Name: Manoj Nagaraja

Batch code: LISUM19

Submission date: 04/04/2023

Submitted to: Week 5: Cloud and API deployment

Github : https://github.com/ManojN7270/Cloud-and-API-Deployment.git

```
import flask
import pickle
```

```
from sklearn import datasets
import pandas as pd
data=datasets.load_iris()
df = pd.DataFrame(data.data,columns=data.feature_names)
df['target']=data['target']
```

```
X=df.loc[:,df.columns!="target"] #let the feature dataframe contain every column of df, except the value we are predicting
y=df.loc[:,df.columns=="target"].values.ravel() #let the target dataframe contain only the value we are predicting
```

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.25, shuffle=True)
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X, y)
pred=knn.predict(X_test)
acc=accuracy_score(y_test, pred)
```

```
from joblib import dump, load
dump(knn, 'model.joblib')
```

```
['model.joblib']
```

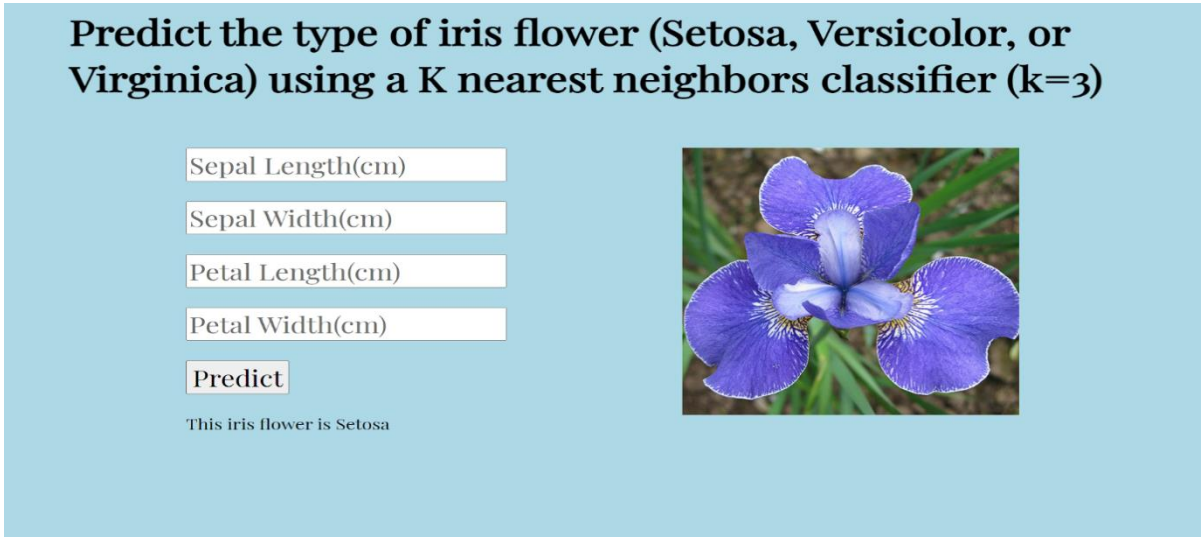
```
l_m = load('model.joblib')
```

```
print(l_m)
```

```
KNeighborsClassifier(n_neighbors=3)
```

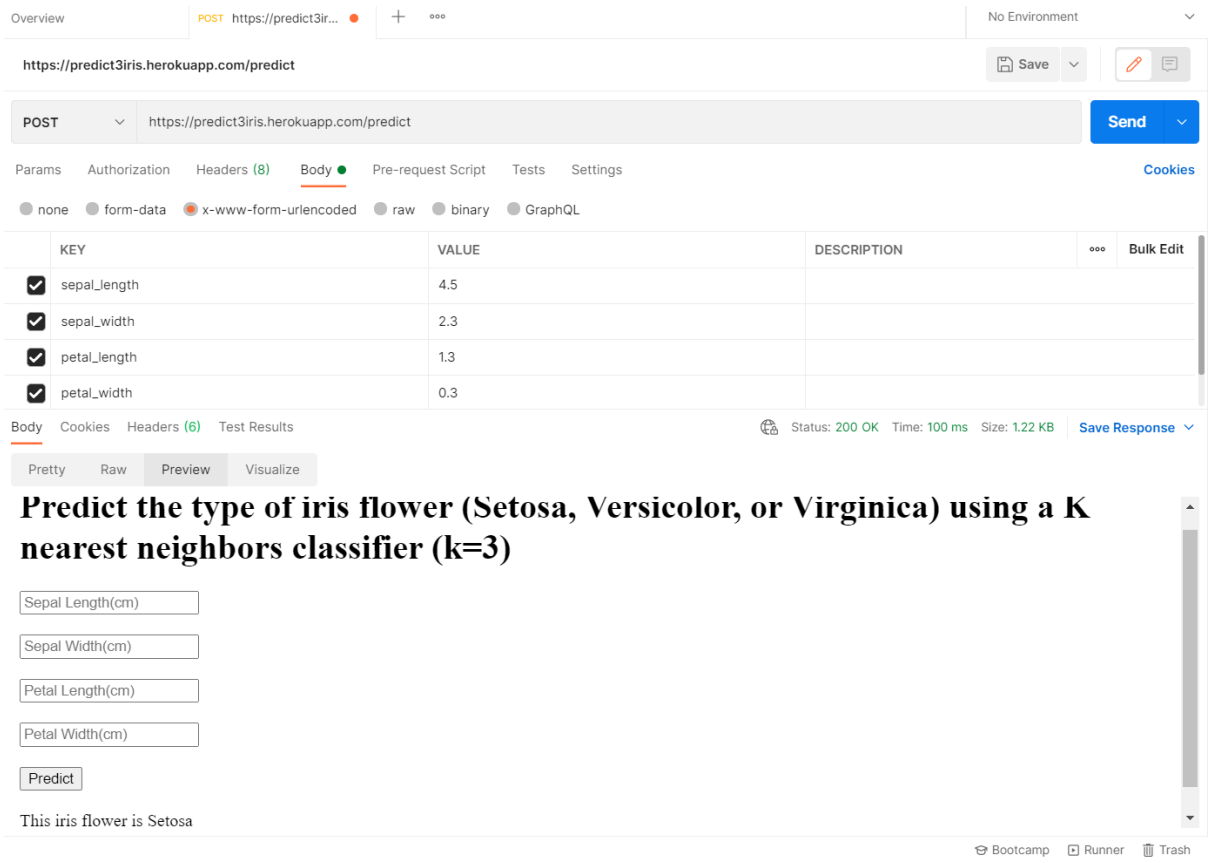
Select data (iris dataset), create and save a simple model (knn classifier):

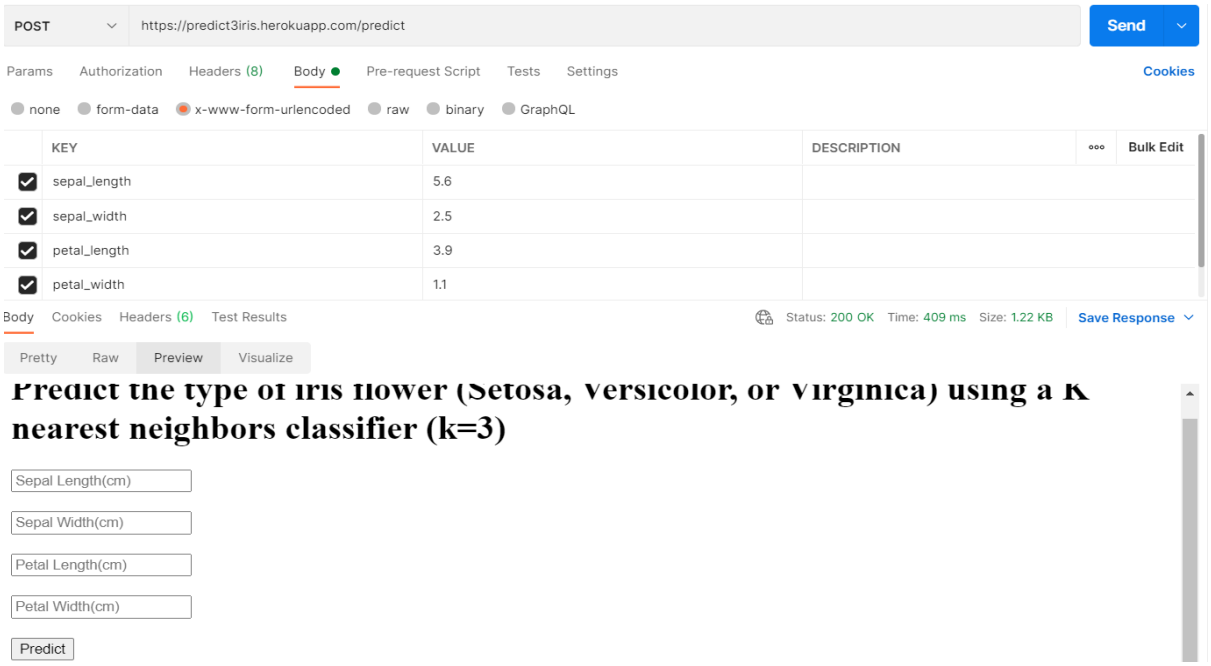
```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:
import numpy as np
from flask import Flask, request, render_template
import joblib
from joblib import load
from sklearn.neighbors import KNeighborsClassifier
import os
images_folder=os.path.join('static', 'images')
app=Flask(__name__)
app.config['UPLOAD_FOLDER'] = images_folder
model=load('model.joblib')

@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict', methods=['POST'])
def predict():
    features=[float(x) for x in request.form.values()]
    final_features=[np.array(features)]
    prediction=model.predict(final_features)
    pred_round=round(prediction[0])
    output=""
    if pred_round==0:
        output+="Setosa"
        file = os.path.join(app.config['UPLOAD_FOLDER'], 'setosa.jpg')
    elif pred_round==1:
        output+="Versicolor"
        file = os.path.join(app.config['UPLOAD_FOLDER'], 'versicolor.jpg')
    else:
        output+="Virginica"
        file = os.path.join(app.config['UPLOAD_FOLDER'], 'virginica.jpg')

    return render_template('index.html', prediction_text='This iris flower is {}'.format(output),
                           iris=file
                          )
if __name__=="__main__":
    app.run(port=5000, debug=True, use_reloader=False)

```

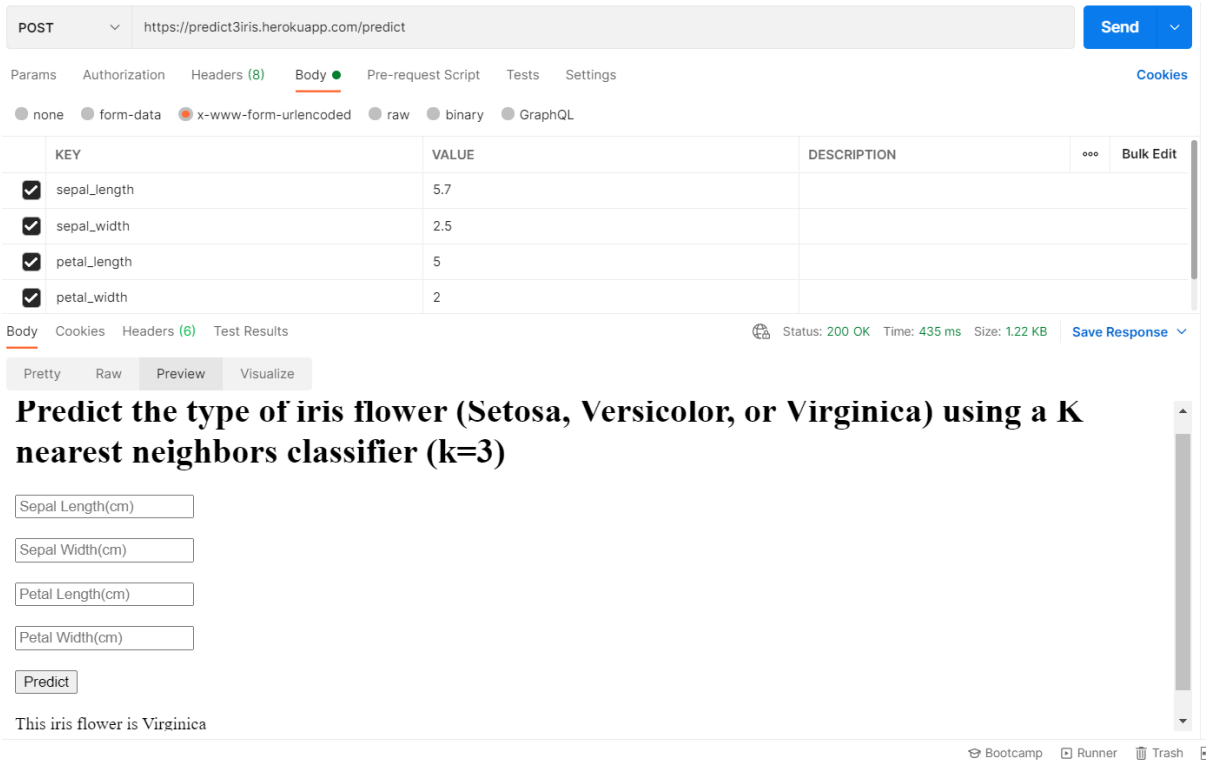


The app is working as intended.

Each type of prediction was tested using Postman:






POST
https://predict3iris.herokuapp.com/predict
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
KEY
VALUE
DESCRIPTION
Bulk Edit
sepal_length
5.7
sepal_width
2.5
petal_length
5
petal_width
2
Body
Cookies
Headers (6)
Test Results
Status: 200 OK
Time: 435 ms
Size: 1.22 KB
Save Response
Pretty
Raw
Preview
Visualize
Predict the type of iris flower (Setosa, Versicolor, or Virginica) using a K nearest neighbors classifier (k=3)
Sepal Length(cm)
Sepal Width(cm)
Petal Length(cm)
Petal Width(cm)
Predict
This iris flower is Virginica
Bootcamp
Runner
Trash