

clinical analytics api

Implementation Explanation with API Documetation and User Guide



August 2, 2021

# **Task 1 Build and Host API**

## **Scope**

Create and host an API that will serve the results from a prebuilt Python or R model (either one API for the Python model **or** one for the R model).

You can decide on your own JSON format for the requests.

The models can score multiple records in a single call and the API should do so as well.

We have provided a CSV file for both languages with a sample of the inputs and outputs required.

In both cases, the ‘Predictions’ column is the output that should be returned by the API.

## **Additional Implementation**

Upload CSV and download csv with predictions [developed locally available in development branch]

**GitHUB:** [**https://github.com/KasiChennupati/Clinical-Analytics-API**](https://github.com/KasiChennupati/Clinical-Analytics-API)

**API: https://clinical-analytics-api.herokuapp.com/ca/getpredictions**

## **Technical Specifications**

### Specifications

|  |  |
| --- | --- |
| **Operating System** | Ubuntu 18.04 |
| **Development Language(s)** | Python==3.6.5 |
| **Cloud Platform** | Heroku/Heroku-18 stack |
| **Packages/Libraries** | numpy==1.15.4  pandas==1.1.0  Flask==1.1.0  scipy==1.5.4  scikit-learn==0.19.1  gunicorn==20.1.0 |

**Note**: The above python version and libraries(numpy, pandas, and scikit-learn) are needed for pickle file deserialisation

### Why the specific requirements?

The development of API have a dependency on successfully deserialising the pickle file and for that process the specific versions of python and scikit-learn are required.

The Heroku-20 stack (Ubuntu-20) has no supported version of python 3.6 available but the Heroku-18 (Ubuntu-18) stack has the support for the required python version.

## **Data I/O Format Design**

The data input and outut of the API

|  |  |
| --- | --- |
| Feature Name | Values |
| Pclass | Integer |
| Sex | Boolean |
| Age | Integer |
| SibSp | Integer |
| Parch | Integer |
| Fare | Float |
| Embarked\_S, Embarked\_C | Boolean |

### API Input JSON

The API expects a JSON format input of the following example structure

{

    "Pclass":[2, 3],

    "Sex": [1, 1],

    "Age":[24, 61],

    "SibSp":[0, 0],

    "Parch":[0, 0],

    "Fare":[13, 6.2375],

    "Embarked\_S":[1, 1],

    "Embarked\_C":[1, 0]

}

The inputs need to be followed the list of the same specified datataypes as mentioned in the above features table not scalars. Check out more acceptable Json input formats in HOW-to-Guide

### API Output JSON

{

    "prediction": [

        0.38441490346019563,

        0.17907895626397127,

        0.4239311987959022,

        0.39193345612918523,

        0.3977242630764239

    ]

}

## **Solution Approach**

1. Develop the ml api using python flask framework
2. Containerise the application using docker
3. Deploy the container into heroku cloud
4. Test the final deployment using post man and requests

## **Cloud Deployment**

The deployment methods available for Heroku are

* Heroku Git (Heroku CLI )
* Github
* Container Registry ( Heroku CLI ) Base Docker 🡨 Used Method

From the above methods the use of Github is not recommended as the current version needs to be deployed on Heroku-18 stack (Ubuntu-18.04) where by default heroku builds on Heroku-20 stack

The suited mode of deployments are Heroku Git and Container Registry using the Heroku CLI

The Container method is the ideal method for the deployment for future proofing the deployment pipeline and ease of development.

### Deployment Steps

Commands in command line

$ heroku login

$ heroku container:login

$ heroku container:push web -a clinical-analytics-api

$ heroku container:release web -a clinical-analytics-api

## **Tests**

### Testing with POSTMAN

The API calls can be tested using the POSTMAN

Step 1 : goto the [Postman API Platform](https://web.postman.co/home)

Step 2 : navigate to the create new section by clicking on “Create new”

Step 3: in the new page select POST and add <https://clinical-analytics-api.herokuapp.com/ca/getpredictions>

In the address bar

Step 4: select the Body tab and datatype JSON paste the input test JSON in the below text area

Step5: Click send

Step 6: in the response if 200 Is found it means its success and in the results body select JSON in the dropdown . The predictions output is found like below image

Graphical user interface, text

Description automatically generated

### Python Requests

The API method can be requested using the python requests package

Graphical user interface, text, application

Description automatically generated

## **API Documentation and User Guide**

The api have method /ca/getpredictions route and the funtion receives the json from the client and respond with a json object.

### **How-to-Guide**

#### **Get predictions for a single record of values**

Import requests

Input\_json = {

    "Pclass": [3],

    "Sex":  [1],

    "Age":[24],

    "SibSp":[0],

    "Parch":[0],

    "Fare":[6.2375],

    "Embarked\_S": [1],

    "Embarked\_C":[1]

}

url = “<https://clinical-analytics-api.herokuapp.com/ca/getpredictions>”

r = requests.post(url,input\_json)

r.json()

#### **Get predictions for a multiple records of values**

Input\_json = {

"Pclass":{"0":2,"1":3,"2":2,"3":2,"4":3},

"Sex":{"0":False,"1":True,"2":False,"3":True,"4":False},

"Age":{"0":24.0,"1":61.0,"2":17.0,"3":18.0,"4":24.0},

"SibSp":{"0":0,"1":0,"2":0,"3":0,"4":0},

"Parch":{"0":0,"1":0,"2":0,"3":0,"4":3},

"Fare":{"0":13.0,"1":6.2375,"2":12.0,"3":11.5,"4":19.2583},

"Embarked\_S":{"0":True,"1":True,"2":False,"3":True,"4":False},

"Embarked\_C":{"0":False,"1":False,"2":True,"3":False,"4":True}

}

Graphical user interface, text

Description automatically generated

#### **Get predictions for a DataFrame**

df = pd.read\_csv('data/test\_data\_for\_candidate\_python.csv')

\_\_input = df[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked\_S', 'Embarked\_C']]

Input\_json = \_\_input.to\_dict()

url = "https://clinical-analytics-api.herokuapp.com/ca/getpredictions"

r = requests.post(url, json = Input\_json)

r.text.strip()

print(r)

print("\n")

print(r.json())

