Data Intake Report

Name: <Week 4 Deployment on Flask>
Report date: <04/07/21>
Internship Batch:<LSUM01>
Version:<1.0>
Data intake by:<H. Melis Tekin Akcin>
Data intake reviewer:<intern who reviewed the report>
Data storage location: <https://github.com/melis-ta/ML-Deployment-on-Flask>

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':**

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

```
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 = 42)
        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]: #constant and coefficients of logistic regression model.
         loj_model.intercept_

Out[11]: array([-5.76722906])

In [12]: loj_model.coef_

Out[12]: array([[ 0.06124462,  0.02617162, -0.01666689, -0.00217604, -0.00028034,
                  0.06420964,  0.24872595,  0.02208779]])

In [13]: #computed the predicted values with test test.
         y_pred=loj_model.predict(X_test)

In [14]: #accuracy score of our model.
         accuracy_score(y_test, y_pred)

Out[14]: 0.7532467532467533
```

Cross-Validated score of the model and pickle/load:

```
In [17]: cross_val_score(loj_model, X_test, y_test, cv=10)

Out[17]: array([0.79166667, 0.82608696, 0.73913043, 0.82608696, 0.73913043,
                0.91304348, 0.7826087 , 0.65217391, 0.65217391, 0.7826087 ])

In [18]: cross_val_score(loj_model, X_test, y_test, cv=10).mean()

Out[18]: 0.7704710144927536

In [19]: pickle.dump(loj_model, open('Diabetes_model.pkl','wb'))

In [20]: loj_model = pickle.load(open('Diabetes_model.pkl','rb'))

In [21]: print(loj_model.predict_proba([[2, 90, 65, 12, 90, 35, 5, 67]]))

         [[0.36659234 0.63340766]]
```

2. HTML codes (index.html)

```
<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>ML API </title>
    <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
  <div class="login">
    <h1> Diabetes Prediction</h1>


     <!--Main Input For Receiving Query to pur ML-->
    <form action="{{ url_for('predict')}}"method="post">
        <input type="text" name="Number of Pregnancies" placeholder="Number of Pregnancies" required="required" />
        <input type="text" name="Glucose" placeholder="Glucose" required="required" />
        <input type="text" name="BloodPressure" placeholder="BloodPressure" required="required" />
        <input type="text" name="SkinThickness" placeholder="SkinThickness" required="required" />
        <input type="text" name="Insulin" placeholder="Insulin" required="required" />
        <input type="text" name="BMI" placeholder="BMI" required="required" />
        <input type="text" name="DiabetesPedigreeFunction" placeholder="DiabetesPedigreeFunction" required="required" />
        <input type="text" name="Age" placeholder="Age" required="required" />
        <button type="submit" class="btn btn-primary btn-block btn-large">Probability of Having Diabetes </button>
    </form>

    <br>
    <br>
    {{ prediction_text }}
</div>
</body>
</html>
```

3. app file

In [1]:
```
import numpy as np
import flask
from flask import Flask, request, jsonify, render_template
import pickle
```

In [2]:
```
app= Flask(__name__)
Diabetes_model = pickle.load(open('diabetes_model.pkl', 'rb'))
```

In [3]:
```
@app.route('/')
def home():
    return render_template('index.html')
```

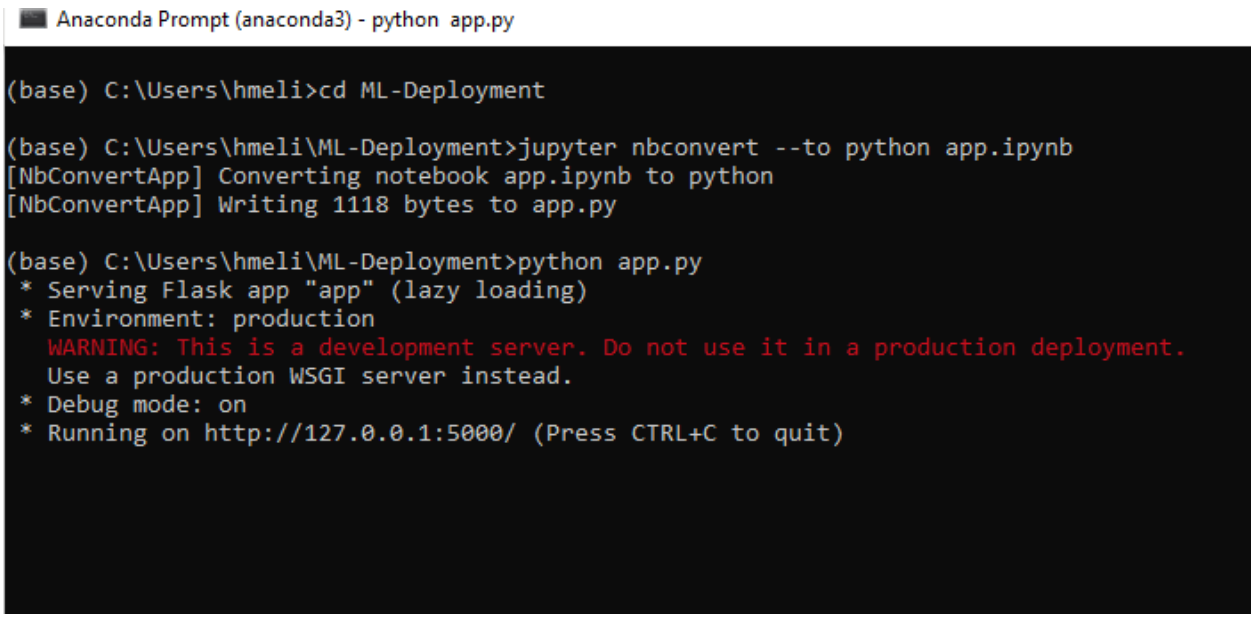
In [4]:
```
@app.route('/predict', methods=['POST'])
def predict():

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

    output = round(prediction[0], 2)

    return (flask.render_template('index.html', prediction_text='The probability that the patient has diabetes is {}'.
```

In [5]:
```
@app.route('/results',methods=['POST'])
def results():

    data = request.get_json(force=True)
    prediction = Diabetes_model.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)
```

In [*]:
```
if __name__ == "__main__":
    app.run(port=5000, 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)
127.0.0.1 - - [04/Jul/2021 04:58:56] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2021 04:58:56] "GET /static/css/style.css HTTP/1.1" 404 -
127.0.0.1 - - [04/Jul/2021 04:59:10] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2021 04:59:10] "GET /static/css/style.css HTTP/1.1" 404 -
127.0.0.1 - - [04/Jul/2021 04:59:56] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2021 04:59:57] "GET /static/css/style.css HTTP/1.1" 404 -
127.0.0.1 - - [04/Jul/2021 05:00:06] "GET /predict HTTP/1.1" 405 -
127.0.0.1 - - [04/Jul/2021 05:00:17] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2021 05:00:18] "GET /static/css/style.css HTTP/1.1" 404 -
127.0.0.1 - - [04/Jul/2021 05:01:14] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2021 05:01:14] "GET /static/css/style.css HTTP/1.1" 404 -
```

4. Converting notebook to .py file and running python code from prompt:



5. An Example of the model:

Diabetes Prediction

2	68	75	21	0	13	23

50 Probability of Having Diabetes

Diabetes Prediction

Number of Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction

Age Probability of Having Diabetes

The probability that the patient has diabetes is 1