



DATA SCIENCE INTERNSHIP AT DATA GLACIER

WEEK 4 – MODEL DEPLOYMENT USING FLASK



SEPTEMBER 27, 2021
KAVINILAVAN MUTHUKUMAR
LISUM13:30

Model deployment using Flask

Introduction

in this project, I am going to creat a API for machine learning deployment using Python Flask Frameworks. As a demonstration, I am taking the Diabeties dataset which classifying wheather the chances of gettting diabeties or Not. This dataset is already cleaned and ready for model building.

Dataset Information

I have taken the diabeties datset which have 8 features and 768 records. The target value is 1 and 0, 1 is person getting high chance of diabeties and 0 is low chance of diabetes. Since, our dataset is already cleaned, there is no null value. The below image showing the dataset.

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

The information of the columns is showing below. It shows the data types of columns, null value counts and shape of the columns.

```
<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 Machine Learning Model Implementation

3.1 Data Splitting

I separate the data for training and testing purpose. Here, 10% of data for testing and 90% for training. Data splitting is used for compute the metrics of the model and identify the model overfitting.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X , Y , test_size = 0.10)
```

3.2 Logistics Regression

For demonstration, I am using logistics regression for model building, it is more efficient for binary classification and perform well with small amount of data. It estimates the probability of event occurring

and not occurring. The target value is bounded between 0 to 1 with threshold of 0.5. if any value lies below 0.5, it considered as negative (low chance of getting diabetes) and above 0.5 considered as positive (high chance of getting diabetes).

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
```

```
>
          precision    recall  f1-score   support

     0       0.94        0.85        0.89         59
     1       0.62        0.83        0.71         18

 accuracy          0.84         77
 macro avg       0.78        0.84        0.80         77
 weighted avg    0.87        0.84        0.85         77
```

3.3 Save the model

After, the model is saved in pickle file for model deployment. For that, I am importing pickle library.

```
import pickle
pickle.dump(model, open('models.pkl','wb'))
model.predict([[0,113,30,38,93,44.3,0.193,40]]))
```

4 Web Application

For deploy the model in web application, python flask framework are used. It can allow us to predict the patient has getting chance of

diabetes or not with simply typing the data in the field. For that, I am using visual studio code editor. Before that I had to create a virtual environment, then, I made the directory called flask. For model development, we need following files.

app.py

index.html

model.pkl

4.1 App.py

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('venv/models.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST', 'GET'])
def predict():

    int_features = [float(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    if prediction==0:
        return render_template('index.html',
                               prediction_text='Patient has Low chance of getting Diabetes'.format(prediction),
                               )
    else:
        return render_template('index.html',
                               prediction_text= 'Patient has High chance of getting Diabetes'.format(prediction)
                               )

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

App.py is the main file which contains the code for python flask. Here, are we going to create API. Let's look at the following terminology for python flask.

Flask application instances by passing `__name__` as the first argument to the `Flask` class. To allowing the Flask to get HTML file which is belonging to same directory.

`@ app.route (/)` the routing technique is used for users to remember application URLs. It is useful to access the desired page directly without having to navigate from the home page.

Renter template is used to navigates the HTML file in the templates folder. It helps us to collect information from outside file instead of hard coding the html file in app.py.

`Debug = True` is used for debugging the code and it shows if we have any error.

4.2 index.html

It is the html file which contains all the text and fields which we going to type the data for prediction.

```
<!DOCTYPE html>
<html >

<head>
  <meta charset="UTF-8">
  <title>Diabetes Prediction</title>

  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

</head>

<body>
<nav class="navbar navbar-expand-sm bg-primary navbar-light">
  <ul class="navbar-nav">
    <li class="nav-item active">
      <a class="nav-link" href="index.html">Diabetes Prediction</a>
    </li>
  </ul>
</nav>

  <br>
  <br>
```

5 Running Code in Local Host

After, all the above procedure is completed, we can run the code in Local Host. The outcome of the code is shown in the web browser.

```
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
C:\Users\Kavinilavan\flask\venv\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator LogisticRegression from version 1.0.2 when using version 1.1.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
* Debugger is active!
* Debugger PIN: 104-650-624
[]
```

← → ↻ 127.0.0.1:5000

Gmail YouTube Maps Shanes Tools Feature-Selection/4... Ridge and Lasso Re... 15:57 Now playing...

Diabetes Prediction

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
Age						

Predict

Finally, we can type our input data to predict the outcome, let assume we have the new data 0, 113, 30, 38, 93, 44.3, 0.193, 40.

Diabetes Prediction

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
Age						

Predict

Patient has Low chance of getting Diabetes

This model shows the prediction of patient has low chance of getting diabetes.

6. Model deployment in Cloud

For model deployment in cloud, I am using Heroku, it is a platform as a service (paas) that enables us to build, run and operate application entirely in cloud. After, testing our model in localhost, next step is deploy in cloud. There are three different ways to deploy in Heroku. One way is to connect with GitHub that people prefer most because it is easy.

There are two files required to deploy in Heroku i.e., `requirements.txt` and `Procfile`, `requirements.txt` contains all the necessary libraries to deploy and another file is `Procfile`. It is a Process file that is required for all Heroku applications. `Procfile` specifies the commands that are executed by the app on start up. The following steps are for deploying the model in Heroku.

Step 1

Upload all necessary files in GitHub repository. The below image showing all the files required for deployment.

main 1 branch 0 tags			Go to file	Add file	Code
kavinilavanM Delete delme			5c584be	yesterday	5 commits
templates	Delete delme	yesterday			
README.md	Initial commit	9 days ago			
app.py	Add files via upload	yesterday			
models.pkl	Add files via upload	yesterday			
procfile	Add files via upload	yesterday			
requirements.txt	Add files via upload	yesterday			

Step 2

After uploaded all the necessary file in github, now log in into heroku. And, click creat new app.

Salesforce Platform

HEROKU Jump to Favorites, Apps, Pipelines, Spaces...

Personal New

Starting November 28th, 2022, free Heroku Dynos, free Heroku Postgres, and free Heroku Data for Redis* will no longer be available. If you have apps using any of these resources, you must upgrade to paid plans by this date to ensure your apps continue to run and to retain your data. For students, see the Heroku for Students program by the end of September. [Learn more](#)

Filter apps and pipelines

dataglacierdiabetesprediction	heroku-22 · United States	☆
dataglacierinternship-api	Python · heroku-22 · United States	☆
fiji-island-cyclone	Python · heroku-20 · United States	☆

Step 3

Now, creat the app name, my app name is dataglacierinternship-api , after create the app name do creat app, the region should be united states.

App name

Choose a region

 United States

Add to pipeline...

Create app

Step 4


After creating app, now time to connect with github. After connecting with github, type the repository where our files are available.


[Overview](#) [Resources](#) [Deploy](#) [Metrics](#) [Activity](#) [Access](#) [Settings](#)

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.


Add this app to a stage in a pipeline to enable additional features



Pipelines let you connect multiple apps together and **promote code** between them.
[Learn more.](#)



Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests.
[Learn more.](#)

Choose a pipeline

Deployment method

 **Heroku Git**
Use Heroku CLI

 **GitHub**
Connect to GitHub

 **Container Registry**
Use Heroku CLI

Deploy using Heroku Git

Install the Heroku CLI



Step 5

After clicking, deploy , app was built successfully in a few min. now we can see the app in <https://dataglacierinternship-api.herokuapp.com/> this url.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more.](#)

Choose a branch to deploy

 master 

Deploy Branch

Receive code from GitHub



Build master 3a3e6e70



Release phase



Deploy to Heroku



Your app was successfully deployed.

 View