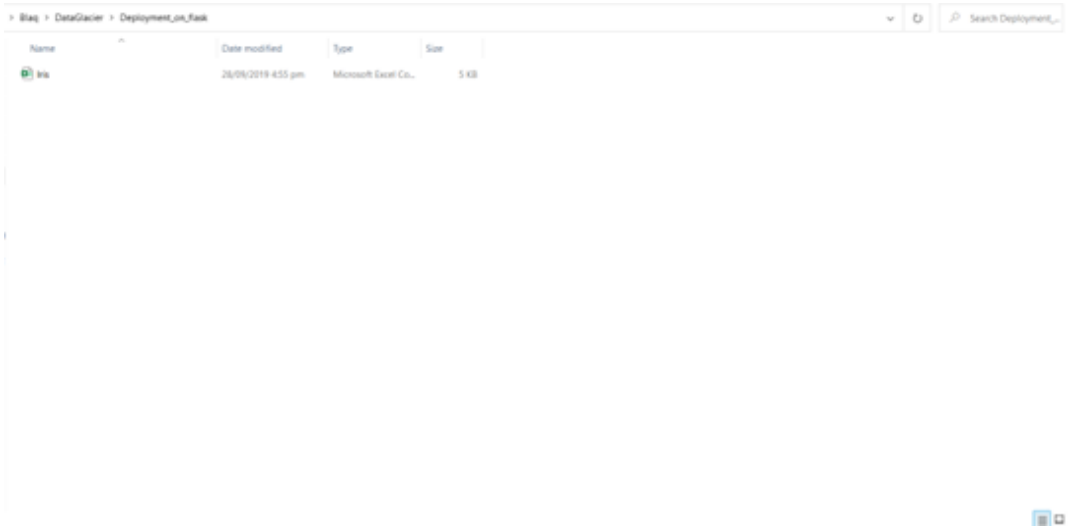
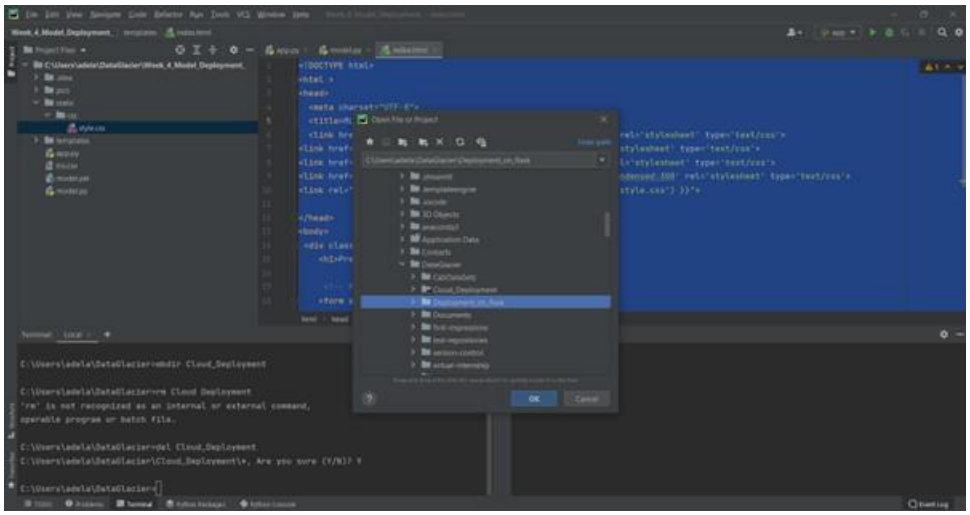
Name:	Babatunde Adelalu
Batch Code:	LISUM10: 30
Submission Date:	06-07-2022
Submission To:	Data Glacier

Create folder

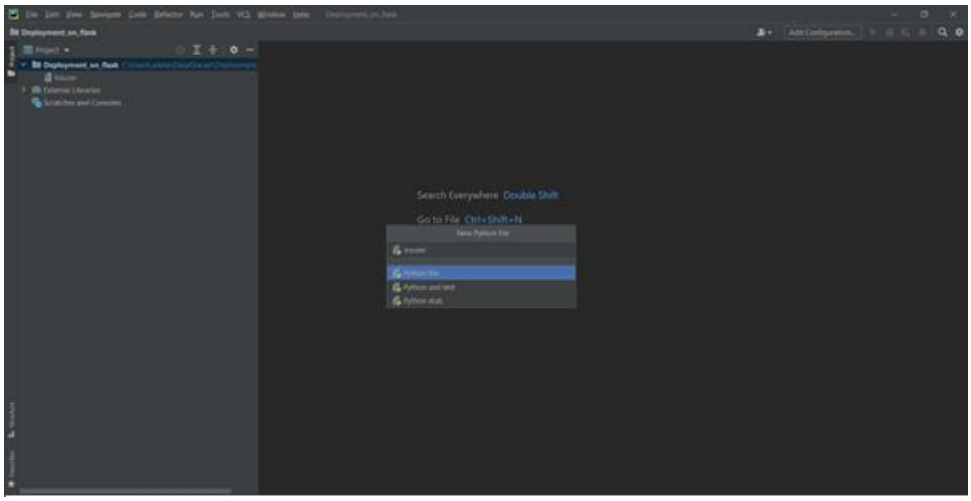


MODEL CREATION

Open folder in IDE



Create model



Import Libraries

```
# Import libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import pickle
```

Import and print dataset columns

```
# Read dataset
iris = pd.read_csv("Iris.csv")
print(iris.columns)          # Print all columns of dataset
print(iris.head())           # Print top 5 rows
```

Define Target feature

```
# Define model features y: Target variable
y = iris['species']
iris.drop(columns='species',inplace=True)
X = iris[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
print('y: ', y.shape, 'X: ', X.shape)
```

Train and print Model accuracy

```
# Training the model
x_train,x_test,y_train,y_test = train_test_split(X,y, test_size=0.3)
model = LogisticRegression()
model.fit(x_train,y_train)
```

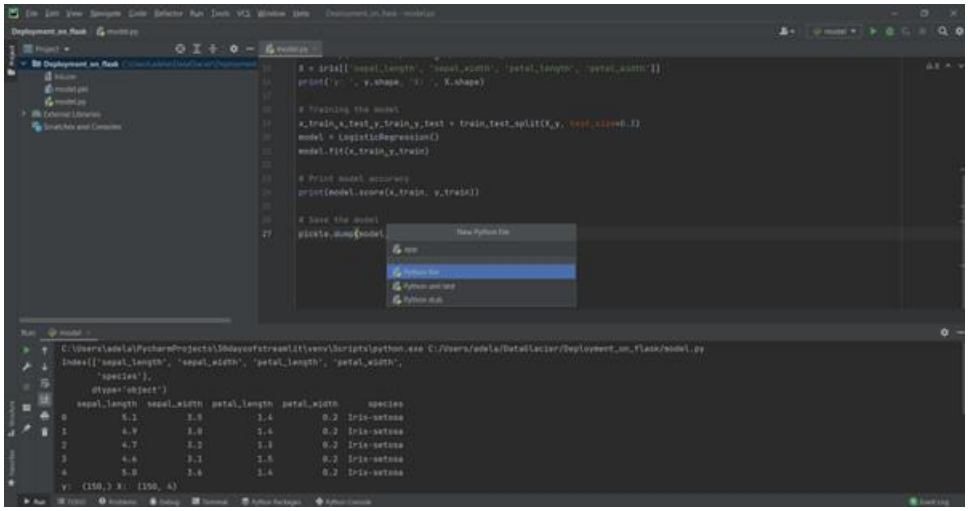
```
# Print model accuracy
print(model.score(x_train, y_train))
```

Save model (pickle file)

```
# Save the model
pickle.dump(model, open('model.pkl','wb'))
```

APPLICTION CREATION

Create application file ‘app.py’;



Import application libraries

```
# import libraries and functions
import numpy as np
from flask import Flask, request, render_template
import pickle
```

Initialize flask application and load the trained model pickle file

```
app = Flask(__name__)  # Initialize flask App
model = pickle.load(open('model.pkl', 'rb'))  # load trained model
```

Set application homepage route

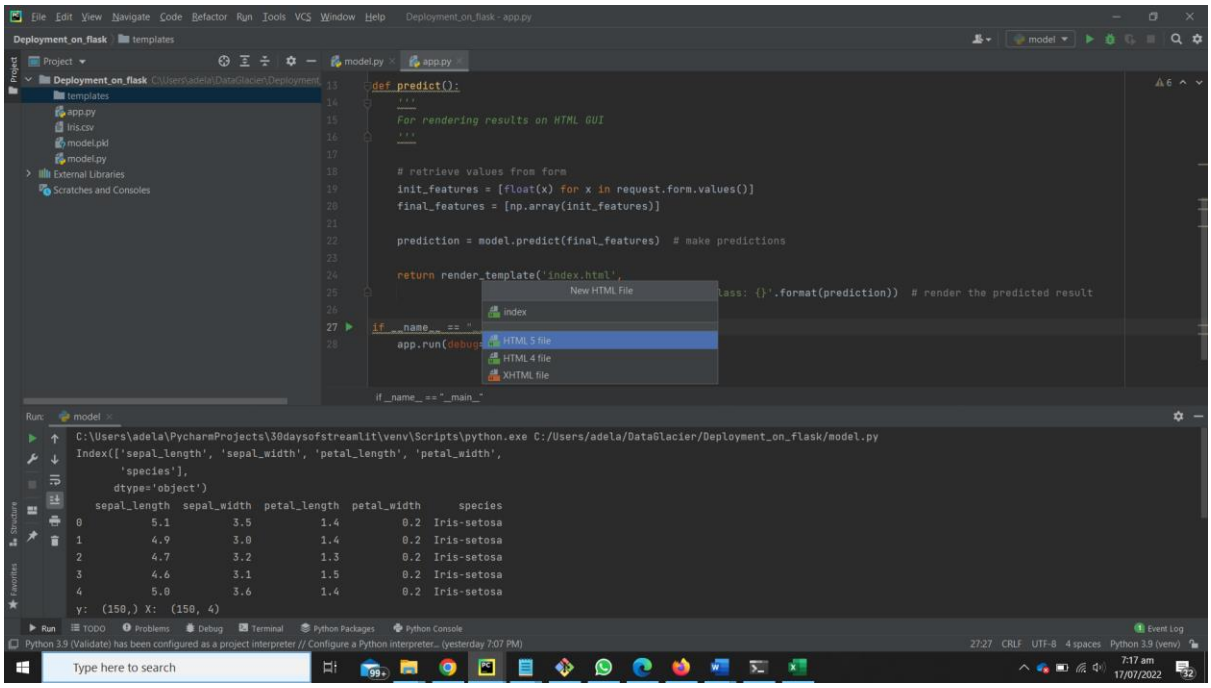
```
@app.route('/')  # Homepage
def home():
    return render_template('index.html')
```

Define prediction route

```
@app.route('/predict', methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''

    # retrieve values from form
    init_features = [float(x) for x in request.form.values()]
    final_features = [np.array(init_features)]

    prediction = model.predict(final_features)  # make predictions

    return render_template('index.html',
                           prediction_text='Predicted Class:
{}'.format(prediction))  # render the predicted result
if __name__ == "__main__":
    app.run(debug=True)
```

USER INTERFACE:

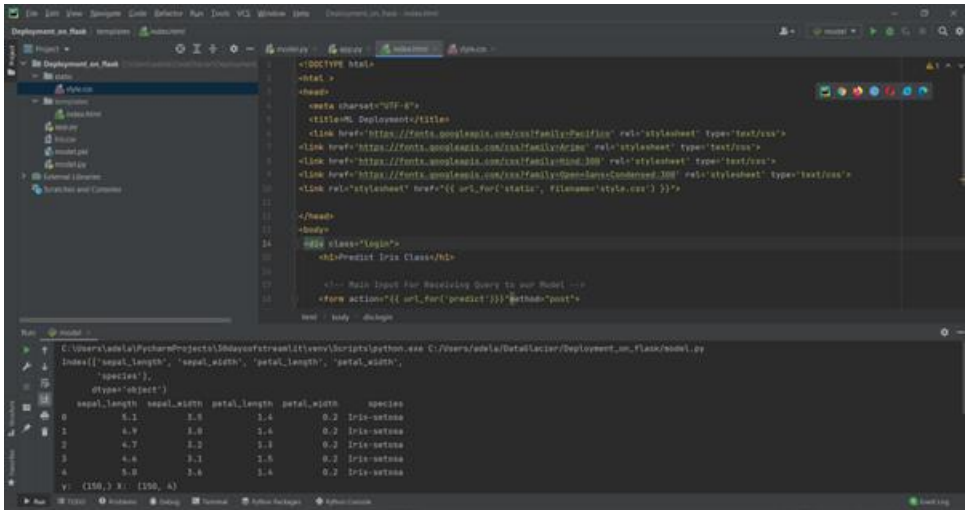
Create homepage HTML file for input forms:



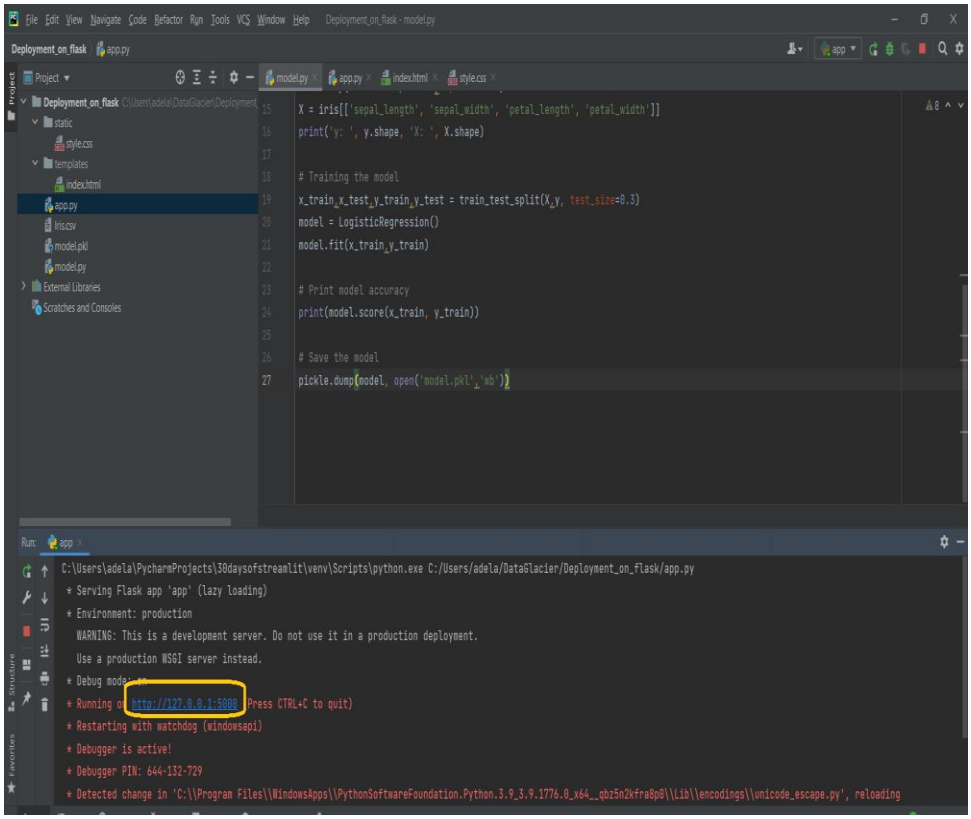
```html
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>ML Deployment</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico'
rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300'
rel='stylesheet' type='text/css'>
<link
href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>


</head>
<body>
 <div class="login">
   <h1>Predict Iris Class</h1>

     <!-- Main Input For Receiving Query to our Model -->
    <form action="{{ url_for('predict')}}"method="post">

       <input type="text" name="sepal_length" placeholder="Sepal Length"
required="required" />
        <input type="text" name="sepal_width" placeholder="Sepal Width"
required="required" />
      <input type="text" name="petal_length" placeholder="Petal Length"
required="required" />
      <input type="text" name="petal_width" placeholder="Petal Width"
required="required" />

        <button type="submit" class="btn btn-primary btn-block btn-
large">Predict</button>
    </form>

   <br>
   <br>
   {{ prediction_text }}
 </div>
</body>
</html>
```

Create and link stylesheet

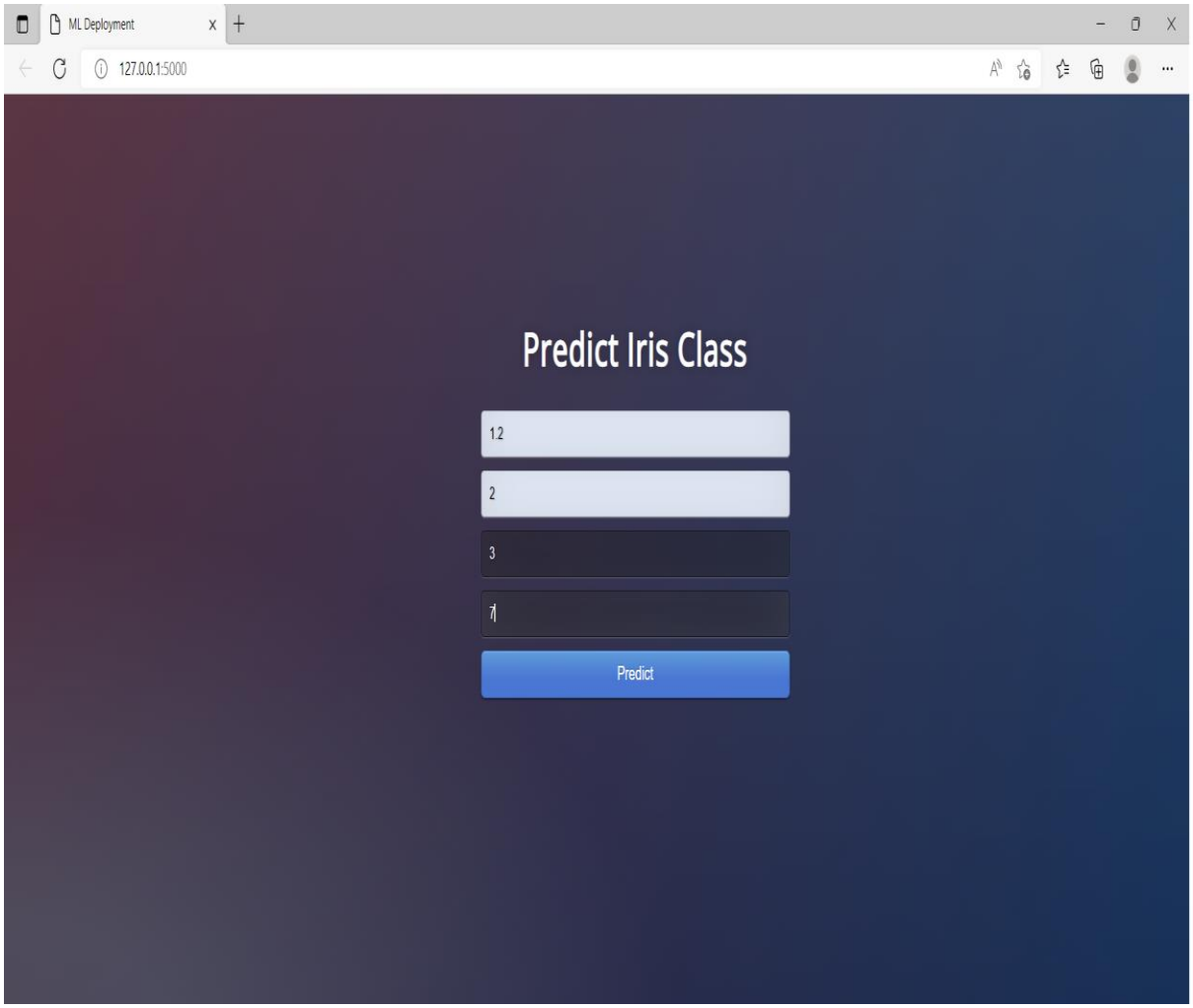
```html
<link rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}">
```

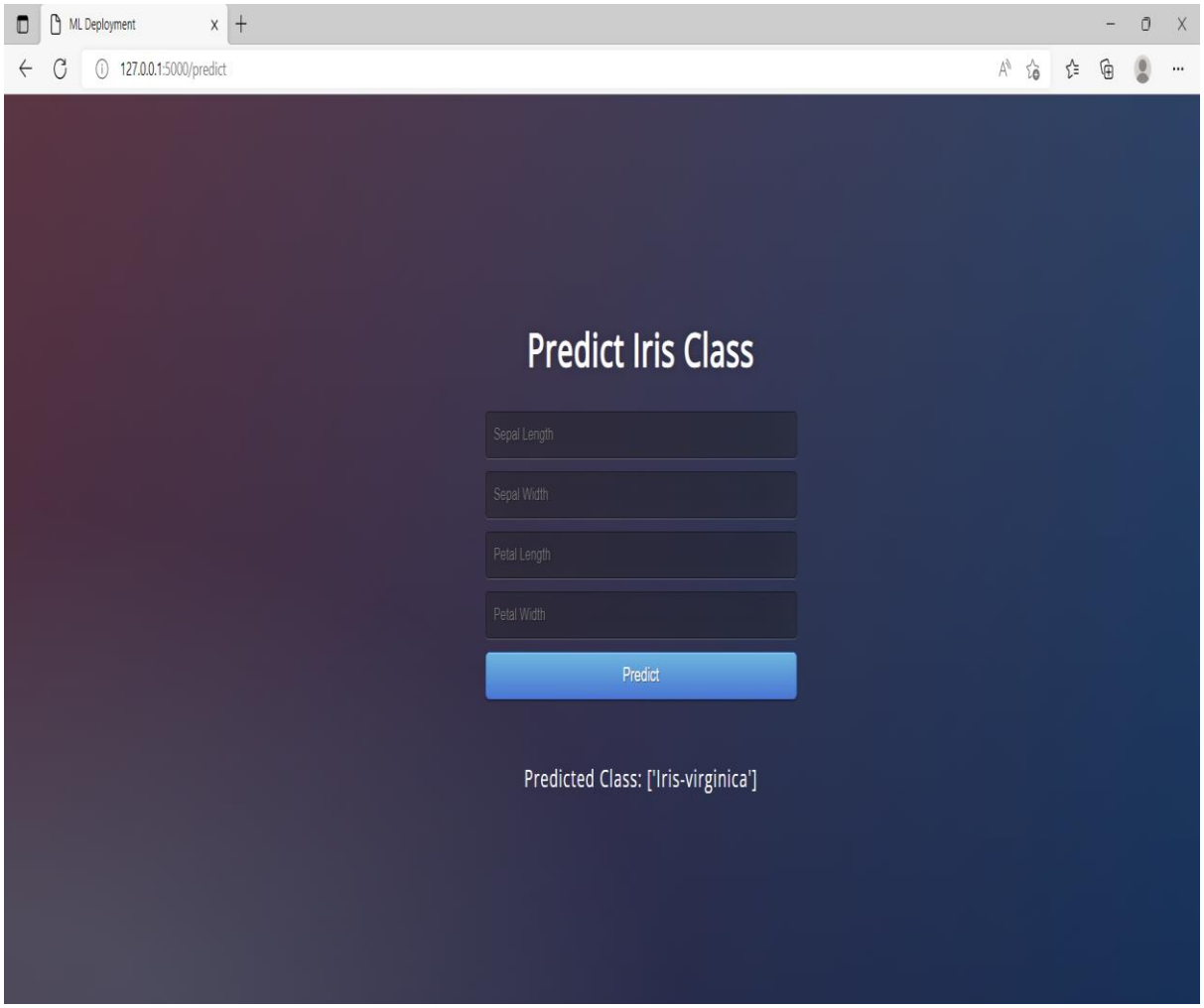


Run application ‘app.py’



Rendered html page:




ML Deployment
127.0.0.1:5000/predict
Predict Iris Class
Sepal Length
Sepal Width
Petal Length
Petal Width
Predict
Predicted Class: ['Iris-virginica']