머신러닝을 이용한 분류 모델 설계

-당뇨병 예측

전자전기공학부

2017111930 김민서

목차

데이터 설명

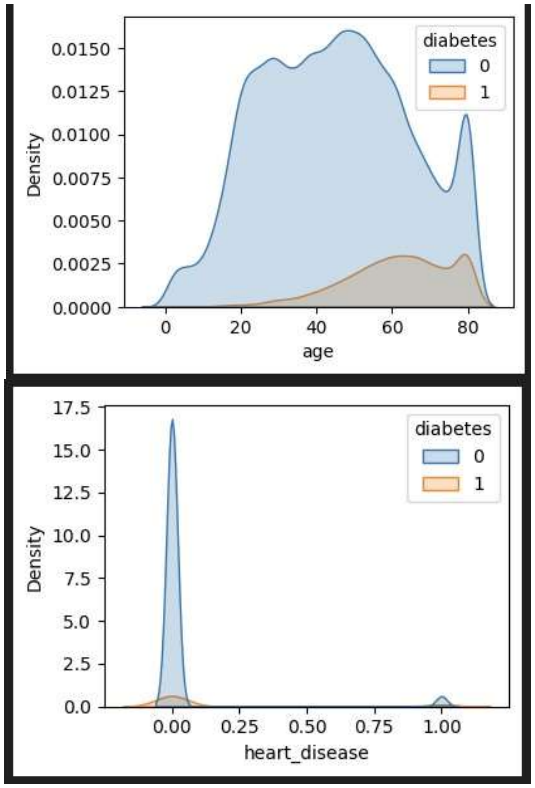
- 데이터셋 이름 : Diabetes prediction dataset
- 출처 : Kaggle
- 100,000개의 샘플과 9개의 variable로 구성
- 당뇨에 걸린 사람과 걸리지 않은 사람으로 구성
- 목표: 데이터셋을 이용하여 당뇨 여부 진단

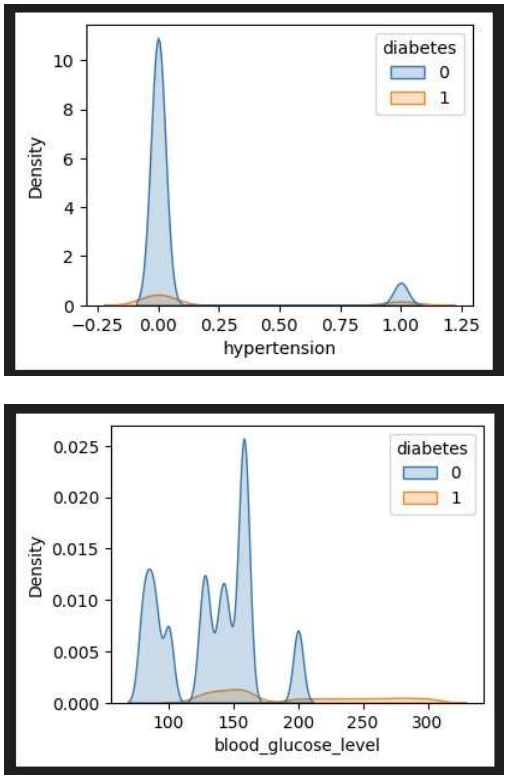
데이터 설명

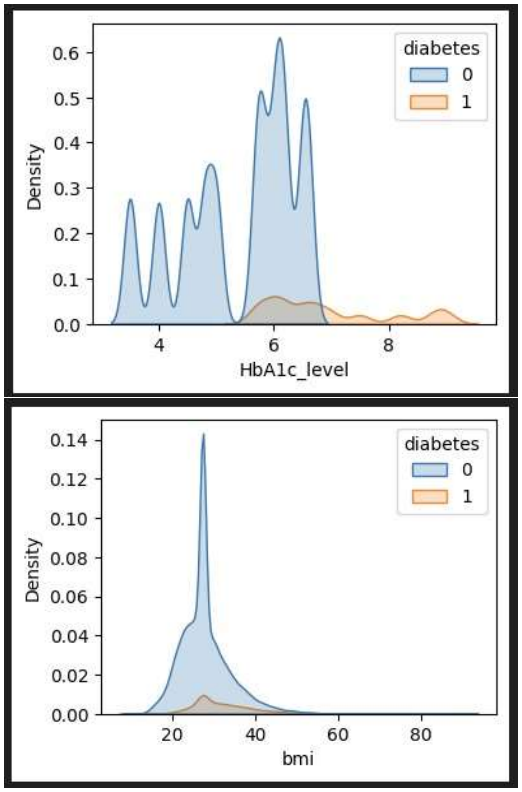
변수명	변수 내용
Gender	성별
Age	나이
Hypertension	고혈압 여부
Heart_disease	심장병 여부
Smoking_history	흡연 이력
Bmi	체질량 지수
HbA1c_level	당화혈색소(평균혈당치의 지표)
Blood_glucose_level	혈당값
Diabetes(Target)	당뇨병이 있는지 여부

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0

데이터 시각화


diabetes
0
1
Density
age
heart_disease


diabetes
0
1
Density
hypertension
blood_glucose_level


diabetes
0
1
Density
HbA1c_level
bmi

데이터 시각화

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes	gender_Female	gender_Male	smoking_history_current	smoking_history_ever
age	1.000000	0.252212	0.238012	0.161190	0.116039	0.116928	0.260850	-0.022214	0.022214	-0.052281	0.034254
ertension	0.252212	1.000000	0.117485	0.130012	0.086949	0.086737	0.192226	-0.028102	0.028102	-0.011503	0.005869
t_disease	0.238012	0.117485	1.000000	0.037773	0.072913	0.078762	0.169614	-0.092126	0.092126	-0.005829	0.037941
bmi	0.161190	0.130012	0.037773	1.000000	0.084901	0.094917	0.204421	-0.011353	0.011353	0.000550	0.013374
A1c_level	0.116039	0.086949	0.072913	0.084901	1.000000	0.194530	0.438897	-0.028875	0.028875	-0.006113	0.003323
ose_level	0.116928	0.086737	0.078762	0.094917	0.194530	1.000000	0.449698	-0.030301	0.030301	0.000116	-0.001953
diabetes	0.260850	0.192226	0.169614	0.204421	0.438897	0.449698	1.000000	-0.056997	0.056997	-0.010143	0.006693

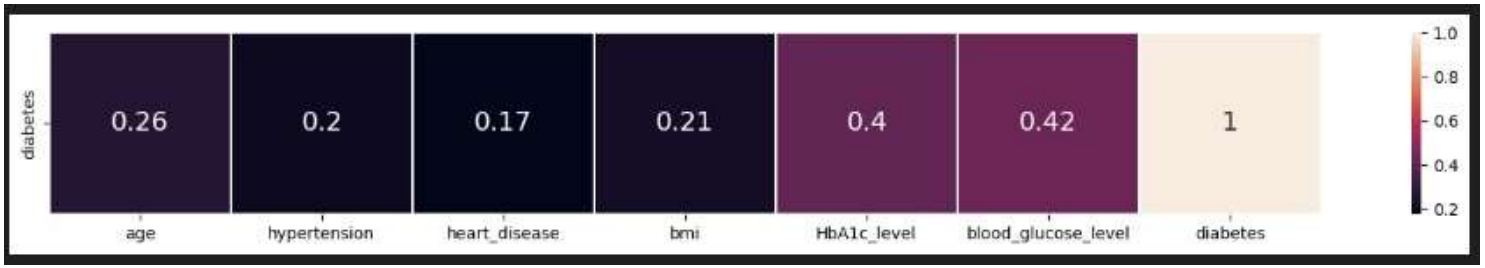
smoking_history_former	smoking_history_never	smoking_history_not current
0.222256	-0.149320	0.020035
0.062914	-0.024418	-0.024683
0.097624	-0.081894	-0.002676
0.075843	-0.053968	-0.011060
0.033020	-0.021201	0.000834
0.035115	-0.022980	-0.001728
0.079555	-0.050770	-0.002819

- 변수들 끼리의 상관계수 낮음 확인

데이터 시각화

- HbA1c_level과 blood_glucose_level이 diabetes와 제일 밀접한 관계가 있다

- Heart_diseas는 크게 영향이 없음을 알 수 있다.



데이터 전처리

- Smoking_history에 ‘No Info’값은 사용하지 못하므로 ‘No Info’ 포함된 행 제거
- Gender에도 Male, Female 외에 Other 개수가 적으므로 제거
- 결측치 확인

```
df = df[df['smoking_history'] != 'No Info']
df = df[df['gender'] != 'Other']
df.head(5)
df.isnull().sum()
```

```
gender                0
age                   0
hypertension          0
heart_disease         0
smoking_history       0
bmi                   0
HbA1c_level           0
blood_glucose_level   0
diabetes              0
dtype: int64
```

데이터 전처리

- Get_dummies를 이용하여 One-Hot encoding(gender, smoking_history)
- 데이터 샘플이 100,000개로 너무 많으므로 2000개의 샘플만 사용(제거한 행 고려)

```
dummy_gender = pd.get_dummies(df[['gender']])
df = pd.concat([df, dummy_gender], axis=1)
dummy_smoking = pd.get_dummies(df[['smoking_history']])
df = pd.concat([df, dummy_smoking], axis=1)
df = df.drop(['gender', 'smoking_history'], axis=1)
df.head(10)
```

데이터 전처리

- Train, test library를 이용하지 못하므로 임의로 설정
- 제거한 값을 고려하여 전체 샘플 개수가 2000개정도 되도록 한 다음
- 행 개수로 train, test size 조절

```
nums = 3100
train_size = 800
```
✓ 0.0s

```
df = df.loc[1:nums]
df
```
✓ 0.0s

```
X_train = df[:800].drop(['diabetes'], axis=1)
y_train = df[:800]['diabetes']
X_test = df[800:].drop(['diabetes'], axis=1)
y_test = df[800:]['diabetes']
```
✓ 0.0s

머신 러닝 모델 선정

- Decision Tree 사용
  - 직관적이고 해석이 쉬움
  - 가장 대중적인 모델
  - Boosting 사용으로 정확도 향상
  - Gini index 기준으로 분할

```
from sklearn.tree import DecisionTreeClassifier


tree = DecisionTreeClassifier(criterion='gini',
                              max_depth = 4,
                              random_state = 1)
```

- Gini, Entropy 사용했을 때 결과값은 동일함
- Max_depth를 조정하면서 test set에서의 정확도 향상

모델 활용 및 해석

- Max_depth 4인 경우

```
Decision tree train/test accuracies(max_depth 4) 0.964/0.963
```

- Max_depth 7인 경우

```
Decision tree train/test accuracies(max_depth 7) 0.983/0.939
```

- Max_depth 10인 경우

```
Decision tree train/test accuracies(max_depth 10) 0.994/0.924
```

모델 활용 및 해석

- Decision Tree를 사용했을 경우에
  - Max_depth가 10처럼 많이 커지는 경우에 train accuracy는 매우 높으나 test accuracy는 점점 감소하는 추세를 보임.
  - 모든 경우를 비교해보았을 때 test accuracy는 Max_depth가 4일때 가장 높았다.
- Decision Tree에 Boosting 적용해보기
  - 성능을 더욱 잘 확인하기 위해서 Max-depth를 1로 한 다음 비교

모델 활용 및 해석

```
from sklearn.ensemble import AdaBoostClassifier

ada = AdaBoostClassifier(base_estimator=tree,
                         n_estimators=100,
                         learning_rate=0.1,
                         random_state=1)
```

✓ 0.0s

- Adaboost 사용
  - Boosti 사용으로 정확도 향상 기대
  - 하이퍼파라미터
    - n_estimators : 반복 수 지정
    - learning_rate : 학습률, 이에 따라 update 정도가 결정되고 너무 커지면 오히려 성능이 안좋아진다.
  - 이 두 파라미터를 조정하여 최대 성능을 이끌어낼 수 있음.

모델 활용 및 해석

- Adaboost(a,b)
  - a: n_estimators, b: learning_rate
- 처음은 100, 1로 시작. 그냥 Decision Tree보다 성능이 좋은 것을 확인할 수 있다.

```
Decision tree train/test accuracies 0.946/0.938
AdaBoost train/test accuracies(100, 0.1) 0.964/0.963
```

- n_estimators, learning_rate 변경하며 정확도 관측

모델 활용 및 해석

- 1. n-estimators 만 바꾸며 관측
  - N = 150일 때

```
Decision tree train/test accuracies 0.946/0.938
AdaBoost train/test accuracies(150, 0.1) 0.964/0.963
```

  - N = 200일 때

```
Decision tree train/test accuracies 0.946/0.938
AdaBoost train/test accuracies(200, 0.1) 0.966/0.961
```

  - N = 500일 때

```
Decision tree train/test accuracies 0.946/0.938
AdaBoost train/test accuracies(500, 0.1) 0.974/0.958
```

모델 활용 및 해석

- 2. learning_rate 만 바꾸며 관측
  - α = 0.15일 때

```
Decision tree train/test accuracies 0.946/0.938
AdaBoost train/test accuracies(100, 0.15) 0.964/0.963
```

  - α = 0.5일 때

```
Decision tree train/test accuracies 0.946/0.938
AdaBoost train/test accuracies(100, 0.5) 0.974/0.953
```

  - α = 3.0일 때

```
Decision tree train/test accuracies 0.946/0.938
AdaBoost train/test accuracies(100, 3.0) 0.634/0.638
```

  - α = 10.0일 때

```
Decision tree train/test accuracies 0.946/0.938
AdaBoost train/test accuracies(100, 10.0) 0.469/0.470
```

모델 활용 및 해석

- 3. 둘 중 가장 정확도가 높았던 조합으로 시도
  - α = 0.15, n = 150으로 했을 때

```
Decision tree train/test accuracies 0.946/0.938
AdaBoost train/test accuracies(150, 0.15) 0.968/0.961
```

- 이전에 가장 높았던 정확도

```
Decision tree train/test accuracies 0.946/0.938
AdaBoost train/test accuracies(100, 0.15) 0.964/0.963
```

모델 활용 및 해석

- 추가 1 – heatmap에서 봤을 때 가장 의미 없었던 heart_disease를 제거하고 측정

```
Decision tree train/test accuracies 0.946/0.938
AdaBoost train/test accuracies(150, 0.15)without heart disease  0.968/0.961
```

  - 이전 성능과 동일한 것을 보아 제일 상관 없는 변수는 제거해도 무관하다.
- 추가 2 – 가장 의미있었던 blood_glucose_level을 제거했을 때

```
Decision tree train/test accuracies 0.946/0.938
AdaBoost train/test accuracies(150, 0.15)without blood_glucose_level  0.948/0.936
```

  - 중요한 변수를 제거했을 경우에는 정확도의 차이가 크다

결론

- 단일 Decision Tree를 사용했을 때보다 Boosting을 사용했을 때 훨씬 성능이 좋아진 것을 알 수 있다.
- Adaboost를 사용하는데 있어 hyperparameter 두가지(n_estimators, learning_rate)를 너무 키웠을 경우에는 오히려 성능이 낮아지므로, 다시 정확성이 안좋아지는 지점 전에 stop해야함
- Heatmap과 변수를 제거해보았을 때 blood_glucose_level과 HbA1c_level이 당뇨병에 큰 영향을 끼치는 것을 알 수 있다.
- Decision Tree overfitting을 해결하기 위해 max_depth를 정확도가 제일 높은 지점에서 멈춰야 한다.(Post_pruning)

감사합니다