**29/11/21**-Amarnath

Flask is built upon Python

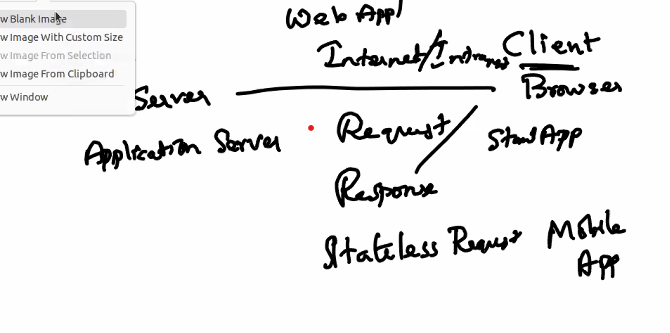
Client can be mobile application, browser, stand-alone application

Request and responses

Deploying ML models on web application and how to access them

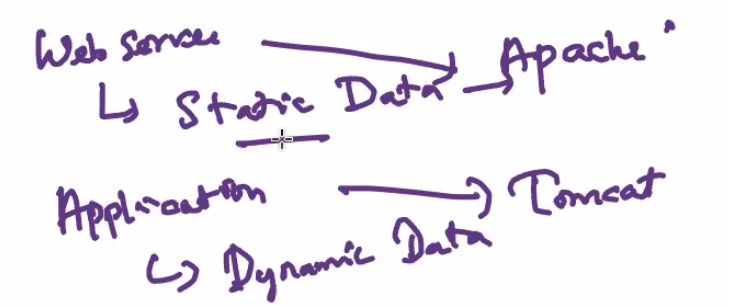
Docker-container

Kubernetes-orchestrating the container



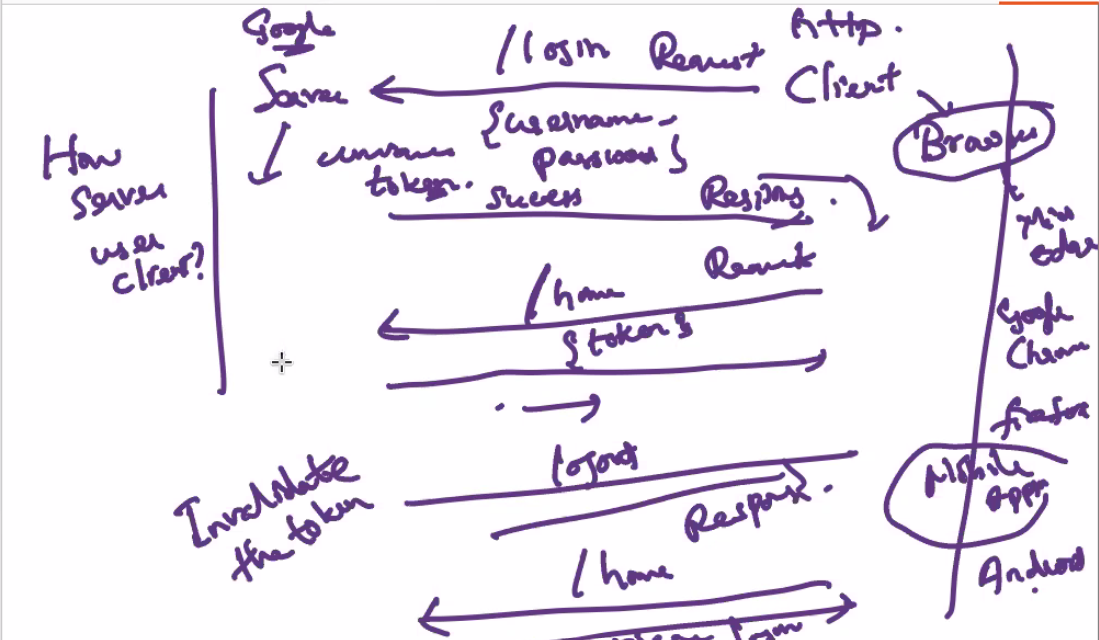
http-stateless server

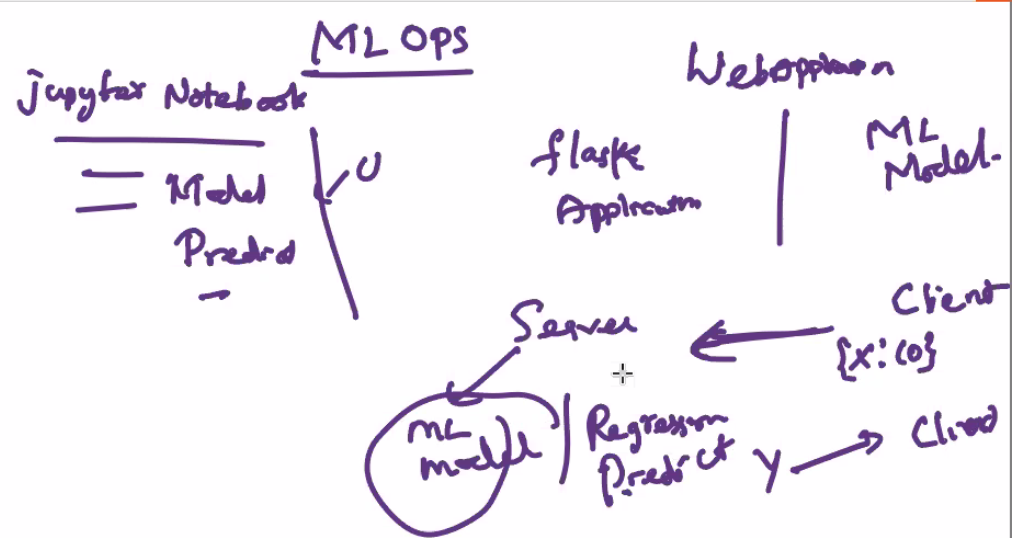
client requests the server and server responds to the client



If home page is requested now, authentication is already done.

Passes unique token back to the client



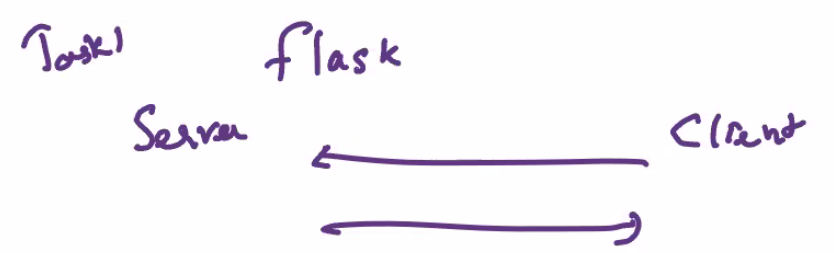


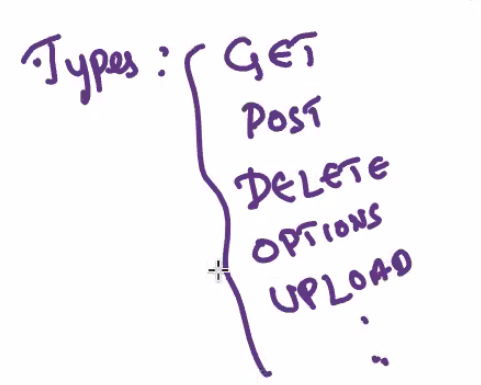
We have to create server,client and ML model.

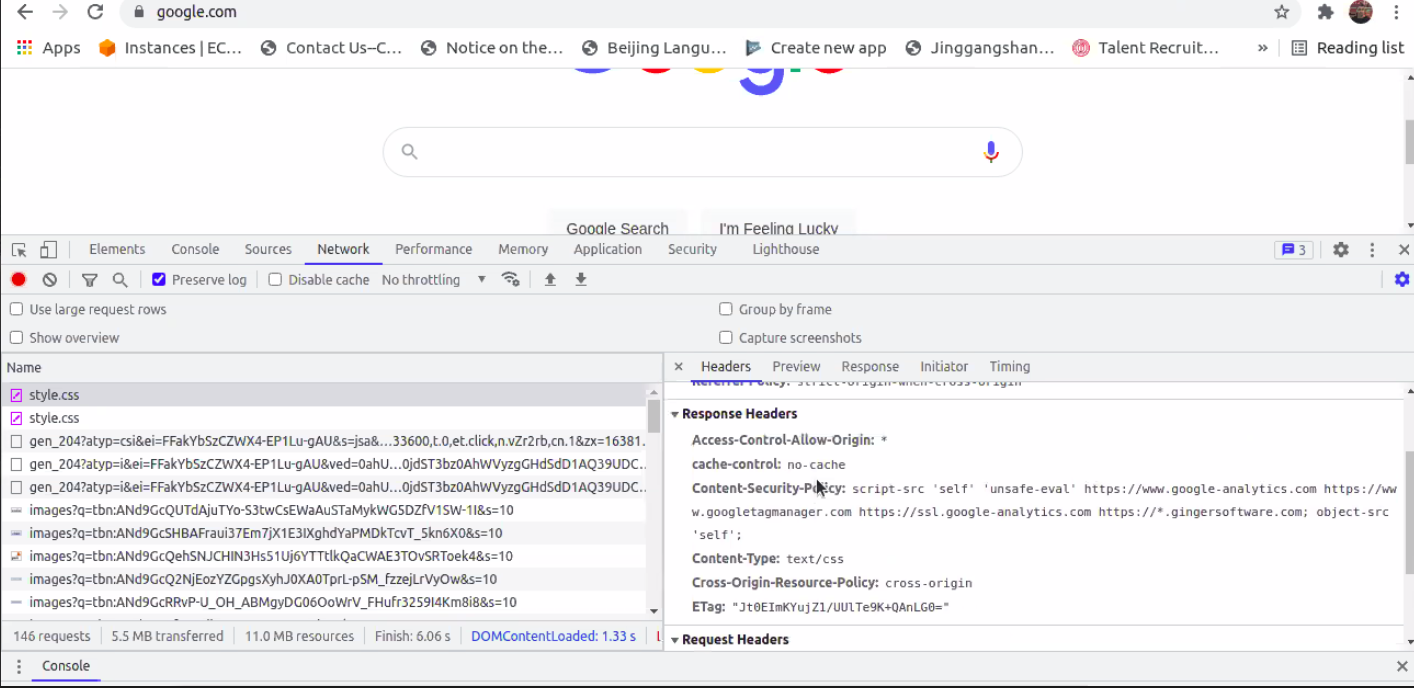
Flask, Django, servelets,structs,hybernets, .net,spring,J2EE is used to access web application

.net is based on C#

J2EE,Spring on Java



these are the types of request

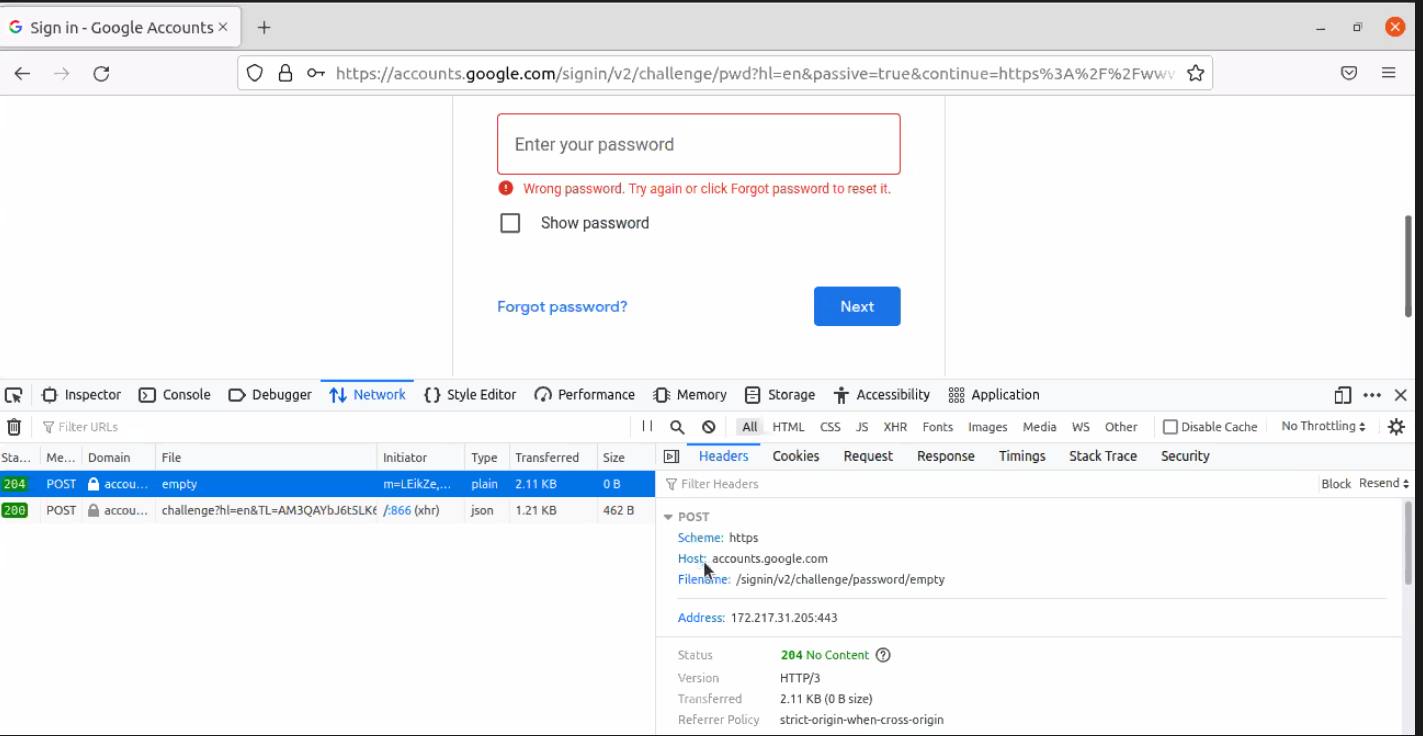


On signing in to a page-POST request

Use inspect to check request and response.

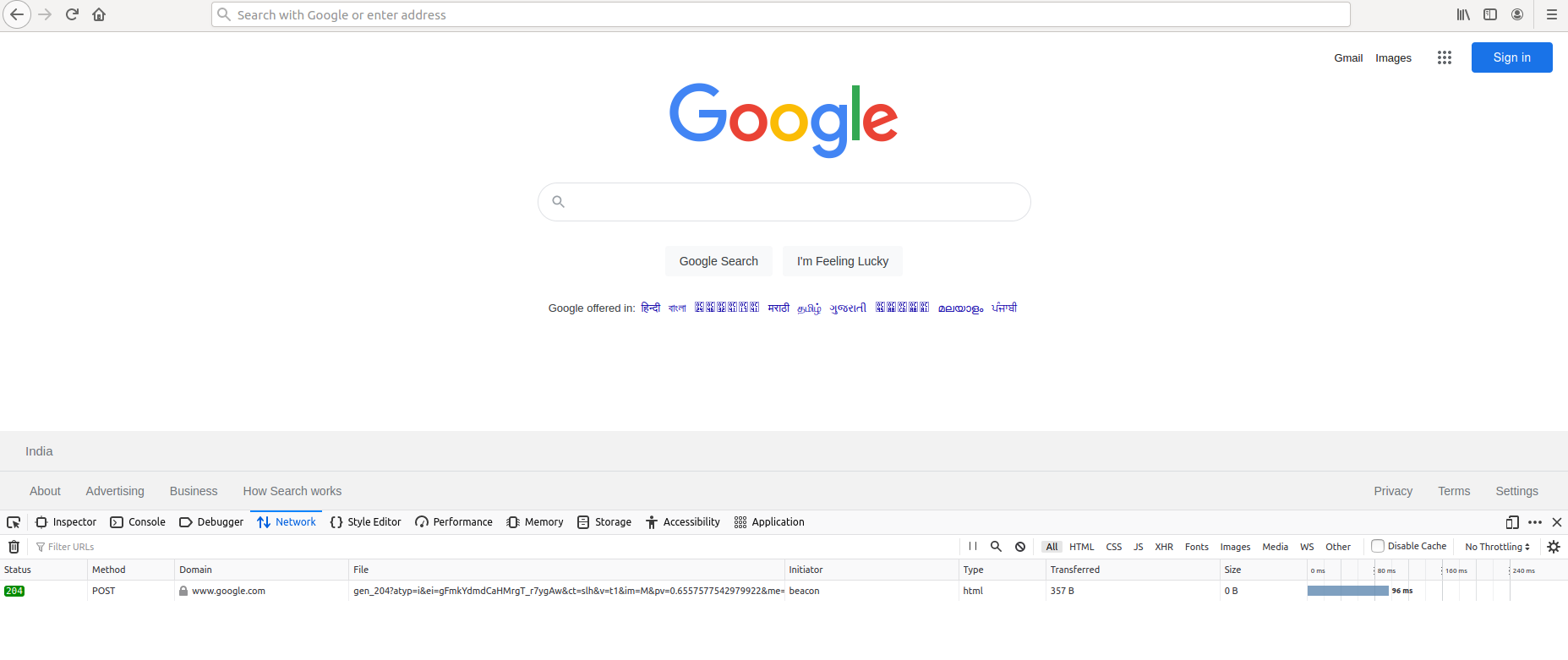
Instead of username, identifier is used.

POST request-confidential info

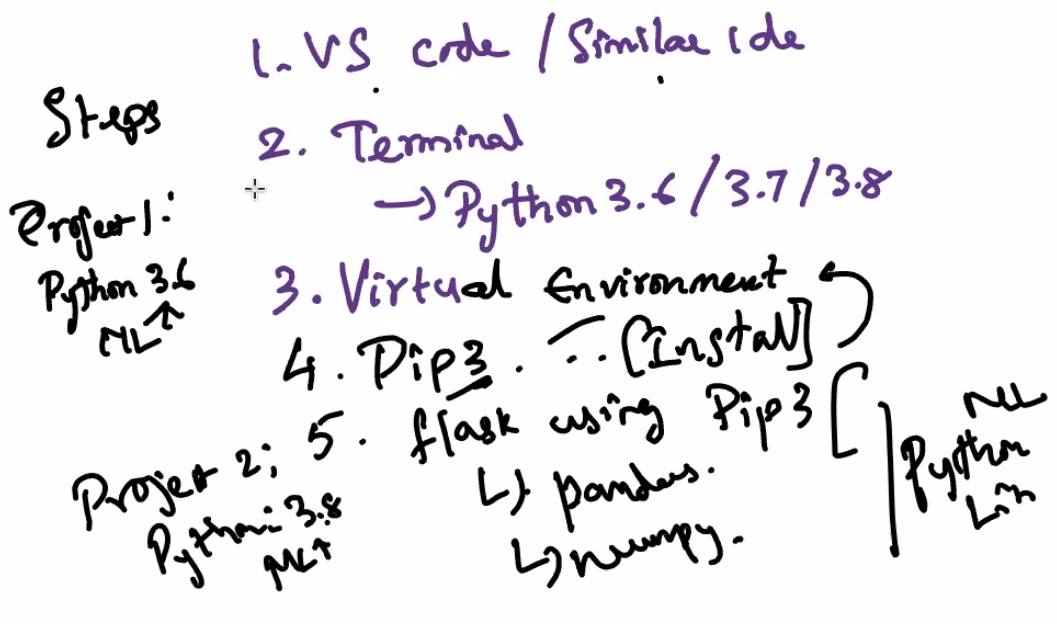


Diff b/w get and post

Search in linux



Using VS code





Pip3 –version

pip3 install Flask

from flask import Flask

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

@app.route('/')

def hello():

return "Hello World!"

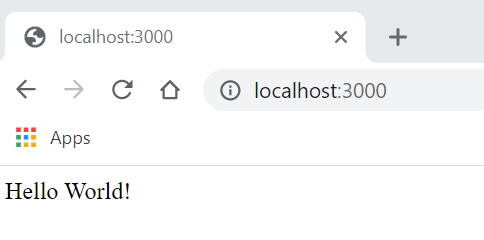
app.run(host='0.0.0.0', debug=True, port=3000)

python3 app.py

pip install -U Flask

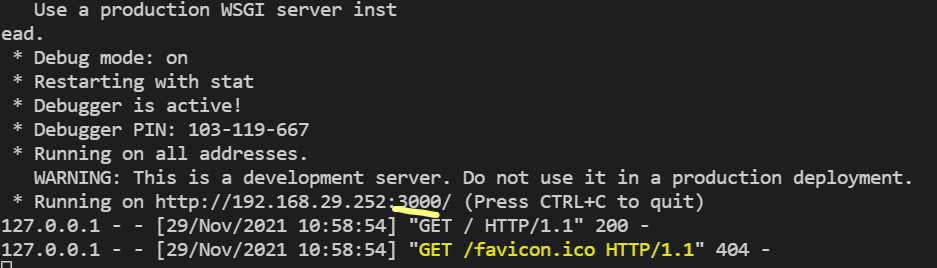
pip3 install Flask==2.0.2

pip install virtualenv



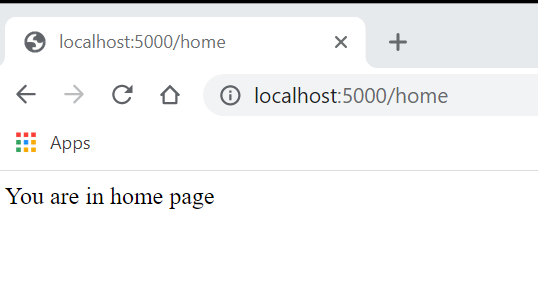
Ctrl Z will skip out running

Ctrl C will quit



For multiple routes-

from flask import Flaskapp = Flask(\_\_name\_\_)if \_\_name\_\_ == '\_\_main\_\_': # Router and Controller @app.route('/') def hello(): return "Hello World !" @app.route('/home') def home(): return "You are in home page" app.run(host='0.0.0.0', debug=True, port=5000)



If no mapping wrt / then Not Found error

Pass Dynamic data-like list

Router followed by the definition

from flask import Flask

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

    # Router and Controller

    @app.route('/home')

    def hello():

        return "Hello World !"

    @app.route('/home/dashboard')

    def home():

        return "You are in home page"

    @app.route('/list')

    def list():

        names = ['amarnath', 'ram', 'goyal']

        return str(names)

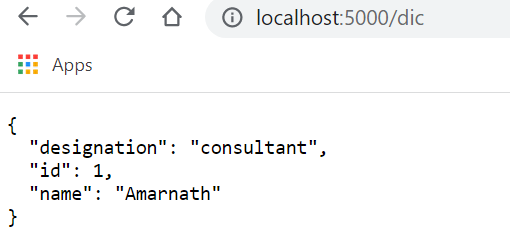
    @app.route('/dic')

    def dic():

        names = {'id':1, 'name':'Amarnath', 'designation':'consultant'}

        return names

    app.run(host='0.0.0.0', debug=True, port=5000)



from flask import Flask

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

    # Router and Controller

    @app.route('/home')

    def hello():

        return "Hello World !"

    @app.route('/home/dashboard')

    def home():

        return "You are in home page"

    @app.route('/list',methods=["GET"])

    def list():

        names = ['amarnath', 'ram', 'goyal']

        return str(names)

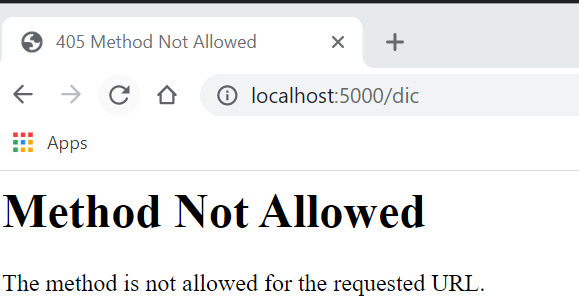
    @app.route('/dic',methods=["POST"])

    def dic():

        names = {'id':1, 'name':'Amarnath', 'designation':'consultant'}

        return names

    app.run(host='0.0.0.0', debug=True, port=5000)



To access control, use methods

Create new folder in root as templates

New file as home.html

<html>

    <body>

        Welcome to the Home page - HTML Page

    </body>

</html>

Render html page instead of returning the names

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

    # Router and Controller

    @app.route('/home')

    def hello():

        return "Hello World !"

    @app.route('/home/dashboard')

    def home():

        return "You are in home page"

    @app.route('/list', methods=["GET"])

    def list():

        names = ['amarnath', 'ram', 'goyal']

        return str(names)

    @app.route('/dic', methods=["GET", "POST"])

    def dic():

        names = {'id':1, 'name':'Amarnath', 'designation':'consultant'}

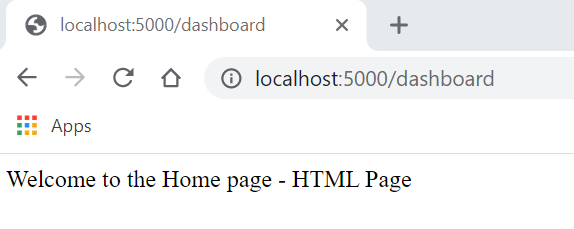
        return names

    @app.route('/dashboard')

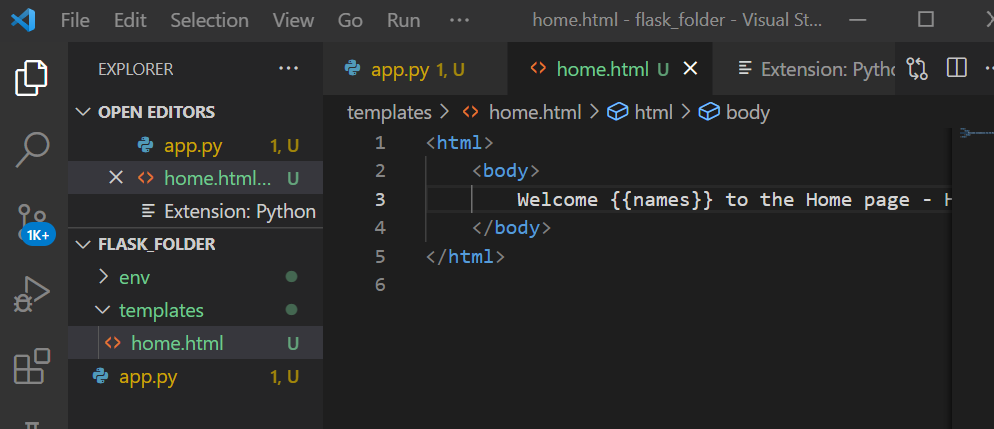
    def dashboard():

        return render\_template("home.html")

    app.run(host='0.0.0.0', debug=True, port=5000)



Transfer Dynamic data to html page



from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

    # Router and Controller

    @app.route('/home')

    def hello():

        return "Hello World !"

    @app.route('/home/dashboard')

    def home():

        return "You are in home page"

    @app.route('/list', methods=["GET"])

    def list():

        names = ['amarnath', 'ram', 'goyal']

        return str(names)

    @app.route('/dic', methods=["GET", "POST"])

    def dic():

        names = {'id':1, 'name':'Amarnath', 'designation':'consultant'}

        return names

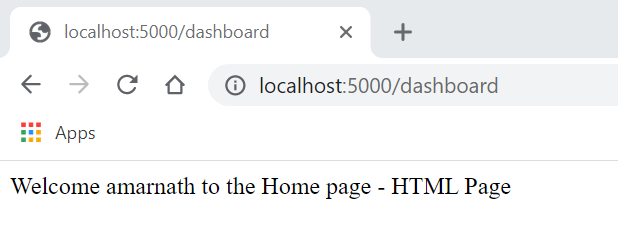
    @app.route('/dashboard')

    def dashboard():

        names = 'amarnath'

        return render\_template("home.html", names = names)

    app.run(host='0.0.0.0', debug=True, port=5000)



from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

    # Router and Controller

    @app.route('/home')

    def hello():

        return "Hello World !"

    @app.route('/home/dashboard')

    def home():

        return "You are in home page"

    @app.route('/list', methods=["GET"])

    def list():

        names = ['amarnath', 'ram', 'goyal']

        return str(names)

    @app.route('/dic', methods=["GET", "POST"])

    def dic():

        names = {'id':1, 'name':'Amarnath', 'designation':'consultant'}

        return names

    @app.route('/dashboard')

    def dashboard():

        name = 'amarnath'

        name\_list = ['Vishwanathan', 'Ramkumar', 'Cyndee']

        return render\_template("home.html", name = name, name\_list= name\_list)

    app.run(host='0.0.0.0', debug=True, port=5000)

<html>

    <body>

        Welcome {{name}} to the Dashboard page - HTML Page

        <br/>

        {% for n in name\_list %}

        <li>

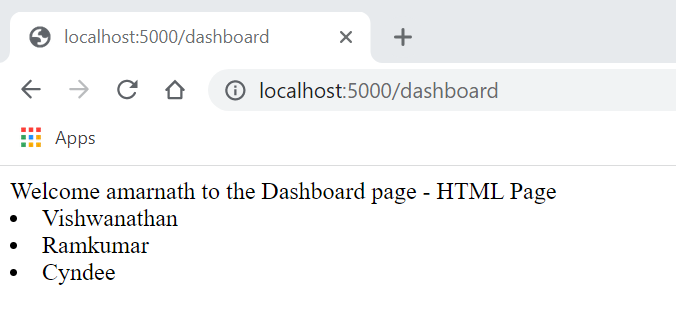
            {{n}}

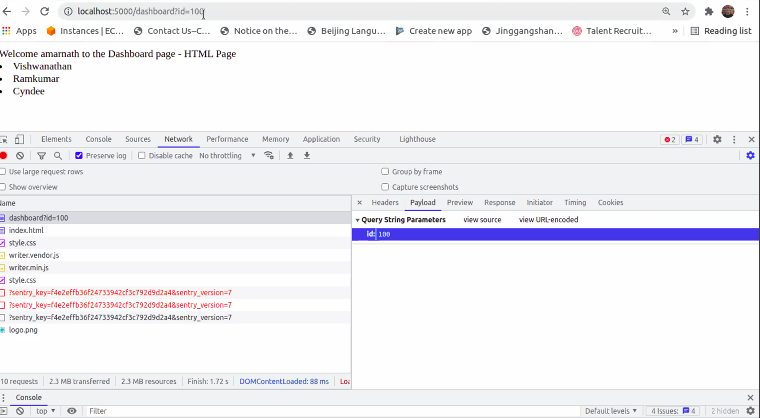
        </li>

        {%endfor%}

    </body>

</html>





from flask import Flask, render\_template, jsonify

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

    # Router and Controller

    @app.route('/home')

    def hello():

        return "Hello World !"

    @app.route('/home/dashboard')

    def home():

        return "You are in home page"

    @app.route('/list', methods=["GET"])

    def list():

        names = ['amarnath', 'ram', 'goyal']

        return str(names)

    @app.route('/dic', methods=["GET", "POST"])

    def dic():

        names = {'id':1, 'name':'Amarnath', 'designation':'consultant'}

        return jsonify(names)

    @app.route('/dashboard')

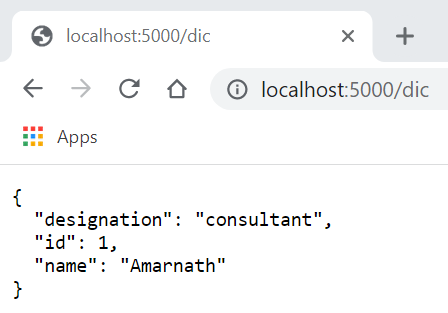
    def dashboard():

        name = 'amarnath'

        name\_list = ['Vishwanathan', 'Ramkumar', 'Cyndee']

        return render\_template("home.html", name = name, name\_list= name\_list)

    app.run(host='0.0.0.0', debug=True, port=5000)



from flask import Flask, render\_template, jsonify,request

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

    # Router and Controller

    @app.route('/home')

    def hello():

        return "Hello World !"

    @app.route('/home/dashboard')

    def home():

        return "You are in home page"

    @app.route('/list', methods=["GET"])

    def list():

        names = ['amarnath', 'ram', 'goyal']

        return str(names)

    @app.route('/dic', methods=["GET", "POST"])

    def dic():

        names = {'id':1, 'name':'Amarnath', 'designation':'consultant'}

        return jsonify(names)

    @app.route('/dashboard')

    def dashboard():

        print(request)

        print(request.args)

        print(request.args['id'])

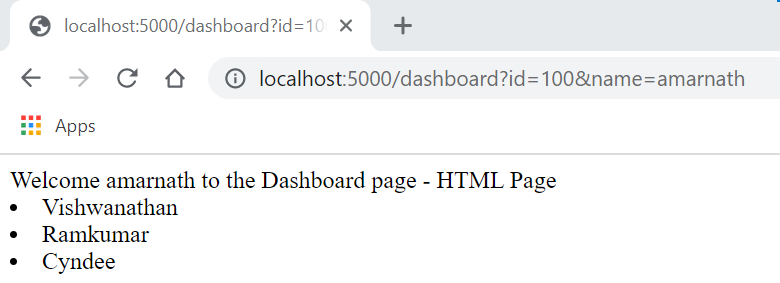
        print(request.args['name'])

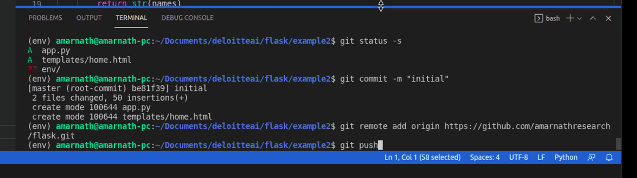
        name = 'amarnath'

        name\_list = ['Vishwanathan', 'Ramkumar', 'Cyndee']

        return render\_template("home.html", name = name, name\_list= name\_list)

    app.run(host='0.0.0.0', debug=True, port=5000)





[amarnathresearch/flask (github.com)](https://github.com/amarnathresearch/flask)

from flask import Flask, render\_template, jsonify,request

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

    # Router and Controller

    @app.route('/home')

    def hello():

        return "Hello World !"

    @app.route('/home/dashboard')

    def home():

        return "You are in home page"

    @app.route('/list', methods=["GET"])

    def list():

        names = ['amarnath', 'ram', 'goyal']

        return str(names)

    @app.route('/dic', methods=["GET", "POST"])

    def dic():

        names = {'id':1, 'name':'Amarnath', 'designation':'consultant'}

        return jsonify(names)

    @app.route('/dashboard')

    def dashboard():

        print(request)

        print(request.args)

        print(request.args['id'])

        print(request.args['name'])

        name=request.args['name']

        reg=request.args['id']

        name = 'amarnath'

        name\_list = ['Vishwanathan', 'Ramkumar', 'Cyndee']

        return render\_template("home.html", name = name,reg=reg, name\_list= name\_list)

    app.run(host='0.0.0.0', debug=True, port=5000)

<html>

    <body>

        Welcome {{name}} [of holding reg id {{reg}}] to the Dashboard page - HTML Page

        <br/>

        {% for n in name\_list %}

        <li>

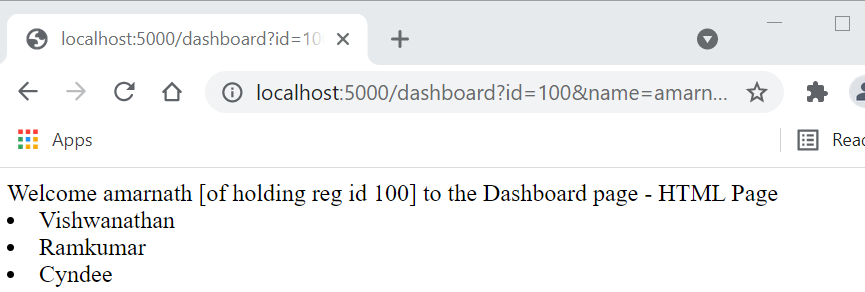
            {{n}}

        </li>

        {%endfor%}

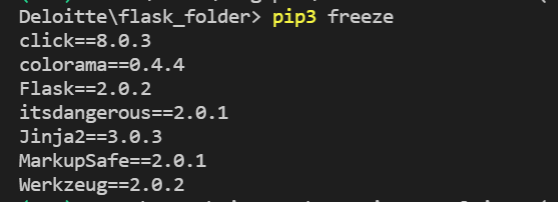
    </body>

</html>



<http://localhost:5000/dashboard?id=100&name=amarnath>

<https://raw.githubusercontent.com/amarnathresearch/flask/master/data/binary.csv>



pip3 install pandas

import pandas as pd-in code

@app.route('/analyse')

def analyse():

df = pd.read\_csv('./data/binary.csv')

print(df.head())

return "hello world"

Add a binary.csv file in data folder

from flask import Flask, render\_template, jsonify,request

import pandas as pd

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

    # Router and Controller

    @app.route('/home')

    def hello():

        return "Hello World !"

    @app.route('/home/dashboard')

    def home():

        return "You are in home page"

    @app.route('/list', methods=["GET"])

    def list():

        names = ['amarnath', 'ram', 'goyal']

        return str(names)

    @app.route('/dic', methods=["GET", "POST"])

    def dic():

        names = {'id':1, 'name':'Amarnath', 'designation':'consultant'}

        return jsonify(names)

    @app.route('/dashboard')

    def dashboard():

        print(request)

        print(request.args)

        print(request.args['id'])

        print(request.args['name'])

        name=request.args['name']

        reg=request.args['id']

        name = 'amarnath'

        name\_list = ['Vishwanathan', 'Ramkumar', 'Cyndee']

        return render\_template("home.html", name = name,reg=reg, name\_list= name\_list)

    @app.route('/analyse')

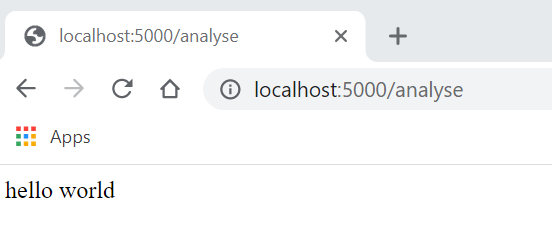
    def analyse():

        df = pd.read\_csv('./data/binary.csv')

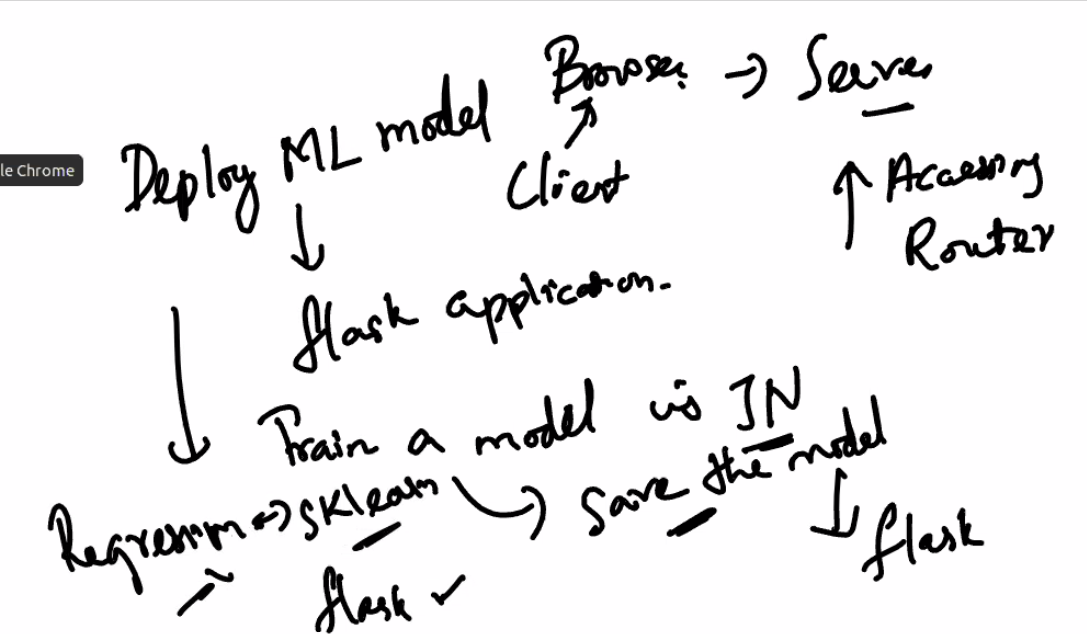
        print(df.head())

        return "hello world"

    app.run(host='0.0.0.0', debug=True, port=5000)



Regression model can be trained in flask also



Pip install sklearn

from flask import Flask, render\_template, jsonify,request

import pandas as pd

from sklearn import linear\_model

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

    # Router and Controller

    @app.route('/home')

    def hello():

        return "Hello World !"

    @app.route('/home/dashboard')

    def home():

        return "You are in home page"

    @app.route('/list', methods=["GET"])

    def list():

        names = ['amarnath', 'ram', 'goyal']

        return str(names)

    @app.route('/dic', methods=["GET", "POST"])

    def dic():

        names = {'id':1, 'name':'Amarnath', 'designation':'consultant'}

        return jsonify(names)

    @app.route('/dashboard')

    def dashboard():

        print(request)

        print(request.args)

        print(request.args['id'])

        print(request.args['name'])

        name=request.args['name']

        reg=request.args['id']

        name = 'amarnath'

        name\_list = ['Vishwanathan', 'Ramkumar', 'Cyndee']

        return render\_template("home.html", name = name,reg=reg, name\_list= name\_list)

    @app.route('/analyse')

    def analyse():

        df = pd.read\_csv('./data/binary.csv')

        print(df.head())

        return "hello world"

    @app.route('/train')

    def train():

        reg = linear\_model.LinearRegression()

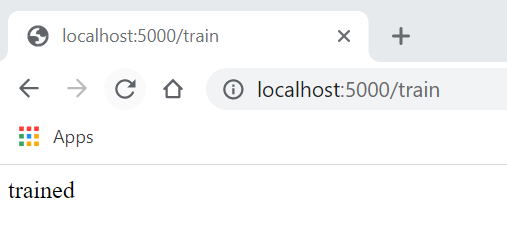
        reg.fit([[0, 0], [1, 1], [2, 2]], [0, 1, 2])

        print(reg.coef\_)

        print(reg.predict([[1, 1]]))

        return "trained"

    app.run(host='0.0.0.0', debug=True, port=5000)



Try to save this model

Pickle is one of the libraries to save the model

from flask import Flask, render\_template, jsonify,request

import pandas as pd

from sklearn import linear\_model

from sklearn.linear\_model import LogisticRegression

from sklearn import model\_selection

import pickle

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

    # Router and Controller

    @app.route('/home')

    def hello():

        return "Hello World !"

    @app.route('/home/dashboard')

    def home():

        return "You are in home page"

    @app.route('/list', methods=["GET"])

    def list():

        names = ['amarnath', 'ram', 'goyal']

        return str(names)

    @app.route('/dic', methods=["GET", "POST"])

    def dic():

        names = {'id':1, 'name':'Amarnath', 'designation':'consultant'}

        return jsonify(names)

    @app.route('/dashboard')

    def dashboard():

        print(request)

        print(request.args)

        print(request.args['id'])

        print(request.args['name'])

        name=request.args['name']

        reg=request.args['id']

        name = 'amarnath'

        name\_list = ['Vishwanathan', 'Ramkumar', 'Cyndee']

        return render\_template("home.html", name = name,reg=reg, name\_list= name\_list)

    @app.route('/analyse')

    def analyse():

        df = pd.read\_csv('./data/binary.csv')

        print(df.head())

        return "hello world"

    @app.route('/train')

    def train():

        url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

        names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

        dataframe = pd.read\_csv(url, names=names)

        array = dataframe.values

        X = array[:,0:8]

        Y = array[:,8]

        test\_size = 0.33

        seed = 7

        X\_train, X\_test, Y\_train, Y\_test = model\_selection.train\_test\_split(X, Y, test\_size=test\_size)

        model = LogisticRegression()

        model.fit(X\_train, Y\_train)

        # save the model to disk

        filename = './models/finalized\_model.sav'

        pickle.dump(model, open(filename, 'wb'))

        return "trained"

    @app.route('/test')

    def test():

        filename = './models/finalized\_model.sav'

        loaded\_model = pickle.load(open(filename, 'rb'))

        url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

        names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

        dataframe = pd.read\_csv(url, names=names)

        array = dataframe.values

        X = array[:,0:8]

        Y = array[:,8]

        test\_size = 0.33

        seed = 7

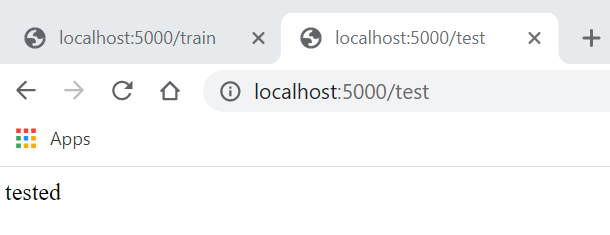
        X\_train, X\_test, Y\_train, Y\_test = model\_selection.train\_test\_split(X, Y, test\_size=test\_size)

        result = loaded\_model.score(X\_test, Y\_test)

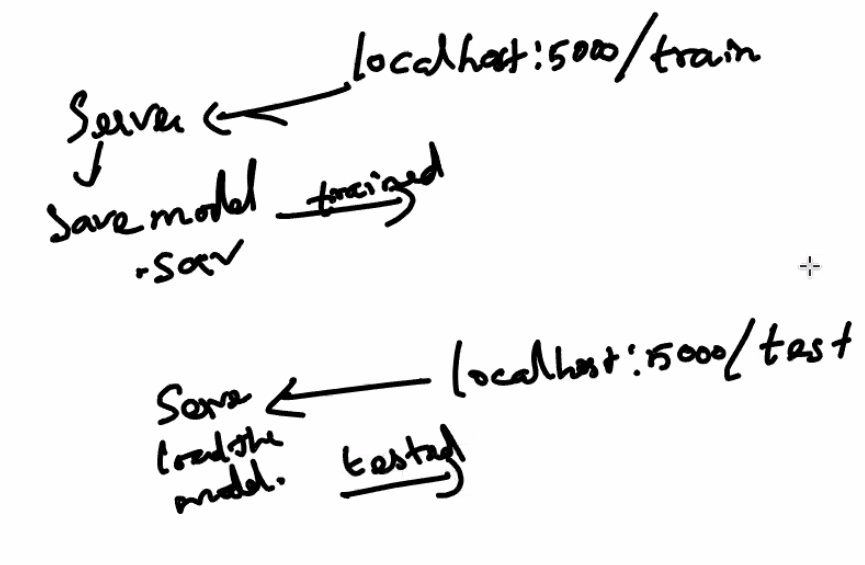
        print(result)

        return "tested"

    app.run(host='0.0.0.0', debug=True, port=5000)

****

<https://github.com/amarnathresearch/flask>



IRIS Dataset

from flask import Flask, render\_template, jsonify, request

import pandas as pd

from sklearn import model\_selection

from sklearn.linear\_model import LogisticRegression

import pickle

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

    # Router and Controller

    @app.route('/home')

    def hello():

        return "Hello World !"

    @app.route('/home/dashboard')

    def home():

        return "You are in home page"

    @app.route('/list', methods=["GET"])

    def list():

        names = ['amarnath', 'ram', 'goyal']

        return str(names)

    @app.route('/dic', methods=["GET", "POST"])

    def dic():

        names = {'id':1, 'name':'Amarnath', 'designation':'consultant'}

        return jsonify(names)

    @app.route('/dashboard')

    def dashboard():

        print(request)

        print(request.args)

        print(request.args['id'])

        print(request.args['name'])

        name = request.args['name']

        registration = request.args['id']

        name\_list = ['Vishwanathan', 'Ramkumar', 'Cyndee']

        return render\_template("home.html", name = name, registration = registration, name\_list= name\_list)

    @app.route('/analyse')

    def analyse():

        df = pd.read\_csv('./data/binary.csv')

        data = df.head()

        print(data)

        return "hello world"

# Using Jupyter nodebook, we will train the model and save

    @app.route('/train')

    def train():

        url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

        names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

        dataframe = pd.read\_csv(url, names=names)

        array = dataframe.values

        X = array[:,0:8]

        Y = array[:,8]

        test\_size = 0.33

        seed = 7

        X\_train, X\_test, Y\_train, Y\_test = model\_selection.train\_test\_split(X, Y, test\_size=test\_size)

        model = LogisticRegression()

        model.fit(X\_train, Y\_train)

        # save the model to disk

        filename = './models/finalized\_model.sav'

        pickle.dump(model, open(filename, 'wb'))

        return "trained"

    @app.route('/test')

    def test():

        filename = './models/finalized\_model.sav'

        loaded\_model = pickle.load(open(filename, 'rb'))

        url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

        names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

        dataframe = pd.read\_csv(url, names=names)

        array = dataframe.values

        X = array[:,0:8]

        Y = array[:,8]

        test\_size = 0.33

        seed = 7

        X\_train, X\_test, Y\_train, Y\_test = model\_selection.train\_test\_split(X, Y, test\_size=test\_size)

        result = loaded\_model.score(X\_test, Y\_test)

        print(result)

        return "tested"

    @app.route('/iris/test')

    def iris\_test():

        pkl\_filename = './models/iris\_model.pkl'

        with open(pkl\_filename, 'rb') as file:

            pickle\_model = pickle.load(file)

        sample = [[6.4, 2.8, 5.6, 2.1]]

        Ypredict = pickle\_model.predict(sample)

        print("Ypredict", Ypredict)

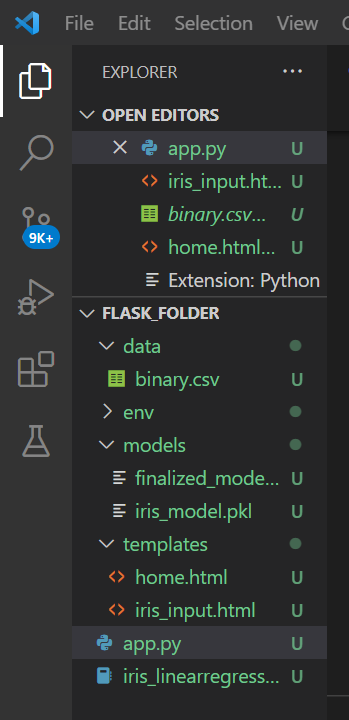
        return str(Ypredict[0])

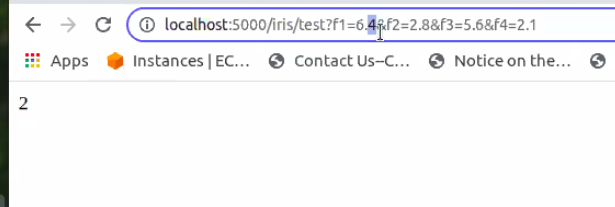
    @app.route('/iris/input')

    def iris\_input():

        return render\_template("iris\_input.html")

    app.run(host='0.0.0.0', debug=True, port=5000)

iris\_model.pkl saved



<http://localhost:5000/iris/test?f1=6.0&f2=1.8&f3=1.6&f4=2.1>

@app.route('/iris/test') def iris\_test(): print(request.args) f1 = request.args['f1'] f2 = request.args['f2'] f3 = request.args['f3'] f4 = request.args['f4'] pkl\_filename = './models/iris\_model.pkl' with open(pkl\_filename, 'rb') as file:

pickle\_model = pickle.load(file)

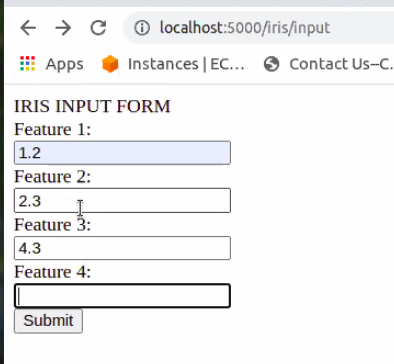
sample = [[f1, f2, f3, f4]]

Ypredict = pickle\_model.predict(sample)

print("Ypredict", Ypredict)

return str(Ypredict[0])

Create iris\_input.html in templates



Submit and see the output

<html>

    <body>

        IRIS INPUT FORM

        <form action="/iris/test">

            <label for="f1">Feature 1:</label><br>

            <input type="text" id="f1" name="f1" ><br>

            <label for="f2">Feature 2:</label><br>

            <input type="text" id="f2" name="f2" ><br>

            <label for="f2">Feature 3:</label><br>

            <input type="text" id="f3" name="f3" ><br>

            <label for="f2">Feature 4:</label><br>

            <input type="text" id="f4" name="f4" ><br>

            <input type="submit" value="Submit">

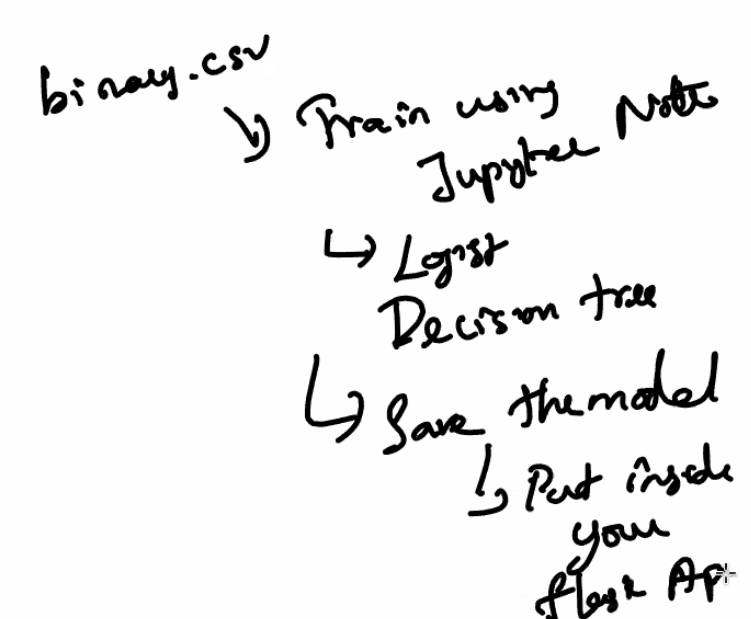
          </form>

    </body>

</html>

Will give the labels as output-0 for setosa, 1 for versicolor, 2 for virginica

BINARY.CSV



<http://localhost:5000/admin/test?gre=300&gpa=3.7&rank=3>

from flask import Flask, render\_template, jsonify, request

import pandas as pd

from sklearn import model\_selection

from sklearn.linear\_model import LogisticRegression

import pickle

app = Flask(\_\_name\_\_)

if \_\_name\_\_ == '\_\_main\_\_':

    # Router and Controller

    @app.route('/home')

    def hello():

        return "Hello World !"

    @app.route('/home/dashboard')

    def home():

        return "You are in home page"

    @app.route('/list', methods=["GET"])

    def list():

        names = ['amarnath', 'ram', 'goyal']

        return str(names)

    @app.route('/dic', methods=["GET", "POST"])

    def dic():

        names = {'id':1, 'name':'Amarnath', 'designation':'consultant'}

        return jsonify(names)

    @app.route('/dashboard')

    def dashboard():

        print(request)

        print(request.args)

        print(request.args['id'])

        print(request.args['name'])

        name = request.args['name']

        registration = request.args['id']

        name\_list = ['Vishwanathan', 'Ramkumar', 'Cyndee']

        return render\_template("home.html", name = name, registration = registration, name\_list= name\_list)

    @app.route('/analyse')

    def analyse():

        df = pd.read\_csv('./data/binary.csv')

        data = df.head()

        print(data)

        return "hello world"

    @app.route('/admit/test', methods=['GET'])

    def admit\_test():

        print(request.args)

        gre = request.args['gre']

        gpa = request.args['gpa']

        rank = request.args['rank']

        pkl\_filename = './models/admit\_model.pkl'

        with open(pkl\_filename, 'rb') as file:

            pickle\_model = pickle.load(file)

        sample = [[gre, gpa, rank]]

        Ypredict = pickle\_model.predict(sample)

        print("Ypredict", Ypredict)

        return str(Ypredict[0])

    @app.route('/admit/input')

    def iris\_input():

        return render\_template("admit\_input.html")

    app.run(host='0.0.0.0', debug=True, port=5000)

<html>

    <body>

        BINARY INPUT FORM

        <form action="/admit/test">

            <label for="f1">GRE:</label><br>

            <input type="text" id="gre" name="gre" ><br>

            <label for="f2">GPA:</label><br>

            <input type="text" id="gpa" name="gpa" ><br>

            <label for="f2">RANK:</label><br>

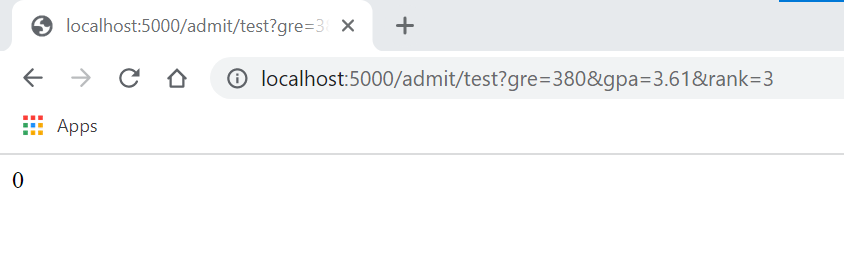
            <input type="text" id="rank" name="rank" ><br>

            <input type="submit" value="Submit">

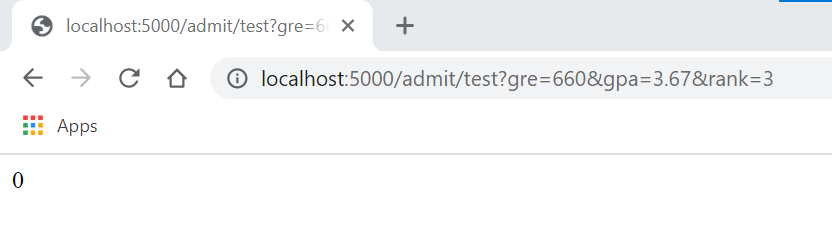
          </form>

    </body>

</html>



Accuracy of the model ~70%.



try to improve the accuracy, wrong prediction for given input here

SENTIMENT ANALYSIS

<https://github.com/amarnathresearch/sentiment_analysis/blob/master/main.ipynb>

For text data

[amar\_sentiment\_analysis.ipynb - Colaboratory (google.com)](https://colab.research.google.com/drive/14GdhrJbOxjTxTe-dsuVk0G0M42W0nxUb?usp=sharing)

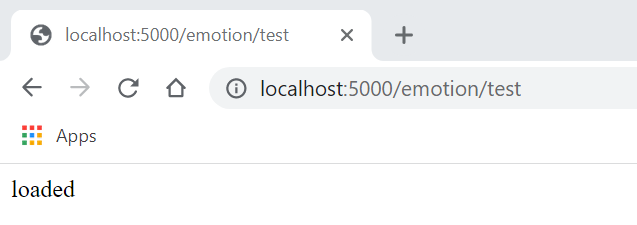
Download the model text\_analysis

#Sentiment Analysis

    @app.route('/emotion/test',methods=['GET'])

    def emotion\_test():

        return ("loaded")

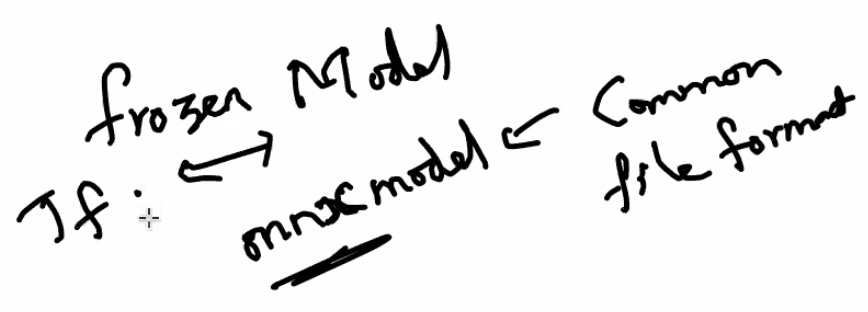


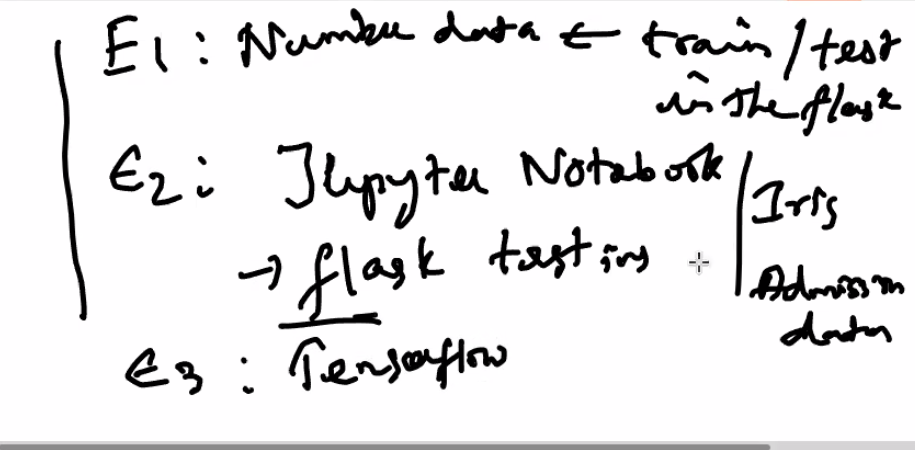
import os

import tensorflow as tf

from tensorflow import keras

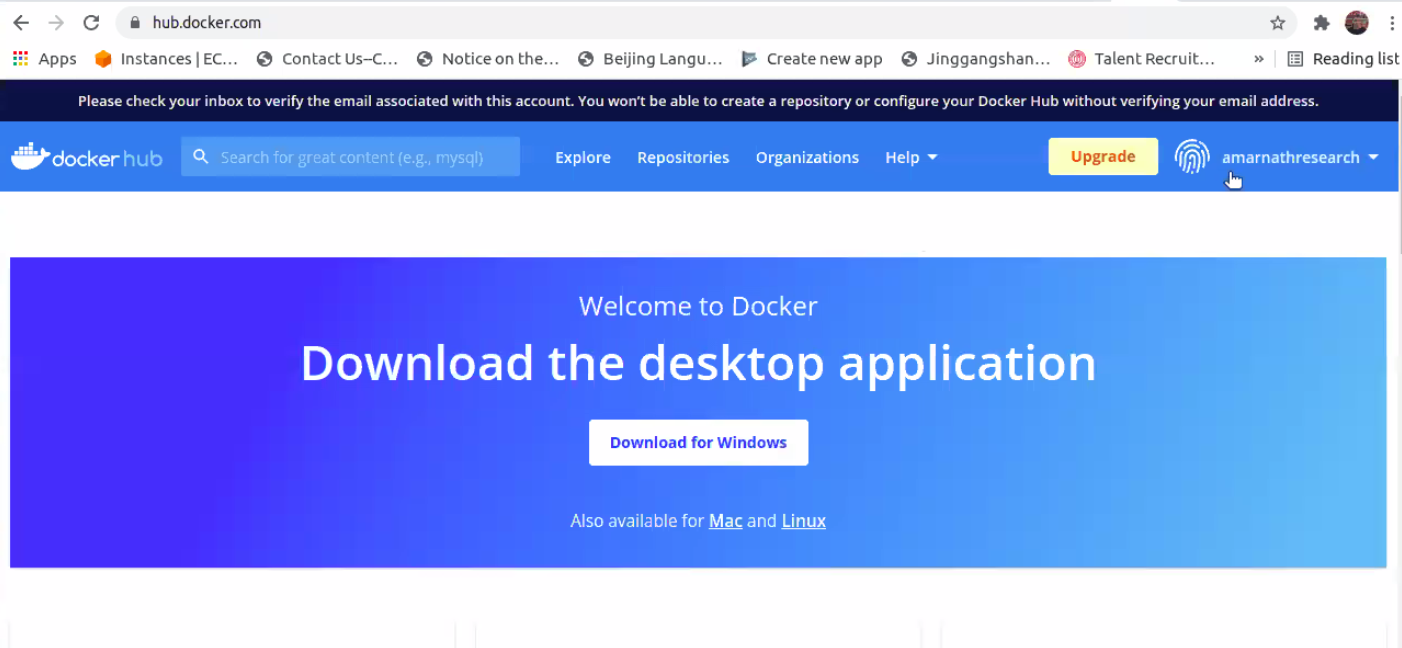
pip install tensorflow





Download Docker for windows 10 enterprise -800MB

<https://docs.docker.com/desktop/windows/install/>



Sign in to <https://hub.docker.com/> and then download

For linux-sudo apt install docker.io