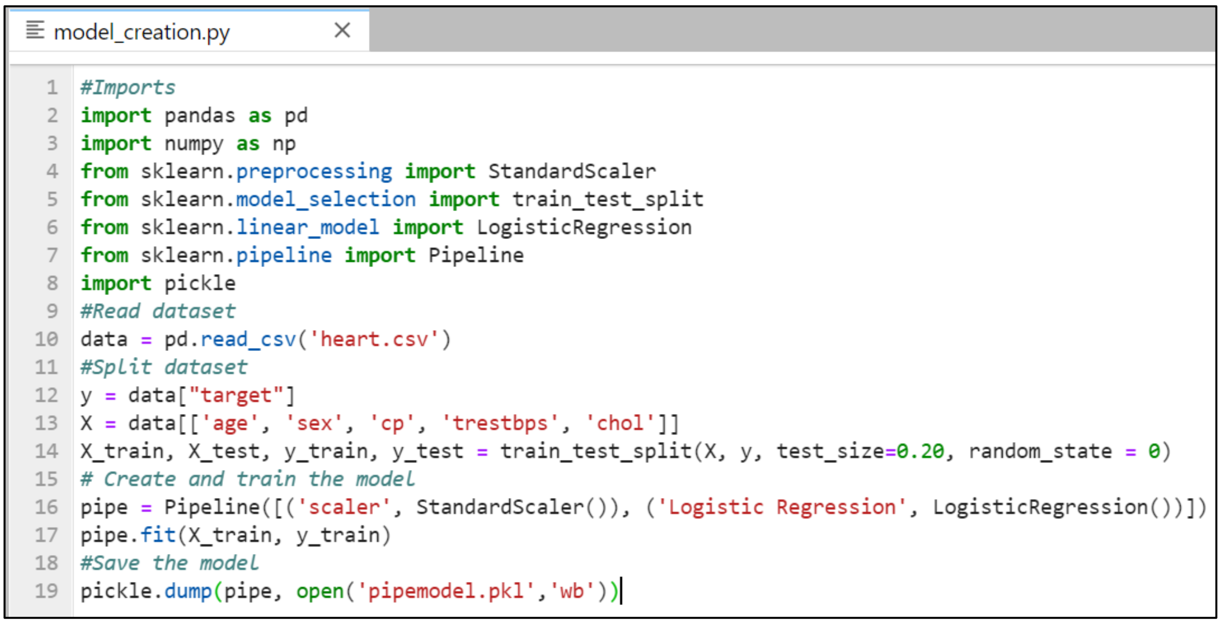
Deployment on Flask and Postman API

In this report, a demo of creating a simple machine learning model and deploy it using Postman API. The used dataset is derived from Kaggle. It discusses the probability of a patient to get a heart attack or not using Logistic regression. The model takes 5 features – various heart rate reading including blood sugar and pressure -. The patient readings are passed from the API to the web app to interact with the model, then return the result back to the API again in json format to be printed in console.

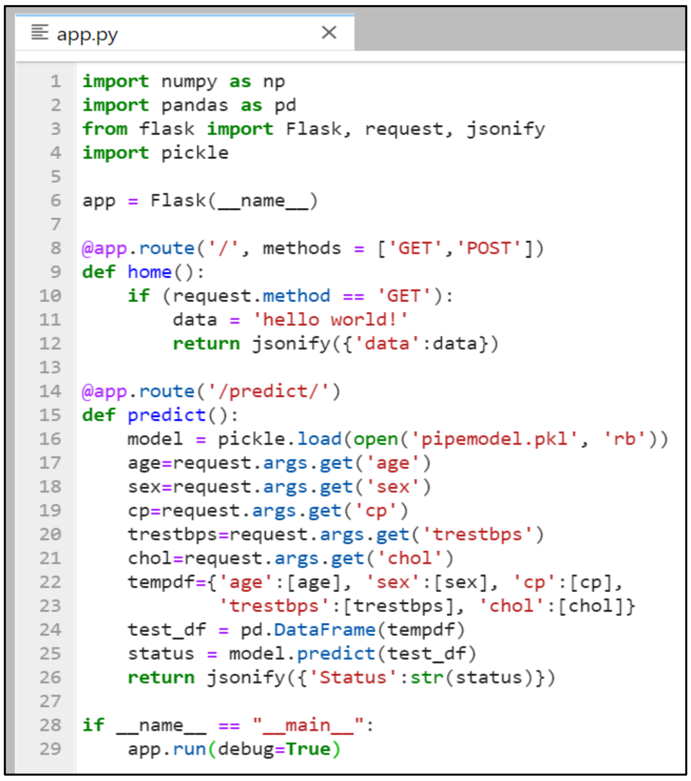
The model code (model_creation.py):



model_creation.py

```python
#Imports
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
import pickle
#Read dataset
data = pd.read_csv('heart.csv')
#Split dataset
y = data["target"]
X = data[['age', 'sex', 'cp', 'trestbps', 'chol']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state = 0)
# Create and train the model
pipe = Pipeline([('scaler', StandardScaler()), ('Logistic Regression', LogisticRegression())])
pipe.fit(X_train, y_train)
#Save the model
pickle.dump(pipe, open('pipemodel.pkl','wb'))
```

Web app code (app.py):

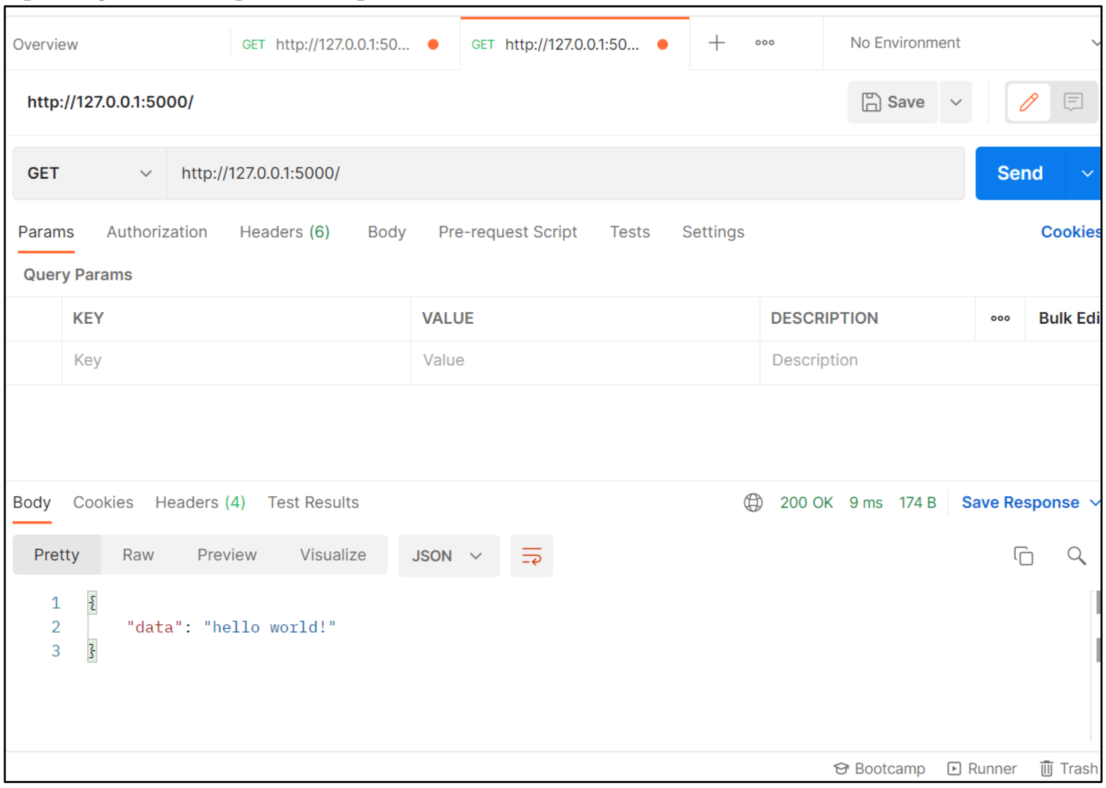


app.py

```python
import numpy as np
import pandas as pd
from flask import Flask, request, jsonify
import pickle

app = Flask(__name__)

@app.route('/', methods = ['GET','POST'])
def home():
    if (request.method == 'GET'):
        data = 'hello world!'
        return jsonify({'data':data})

@app.route('/predict/')
def predict():
    model = pickle.load(open('pipemodel.pkl', 'rb'))
    age=request.args.get('age')
    sex=request.args.get('sex')
    cp=request.args.get('cp')
    trestbps=request.args.get('trestbps')
    chol=request.args.get('chol')
    tempdf={'age':[age], 'sex':[sex], 'cp':[cp],
            'trestbps':[trestbps], 'chol':[chol]}
    test_df = pd.DataFrame(tempdf)
    status = model.predict(test_df)
    return jsonify({'Status':str(status)})

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

Output:

1. Opening a new request and post the URL



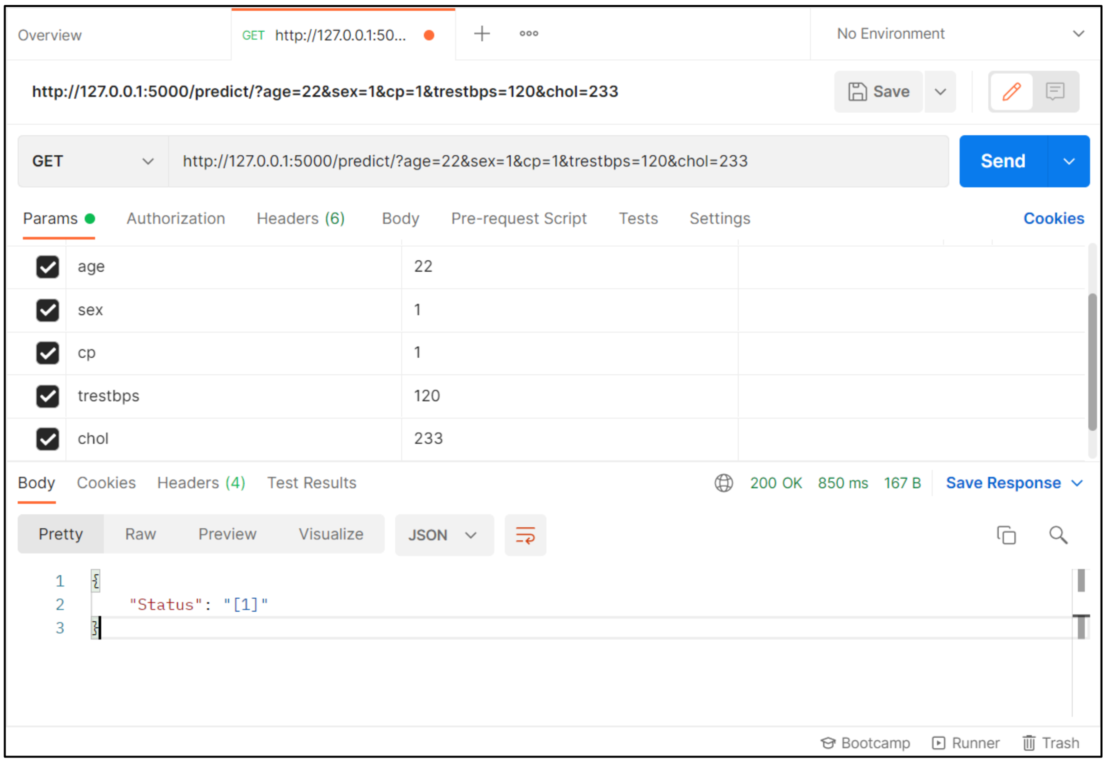
http://127.0.0.1:5000/

GET http://127.0.0.1:5000/

200 OK 9 ms 174 B

```
{
    "data": "hello world!"
}
```

2. Modifying the URL to get the prediction and add the keys and values



http://127.0.0.1:5000/predict/?age=22&sex=1&cp=1&trestbps=120&chol=233

GET http://127.0.0.1:5000/predict/?age=22&sex=1&cp=1&trestbps=120&chol=233

✓	age	22
✓	sex	1
✓	cp	1
✓	trestbps	120
✓	chol	233

200 OK 850 ms 167 B

```
{
    "Status": "[1]"
}
```