

Data Intake Report

Name: <Week 5 Cloud and API Deployment>

Report date: <11/07/21>

Internship Batch:<LSUM01>

Version:<1.0>

Data intake by:<H. Melis Tekin Akcin>

Data intake reviewer:<Data Glacier>

Data storage location: <<https://github.com/melis-ta/Model-Deployment-in-Heroku>>

Tabular data details:

Total number of observations	<768>
Total number of files	<1>
Total number of features	<9>
Base format of the file	<.csv>
Size of the data	<54.1 KB>

1. Modeling the dataset ‘Diabetes’ : The model that I have created on Week 4.

Modelling

Creating the dependent and independent variables to apply Logistic regression model.

```
In [6]: df["Outcome"].value_counts()
```

```
Out[6]: 0    500  
        1    268  
        Name: Outcome, dtype: int64
```

```
In [7]: X= df.iloc[:, :8]  
        y=df["Outcome"]
```

```
In [8]: #check the partition X.  
        X.head()
```

```
Out[8]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

Splitting the data set as train and test set. And training the model.

```
In [9]: #splitting the dataset as train set and test set by using sklearn.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.30, random_state = 101)
loj=LogisticRegression(solver="liblinear")

In [10]: #fitting the model with training set.
loj_model= loj.fit(X_train,y_train)
loj_model

Out[10]: LogisticRegression(solver='liblinear')

In [11]: pickle.dump(loj, open('model.pickle','wb'))
```

2. Converting the notebook to python file and saving the model on our disk:

```
(base) C:\Users\hmeli\DataGlacier_Week5_API_Cloud_Deployment>jupyter nbconvert --to python model.ipynb
[NbConvertApp] Converting notebook model.ipynb to python
[NbConvertApp] Writing 1178 bytes to model.py

(base) C:\Users\hmeli\DataGlacier_Week5_API_Cloud_Deployment>python model.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   Pregnancies           768 non-null    int64
 1   Glucose               768 non-null    int64
 2   BloodPressure         768 non-null    int64
 3   SkinThickness         768 non-null    int64
 4   Insulin               768 non-null    int64
 5   BMI                   768 non-null    float64
 6   DiabetesPedigreeFunction 768 non-null    float64
 7   Age                   768 non-null    int64
 8   Outcome               768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

3. app file

```
In [1]: #importing necessary libraries
import pandas as pd
import flask
from flask import Flask, jsonify, request
import pickle
```

```
In [2]: #creating flask app
app= Flask(__name__)
```

```
In [3]: @app.route('/', methods=['GET', 'POST'])
def home():
    if (request.method=='GET'):
        data= "Enter your test results!"
        return jsonify({'data': data})
```

```
In [4]: @app.route('/predict/', endpoint='Diabetes_prediction')
def Diabetes_prediction():
    model= pickle.load(open('model.pickle', 'rb'))
    Pregnancies= request.args.get('Pregnancies')
    Glucose=request.args.get('Glucose')
    BloodPressure=request.args.get('BloodPressure')
    SkinThickness=request.args.get('SkinThickness')
    Insulin=request.args.get('Insulin')
    BMI=request.args.get('BMI')
    DiabetesPedigreeFunction=request.args.get('DiabetesPedigreeFunction')
    Age=request.args.get('Age')

    test_df= pd.DataFrame({'Pregnancies': [Pregnancies], 'Glucose': [Glucose],
                           'BloodPressure': [BloodPressure], 'SkinThickness': [SkinThickness],
                           'Insulin': [Insulin], 'BMI': [BMI],
                           'DiabetesPedigreeFunction': [DiabetesPedigreeFunction], 'Age': [Age]})
    pred_diabetes= model.predict(test_df)
    return jsonify({'Diabetes Predicted': str(pred_diabetes)})
```

```
In [*]: if __name__=='__main__':
        app.run(debug=True, use_reloader=False)

* Serving Flask app "__main__" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

4. Running app.py on computer.

Firstly, I run in my computer.

```
Microsoft Windows [Version 10.0.19041.1083]
(c) Microsoft Corporation. All rights reserved.

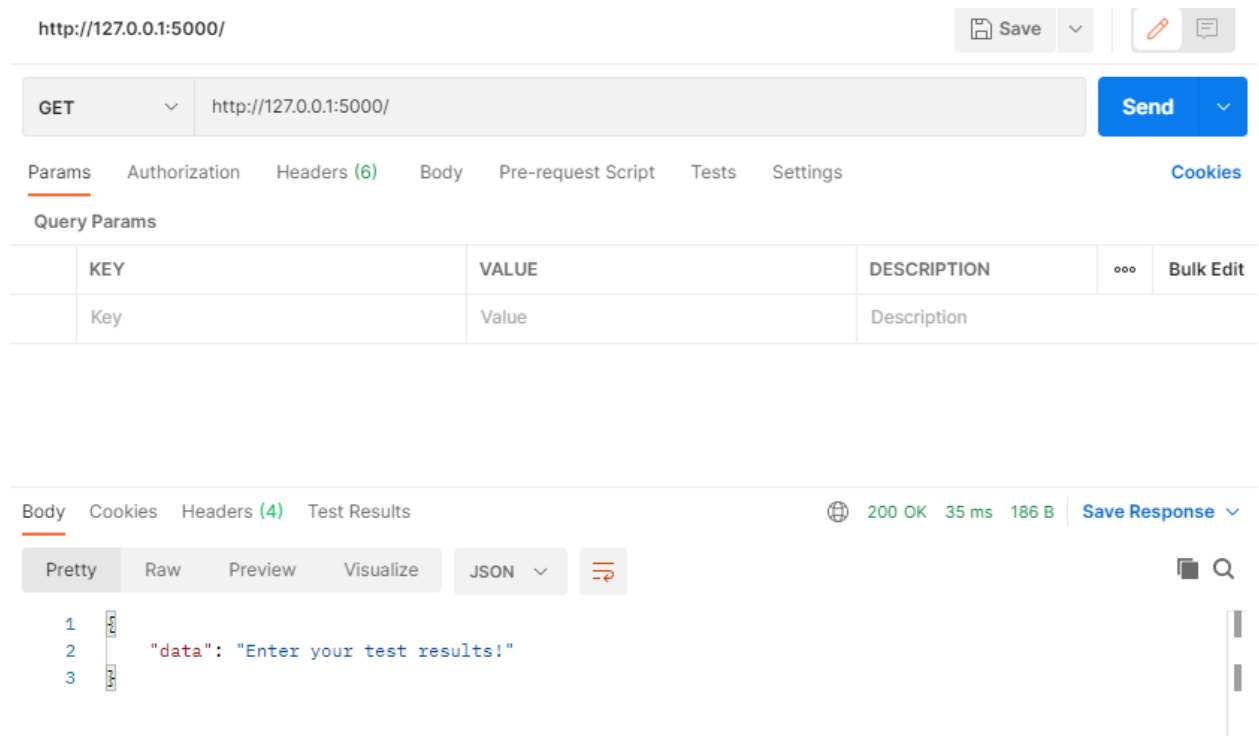
C:\Users\hmel>cd DataGlacier_Week5_API_Cloud_Deployment

C:\Users\hmel\DataGlacier_Week5_API_Cloud_Deployment>jupyter nbconvert --to python app.ipynb
[NbConvertApp] Converting notebook app.ipynb to python
[NbConvertApp] Writing 883 bytes to app.py



C:\Users\hmel\DataGlacier_Week5_API_Cloud_Deployment>python app.py
C:\Users\hmel\Python\Python39\lib\site-packages\sklearn\base.py:310: UserWarning: Trying to unpickle estimator Logistic
Regression from version 0.24.1 when using version 0.24.2. This might lead to breaking code or invalid results. Use at yo
ur own risk.
  warnings.warn(
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```


5. Applying Postman:


Get:



Predicted:




http://127.0.0.1:5000/predict/?Pregnancies=2&Glucose=65&BloodPressure=72&SkinThickness=2&Insu...  Save 



GET  http://127.0.0.1:5000/predict/?Pregnancies=2&Glucose=65&BloodPressure=72&SkinThickness=2&Insu...

Params  Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	Pregnancies	2	
<input checked="" type="checkbox"/>	Glucose	65	
<input checked="" type="checkbox"/>	BloodPressure	72	
<input checked="" type="checkbox"/>	SkinThickness	2	

Body  Cookies Headers (4) Test Results  200 OK 30 ms 179 B 

Pretty Raw Preview Visualize JSON  

```
1 {  
2   "Diabetes Predicted": "[1]"  
3 }
```

6. Connecting HEROKU with my GITHUB Account

Connected to  [melis-ta/Cloud API Deployment](#) by  [melis-ta](#)

 Releases in the [activity feed](#) link to GitHub to view commit diffs

Prepared requirements.txt and procfile files and uploaded to github. Connected via my github account and deployed by model in HEROKU.

```
Installing collected packages: click, Werkzeug, itsdangerous, MarkupSafe, Jinja2, Flask, gunicorn, numpy, six, python-dateutil, pytz, pandas, joblib, threadpoolctl, scipy, scikit-learn, idna, chardet, certifi, urllib3, requests
Successfully installed Flask-1.1.2 Jinja2-3.0.1 MarkupSafe-2.0.1 Werkzeug-2.0.1 certifi-2021.5.30 chardet-4.0.0 click-8.0.1 gunicorn-20.1.0 idna-2.10 itsdangerous-2.0.1 joblib-1.0.1 numpy-1.20.1 pandas-1.2.4 python-dateutil-2.8.1 pytz-2021.1 requests-2.25.1 scikit-learn-0.24.1 scipy-1.7.0 six-1.16.0 threadpoolctl-2.1.0 urllib3-1.26.6
-----> Discovering process types
Procfile declares types -> web
-----> Compressing...
```

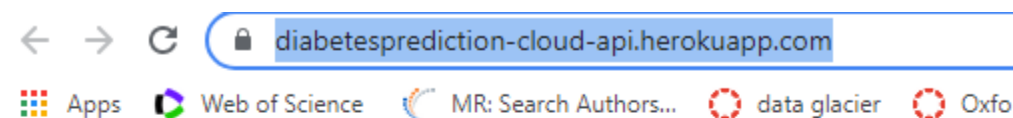
☒ Autoscroll with output [View build log](#)

Build main c983644d

```
-----> Installing requirements with pip
-----> Discovering process types
Procfile declares types -> web
-----> Compressing...
Done: 139.8M
-----> Launching...
Released v27
https://diabetesprediction-cloud-api.herokuapp.com/ deployed to Heroku
```

☒ Autoscroll with output [View build log](#)

Release phase



```
{"data": "Enter your test results!"}
```