SUMMER INTERNSHIP REPORT ON IRIS FLOWER CLASSIFICATION USING FLASK AND MACHINE LEARNING

At

Plasmid

IN PARTIAL FULFILLMENT FOR

THE AWARDOF THE DEGREE OF BCA

Teerthanker Mahaveer University, Moradabad.

Uttar Pradesh

Submitted By:

ANKIT SAINI

TCA2201092

5th Semester 3rd year

Submitted To:

Mr. Ajay Rastogi

Under Supervision of

TO

Mr. AJAY RASTOGI

Declaration

I ANKIT SAINI (TCA2201092), Student of BCA, 5th Semester, studying at Teerthanker Mahaveer University, Moradabad (UP), hereby declare that the Internship Report on "**IRIS FLOWER CLASSIFICATION USING FLASK AND MACHINE LEARNING**" submitted in partial fulfillment of BCA is the innovative exertion showed thru us, is a record of an original work done by me under the guidance of **MR.AJAY RASTOGI.**

The description of internship Report is not presence succumbed to any other University for award of any other Degree, Diploma and Fellowship.
Place: Moradabad

Submitted By:

ANKIT SAINI(TCA2201092)
Date:- 28 Dec 2024

CERTIFICATE

PROJECT TITLE

Iris Flower Classification Using Flask and Machine Learning

Problem Statement

The classification of Iris flowers based on their physical characteristics has been a benchmark problem in machine learning for decades. This project aims to address the challenge of automating the classification process for the three Iris species: Setosa, Versicolor, and Virginica. Using machine learning, we aim to develop a robust and efficient model that can predict the species based on sepal and petal dimensions, making the process fast, accurate, and user-friendly.

Project Description

This project focuses on developing a machine learning model to classify Iris flower species and deploying it using a Flask-based web application. The following steps were undertaken:

Scope of the Work

- **In-Scope:**
    - Development of a machine learning model for classification.
    - Deployment of the model using Flask.
    - Creation of a user-friendly web interface for predictions.

- **Out-of-Scope:**
    - Integration with external databases.
    - Advanced visualization dashboards.

Project Modules

- **Data Preparation:**
    - Loading and preprocessing the Iris dataset.

- **Model Development:**
    - Training and evaluation of the classification model.

- **Web Application:**
    - Creation of a Flask application for user interaction.
    - HTML and CSS for front-end design.

Context Diagram (High Level)

- **Inputs:** Sepal length, Sepal width, Petal length, Petal width.
- **Outputs:** Predicted Iris species.

Implementation Methodology

1. **Data Preparation:**
    - Utilized the Iris dataset from the UCI Machine Learning Repository.
    - Split the data into training and testing sets.

2. **Model Training:**
    - Trained a classification model using Scikit-learn's DecisionTreeClassifier.
    - Evaluated model accuracy and tuned hyperparameters.

3. **Web Application Development:**
    - Used Flask for the backend to serve predictions.
    - Designed a responsive front-end interface using HTML and CSS.

4. **Deployment:**
    - Packaged the trained model using Pickle.
    - Integrated the model into the Flask application.

Technologies to be Used

Software Platform

- **Front-end:** HTML, CSS
- **Back-end:** Python, Flask
- **Machine Learning Library:** Scikit-learn

Hardware Platform

- RAM: 8GB
- Hard Disk: 256GB SSD
- Operating System: Windows 10 or Ubuntu
- Editor: Visual Studio Code
- Browser: Chrome/Edge

Tools, if any

- Pickle for model serialization.
- Jupyter Notebook for model development and analysis.

Advantages of this Project

- Simplifies the process of Iris species classification.
- Reduces manual effort and improves accuracy.
- Provides a user-friendly interface for predictions.

Assumptions, if any

None.

Future Scope and Further Enhancement of the Project

- Integration with mobile applications for wider accessibility.
- Use of cloud platforms for scalability.
- Incorporation of additional datasets to expand classification capabilities.

Project Repository Location

Repository Name: Iris_Classification_Project

- **Source Code Location:**
- **Documentation Location:**

Definitions, Acronyms, and Abbreviations

- **Flask:** A micro web framework for Python.
- **Scikit-learn:** A machine learning library in Python.
- **Pickle:** A Python module for serializing objects.

- **UCI:** University of California, Irvine.

Conclusion

This project successfully demonstrates the use of machine learning to classify Iris flowers with high accuracy. The deployment using Flask ensures ease of access and usability, making it suitable for real-world applications. Future enhancements can further improve the scope and functionality of the system.

References

1. Fisher, R. A. "The Use of Multiple Measurements in Taxonomic Problems." 1936.
2. Scikit-learn Documentation. https://scikit-learn.org
3. Flask Documentation. https://flask.palletsprojects.com

Files

deploy.py

```
from flask import Flask, render_template, request
import pickle

app = Flask(__name__)
# load the model
model = pickle.load(open('iris01_model.sav', 'rb'))

@app.route('/')
def home():
    result = ''
    return render_template('index.html', **locals())



@app.route('/predict', methods=['POST', 'GET'])
def predict():
    sepal_length = float(request.form['sepal_length'])
    sepal_width = float(request.form['sepal_width'])
    petal_length = float(request.form['petal_length'])
    petal_width = float(request.form['petal_width'])
    result = model.predict([[sepal_length, sepal_width, petal_length, petal_width]])[0]
    return render_template('index.html', **locals())

if __name__ == '__main__':
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Iris Flower Prediction</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600;700&display=swap"
rel="stylesheet">
</head>
<body>
    <div class="container">
        <h1>Iris Predictor</h1>
        <form action="/predict" method="POST">
            <label for="sepal_length">Sepal Length</label>
            <input type="text" id="sepal_length" name="sepal_length" placeholder="Enter Sepal Length"
required>

            <label for="sepal_width">Sepal Width</label>
            <input type="text" id="sepal_width" name="sepal_width" placeholder="Enter Sepal Width"
required>

            <label for="petal_length">Petal Length</label>
            <input type="text" id="petal_length" name="petal_length" placeholder="Enter Petal Length"
required>

            <label for="petal_width">Petal Width</label>
            <input type="text" id="petal_width" name="petal_width" placeholder="Enter Petal Width"
required>

            <button type="submit">Predict</button>
        </form>
        {% if result %}
        <h3>Prediction Result: {{ result }}</h3>
```

```
        {% endif %}
    </div>
</body>
</html>
```

Style.css

```
/* Apply a red background to the entire body */
body {
    background: linear-gradient(135deg, #ff0000, #ff4e50);
    font-family: 'Poppins', sans-serif;
    color: #333;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    overflow: hidden;
}

/* Card-style container */
.container {
    width: 400px;
    padding: 20px 30px;
    background: white;
    border-radius: 15px;
    box-shadow: 0 8px 20px rgba(0, 0, 0, 0.3);
    text-align: center;
    animation: fadeIn 1s ease-in-out;
```

```
}

/* Heading styles */
h1 {
    font-size: 2rem;
    color: #ff0000;
    margin-bottom: 15px;
    font-weight: 700;
}

/* Form labels */
form label {
    display: block;
    margin: 10px 0 5px;
    font-size: 14px;
    font-weight: 500;
    color: #555;
}

/* Input fields */
input[type="text"] {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ddd;
    border-radius: 5px;
    font-size: 14px;
    transition: border-color 0.3s;
}

input[type="text"]:focus {
```

```
    outline: none;
    border-color: #ff0000;
    box-shadow: 0 0 8px rgba(255, 0, 0, 0.5);
}

/* Submit button */
button {
    width: 100%;
    background: linear-gradient(90deg, #ff6f61, #ff0000);
    color: white;
    font-size: 16px;
    font-weight: 600;
    padding: 12px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background 0.3s, transform 0.2s;
}

button:hover {
    background: linear-gradient(90deg, #ff0000, #d00000);
    transform: scale(1.05);
}

/* Prediction result styling */
h3 {
    margin-top: 20px;
    color: #333;
    font-weight: 600;
    background: linear-gradient(90deg, #ff4e50, #d00000);
    padding: 10px;
```

```
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
}

/* Animation */
@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(-20px);
    }

    to {
        opacity: 1;
        transform: translateY(0);
    }
}
```

Jupyter file

```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')


df = pd.read_csv('Iris.csv')
df.head()
```

```
df = df.drop(columns = ['Id'])
df.head()


# to display stats about data
df.describe()


# to basic info about datatype
df.info()


# to display no. of samples on each class
df['Species'].value_counts()


# check for null values
df.isnull().sum()


# histograms
df['SepalLengthCm'].hist()


df['SepalWidthCm'].hist()
```

```
df['PetalLengthCm'].hist()


df['PetalWidthCm'].hist()


# scatterplot
colors = ['red', 'orange', 'blue']
species = ['Iris-virginica','Iris-versicolor','Iris-setosa']


for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['SepalLengthCm'], x['SepalWidthCm'], c = colors[i], label=species[i])
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.legend()


for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['PetalLengthCm'], x['PetalWidthCm'], c = colors[i], label=species[i])
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
plt.legend()


for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['SepalLengthCm'], x['PetalLengthCm'], c = colors[i], label=species[i])
```

```
plt.xlabel("Sepal Length")
plt.ylabel("Petal Length")
plt.legend()



for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['SepalWidthCm'], x['PetalWidthCm'], c = colors[i], label=species[i])
plt.xlabel("Sepal Width")
plt.ylabel("Petal Width")
plt.legend()



df.corr()



corr = df.corr()
fig, ax = plt.subplots(figsize=(5,4))
sns.heatmap(corr, annot=True, ax=ax, cmap = 'coolwarm')



from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()



df['Species'] = le.fit_transform(df['Species'])
df.head()
```

```
from sklearn.model_selection import train_test_split
# train - 70
# test - 30
X = df.drop(columns=['Species'])
Y = df['Species']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30)



# logistic regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()



# model training
model.fit(x_train, y_train)



# print metric to get performance
print("Accuracy: ",model.score(x_test, y_test) * 100)



# knn - k-nearest neighbours
from sklearn.neighbors import KNeighborsClassifier
```

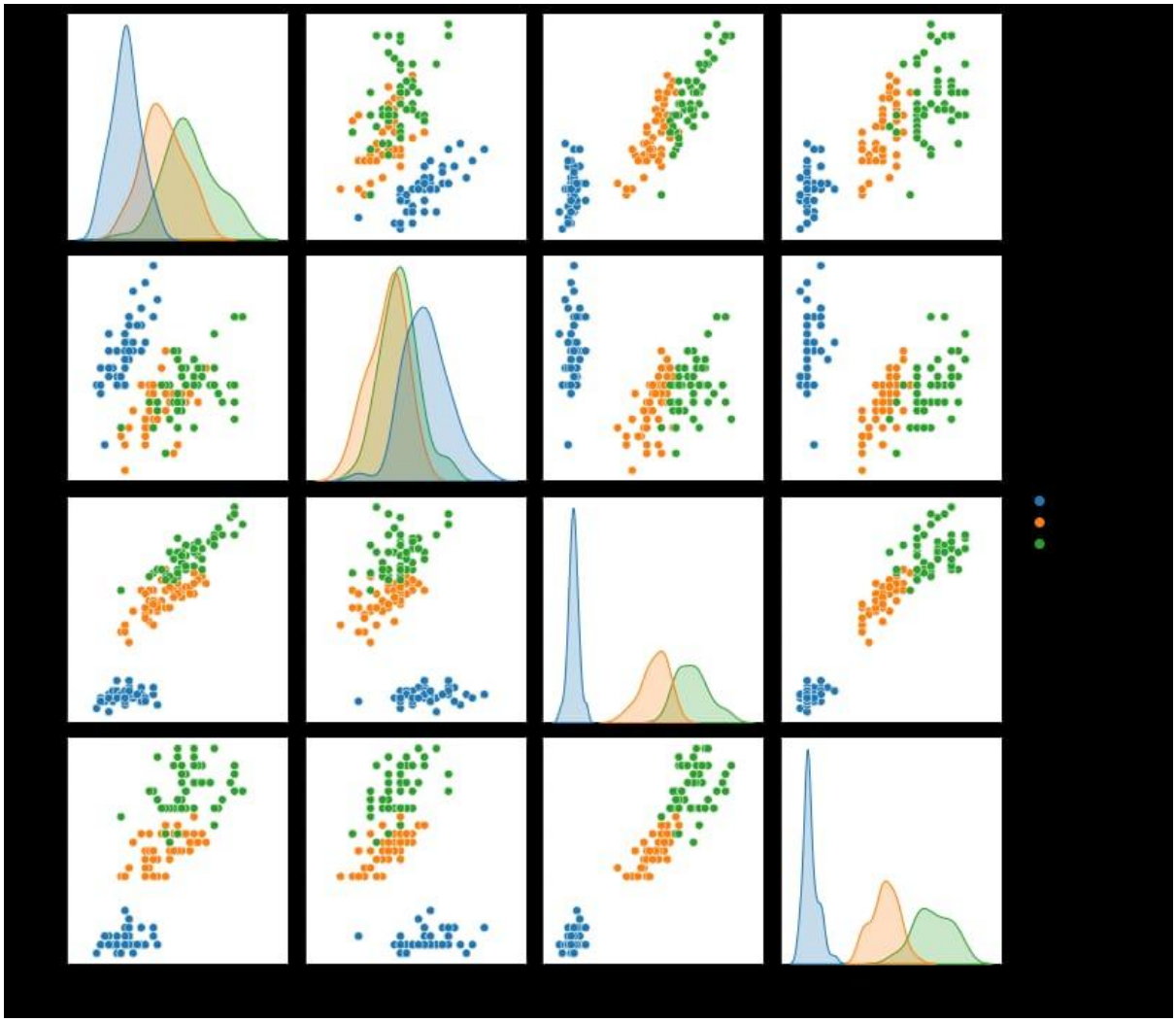
```
model = KNeighborsClassifier()



model.fit(x_train, y_train)



# print metric to get performance
print("Accuracy: ",model.score(x_test, y_test) * 100)



# decision tree
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()



model.fit(x_train, y_train)



# print metric to get performance
print("Accuracy: ",model.score(x_test, y_test) * 100)
```

Appendix

A: Data Flow Diagram (DFD)

High-Level DFD:

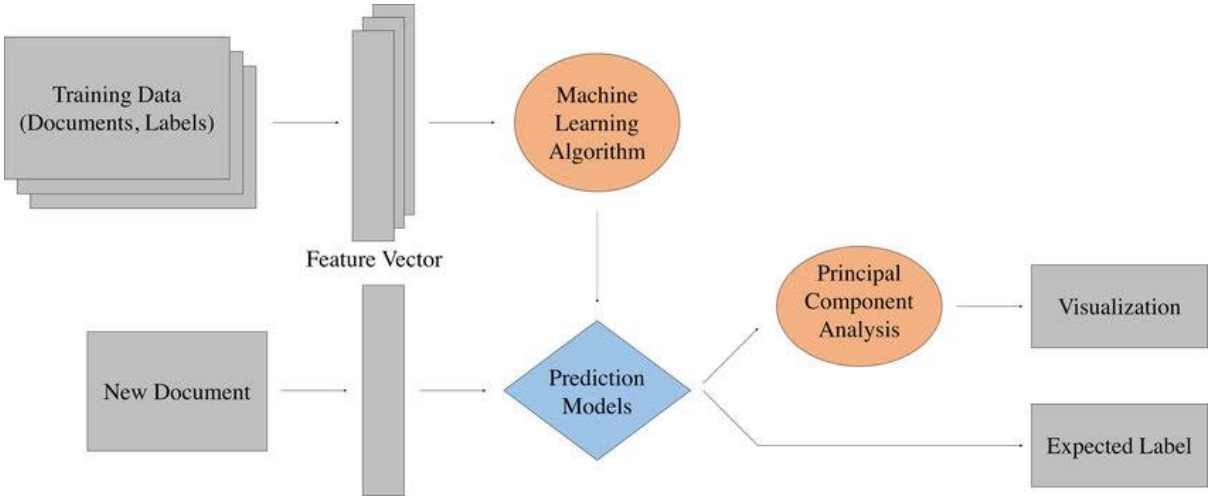
- **Input:** User inputs (Sepal length, Sepal width, Petal length, Petal width).
- **Process:** Prediction model processes the input.
- **Output:** Predicted Iris species.



B: Entity Relationship Diagram (ERD)

ERD:

- Entities: Input features (Sepal, Petal dimensions), Prediction Model.
- Relationships: User inputs are processed by the model to generate predictions.

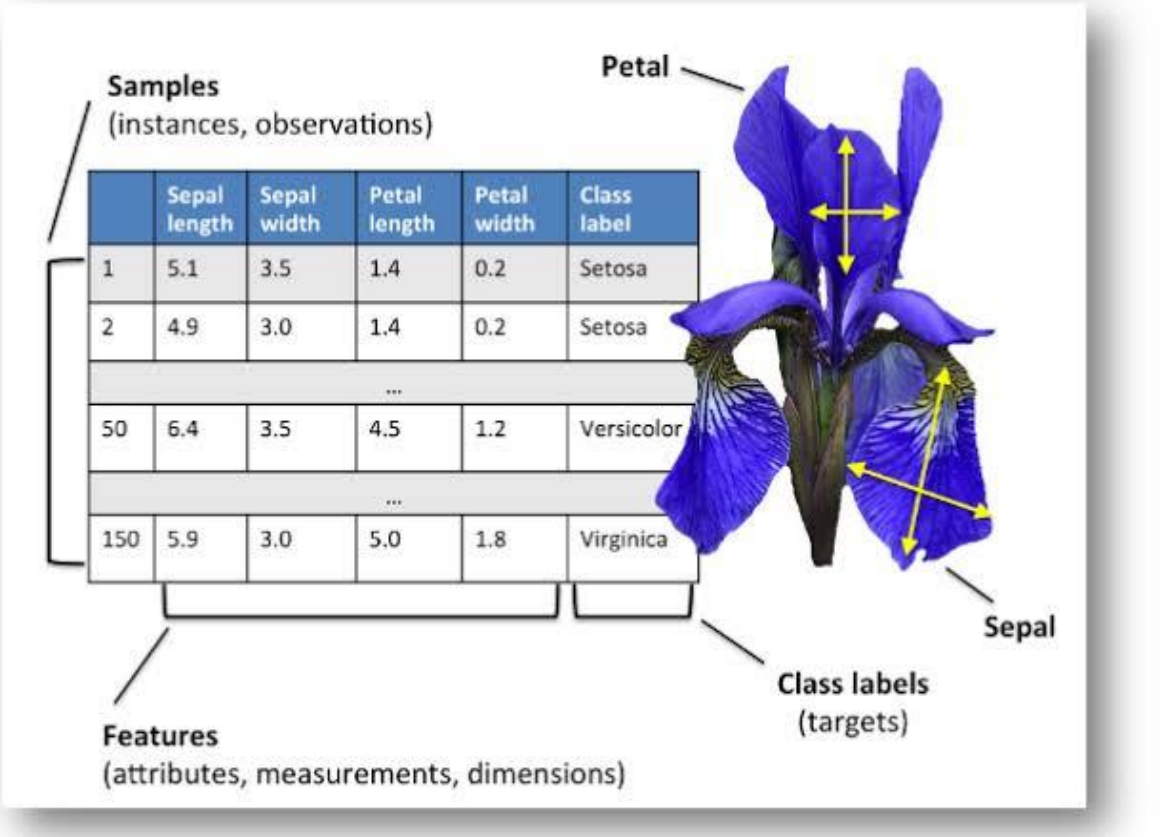


C: Use Case Diagram (UCD)

Use Case:

- Actor: User

- System: Flask web application
- Interaction: User submits input values, system returns prediction.



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

D: Data Dictionary (DD)

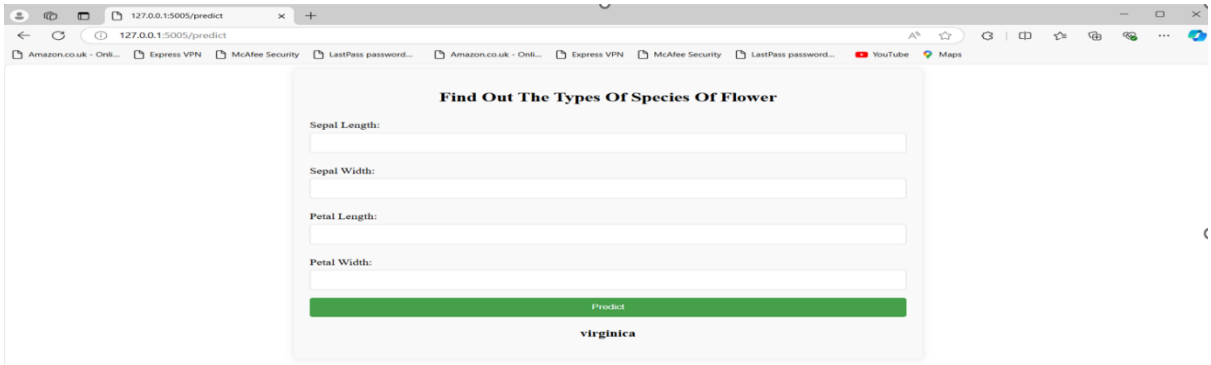
Example:

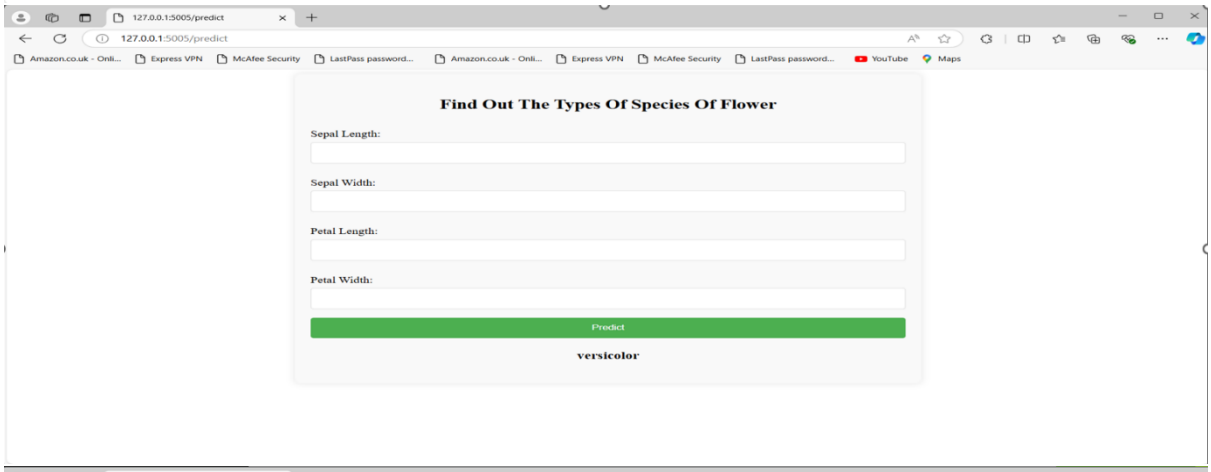
Field Name	Data Type	Description
sepal_length	float	Length of the sepal in cm
sepal_width	float	Width of the sepal in cm
petal_length	float	Length of the petal in cm
petal_width	float	Width of the petal in cm
predicted_class	string	Predicted Iris species

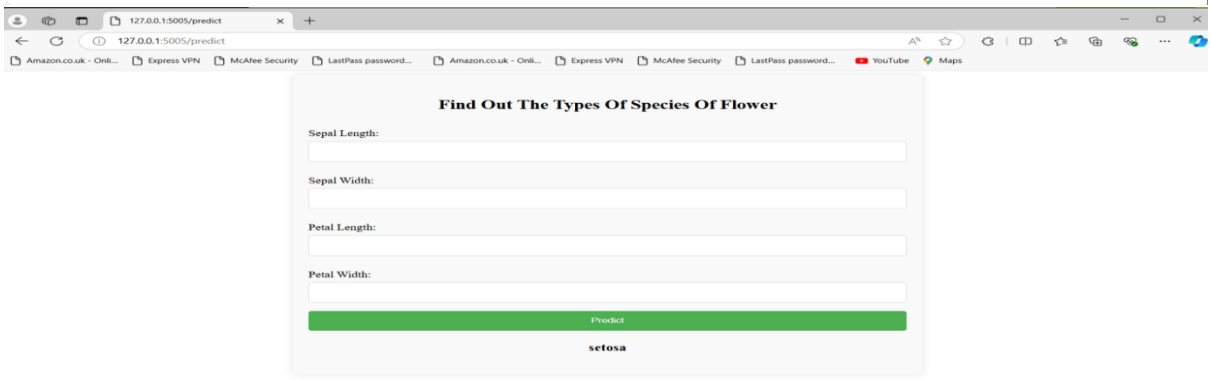
E: Screen Shots

Example Screens:

1. **Prediction Page:**
    - Displays the prediction result.


127.0.0.1:5005/predict
Find Out The Types Of Species Of Flower
Sepal Length:
Sepal Width:
Petal Length:
Petal Width:
Predict
virginica


127.0.0.1:5005/predict
Find Out The Types Of Species Of Flower
Sepal Length:
Sepal Width:
Petal Length:
Petal Width:
Predict
versicolor


127.0.0.1:5005/predict
Find Out The Types Of Species Of Flower
Sepal Length:
Sepal Width:
Petal Length:
Petal Width:
Predict
setosa