DEPLOY MACHING LEARNING USING FLASK

Present by Pariwat Prathanrat

MEACHINE LEARNING FLOW

PREPARE DATA
(collect, ETL)

CREATE MODEL
(train, test, predict)

DEPLOY
(form, dashboard)
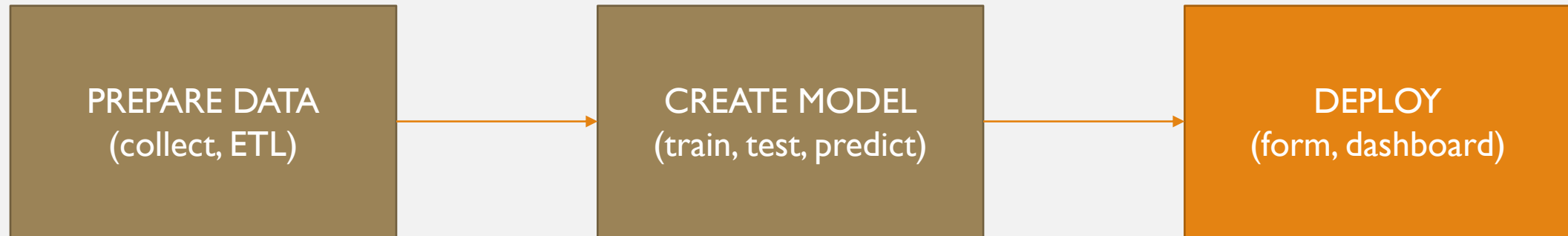
DEPLOY REQUIREMENTS

- Visual Studio code

- Python 3.6^ (sklearn, pandas, numpy, flask)

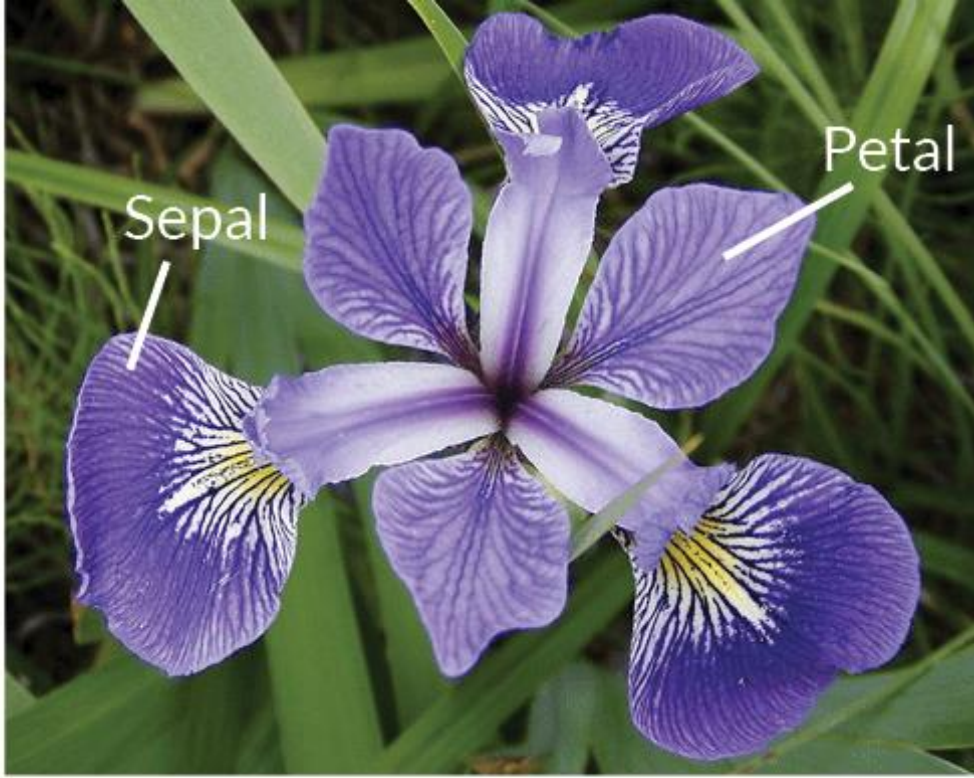- Terminal (Mac & Linux OS), Windows Termina, CMD (Window OS),

GIT INSTALL

- https://git-scm.com/download

GIT CLONE

- https://github.com/zooksonline/flask_learn.git
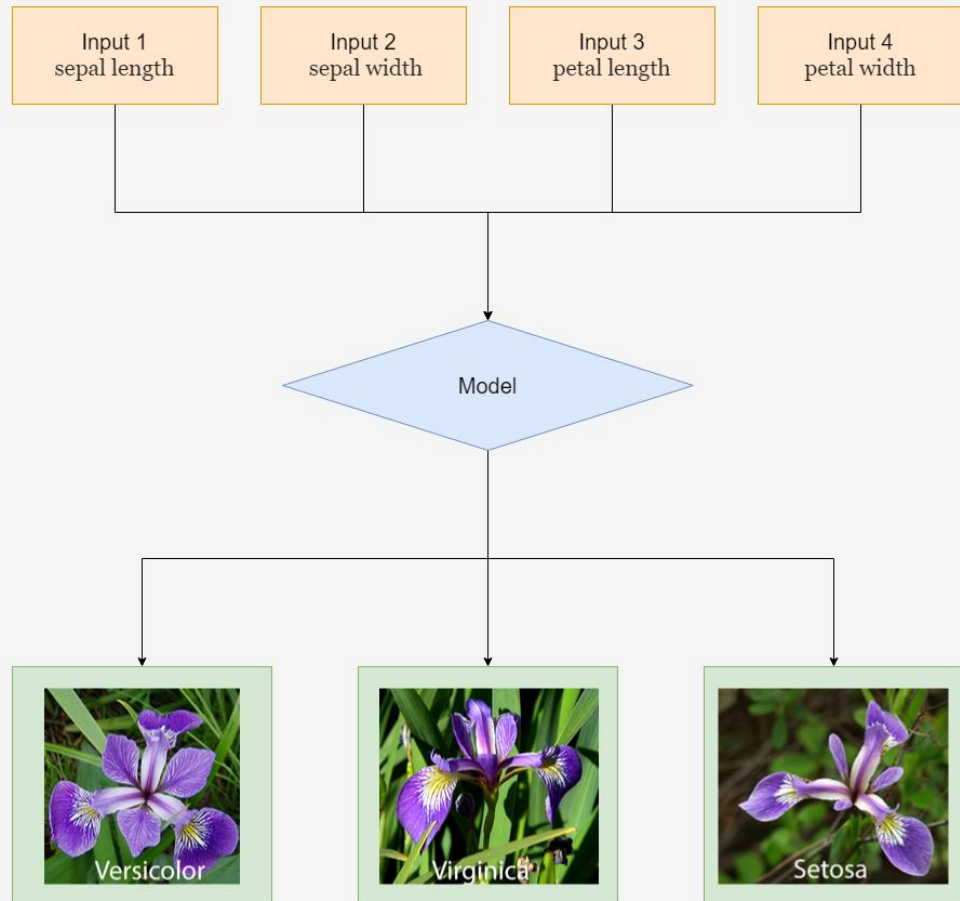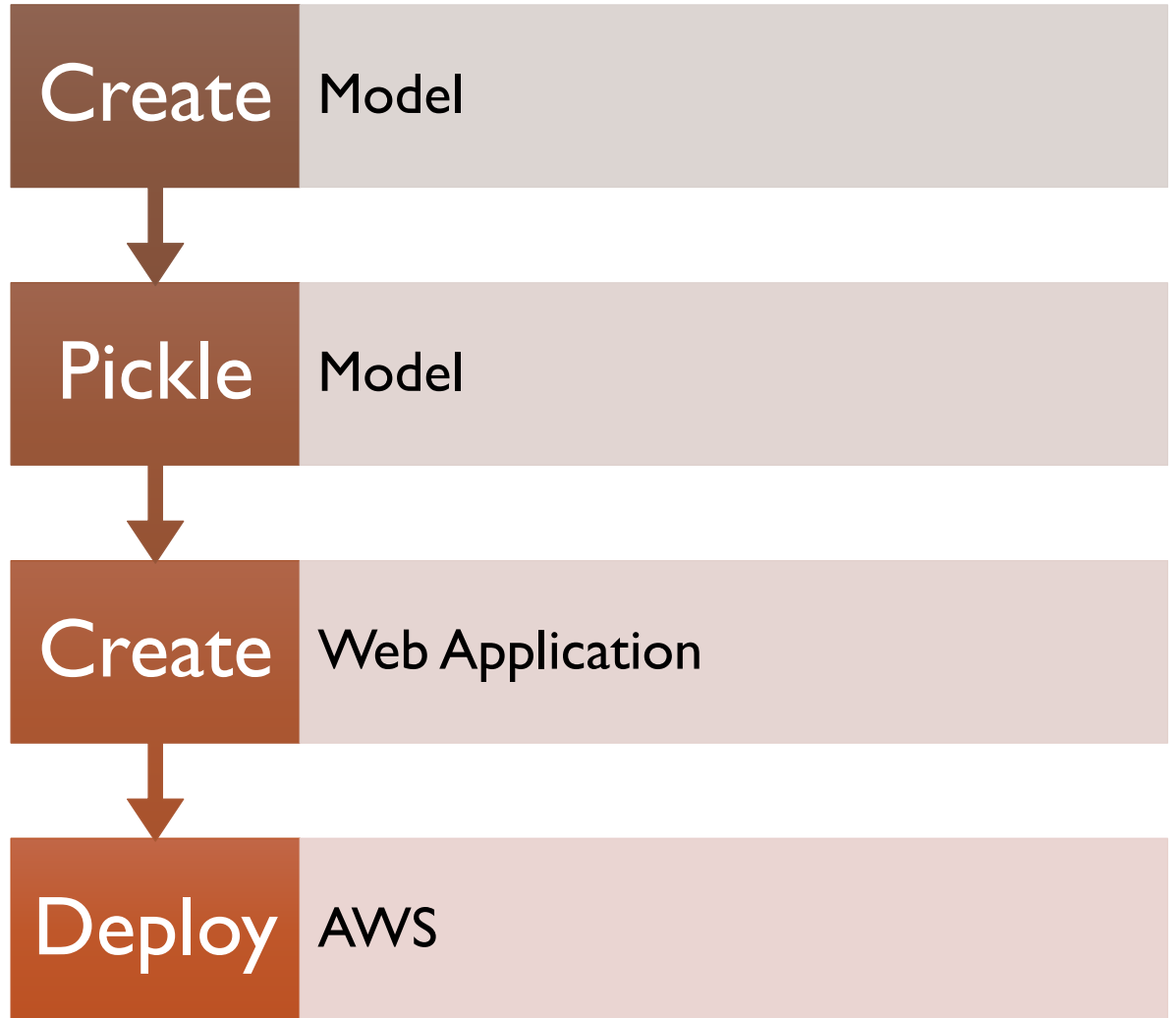
Iris Versicolor

Iris Setosa

Iris Virginica

Sepal

Petal

GOAL

Create the Form that User can input values to predict the answer what kind of the Iris flower is

STEP PROCESS

STEP DEPLOY

Create Model

Pickle Model

Create Web Application

Deploy AWS

CREATE MODEL

dataset - iris

machine learning model - LogisticRegression

CREATE MODEL

- open jupyter notebook and create model_iris.ipynb

- in file – type below to import libraries

```python
from sklearn.metrics import accuracy_score
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
```

- load iris dataset

```python
dataset = load_iris()
```

CREATE MODEL

- create machine learning model

```
names = dataset.feature_names

features = dataset.data
labels = dataset.target

feature_train, feature_test, label_train, label_test = train_test_split(features, labels, test_size=0.2,
random_state=42)

model = LogisticRegression(max_iter=500)
```

CREATE MODEL

- test model

```
model.fit(feature_train, label_train)

label_pred = model.predict(feature_test)

accuracy_score(label_pred, label_test)
print(accuracy_score(label_pred, label_test))
```
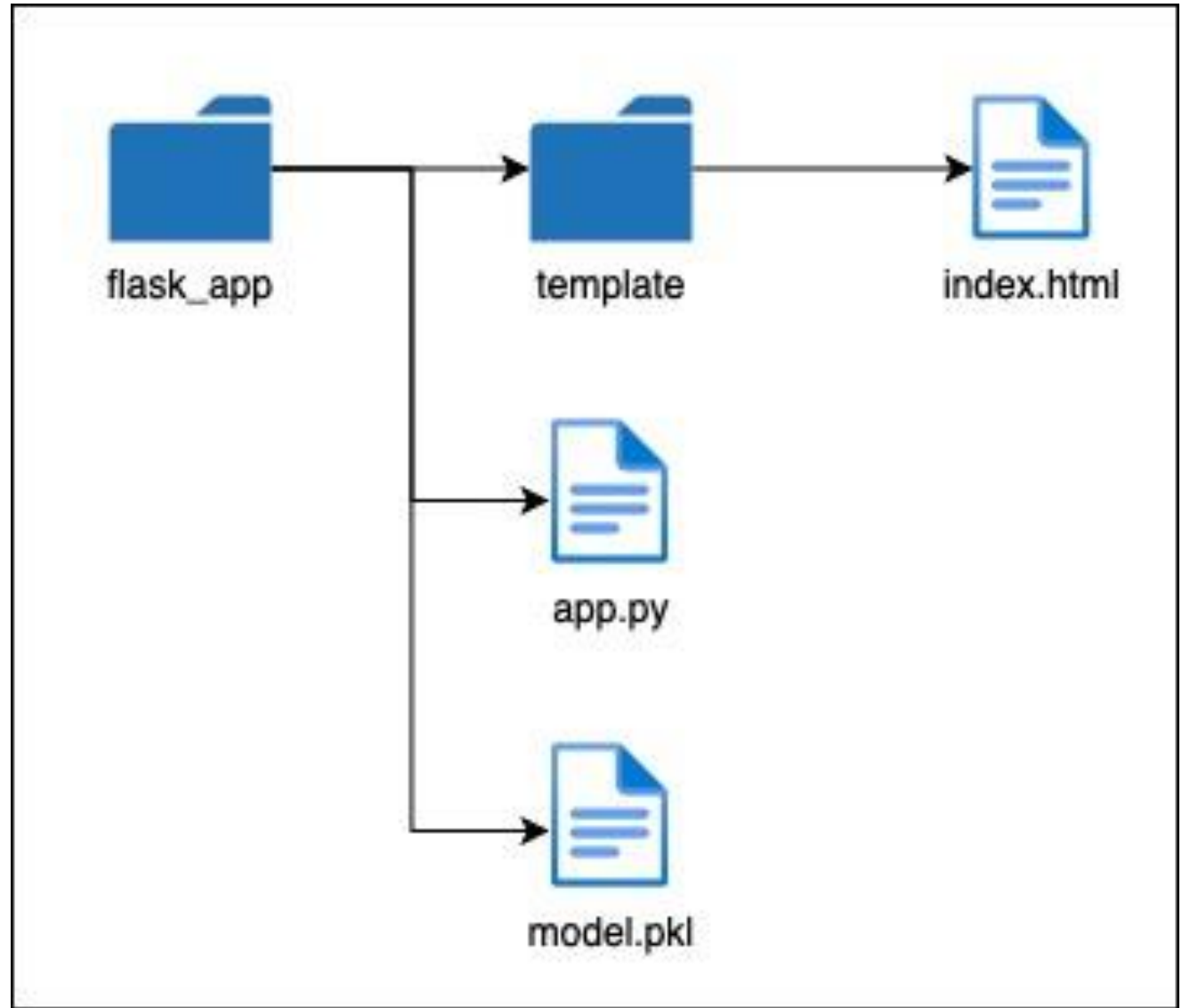
- The Result should be 1.0

PICKLE MODEL

- continue in model_iris file type:

```
import pickle

pickle.dump(model,open('model.pkl','wb+'))
```
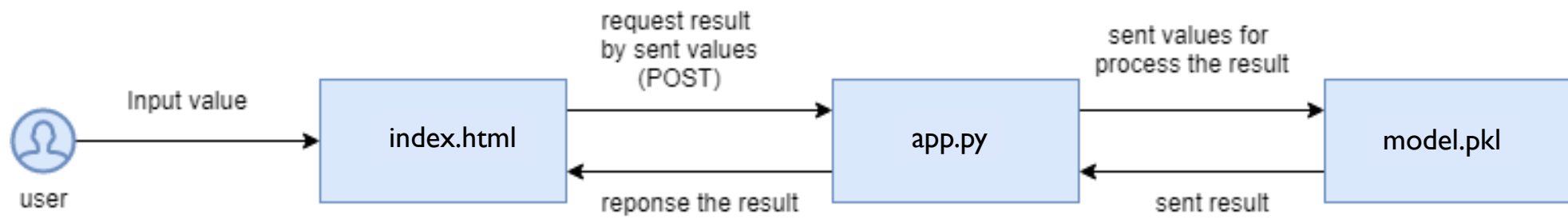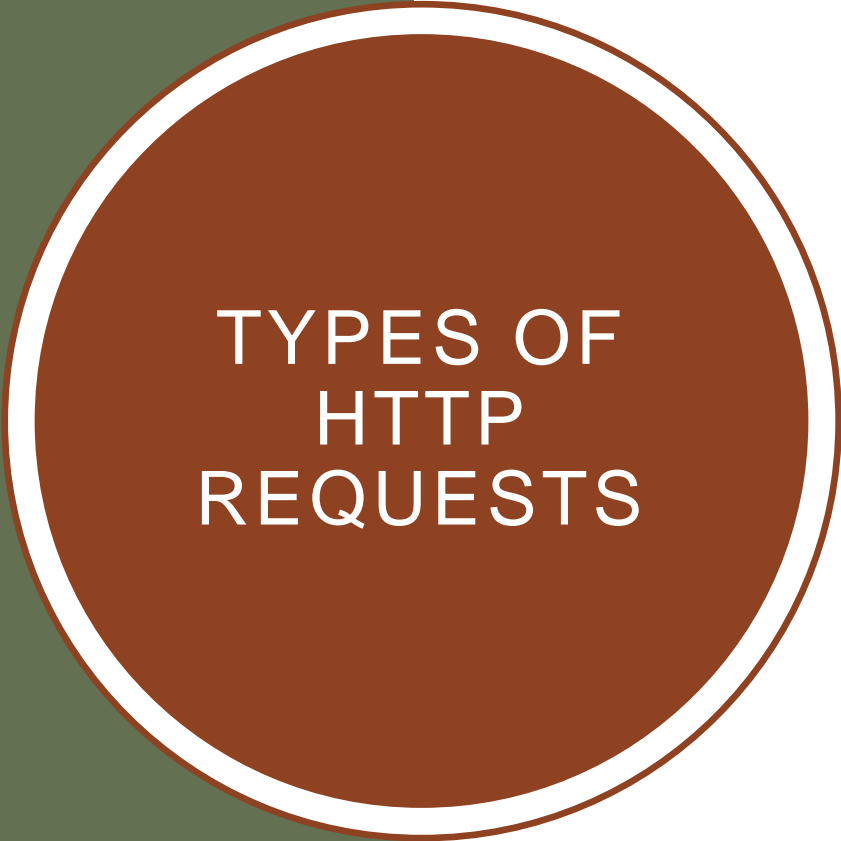
- A file named model.pkl file should be created.

FLASK-APP TREE

MAIN IDEA

TYPES OF HTTP REQUESTS

- **GET** - requests a representation of the specified resource. Note that GET should not be used for operations that cause side-effects, such as using it for taking actions in web applications. One reason for this is that GET may be used arbitrarily by robots or crawlers, which should not need to consider the side effects that a request should cause.

- **POST** - submits data to be processed to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both.

- https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

APP.PY

- import libraries

```python
from flask import Flask,render_template,request
import joblib
```

- Initial folder for run flask-app

```python
app = Flask(__name__, template_folder='templates')
```

- load model

```python
model = joblib.load('model.pkl')
```

APP.PY

- route '/' for main page

```python
@ app.route('/', methods=['POST', 'GET'])
def main():
    if request.method == 'GET':
        return render_template('index.html')
```

APP.PY

- route '/predict' for predict model

```python
@ app.route('/predict', methods=['POST', 'GET'])
def predict():
    if request.method == 'GET':
        return render_template('index.html')
    if request.method == 'POST':
        features = [float(x) for x in request.form.values()]
    print(features)
    labels = model.predict([features])
    species = labels[0]
    if species == 0:
        s = "It is Iris Setosa"
    elif species == 1:
        s = "It is Iris VersiColor"
    else:
        s = "It is Iris Virginica"
    return s
```

APP.PY

- Initial host with port 8080

```python
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

INDEX.HTML

type structure:

```
<!DOCTYPE html>
<html>
  <head></head>
  <body></body>
</html>
```

INDEX.HTML

- In header type title and import ajax:

```
<title>Iris Classifier</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
></script>
```

INDEX.HTML

- In body type header, form, button and result:

```html
<h1><center>Iris Classifier</center></h1>
    <form id="myform" method="POST">
      <input type="text" name="sl"
        placeholder="Enter Sepal Length in cm"
      /><br /><br />
      <input
        type="text"
        name="sw"
        placeholder="Enter Sepal Width in cm"
      /><br /><br />
      <input
        type="text"
        name="pl"
        placeholder="Enter Petal Length in cm"
      /><br /><br />
      <input
        type="text"
        name="pw"
        placeholder="Enter Petal Width in cm"
      /><br /><br />
    </form>
    <button id="predict">Predict</button>
    <h2 id="result"></h2>
```

INDEX.HTML

- Below </html> addition ajax part:

```html
<script type="text/javascript">
  $(function () {
    $("#predict").click(function () {
      event.preventDefault();
      var form_data = new FormData($("#myform")[0]);
      console.log(form_data);
      $.ajax({
        type: "POST",
        url: "/predict",
        data: form_data,
        contentType: false,
        processData: false,
      })
        .done(function (data, textStatus, jqXHR) {
          $("#result").text(data);
        })
        .fail(function (data) {
          alert("error!");
        });
    });
  });
</script>
```

TEST RUN APP

- You can test run app by type command in terminal in flask_app folder

```
python3 app.py
```

DEPLOY TO AWS

- Create instance in EC2

  - AMI(Free Tier): Ubuntu 20.04 LTS SSD Volume Type (64-bit(x86))

  - Instance Type: t2.micro

  - Instance Details: Auto-assign Public IP – Use subnet setting (Enable)

  - Storage: 8GB, General Purpose SSD (gp2)

  - Security Group: Create new security group and add rule

    - SSH Source Anywhere

    - Custom TCP port 8080 Source Anywhere

REMOTE & TRANSFER FILE TO INSTANCE

- Remote
  - chmod 400 key.pem
  - ssh –i "key.pem" ubuntu@PublicDNS
  - sudo apt-get update
  - sudo apt install python3-pip
  - pip3 install numpy flask sklearn
- Transfer file
  - scp -r -i key.pem ./path-flask-app ubuntu@PublicDNS:~

RUN FLASK-APP IN EC2

- Remote to instance
- Move to flask-app folder
- Type python3 app.py to run flask-app
- In browser type Public DNS and port 8080
  - Example "ec2-12-345-678-90. ap-southeast-1.compute.amazonaws.com:8080"

REFERENCE

- https://medium.com/shapeai/deploying-flask-application-with-ml-models-on-aws-ec2-instance-3b9a1cec5e13