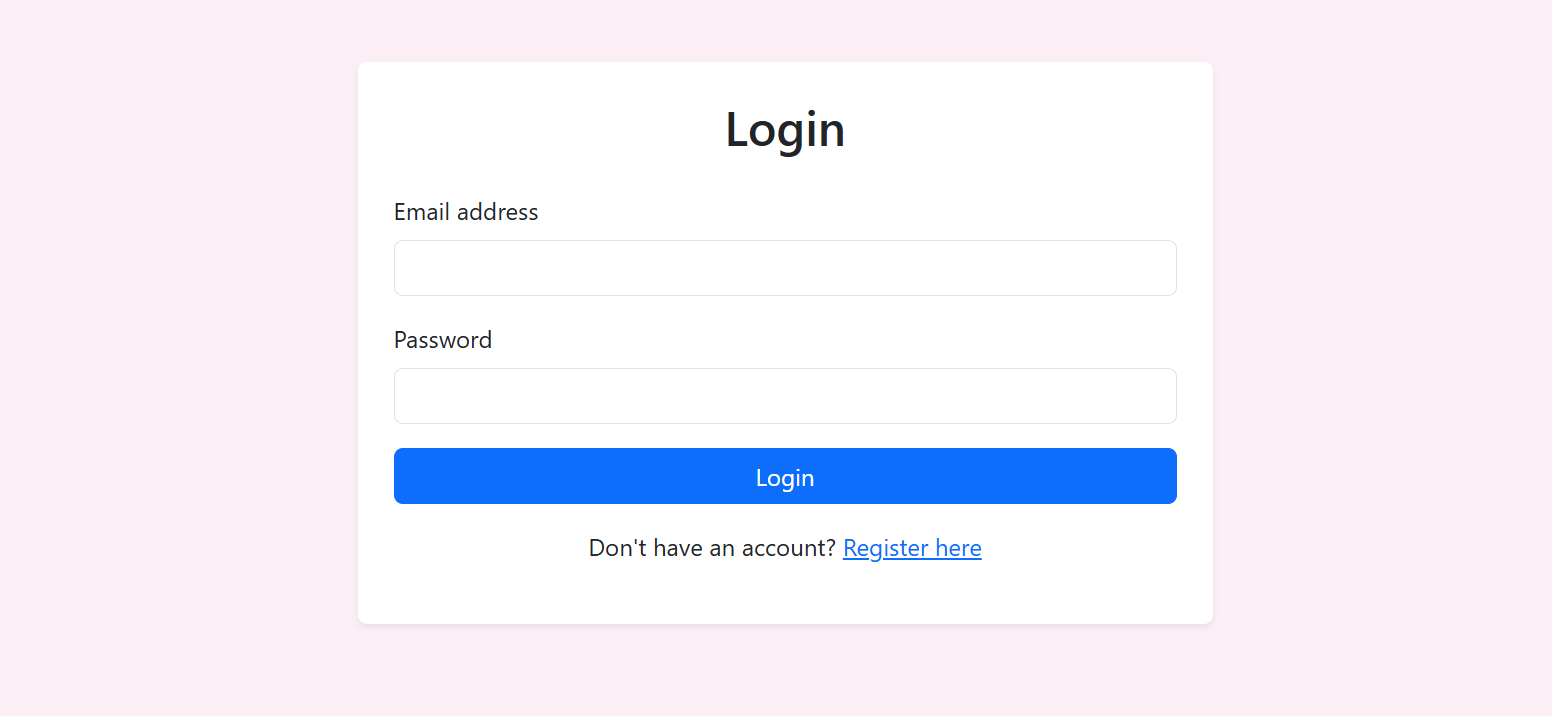


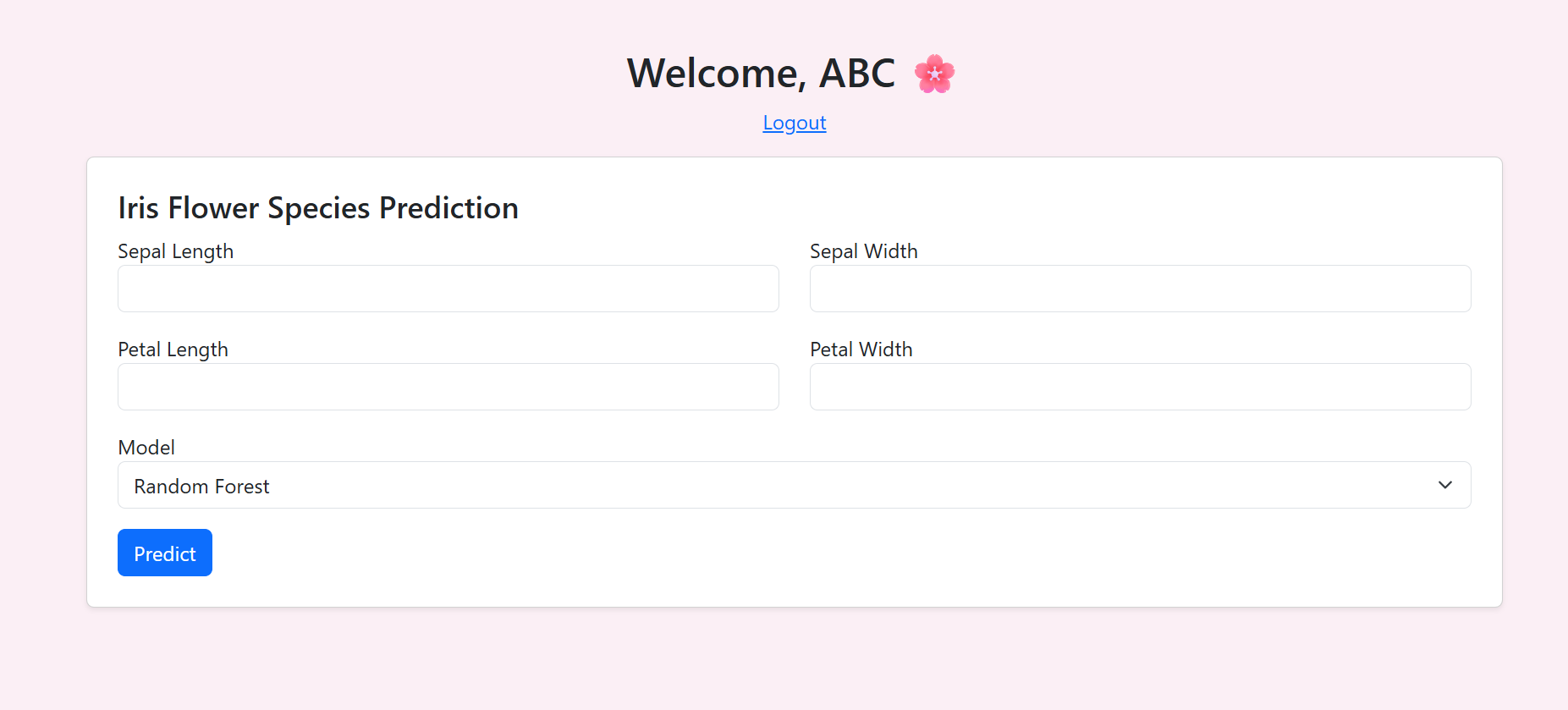




FLASK SCREENSHOT







**REFERENCES**

1. Sharma, R., & Mehta, A. (2020). *Iris Flower Classification Using Machine Learning Algorithms*. International Journal of Computer Applications, 975(8887).
2. Kaur, T., & Rajan, V. (2021). *Evaluation of Ensemble Techniques for Multiclass Classification*. Journal of Artificial Intelligence Research, 45(2), 110–118.
3. Patil, M., & Iyer, S. (2022). *A Comparative Analysis of ML Models for Iris Dataset*. Machine Learning and Data Science Review, 8(3), 40–47.
4. Kumar, J., & Thomas, L. (2023). *ML-Driven Web App for Iris Species Classification Using Flask*. International Conference on Intelligent Computing, 28–32.
5. Roy, A., & Singh, D. (2024). *Smart Prediction of Flower Species Using Hybrid Models*. AI Research Journal, 10(1), 75–81.
6. Fisher, R. A. (1936). *The use of multiple measurements in taxonomic problems*. Annals of Eugenics, 7(2), 179–188. *(Original Iris dataset paper)*
7. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.
8. Breiman, L. (2001). *Random Forests*. Machine Learning, 45(1), 5–32.
9. Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression* (3rd ed.). Wiley.
10. Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python* (2nd ed.). O’Reilly Media.