Data Intake Report

Name: Predict iris species
Report date: 30/Jun/2022
Internship Batch: LISUM10: 30
Version:<1.0>
Data intake by: Moath Bin Musallam
Data intake reviewer:
Data storage location: local storage

Tabular data details:

Total number of observations	150
Total number of files	1
Total number of features	5
Base format of the file	.csv
Size of the data	4,49 KB

Machine learning model:

Using knn because it is better model.

```
#split to test the model
# from sklearn.model_selection import train_test_split

#Wit stratification to balance the output y
# X_train,X_test, y_train, y_test= train_test_split (X,Y, test_size=0.3,random_state=1,stratify=Y)

###Train the model
from sklearn.neighbors import KNeighborsClassifier

model_knn = KNeighborsClassifier(n_neighbors=4,weights='uniform',algorithm='ball_tree', p=1)

# model.fit(X_train, y_train) #Training the model
# #Test the model
# predictions = model.predict(X_test)
# print( classification_report(y_test, predictions) )
# print( accuracy_score(y_test, predictions))

model_knn.fit(X,Y)

# Saving model to disk

pickle.dump(model_knn,open('model.pkl','wb'))

# Loading model to compare the results
model = pickle.load(open('model.pkl','rb'))
print(model.predict([[5.1,3.5,1.4,0.2]]))
```

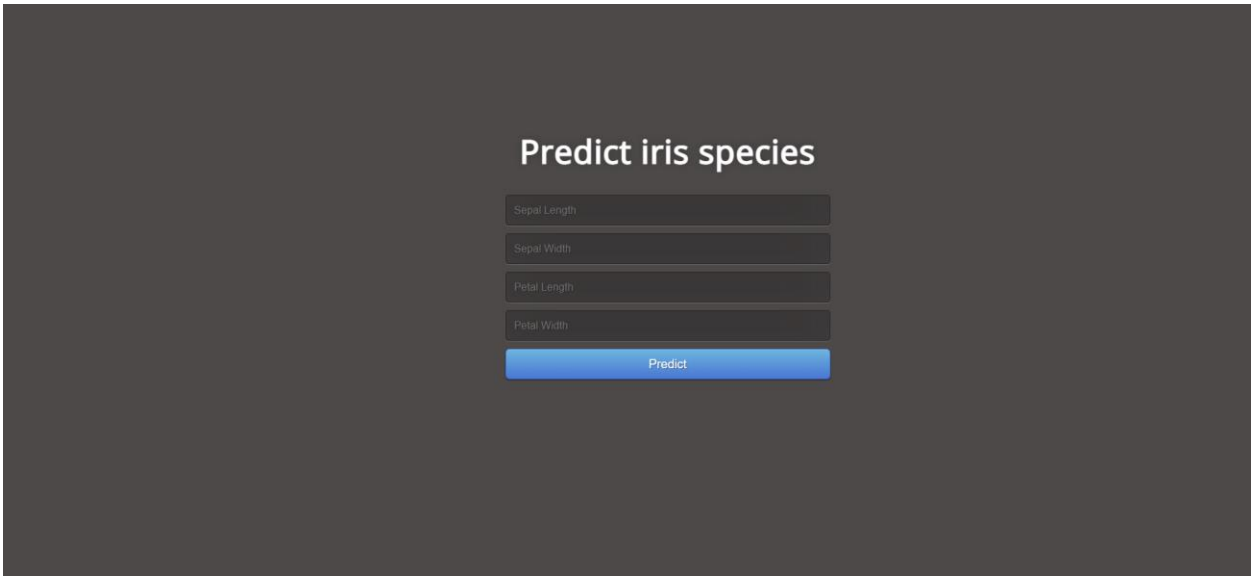
Flask app:

```python
model = pickle.load(open('model.pkl', 'rb'))


app = Flask(__name__)
@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [float(x) for x in request.form.values()]

    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output =prediction[0]
    if output ==0:
        output ='Iris-setosa'
    elif output ==1:
        output ='Iris-versicolor'
    elif output ==2:
        output ='Iris-virginica'

    return render_template('index.html', prediction_text='The Flower is {}'.format(output))
```

HTML:

```html
<!DOCTYPE html>
<html >
<head>

<meta charset="UTF-8">
<title>ML APP</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/c
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>
<body>
<div class="login">
<h1>Predict iris species</h1>
<!-- Main Input For Receiving Query to our ML -->
<form action="{{ url_for('predict')}}"method="post">
<input type="text" name="SepalLength" placeholder="Sepal Length" required="required" />
<input type="text" name="SepalWidth" placeholder="Sepal Width" required="required" />
<input type="text" name="PetalLength" placeholder="Petal Length" required="required" />
<input type="text" name="PetalWidth" placeholder="Petal Width" required="required" />
<button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
</form>
<br>
<br>
```

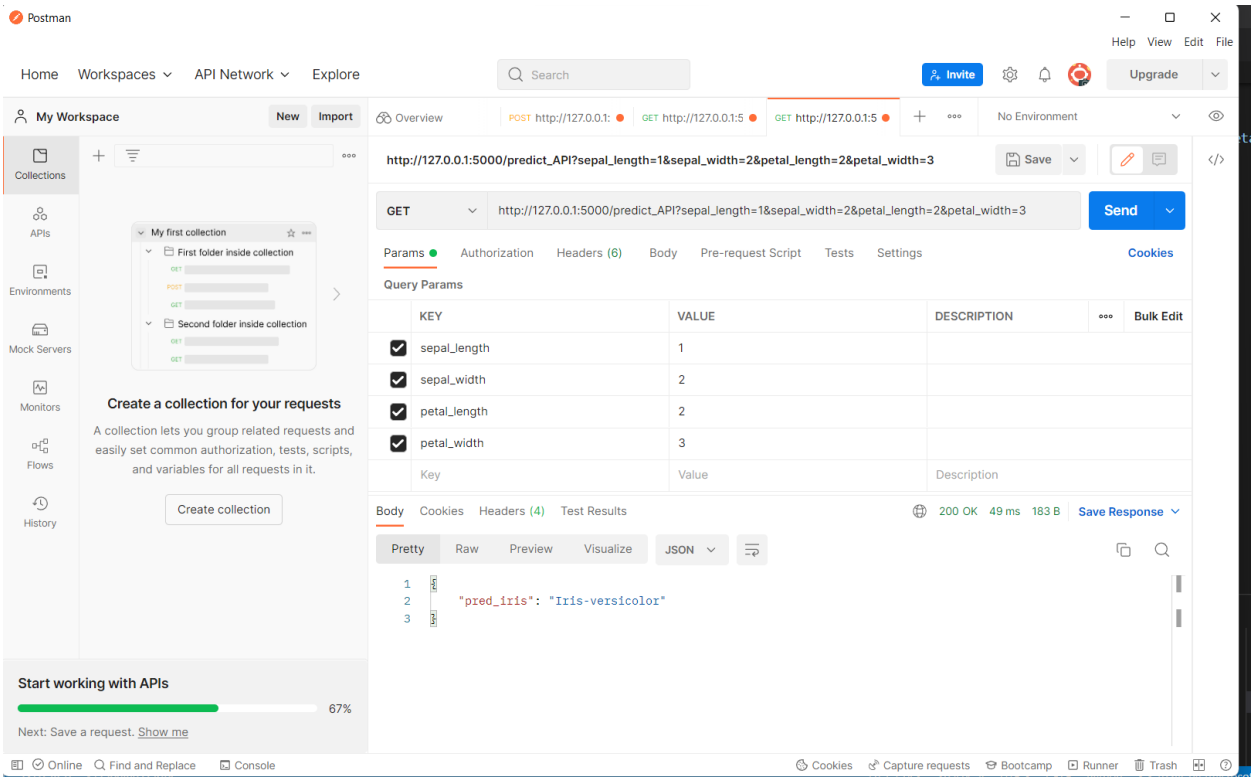
final product:
after deploying the project to Heroku



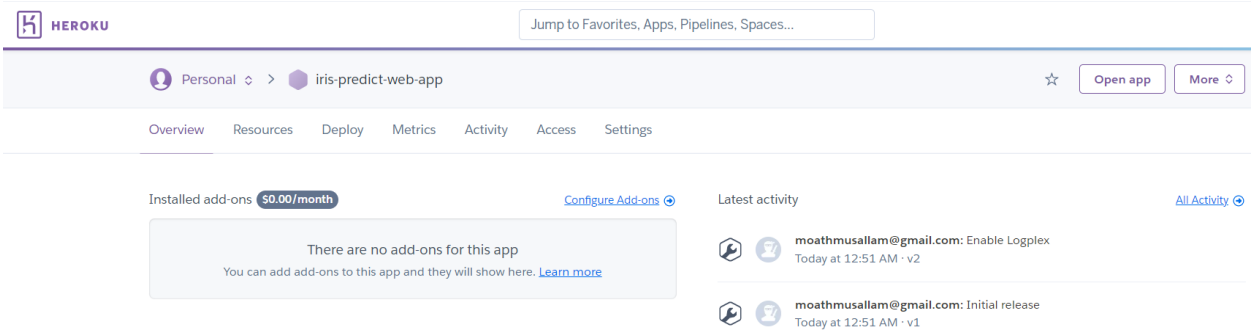
Create API method

```python
@app.route('/predict_API')
def predict_API():
    '''
    For rendering results on postman
    '''
    model = pickle.load(open('model.pkl', 'rb'))
    sepal_length = request.args.get('sepal_length')
    sepal_width = request.args.get('sepal_width')
    petal_length = request.args.get('petal_length')
    petal_width = request.args.get('petal_width')
    test_df = pd.DataFrame({'sepal_length':[sepal_length],
    'sepal_width':[sepal_width], 'petal_length':[petal_length],
    'petal_width':[petal_width]})

    pred_iris = model.predict(test_df).tolist()
    output =pred_iris[0]
    if output ==0:
        output ='Iris-setosa'
    elif output ==1:
        output ='Iris-versicolor'
    elif output ==2:
        output ='Iris-virginica'
    return jsonify({'pred_iris': output})

# if name = main the app will start

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

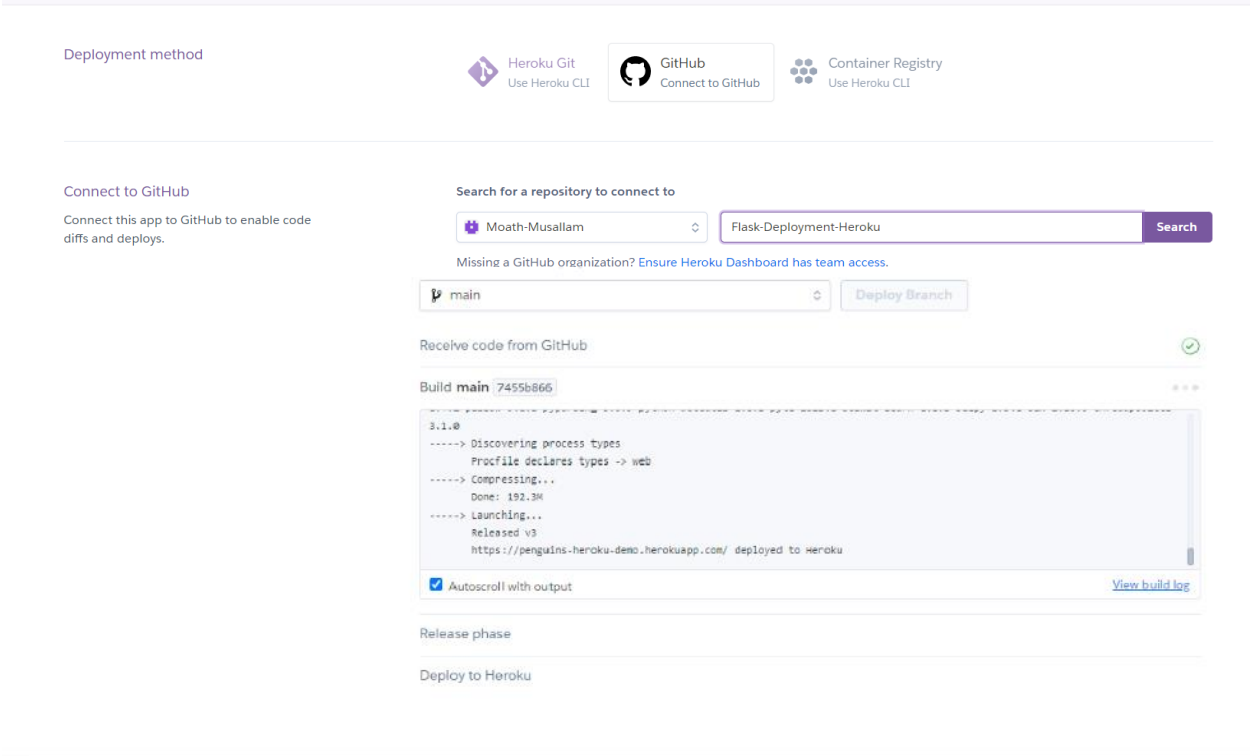
Try the API method with Postman, and its work.

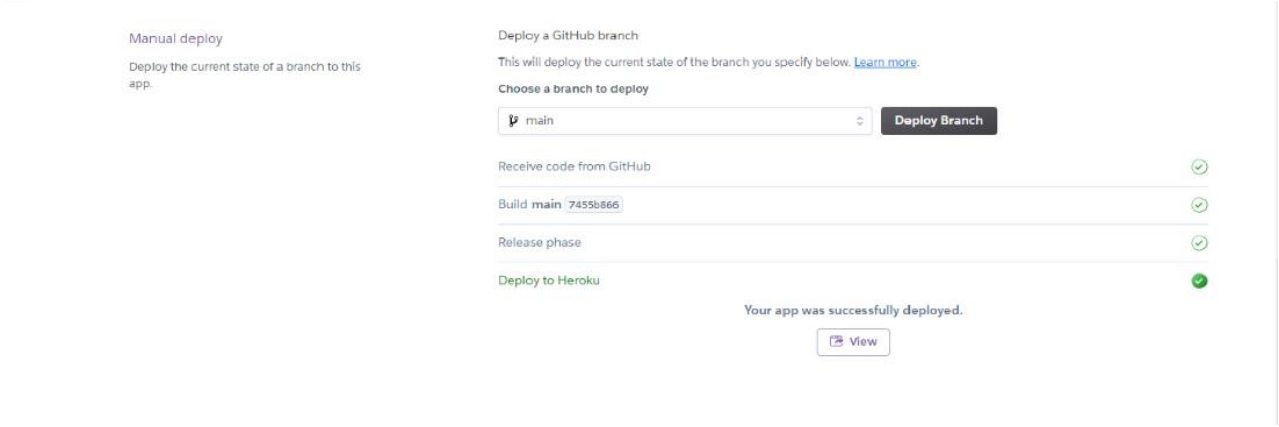


Create new project in Heroku called iris-predict-web-app:



Deploy from GitHub:





project file:

الاسم	تاريخ التعديل	النوع	الحجم
static	١١:٣٦ م ٤٣/١٢/١٣	مجلد ملفات	
templates	١١:٣٦ م ٤٣/١٢/١٣	مجلد ملفات	
app	١٢:٤٠ ص ٤٣/١٢/١٤	ملف PY	٢ كيلوبايت
Heroku	٠٦:١٦ ص ٤٣/٠٩/٢٢	مستند Microsoft ...	٤٣٠ كيلوبايت
Heroku	٠٦:١٦ ص ٤٣/٠٩/٢٢	Microsoft Edge PD...	١٨٥ كيلوبايت
iris	٠٧:٠١ م ٤٢/٠٧/٠٣	ملف القيم المفصو...	٥ كيلوبايت
model.pkl	١٠:١٨ م ٤٣/١٢/١٣	ملف PKL	١٣ كيلوبايت
model	١٢:٥٠ ص ٤٣/١٢/١٤	ملف PY	٢ كيلوبايت
Procfile	٠٥:٢٧ ص ٤٣/٠٩/٢٢	ملف	١ كيلوبايت
README	١٢:٣٣ ص ٤٣/١٢/١٤	ملف MD	١ كيلوبايت
requirements	١٢:٤٦ ص ٤٣/١٢/١٤	مستند نصي	٥ كيلوبايت
test	٠٢:٤١ ص ٤٣/٠٩/٢١	ملف PY	١ كيلوبايت