**Main function**

from fastapi import FastAPI, HTTPException, status, Response, Request, Form

from fastapi.responses import HTMLResponse

from fastapi.templating import Jinja2Templates

from pydantic import BaseModel

import pickle

import uvicorn

app = FastAPI()

templates = Jinja2Templates(directory="templates")

pickle\_in = open("classifier.pkl", "rb")

classifier = pickle.load(pickle\_in)

class Prostate(BaseModel):

    radius: int

    texture: int

    perimeter: int

    area: int

    smoothness: float

    compactness: float

    symmetry: float

    fractal\_dimension: float

# main api function that get triggered when receving a request

@app.post("/prostates", status\_code=status.HTTP\_201\_CREATED)

def create\_heart(request: Request, Radius: str = Form(...), Texture: str = Form(...), Perimeter: str = Form(...), Area: str = Form(...), Smoothness: str = Form(...), Compactness: str = Form(...), Symmetry: str = Form(...), Fractal\_dimension: str = Form(...)):

    prediction = classifier.predict([[Radius, Area,

                                    Smoothness, Compactness, Symmetry]])

    if prediction[0] == 0:

        return templates.TemplateResponse("result.html", {"request": request, "result": "Malignant Tumor"})

    else:

        return templates.TemplateResponse("result.html", {"request": request, "result": "Benign Tumor"})

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

    uvicorn.run(app, host='0.0.0.0', port=8000)

**Prostate model code**

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report

import os

from sklearn.metrics import accuracy\_score

import pickle

# loading the dataset

data = pd.read\_csv('./Prostate\_Cancer.csv')

# getting the first 10 lines of the dataset

data.head()

# descripting the dataset

data.describe()

# droping the id column

data = data.drop(['id'], axis=1)

# replace Malignant  with 0 and Benign  with 1

data['diagnosis\_result'].replace({'M':0,'B':1},inplace=True)

# getting the corolation matrix of the data

corr\_metrics = data.corr()

corr\_metrics.style.background\_gradient()

# eliminate the features that have a low correlation like Fracture dimension, Texture, preimeter

data = data.drop(['fractal\_dimension', 'texture', 'perimeter'], axis=1)

y = data['diagnosis\_result'] # Labels

# splitting the data into train and test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state=10)

# importing the random forest classifier

forest = RandomForestClassifier(n\_estimators = 50)

# training the model with the imported data

forest.fit(X\_train,y\_train)

pred\_forest = forest.predict(X\_test)

# getting the classification report

class\_rep\_forest = classification\_report(y\_test, pred\_forest)

print("Forest Classifier: \n", class\_rep\_forest)

# getting the model accurecy

score=accuracy\_score(y\_test,pred\_forest)

# saving the model for future use

pickle\_out = open("classifier.pkl","wb")

pickle.dump(forest, pickle\_out)

pickle\_out.close()