



**Data Glacier**

Your Deep Learning Partner

# Model Deployment with Flask

LISUM02

**19<sup>th</sup> January 2022**

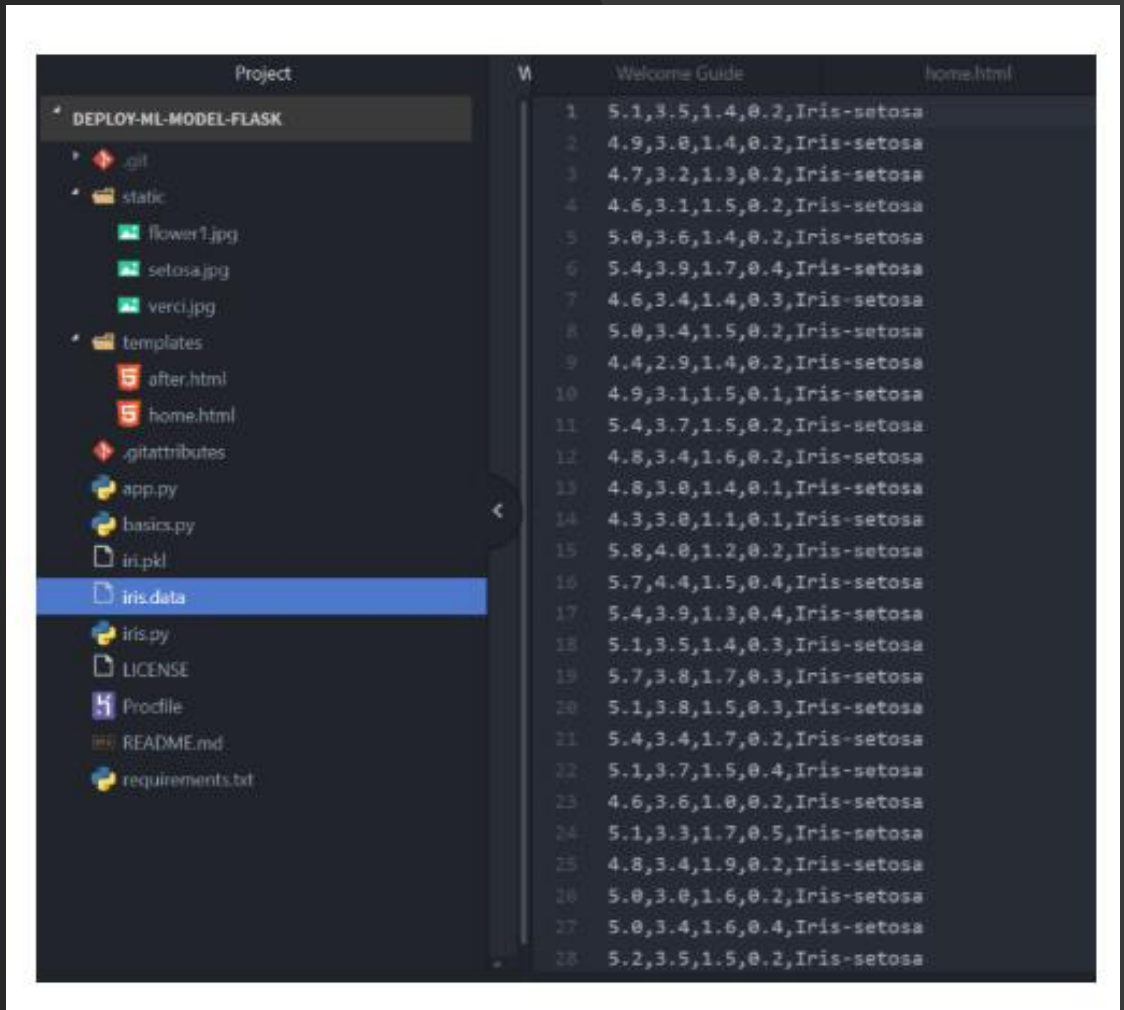
**By,**

**Kelvin Mpofu**

**Submitted to:**

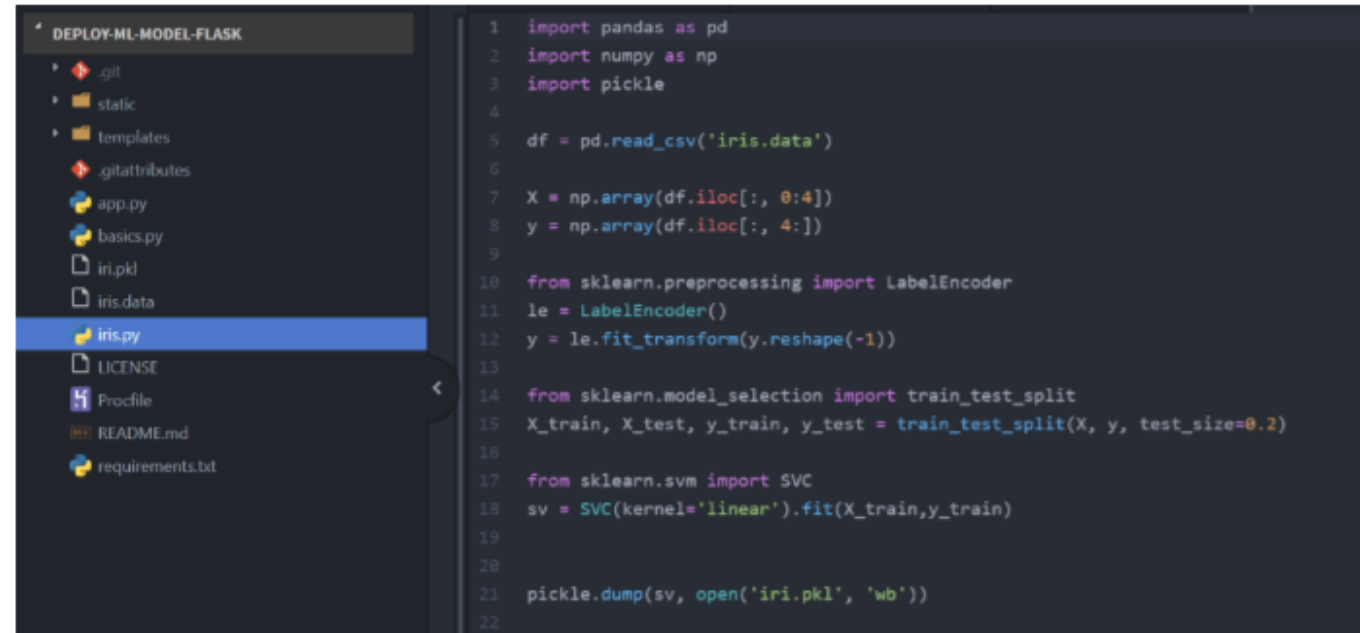
# Tasks Involved

- The data set consists of 50 samples from each of three species of *Iris* (*Iris setosa*, *Iris virginica* and *Iris versicolor*). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.
- Trained a **support vector machine model** to classify the different types of flowers based on their features
- Finally, deployed Support vector machine model to a web-app using **Flask API**.
- This way, we can predict potential customers using the web-app.



# Model Validation

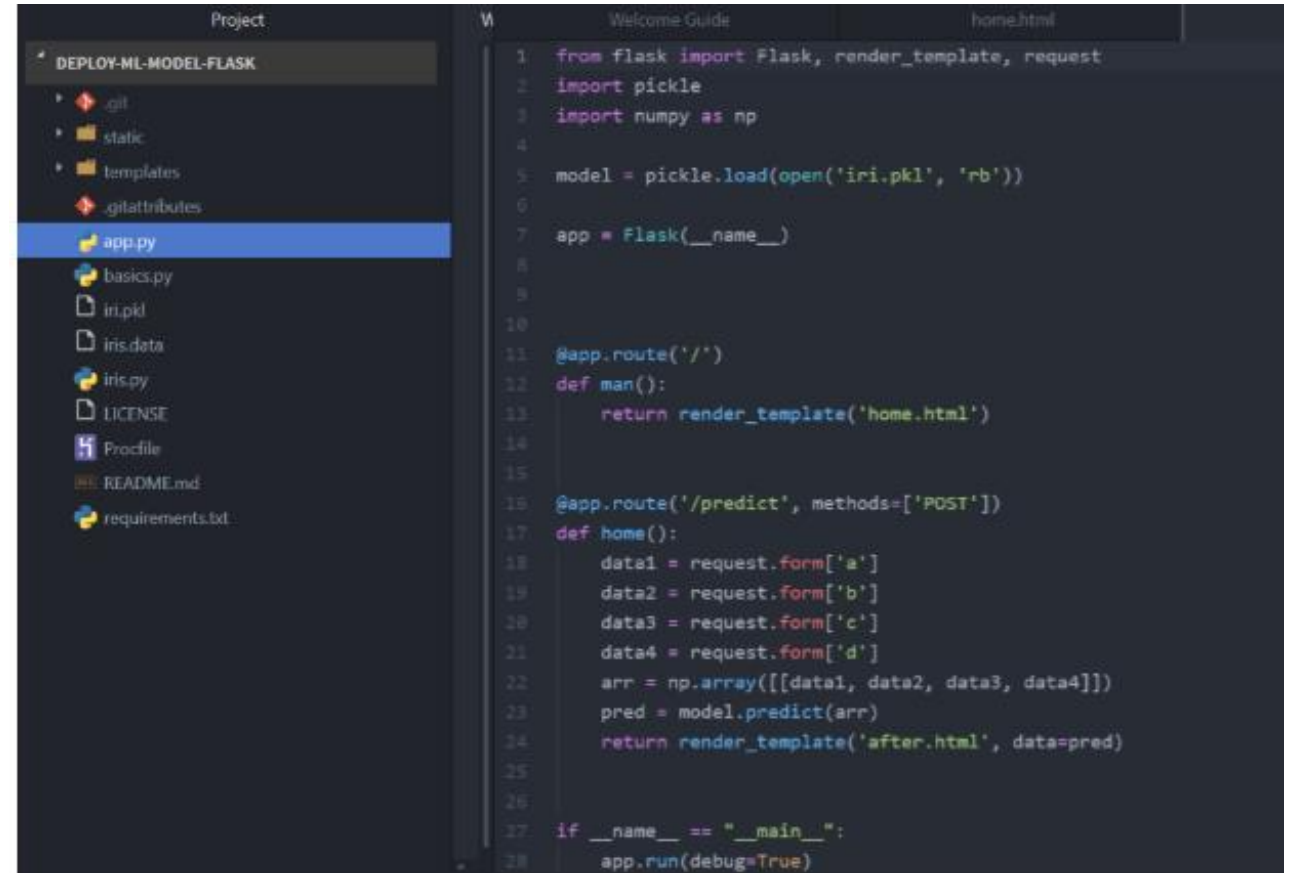
- After splitting the data into train and test set, Support vector machine model was used for predicting on test set.
- The model achieved a high **accuracy of 82.5%**.



```
1 import pandas as pd
2 import numpy as np
3 import pickle
4
5 df = pd.read_csv('iris.data')
6
7 X = np.array(df.iloc[:, 0:4])
8 y = np.array(df.iloc[:, 4:])
9
10 from sklearn.preprocessing import LabelEncoder
11 le = LabelEncoder()
12 y = le.fit_transform(y.reshape(-1))
13
14 from sklearn.model_selection import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
16
17 from sklearn.svm import SVC
18 sv = SVC(kernel='linear').fit(X_train, y_train)
19
20
21 pickle.dump(sv, open('iri.pkl', 'wb'))
22
```

# Saving Model & Creating Web-App using Flask API

- The model was then trained on the whole dataset before saving the model to **pickle format**.
- Pickling is done to **convert python object to character object**.
- Next, created a python file to create the web app using **Flask API** module.

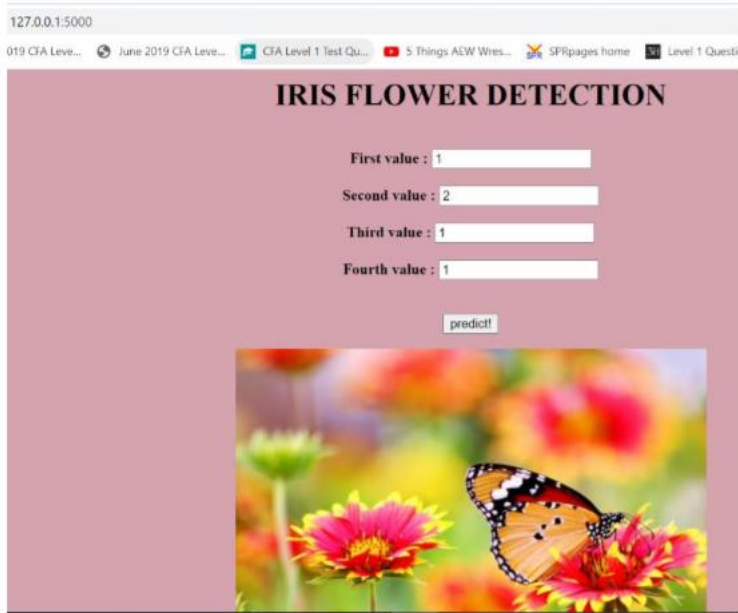


```
1 from flask import Flask, render_template, request
2 import pickle
3 import numpy as np
4
5 model = pickle.load(open('iri.pkl', 'rb'))
6
7 app = Flask(__name__)
8
9
10
11 @app.route('/')
12 def man():
13     return render_template('home.html')
14
15
16 @app.route('/predict', methods=['POST'])
17 def home():
18     data1 = request.form['a']
19     data2 = request.form['b']
20     data3 = request.form['c']
21     data4 = request.form['d']
22     arr = np.array([[data1, data2, data3, data4]])
23     pred = model.predict(arr)
24     return render_template('after.html', data=pred)
25
26
27 if __name__ == "__main__":
28     app.run(debug=True)
```

# HTML

```
1 <html>
2   <body bgcolor=#d4a3ae>
3
4     <center>
5
6     <h1> IRIS FLOWER DETECTION </h1><br>
7
8     <form method="POST", action="{{url_for('home')}}">
9       <b> First value : <input type="text", name='a', placeholder="enter 1"> <br><br>
10       Second value : <input type="text", name='b', placeholder="enter 2"> <br><br>
11       Third value : <input type="text", name='c', placeholder="enter 3"> <br><br>
12       Fourth value : <input type="text", name='d', placeholder="enter 4"> <br><br><br></b>
13       <input type="submit" , value='predict!' >
14     </form>
15
16     <img src='static\flower1.jpg' alt="flower">
17
18   </center>
19
20 </body>
21 </html>
```

# Model Deployment



- Finally, created the web-app and deployed the model into the web-app.
- Based on the model, we can now use the web application to predict the type of flower based on features.

# The End