**== REPORT ==**

**< Lab 2 >**

Server-based Machine Learning Model Deployment

|  |  |
| --- | --- |
|  |  |
| 과 목 | 머신 러닝 |
| 담당 교수 | 정용진 교수님 |
| 제출 일자 | 2021-10-11 |
| 전공 | 전자통신공학과 |
| 학번 | 2019707044 |
| 이름 | 전영우 |

**- 목 차 -**

1. **Model Implementation**

**2. Server Deployment**

1. oneoff\_app.py

2. batch\_app.py

3. realtime\_app.py

1. **Webpage Implementation**
2. **Result**

1. oneoff\_app.py

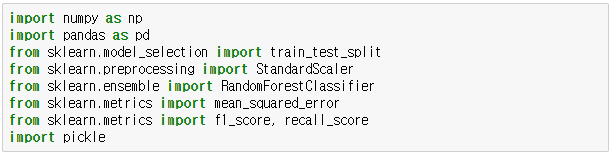
2. batch\_app.py

3. realtime\_app.py

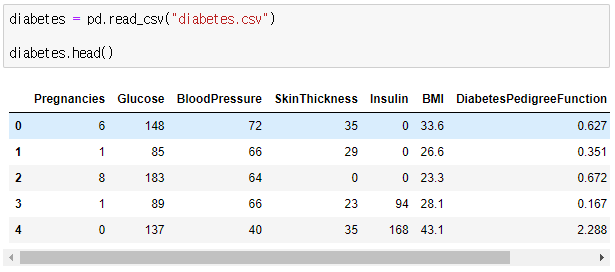
1. **Conclusion**
2. **Model Implementation**

Diabetes 분류를 위해 생성한 모델을 구현 순서에 따라 소개합니다.

* Setup

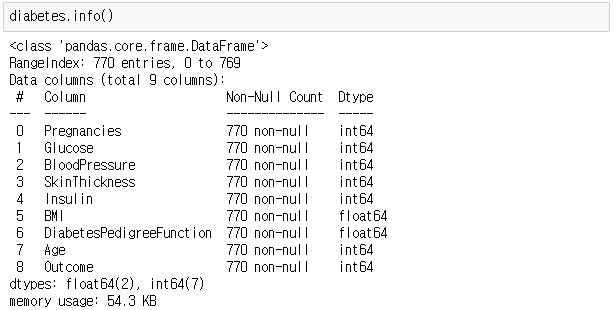


* Scaling을 위해 StandardScaler를 사용합니다.
* 분류 모델로는 RandomForestClassifier를 사용합니다.
* 모델의 성능을 테스트하기 위해 mean\_squared\_error, f1\_score, recall\_score 를 사용합니다.
* 마지막으로 학습된 모델을 파일로 저장하기 위해 pickle을 사용합니다.
* Import dataset



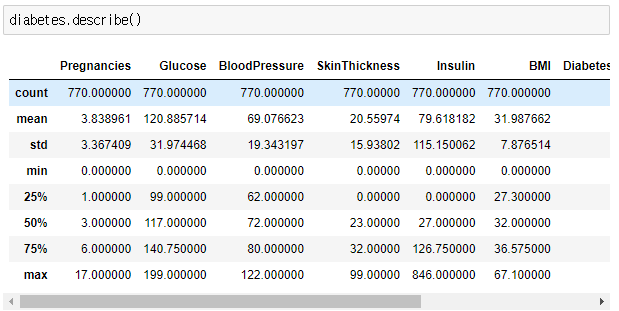
데이터셋으로 사용할 csv 파일을 불러옵니다.

이 데이터셋에 대한 간단한 정보를 살펴보겠습니다.



위 정보에서 주목해야 할 것은, 모두 numerical data라는 것입니다.

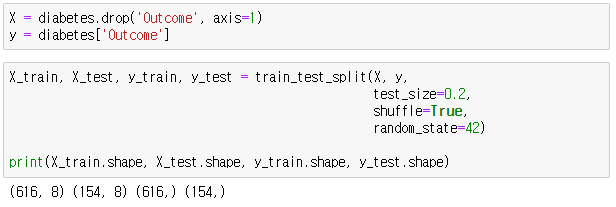
Categorical data가 없기 때문에 원-핫 인코딩은 필요하지 않습니다.



그리고 위 정보를 보면 데이터셋에 결측치가 없음을 알 수 있습니다.

따라서 데이터 전처리 과정에서는 Scaling만을 수행합니다.

* Split train/test set

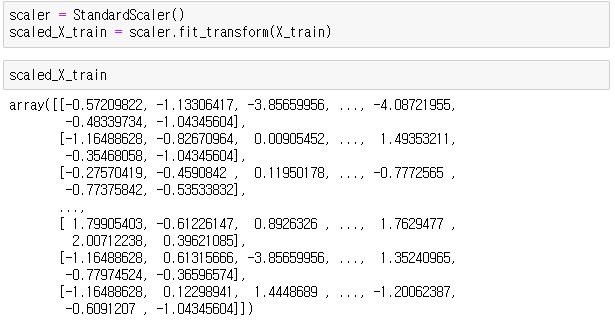


데이터셋을 훈련 데이터셋과 테스트 데이터셋으로 나누고, feature와 label로 나눕니다.

* Data Preprocessing

앞에서 말한 것처럼, 데이터 전처리 과정에서는 scaling만이 요구됩니다.

Imputing, Encoding 등의 과정은 여기서 필요하지 않습니다.



Standard Scaler를 사용하여 데이터를 scaling합니다.

* Train Model

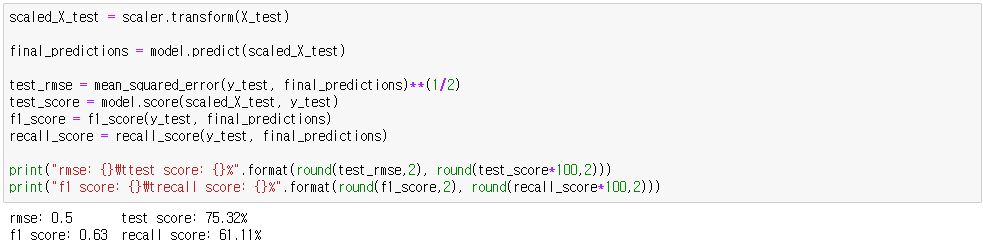


이제 데이터 준비를 마쳤으니 모델을 훈련합니다.

모델로는 머신 러닝에서 성능이 가장 좋은 편에 속하는 RandomForestClassifier를 사용합니다.

* Evaluate Model

모델이 잘 학습했는지 검사해보겠습니다.



약 75%의 정확도를 보입니다.

이로부터 모델이 제대로 학습했다는 것을 알 수 있습니다.

데이터가 더 많았다면 정확도를 더욱 올릴 수 있었겠지만, 샘플이 약 770개 밖에 되지 않아 성능을 더 올리기 어렵습니다.

* Save Model

학습된 모델과 전처리 기구를 pkl 파일로 저장합니다.

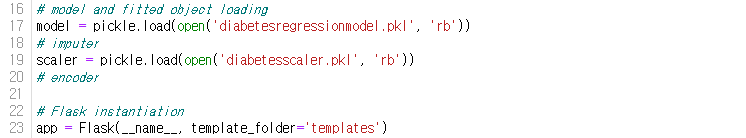




**2. Server Deployment**

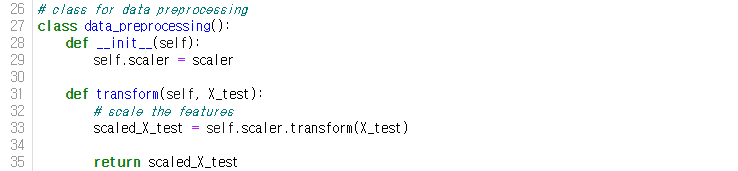
1. oneoff\_app.py

주요 코드를 중심으로 설명하겠습니다.

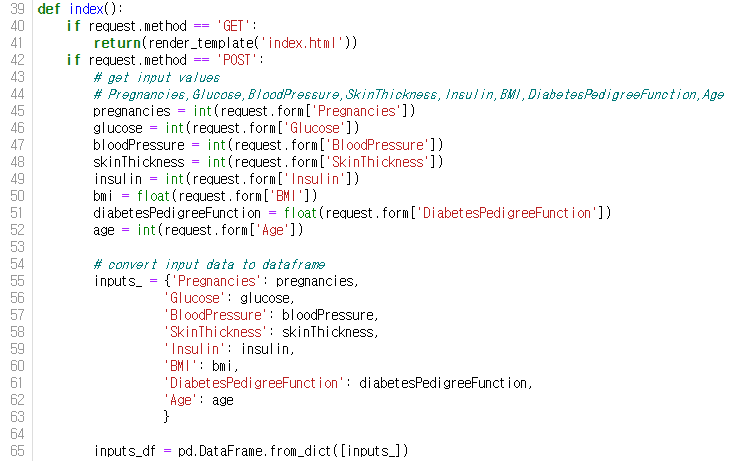


앞에서 pkl 파일로 저장했던 모델 파일과 스케일러 파일을 불러옵니다.

웹 페이지를 랜더링하기 위해 Flask를 사용하고 코드가 들어있는 디렉터리로는 templates를 지정합니다.



데이터 전처리 클래스에서는 scaling만 진행합니다.

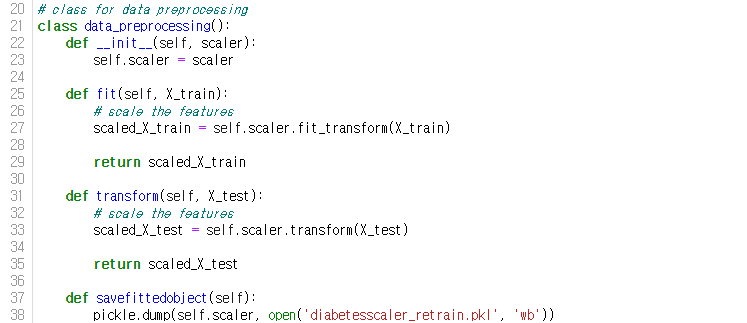


웹 페이지 랜더링을 수행하는 index 함수는 두 가지 일을 합니다.

첫째로, 웹 페이지에 사용자가 접속하여 “GET” 신호를 받아서 웹 페이지를 랜더링합니다.

둘째로, 사용자가 입력 값을 입력하고 [제출]을 클릭하면 해당 데이터를 가져와서 전처리하고 예측을 수행합니다.

1. batch\_app.py



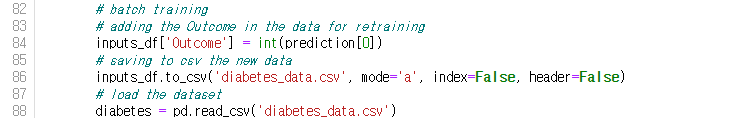
Oneoff\_app.py 에서의 data\_preprocessing 클래스와 다른 점은 fit 메서드가 있다는 것입니다.

배치 훈련 모드에서 모델은 일정 개수 이상의 입력 값이 모일 때마다 재학습을 수행합니다.

그리고 savefittedobject 메서드에서 재학습된 전처리 기구들을 pkl 파일로 저장합니다.

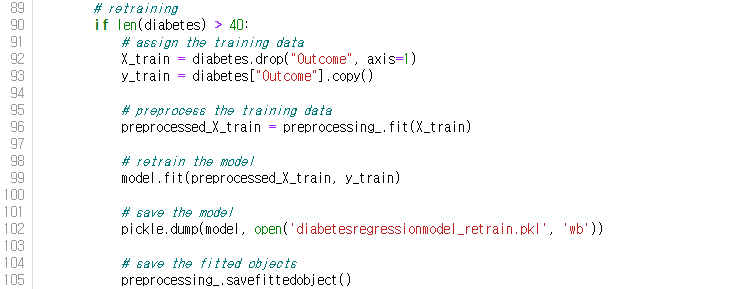
재학습된 모델은 index 함수에서 저장합니다.

다음으로 index 함수에서 다른 점을 살펴보겠습니다.



배치 훈련을 위해 사용자가 입력한 입력 값과 예측 값을 새로운 csv 파일로 저장합니다.

이 csv 파일은 새로 입력된 입력 값들을 저장하는 파일입니다.



그리고 새로운 입력 데이터가 40개 이상 모이면 배치 훈련을 수행합니다.

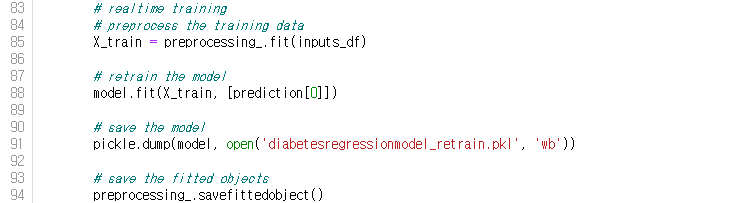
재학습된 모델과 전처리 기구들을 pkl 파일로 저장합니다.

3. realtime\_app.py

Realtime\_app.py의 코드는 batch\_app.py의 코드와 유사한데, 한 부분이 다릅니다.

바로 입력 값이 일정 개수 이상 모였을 때 재학습을 수행하는 것이 아닌, 매 입력 값이 들어올 때 마다 재학습을 수행하는 것입니다.

아래 코드는 index 함수 코드의 일부분입니다.



모든 개별적인 입력 데이터에 대해 재학습을 진행하고 pkl 파일로 저장합니다.

1. **Webpage Implementation**

앞의 oneoff\_app.py 코드에서 보았듯이, 웹 페이지에 랜더링 될 코드가 필요합니다.

이 코드는 templates 디렉터리 아래에 index.html로 작성되어 있습니다.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Diabetes Predictor</title>  <style>  fieldset{      margin: 15px 10px;  }  fieldset legend{      font-weight: bold;      font-size: 18px;      color: blue  }  </style>  </head>  <form action="{{ url\_for('index') }}" method="POST">      <fieldset>          <legend>Input values:</legend>          <p>              Pregnancies:              <input name="Pregnancies" type="number" step=1 required />          </p>          <p>              Glucose:              <input name="Glucose" type="number" step=1 required />          </p>          <p>              Blood Pressure:              <input name="BloodPressure" type="number" step=1 required />          </p>          <p>              Skin Thickness:              <input name="SkinThickness" type="number" step=1 required />          </p>          <p>              Insulin:              <input name="Insulin" type="number" step=1 required/>          </p>          <p>              BMI:              <input name="BMI" type="number" step=0.1 required />          </p>          <p>              Diabetes Pedigree Function:              <input name="DiabetesPedigreeFunction" type="number" step=0.001 required />          </p>          <p>              Age:              <input name="Age" type="number" step=1 required />          </p>            <input type="submit" />      </fieldset>  </form>  <br>  <div>      {% if result == 0 %}          Predicted: <p style="font-size:20px; margin-left:30px;">not diabetes</p>      {% elif result == 1 %}          Predicted: <p style="font-size:20px; margin-left:30px;">diabetes</p>      {% endif %}  </div>  </html> |

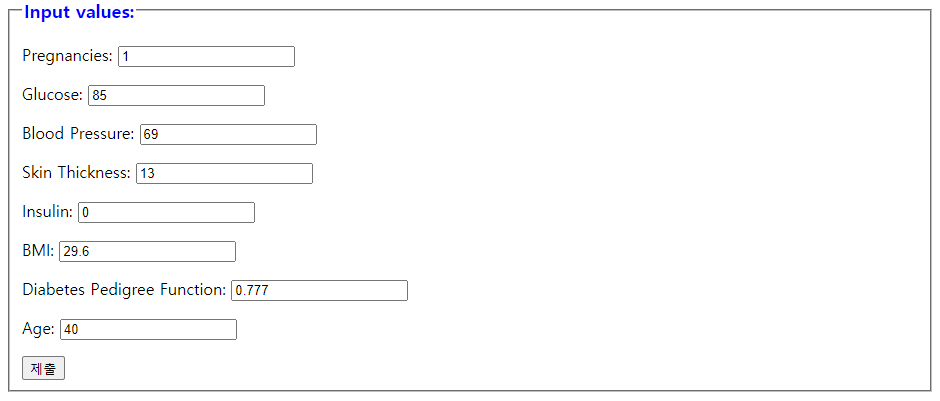
데이터 양식에 맞게 각 feature의 fieldset을 생성합니다.

사용자가 입력 값을 제출하면 모델이 그 입력 값으로 예측을 수행합니다.

모델이 예측 값을 반환하면 그 값이 0이면 ‘not diabetes’를, 1이면 ‘diabetes’를 웹 페이지에 출력하도록 했습니다.

1. **Result**
2. oneoff\_app.py

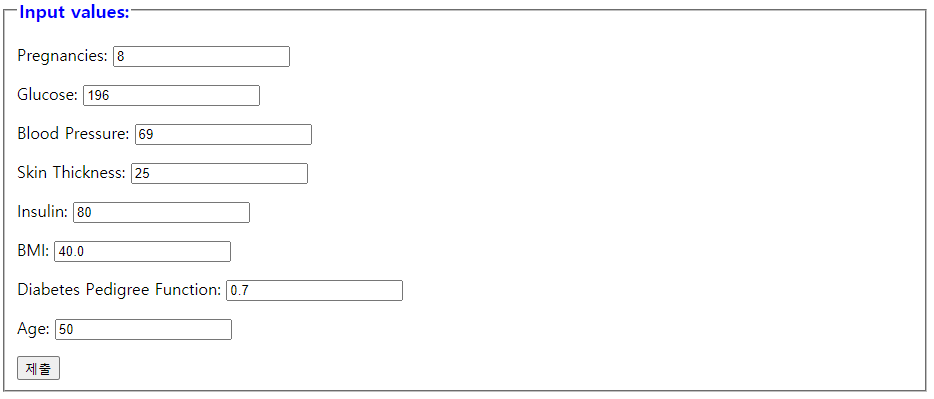
oneoff\_app.py는 가장 기본적인 모델로, 사용자가 값을 입력하고 제출하면 그 입력 값을 이용하여 예측을 수행하고 예측 값을 출력합니다.





1. batch\_app.py

batch\_app.py에서는 사용자가 입력 값을 제출하면 그 값을 새로운 csv 파일로 저장합니다. 이 csv 파일에 40개 이상의 샘플이 모이면 재학습을 수행합니다.

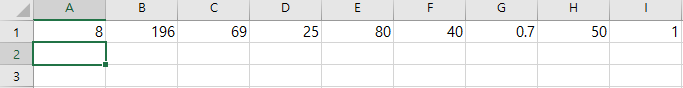




프로그램이 실행된 파일을 보면 새로운 데이터가 저장되는 csv 파일이 생성되어 있습니다.

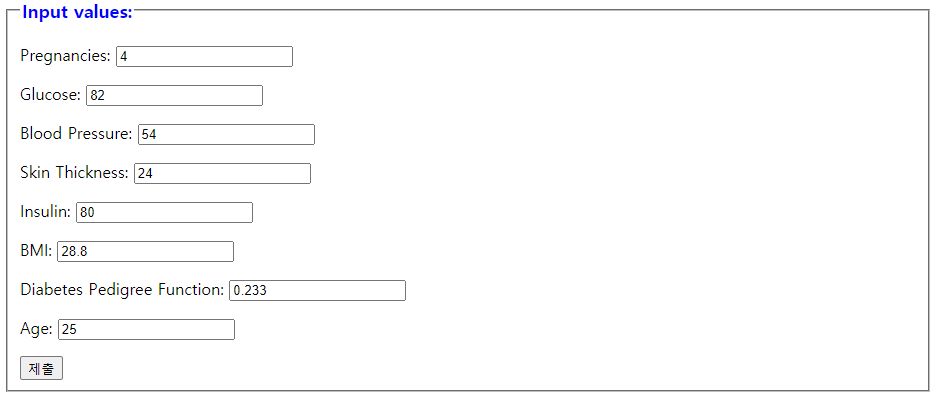


그리고 내용을 확인해보면 사용자가 입력한 값들이 저장되어 있습니다.



3. realtime\_app.py

Realtime\_app.py는 실시간 학습을 수행하기 때문에 사용자가 입력 값을 한 번만 제출해도 재학습된 모델과 전처리 기구들이 저장됩니다.



****

재학습된 model과 scaler가 파일로 저장됩니다.

****

****

1. **Conclusion**

지금까지 분류 모델을 웹 페이지에 deploy하는 예제를 수행해보았습니다.

과정은 다음과 같습니다.

1. 모델 생성
2. 모델 저장
3. 웹 페이지 랜더링
4. 입력 값으로 예측 수행
5. 예측 값 출력

그리고 단순히 예측만 하는 것을 넘어서 입력 받은 데이터로 재학습을 시키는 것도 가능했습니다.

Batch\_app.py, realtime\_app.py에서 각각 배치 훈련과 실시간 훈련을 수행하는 것을 확인할 수 있었습니다.

이번 예제로 인공지능 모델의 적용 범위가 생각했던 것보다 훨씬 넓으며, 다양한 적용이 가능하겠다는 생각을 할 수 있었습니다.