

<b>TEAM ID</b>	PNT2022TMID42133
<b>PROJECT NAME</b>	Statistical Machine Learning Approaches to Liver Disease Prediction
<b>COLLEGE NAME</b>	AVS College Of Technology

## Build python code

Python package builds are the product of coordination between a few different tools driven by a standardized process. One of the biggest choices you have as a package author is which set of tools to use. It can be difficult to assess the nuances of each, especially if you're new to packaging. Fortunately, tools are standardizing around the same core workflow, so once you learn it you've got the agility to switch between tools with minimal effort. This article covers what you first need to learn about the pieces of the Python build system itself.

### Importing Libraries

```
from flask import Flask, render_template, request # Flask is a application
# used to run/serve our application
# request is used to access the file which is uploaded by the user in our application
# render_template is used for rendering the html pages
import pickle # pickle is used for serializing and de-serializing Python object structures
```

Libraries required for the app to run are to be imported.

Creating our flask app and loading the model

```
app=Flask(__name__) # our flask app
```

Now after all the libraries are imported, we will be creating our flask app. and then load our model into our flask app.

## Routing to the html Page:

@app.route is used to route the application where it should route to.

‘/’ URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page is rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Here, “home.html” is rendered when the home button is clicked on the UI and “index.html” is rendered when the predict button is clicked.

Firstly, we are rendering the home.html template and from there we are navigating to our prediction page that is upload.html. We enter input values here and these values are sent to the loaded model and the resultant output is displayed on index.html.

```
@app.route('/data_predict', methods=['POST']) # route for our prediction
def predict():
    age = request.form['age'] # requesting for age data
    gender = request.form['gender'] # requesting for gender data
    tb = request.form['tb'] # requesting for Total_Bilirubin data
    db = request.form['db'] # requesting for Direct_Bilirubin data
    ap = request.form['ap'] # requesting for Alkaline_Phosphotase data
    aa1 = request.form['aa1'] # requesting for Alamine_Aminotransferase data
    aa2 = request.form['aa2'] # requesting for Aspartate_Aminotransferase data
    tp = request.form['tp'] # requesting for Total_Protiens data
    a = request.form['a'] # requesting for Albumin data
    agr = request.form['agr'] # requesting for Albumin_and_Globulin_Ratio data

    # covertng data into float format
    data = [[float(age), float(gender), float(tb), float(db), float(ap), float(aa1), float(aa2), float(tp),
    float(a), float(agr))]

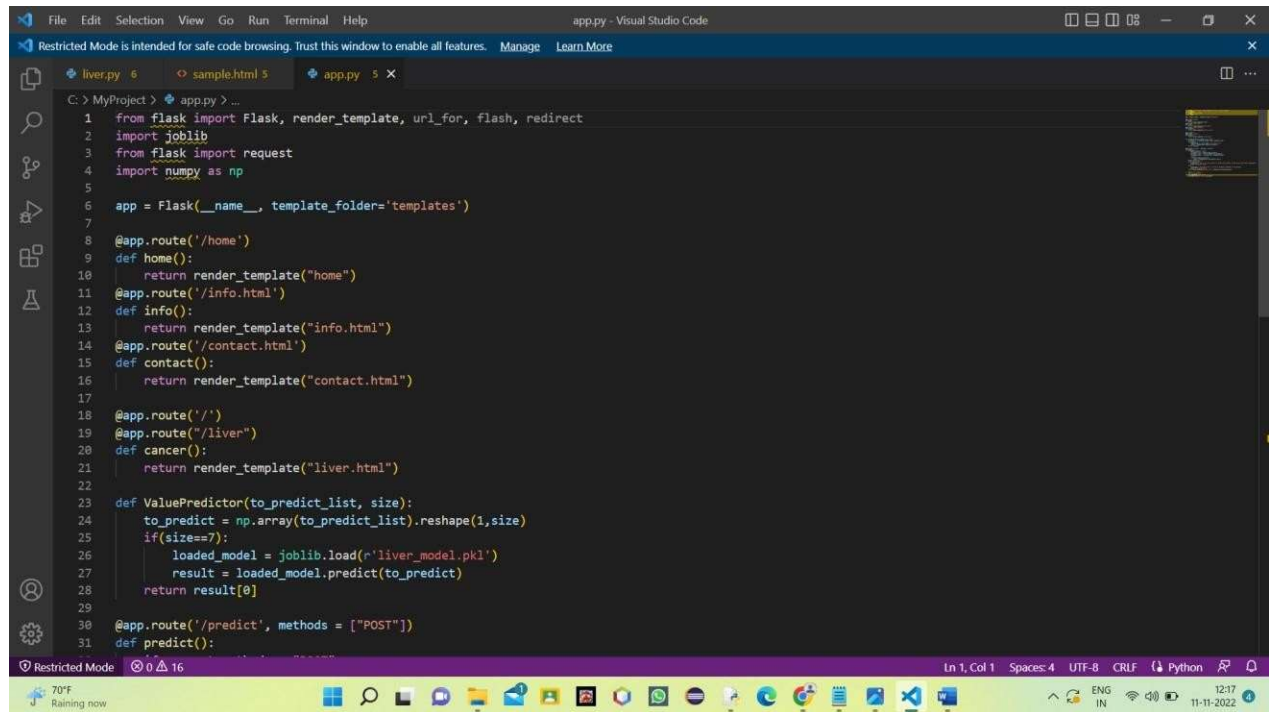
    # Loading model which we saved
    model = pickle.load(open('liver_analysis.pkl', 'rb'))

    prediction= model.predict(data)[0]
    if (prediction == 1):
        return render_template('noChance.html', prediction='You have a liver desease problem, You must and :
    else:
        return render_template('chance.html', prediction='You dont have a liver desease problem')

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

Here the route for prediction is given and necessary steps are performed in order to get the predicted output.

Lastly, we run our app on the local host. Here we are running it on localhost:8087



The screenshot shows a Visual Studio Code editor window titled 'app.py - Visual Studio Code'. The editor is in 'Restricted Mode' and displays a Python file named 'app.py'. The code is a Flask web application with several routes and a machine learning prediction function. The routes include a home page, an info page, a contact page, a liver page, and a prediction endpoint. The prediction endpoint uses a pre-trained model to predict the value of a liver based on input features.

```
1 from flask import Flask, render_template, url_for, flash, redirect
2 import joblib
3 from flask import request
4 import numpy as np
5
6 app = Flask(__name__, template_folder='templates')
7
8 @app.route('/home')
9 def home():
10     return render_template("home")
11
12 @app.route('/info.html')
13 def info():
14     return render_template("info.html")
15
16 @app.route('/contact.html')
17 def contact():
18     return render_template("contact.html")
19
20 @app.route('/')
21 @app.route("/liver")
22 def cancer():
23     return render_template("liver.html")
24
25 def ValuePredictor(to_predict_list, size):
26     to_predict = np.array(to_predict_list).reshape(1,size)
27     if(size==7):
28         loaded_model = joblib.load(r'liver_model.pkl')
29         result = loaded_model.predict(to_predict)
30         return result[0]
31
32 @app.route('/predict', methods = ["POST"])
33 def predict():
```