뇌졸중 예방을 위한 분류 모델 비교 및 최적화: 데이터 마이닝

Korea University

나상우
빅데이터 전공
2018380702

Abstract

세계적으로 뇌졸중 발생률이 증가함에 따라, 초기 예측과 정확한 진단의 중요성이 더욱 부각되고 있습니다. 본 연구에서는 기계 학습 기법을 활용하여 성별, 나이, 고혈압, 심장병, 결혼 여부, 직업 유형, 거주지 유형, 평균 할당 수준, BMI, 흡연 상태 등 10가지 위험 요인을 바탕으로 뇌졸중 발생을 예측하였습니다. 여기에는 Decision Tree, Random Forest, K-Nearest Neighbors(Knn), Naive Bayes, Logistic Regression, Support Vector Machine, AdaBoost, 그리고 Multi-Layer Perceptron 등 다양한 기계 학습 모델이 테스트 되었습니다. 이러한 모델들의 성능은 일련의 측정 기준 및 통계 검정을 통해 비교하였습니다. Decision Tree, random Forest, 그리고 K-Nearest neighbors 모델이 통계적으로 유의미한 차이를 보여주며 다른 모델들을 능가하였습니다. 본 연구는 이들 모델이 뇌졸중 예측 등 의료 분야에서의 가능성을 강조하며, 향후 유사한 문제나 데이터셋에 대해 적용될 수 있음을 제안합니다.

1 서론 및 동기

뇌졸중[1]은 뇌에 혈액을 공급하는 혈관이 막히거나 터져서 뇌 손상이 오는 뇌혈관 질환으로, 세계적으로 주요한 사망 원인 중 하나입니다. 뇌졸중으로 인한 뇌 손상은 심각한 신체 장애를 야기할 수 있으며, 심한 경우 식물인간 상태 또는 사망에 이를 수 있습니다. 따라서 뇌졸중의 발생을 미리 예측하고 예방하는 것이 중요하며, 이를 위해 뇌졸중을 일으키는 위험인자를 미리 파악하는 것이 필요합니다.

이 보고서에서는 데이터 마이닝을 활용하여 뇌졸중의 위험 요인들을 파악하고 뇌졸중 발생 가능성을 예측하는 모델을 탐구합니다. 이를 위해 환자의 성별, 나이, 고혈압 유무, 심장병 유무, 결혼 여부, 직업 유형, 거주지 유형, 평균 혈당 수준, 체질량 지수 그리고 흡연 상태, 총 10개의 특성중 중요한 연관성을 가지는 특성을 고려하였습니다. 이와 같은 특성을 활용하여 뇌졸중 발생을 예측하는 구축합니다.

본 보고서에서는 데이터를 수집하고, 전처리하며, 필요한 특성을 선택하는 초기 단계부터 모델링 및 하이퍼파라미터 튜닝에 이르기까지의 과정을 상세히 기술합니다. 여러 가지 머신러닝 알고리즘을 적용하고 이들의 성능을 비교하며, 최적의 예측 모델을 도출하는데 중점을 둡니다.

이를 통해, 데이터 마이닝과 머신러닝을 활용하여 복잡한 의료 데이터에서 유의미한 통찰력을 얻을 수 있습니다.

2 데이터 설명 및 기초분석

본 연구에서 사용된 데이터셋은 Kaggle에서 제공하는 'stroke_data.csv'[2]로, 총 40910개의 샘플과 11개의 변수로 구성되어 있습니다. 모든 변수는 수치형 데이터를 가지고 있으며, 각 변수의 설명은 다음과 같습니다.

- sex: 환자의 성별 (1: 남성, 0: 여성)
- age: 환자의 나이.
- hypertension: 고혈압 유무 (1: 있음, 0: 없음).
- heart_disease: 심장 질환 유무 (1: 있음, 0: 없음).
- ever_married: 결혼 여부 (1: 결혼함, 0: 결혼하지 않음).
- work_type: 직업 유형 (work_type: 0:never_worked, 1:children, 2:Govt job, 3:self-employed, 4:private).
- residence_type: 거주 지역 (1: 도시, 0: 시골).
- avg_glucose_level: 평균 혈당 수치.
- bmi: 체질량지수 (BMI).
- smoking_status: 흡연 여부 (1: 흡연, 0: 비흡연).
- stroke: 뇌졸중 여부 (1: 있음, 0: 없음). 이는 본 연구의 타겟 변수입니다.

Table 1.은 'stroke' 데이터셋의 각 변수에 대한 정보를 제공합니다. 각 변수의 값의 수와 데이터 타입을 나타냅니다. 이때, 'sex'변수에서, 3개의 결측치가 보입니다. 데이터의 크기가 충분히 크기 때문에, 결측치가 있는 행은 제거하였습니다.

Column	Non-Null Count	Dtype
sex	40907	float64
age	40910	float64
hypertension	40910	int64
heart_disease	40910	int64
ever_married	40910	int64
work_type	40910	int64
Residence_type	40910	int64
avg_glucose_level	40910	float64
bmi	40910	float64
smoking_status	40910	int64
stroke	40910	int64

Table 1: Stroke Dataset Variables

2.1 변수 분포 분석

Figure 1.을 통해 각 변수들의 히스토그램을 시각화하여 분포를 살펴보았습니다. 결과로부터 'avg_glucose_level'과 'bmi'의 분포가 균일하지 않음을 확인하습니다. 이 두 변수는 왜곡된 분포를 가지고 있어 데이터의 이해와 모델의 성능 향상을 위해 정규화가 필요함을 시사합니다.

그 외의 변수들은 대체로 균일하게 분포되어 있음을 확인하였습니다. 특히 목표 변수인 'stroke' 클래스는 균등하게 분포되어 있습니다. 이는 각 클래스 별 데이터의 균형이 잘 이루어져 있음을 나타냅니다.

2.2 평가지표 선택

균등하게 분포된 'stroke' 클래스를 기반으로, 예측 모델의 성능 평가 지표를 선택하였습니다. 여기서는 클래스의 분포가 균등하므로 정확도(accuracy)를 주요 평가 지표로 선정하였습니다. F1 score는 각 클래스의 불균형을 보정하는데 유용하나, 본 데이터셋에서는 'stroke' 클래스가 균등하게 분포하므로 accuracy를 사용하는 것이 적합합니다.

다만, 선택한 평가지표가 항상 모델의 성능을 완벽하게 반영하는 것은 아니기 때문에, 모델의 성능을 평가하는 다양한 지표들을 함께 고려합니다. 즉, 정밀도(Precision), 재현율(Recall), F1 score도 부가적으로 함께 고려합니다.

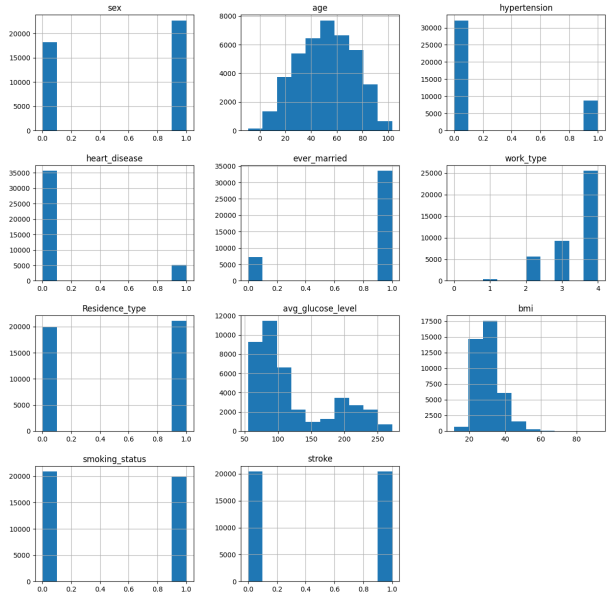

Figure 1: Histogram of Each Variable

2.3 상관관계 분석

변수 간의 상관관계를 파악하기 위해, 데이터 셋에 포함된 모든 변수들 사이의 상관 계수를 계산하였습니다. 이를 시각화하는 도구로서 히트맵을 사용하였습니다. 계산 결과는 다음과 같습니다.

Figure 2.에서 변수 'avg_glucose_level'은 'stroke', 'bmi', 'ever_married', 'heart_disease', 그리고 'hypertension'과의 상관 계수가 0.15 이상으로 나타났습니다. 특히 'stroke'는 'avg_glucose_level', 'ever_married', 'heart_disease', 그리고 'hypertension'과 높은 상관관계를 보였습니다. 이 결과는 'avg_glucose_level'이 'stroke' 발병과 밀접한 연관성을 가진 변수들과의 상관계수가 높음을 의미하며, 이는 'avg_glucose_level'이 'stroke' 발생 예측에 중요한 변수로 작용할 가능성이 높다는 것을 시사합니다. 또한, 'avg_glucose_level'은 'bmi'와도 일정 수준의 상관관계를 보여 추가적인 연관성을 나타냈습니다.

반면에 Figure 3.에서 'stroke'는 'age', 'work_type', 'Residence_type', 'bmi', 'smoking_status'와의 상관 계수가 0.10 이하로 나타났습니다. 이는 해당 변수들이 'stroke'와는 낮은 상관관계를 가짐을 의미합니다. 그 중에서도 'bmi'는 'avg_glucose_level'과 높은 상관관계를 보여주었지만, 'stroke'와의 상관관계는 낮았습니다. 이러한 결과를 바탕으로, 'age', 'work_type', 'Residence_type', 'smoking_status'의 feature는 'stroke' 예측에 필요하지 않다는 결론을 내릴 수 있습니다.

3 데이터 전처리

3.1 변수 선택

상관관계 분석을 통해 'stroke'와 상대적으로 낮은 상관관계를 가진 변수들은 모델의 복잡도를 증가시키며, 예측 성능에는 크게 기여하지 않을 것으로 판단되어 제외하였습니다. 따라서, 'age', 'work_type', 'Residence_type', 'somking_status'는 분석 대상에서 제외하였습니다. 결과적으로 총 6개의 종속 변수와 1개의 독립 변수인 'stroke'를 가지는 새로운 데이터프레임을 생성하였습니다. Table 2.는 그에 따른 기술 통계량을 나타냅니다. count는 해당 변수의 총 개수, mean은 평균값, std는 표준편차, min은 최소값, max는 최대값을 의미합니다.

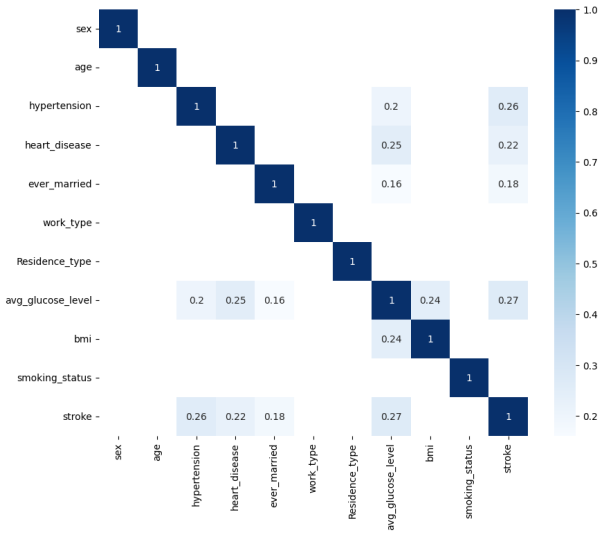

Figure 2: Correlation coefficient is greater than |0.15|

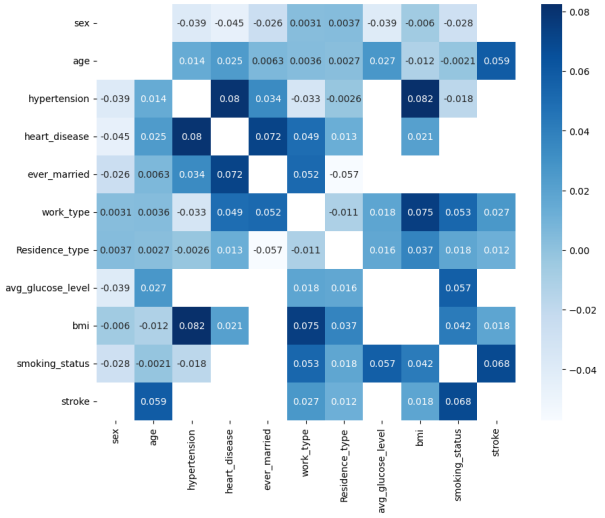

Figure 3: Correlation coefficient is less than |0.1|

3.2 정규화

Table 2.에서 변수 'avg_glucose_level'과 'bmi'는 그 분포의 표준 편차가 크며, 최소값과 최대값의 차이도 큽니다. 이렇게 스케일이 큰 변수는 모델의 학습에 영향을 미치며, 특히 경사하강법 기반 알고리즘에서는 학습의 수렴을 방해하고 지역 최솟값에 갇힐 가능성을 높입니다. 따라서, 'avg_glucose_level'과 'bmi' 두 변수는 MinMaxScaler를 사용하여 0과 1사이의 값으로 정규화하였습니다.

3.3 데이터 분할

데이터는 학습 데이터와 테스트 데이터로 분할하였습니다. 테스트 데이터는 전체 데이터의 20%를 차지하며, 이를 통해 학습된 모델의 성능을 검증할 예정입니다. 이때, 분할은 랜덤하게 이루어지지만 동일한 실험을 재현을 위해 random_state는 42로 설정하였습니다.

	sex	hypertension	heart_disease	ever_married	avg_glucose_level	bmi	stroke
count	40907.000	40907.000	40907.000	40907.000	40907.000	40907.000	40907.000
mean	0.555	0.214	0.128	0.821	122.080	30.406	0.500
std	0.497	0.410	0.334	0.383	57.562	6.835	0.500
min	0.000	0.000	0.000	0.000	55.120	11.500	0.000
25%	0.000	0.000	0.000	1.000	78.750	25.900	0.000
50%	1.000	0.000	0.000	1.000	97.920	29.400	1.000
75%	1.000	0.000	0.000	1.000	167.590	34.100	1.000
max	1.000	1.000	1.000	1.000	271.740	92.000	1.000

Table 2: Descriptive statistics of the stroke dataset after preprocessing

4 방법론 설명

다음으로, 총 8개의 머신러닝 알고리즘을 사용하여 뇌졸중 예측 모델을 구현했습니다. 사용된 알고리즘은 Decision Tree, K-Nearest Neighbors(Knn), Naive Bayes Classification, Logistic Regression, Support Vector Machine(SVM), Random Forest, AdaBoost, 그리고 Multi-layer Perceptron(MLP) 입니다.

- Decision Tree: 결정 트리는 데이터의 복잡한 구조를 트리 모양의 그래픽으로 단순화하여 표현하고, 결정 경계를 생성해 분류 문제를 해결하는 방법입니다. 이 모델은 각 특성에 대한 질문을 통해 데이터를 세분화하며, 리프 노드에 도달할 때까지 이 과정을 반복합니다. 최종적으로 리프 노드에서 데이터 포인트에 클래스 레이블을 할당합니다. 해당 모델은 해석력이 뛰어납니다. 따라서, 해석이 필요한 의료데이터에 잘 어울리는 모델이 될 수 있습니다.
- Knn: Knn 알고리즘은 주어진 데이터 포인트와 가장 가까운 'k' 개의 훈련 데이터 포인트를 찾고, 이 데이터 포인트들이 속한 클래스 중 가장 많은 클래스를 예측 결과로 선택하는 방식입니다. 거리 측정 방법에 따라 이웃이 결정되며, 흔히 사용되는 방법으로는 유클리디안 거리나 맨해튼 거리가 있습니다. 특성 공간에서 복잡한 패턴을 가진 데이터에 잘 동작합니다.
- Naive Bayes Classification: 나이브 베이즈 분류는 특성 간의 독립성을 가정하고 베이즈 정리를 사용하여 분류를 수행합니다. 이 모델은 효율적인 학습 속도를 가지고 있습니다. 데이터셋의 크기가 커질수록, 특히 본 연구의 데이터와 같이 클수록 효율이 좋습니다.
- Logistic Regression: 로지스틱 회귀는 이진 분류 문제에서 주로 사용되는 기법입니다. 이 알고리즘은 로지스틱 함수를 사용하여 확률을 예측하며, 주어진 입력 특성들을 기반으로 특정 크래스에 속할 확률을 추정합니다. 확률이 0.5 이상이면 해당 클래스로, 그렇지 않으면 다른 클래스로 분류합니다. 특성과 목표 변수 간의 어느 정도 높은 상관관계를 확인하였으므로, 적합하다고 판된됩니다.
- SVM: SVM은 주어진 데이터를 가장 잘 구분하는 초평면을 찾는 알고리즘입니다. 이 초평면은 결정 경계와 가장 가까운 데이터 간의 거리인 마진을 최대화하는 방향으로 선택됩니다. 특성간의 복잡한 관계를 포착하기 위해 이 방법을 사용하였습니다.
- Random Forest: 랜덤 포레스트는 앙상을 기법입니다. 복원 추출 방법으로 여러 개의 훈련 데이터 셋을 만들고, 이를 각각의 결정 트리가 학습합니다. 최종 분류는 각 트리의 예측을 모아 투표를 통해 결정됩니다. 각 트리가 각기 다른 특성 부분 집합을 기반으로 학습하므로, 특성간 상호작용을 잘 포착할 수 있으므로, 복잡한 상호작용을 가지는 의료 데이터에 잘 어울립니다.
- AdaBoost: AdaBoost는 약한 학습기를 순차적으로 학습하면서 각 학습기가 만든 오차를 다음 학습기가 보완하는 방식으로 작동하는 앙상블 기법입니다. 모델의 성능은 각 학습기의 가중치를 적용한 결과를 결합하여 얻어집니다. 이 방법은 복잡한 분류 문제에 대해 높은 정확도를 제공합니다.
- MLP: MLP는 심층 인공 신경망의 한 형태로, 입력 계층, 여러 개의 은닉 계층, 그리고 출력 계층으로 구성됩니다. 각 계층은 뉴런으로 구성되어 있으며, 각 뉴런은 가중치, 편향, 활성화 함수를 가지고 있습니다. 이를 통해 복잡한 패턴을 학습할 수 있습니다. 따라서, 특성간 복잡한 상호작용이 예상되는 데이터 이므로 MLP를 사용하였습니다.

이렇게 총 8개의 다양한 알고리즘을 사용하여 모델링을 진행하였고, 이를 바탕으로 뇌졸중 예측을 진행했습니다.

5 실험

5.1 평가 방법

모델의 성능을 평가하기 위한 여러 지표를 사용하였습니다. 이 지표들은 모델이 얼마나 잘 예측하는지를 측정하기 위한 것으로, 정확도(accuracy), 정밀도(precision), 재현율(recall), F1 점수(f1 score)를 사용하였습니다.

- 정확도: 모델이 올바르게 분류한 샘플의 비율을 나타냅니다.
- 정밀도: 양성 클래스로 예측한 샘플 중 실제로 양성 클래스인 샘플의 비율을 나타냅니다. 모델이 만든 양성 클래스 예측의 정확성을 평가합니다.
- 재현율: 실제 양성 클래스 중 모델이 양성 클래스로 올바르게 예측한 샘플의 비율을 나타냅니다. 모델이 실제 양성 클래스를 얼마나 잘 찾아내는지를 평가합니다.
- F1 점수: 정밀도와 재현율의 조화 평균을 나타냅니다. 이 두 지표의 균형을 유지하는 모델의 성능을 평가합니다.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{4}$$

이러한 지표들은 GridSearchCV[3] 객체를 만들 때 'scoring' 매개변수를 통해 지정되었습니다. 이후, 교차 검증(cross-validation)을 사용하여 각 모델의 최적의 하이퍼파라미터를 찾고, 이를 통해 각 모델의 성능을 평가하였습니다. 교차 검증에서는 데이터를 10개의 부분으로 나눈 후, 각 부분을 테스트 세트로 사용하여 모델을 평가하였습니다. 이 방법은 모델이 새로운 데이터에 대해 얼마나 잘 일반화되는지 평가하는 데 효과적입니다. 최종적으로는 정확도를 기준으로 최적의 모델을 선택하였습니다.

또한, 테스트 데이터셋을 사용하여 각 모델의 성능을 직접 검증하였습니다. 이를 통해 모델의 베스트 파라미터와 각 성능 지표(정확도, 정밀도, 재현율, F1 점수)를 출력하였습니다. 이후, Confusion Matrix를 통해 각 모델의 성능을 시각적으로 비교하였습니다.

마지막으로, 각 모델의 성능을 통계적으로 비교하기 위해, t-검정을 사용하였습니다. t-검정은 두 집단 간의 평균 차이가 통계적으로 유의미한지를 판단하는 검증 방법입니다. 이를 통해 각 모델의 예측 결과가 통계적으로 유의미하게 차이가 있는지를 판별하였습니다. 이 방법은 각 모델의 성능 차이가 우연에 의한 것인지, 아니면 실제로 차이가 있는지를 확인하는 데 유용합니다.

5.2 실험 환경

본 연구는 Google Colab에서 수행되었습니다. 사용된 컴퓨팅 리소느는 CPU: Intel Xeon 2.2GHz, RAM: 13GB, 저장공간: 33GB입니다. 본 실험에서 다양한 머신러닝 모델들을 학습하고 평가하였습니다. SVM 모델의 경우 학습 시간이 가장 오래걸렸으며, 4시간 이상 소요되었습니다. 그 외의 모델들은 대체로 1시간 미만의 시간이 소요 되었습니다.

5.3 Decision Tree

이 실험에서는 Decision Tree 분류기의 최적의 하이퍼파라미터를 찾는 것이 목표입니다. 'max_depth'와 'criterion' 두 가지 하이퍼파라미터를 대상으로 최적의 하이퍼 파라미터를 찾는 과정을 진행했습니다. 'max_depth'는 트리의 최대 깊이를 나타내며, 이는 오버피팅을 방지하는데 도움이 됩니다. 'max_depth'는 [1, 5, 10, 20, None] 중에서 선택되었는데, 이는 트리의 복잡도와 일반화 능력 사이의 균형을 찾기 위한 것입니다. 'None'은 노드가 모든 클래스가 순수해질 때까지 확장되는 트리의 제한이 없음을 의미합니다. 'criterion'은 노드 분할을 위한 기준을 설정합니다. 우리는 'gini'와 'entropy'를 비교하여 사용하였는데, 'gini'는 잘못 분류될 확률을 최소화하며, 'entropy'는 정보 이득을 최대화하는 방식으로 노드를 분할합니다. Table 3. 결과 최적의 하이퍼파라미터는 'criterion': 'gini', 'max_depth': None 일 때 입니다.

param_max_depth	param_criterion	mean_test_accuracy	mean_test_precision	mean_test_recall	mean_test_f1
1	gini	0.620	0.660	0.621	0.596
5	gini	0.705	0.718	0.705	0.701
10	gini	0.825	0.835	0.825	0.824
20	gini	0.991	0.991	0.991	0.991
None	gini	1.000	1.000	1.000	1.000
1	entropy	0.620	0.660	0.621	0.596
5	entropy	0.699	0.710	0.699	0.695
10	entropy	0.818	0.819	0.818	0.818
20	entropy	0.977	0.977	0.977	0.977
None	entropy	1.000	1.000	1.000	1.000

Table 3: Grid Search CV results for Decision Tree Classifier

5.4 Knn

이 실험에서는 Knn 분류기의 최적의 하이퍼파라미터를 찾는 것이 목표입니다. 'n_neighbors'와 'weights'라는 두 가지 주요 하이퍼파라미터에 초점을 맞췄습니다. 'n_neighbors'는 분류 결정에 참여하는 가장 가까운 이웃의 수를 나타냅니다. 이것은 [3, 5, 7, 10] 사이에서 선택하였습니다. 'weights'는 이웃에 부여되는 가중치를 나타내며 'uniform'과 'distance' 중에서 선택하였습니다. 'uniform'은 모든 이웃에 동일한 가중치를 부여하며, 'distance'는 가까운 이웃에 더 큰 가중치를 부여합니다. Table 4. 결과 최적의 하이퍼파라미터는 'n_neighbors': 3, 'weights': 'distance' 일 때 입니다.

param_n_neighbors	param_weights	mean_test_accuracy	mean_test_precision	mean_test_recall	mean_test_f1
3	uniform	0.993	0.993	0.993	0.993
3	distance	0.999	0.999	0.999	0.999
5	uniform	0.982	0.983	0.982	0.982
5	distance	0.999	0.999	0.999	0.999
7	uniform	0.974	0.975	0.974	0.974
7	distance	0.999	0.999	0.999	0.999
10	uniform	0.965	0.968	0.965	0.965
10	distance	0.999	0.999	0.999	0.999

Table 4: Grid Search CV results for K-Nearest Neighbors Classifier

5.5 Naive Bayes Classification

Gaussian Naive Bayes 분류기는 특성들이 서로 독립적이라는 "Naive" 가정에 기반하며, 특성의 가능한 값들이 가우스 분포(정규 분포)를 따른다고 가정하는 분류기입니다. 이 모델은 특별한 하이퍼파라미터가 없습니다.

Metric	Score
Accuracy	0.670
Precision	0.682
Recall	0.670
F1 Score	0.664

Table 5: CV results for Gaussian Naive Bayes Classifier

5.6 Logistic Regression

이 실험에서는 Logistic Regression 분류기의 최적의 하이퍼파라미터를 찾는 것이 목표입니다. 'C'라는 주요 하이퍼파라미터가 있는데, 이는 정규화 강도를 제어합니다. 높은 'C' 값은 모델에 더 많은 자유도를 제공하며, 낮은 'C' 값은 모델을 더 강력하게 정규화합니다. 'C'값은 [0.01, 0.1, 1, 10, 100] 중에서 선택하였습니다. Table 6. 결과 최적의 하이퍼파라미터는 'C': 0.01 일 때 입니다.

5.7 Support Vector Machine

이 실험에서는 SVM 분류기의 최적의 하이퍼파라미터를 찾는 것이 목표입니다. 서포트 벡터 머신에는 'C', 'gamma', 'kernel'이라는 주요 하이퍼파라미터가 있습니다. 'C'는 분류 오류를 허용하는 정도를 제어하는 매개변수입니다. 'C' 값이 높을수록, 분류 오류를 덜 허용하며, 'C' 값이 낮을수록, 분류 오류를 더 많이 허용합니다. 이 실험에서는 'C'를 [1, 10, 100] 중에서 선택

param_C	mean_test_accuracy	mean_test_precision	mean_test_recall	mean_test_f1
0.01	0.680	0.686	0.680	0.678
0.1	0.678	0.683	0.678	0.676
1	0.679	0.683	0.679	0.677
10	0.679	0.683	0.679	0.677
100	0.679	0.683	0.679	0.677

Table 6: Grid Search CV results for Logistic Regression Classifier

하였습니다. 'gamma'는 'rbf', 'poly' 및 'sigmoid' 커널에 대한 계수입니다. 감마가 높을수록 모델이 복잡해지며, 감마가 낮을수록 모델이 단순해집니다. 이 실험에서는 'gamma'를 [1, 0.1, 0.01] 중에서 선택하였습니다. 'kernel'은 사용할 커널 유형을 정의합니다. 이 실험에서는 'rbf' 커널만 사용하였습니다. Table 7. 결과 최적의 하이퍼파라미터는 'C': 100, 'gamma': 1, 'kernel': 'rbf' 일 때 입니다.

param_C	param_gamma	param_kernel	mean_test_accuracy	mean_test_precision	mean_test_recall	mean_test_f1
1	1	rbf	0.686	0.689	0.686	0.685
1	0.1	rbf	0.680	0.689	0.681	0.677
1	0.01	rbf	0.655	0.684	0.655	0.641
10	1	rbf	0.692	0.693	0.692	0.692
10	0.1	rbf	0.684	0.691	0.684	0.681
10	0.01	rbf	0.659	0.685	0.659	0.646
100	1	rbf	0.695	0.696	0.695	0.695
100	0.1	rbf	0.686	0.693	0.686	0.684
100	0.01	rbf	0.680	0.689	0.680	0.676

Table 7: Grid Search CV results for Support Vector Machine

5.8 Random Forest

이 실험에서는 Random Forest 분류기의 최적의 하이퍼파라미터를 찾는 것이 목표입니다. 주요 하이퍼파라미터로는 'n_estimators'와 'max_depth'가 있습니다. 'n_estimators'는 사용할 트리의 개수를 결정하는 매개변수로, 이 실험에서는 [10, 50, 100, 200] 중에서 선택하였습니다. 'max_depth'는 트리의 최대 깊이를 결정하는 매개변수입니다. 이 실험에서는 [1, 5, 10, 20, None] 중에서 선택하였습니다. 여기서 None은 깊이에 제한이 없음을 의미합니다. Table 8. 결과 최적의 하이퍼파라미터는 'max_depth': None, 'n_estimators': 200 일 때 입니다.

param_n_estimators	param_max_depth	mean_test_accuracy	mean_test_precision	mean_test_recall	mean_test_f1
10	1	0.664172	0.677851	0.664392	0.657361
50	1	0.675263	0.682794	0.675432	0.672008
100	1	0.676271	0.682985	0.676430	0.673412
200	1	0.678839	0.684958	0.678988	0.676282
10	5	0.708510	0.718683	0.708688	0.705188
50	5	0.705974	0.716463	0.706157	0.702484
100	5	0.705668	0.715934	0.705849	0.702250
200	5	0.705760	0.716383	0.705944	0.702232
10	10	0.870588	0.870943	0.870597	0.870556
50	10	0.877250	0.877666	0.877272	0.877220
100	10	0.877983	0.878479	0.878008	0.877947
200	10	0.878227	0.878574	0.878250	0.878203
10	20	0.998289	0.998298	0.998286	0.998289
50	20	0.999358	0.999361	0.999357	0.999358
100	20	0.999328	0.999330	0.999327	0.999328
200	20	0.999328	0.999330	0.999327	0.999328
10	None	0.999328	0.999331	0.999327	0.999328
50	None	0.999450	0.999452	0.999449	0.999450
100	None	0.999419	0.999422	0.999418	0.999419
200	None	0.999481	0.999482	0.999480	0.999481

Table 8: Grid Search CV results for Random Forest

5.9 AdaBoost

이 실험에서는 AdaBoost 분류기의 최적의 하이퍼파라미터를 찾는 것이 목표입니다. 주요 하이퍼파라미터로는 'n_estimators'와 'learning_rate'가 있습니다. 'n_estimators'는 사용할 weak learner(약한 학습기)의 개수를 결정하는 매개변수로, 이 실험에서는 [10, 50, 100, 200] 중에서 선택하였습니다. 'learning_rate'는 학습률을 조절하는 매개변수로, 각 약한 학습기가 학습에 기여하는 정도를 결정합니다. 이 실험에서는 [0.01, 0.1, 1] 중에서 선택하였습니다.

Table 9. 결과 최적의 하이퍼파라미터는 'learning_rate': 1, 'n_estimators': 200 일 때 입니다.

param_n_estimators	param_learning_rate	mean_test_accuracy	mean_test_precision	mean_test_recall	mean_test_f1
10	0.01	0.620	0.659	0.621	0.596
50	0.01	0.659	0.671	0.659	0.653
100	0.01	0.680	0.686	0.680	0.678
200	0.01	0.681	0.686	0.681	0.679
10	0.1	0.681	0.687	0.681	0.679
50	0.1	0.681	0.689	0.681	0.677
100	0.1	0.679	0.689	0.679	0.675
200	0.1	0.690	0.698	0.691	0.688
10	1	0.680	0.683	0.680	0.678
50	1	0.715	0.717	0.715	0.715
100	1	0.735	0.736	0.735	0.734
200	1	0.761	0.762	0.762	0.761

Table 9: Grid Search CV results for AdaBoost

5.10 Multi-Layer Perceptron

이 실험에서는 MLP(Multi-layer Perceptron) 분류기의 최적의 하이퍼파라미터를 찾는 것이 목표입니다. 주요 하이퍼파라미터로는 'hidden_layer_sizes'와 'activation'이 있습니다. 'hidden_layer_sizes'는 은닉층의 크기를 결정하는 매개변수로, 이 실험에서는 [(10,), (50,), (100,), (20, 10,)] 중에서 선택하였습니다. 'activation'는 활성화 함수를 결정하는 매개변수로, 이 실험에서는 ['relu', 'tanh', 'logistic'] 중에서 선택하였습니다. Table 10.의 결과 최적의 하이퍼파라미터는 'activation': 'relu', 'hidden_layer_sizes': (100,) 일 때 입니다.

param_hidden_layer_sizes	param_activation	mean_test_accuracy	mean_test_precision	mean_test_recall	mean_test_f1
(10,)	relu	0.691	0.692	0.691	0.691
(50,)	relu	0.700	0.701	0.700	0.699
(100,)	relu	0.705	0.707	0.705	0.705
(20, 10)	relu	0.702	0.703	0.702	0.702
(10,)	tanh	0.694	0.695	0.695	0.694
(50,)	tanh	0.695	0.697	0.695	0.694
(100,)	tanh	0.695	0.697	0.695	0.695
(20, 10)	tanh	0.702	0.704	0.702	0.702
(10,)	logistic	0.681	0.683	0.681	0.680
(50,)	logistic	0.685	0.686	0.685	0.684
(100,)	logistic	0.684	0.686	0.684	0.683
(20, 10)	logistic	0.682	0.684	0.682	0.681

Table 10: Grid Search CV results for Multi-Layer Perceptron

6 결과 및 해석

Table 11.에서 보시다시피, 여러 머신러닝 모델들을 이용하여 예측 모델을 구축하고 평가하였습니다. 각 모델의 성능은 Accuracy, Precision, Recall 및 F1-Score 기준으로 측정되었습니다. 이러한 메트릭들은 모델의 성능을 종합적으로 이해하는 데 중요한 역할을 합니다. 세 가지 모델, 즉 Decision Tree, Random Forest, 그리고 Knn 모델의 성능은 특히 독보적인 것으로 확인되었습니다. 이와 반대로, Naive Bayes, Logistic Regression, Support Vector Machine, AdaBoost, Multi-Layer Perceptron과 같은 다른 모델들은 상대적으로 낮은 점수를 기록하였습니다. 이들 모델은 성능 면에서 위에서 언급한 세 가지 모델에 비해 더 많은 개선의 여지가 있음을 시사합니다.

통계적 유의성 검정인 T-test 결과 Table 12.를 살펴보면, 세 모델 간의 성능 차이가 모두 통계적으로 유의미하다는 것을 확인할 수 있습니다. 이는 모든 경우에서 p-value가 0.05 미만으로 나타난 것을 통해 확인되었습니다. 이러한 결과는 Decision Tree가 가장 높은 성능을 보였고, 이를 차례로 Random Forest와 Knn이 뒤따른다는 것을 의미합니다.

Figure 4, 5, 6.은 각각 Decision Tree, Random Forest, 그리고 Knn 모델의 혼동 행렬 (Confusion Matrix)을 보여줍니다. 이 혼동 행렬들을 통해 모델의 성능을 더욱 세밀하게 이해할 수 있습니다. 특히, Random Forest와 Knn 모델에서는 실제로는 음성인 샘플을 양성으로 예측하는 경우가 일부 발생하였습니다. 의학적 문맥에서 이렇게 '거짓 양성'이 발생하는 것은 대체로 '거짓 음성' 발생보다 더 선호되는 경향이 있습니다. 이는 질병이 있는 환자를 놓치는 것보다는 질병이 없는 환자를 조금 더 세심하게 검사하는 것이 더 큰 문제를 방지하기 때문입니다. 따라서, 이러한 결과는 Random Forest와 Knn 모델이 이러한 의학적 문맥에서도 유용하게 사용될 수 있음을 보여줍니다.

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	1.000	1.000	1.000	1.000
K-Nearest Neighbors	0.999	0.999	0.999	0.999
Naive Bayes	0.6733	0.6844	0.6727	0.6678
Logistic Regression	0.6843	0.6892	0.6839	0.6819
Support Vector Machine	0.6969	0.6973	0.6968	0.6966
Random Forest	0.9996	0.9996	0.9996	0.9996
AdaBoost	0.7723	0.7725	0.7722	0.7722
Multi-Layer Perceptron	0.7085	0.7086	0.7084	0.7084

Table 11: Performance metrics for different models

Model 1	Model 2	t-statistic	p-value
Decision Tree	Knn	-10.794	5.57×10^{-27}
Decision Tree	Random Forest	-3.210	0.00133
Knn	Random Forest	9.763	2.14×10^{-22}

Table 12: Comparison of model performance using paired t-test

종합적으로, 이 분석을 통해 Decision Tree, Random Forest, 그리고 Knn 모델이 본 연구에서 사용된 데이터셋에 대해 우수한 성능을 보였음을 확인하였습니다. 특히, Decision Tree 모델이 가장 높은 정확도를 보였으며, 그 다음으로 Random Forest와 Knn 모델이 뒤를 이었습니다. 또한, 모든 모델이 의학적 문맥에서 중요한 특성인 '거짓 양성'에 대해 적절히 대응하였음을 확인하였습니다. 이런 결과는 이들 모델이 향후 유사한 문제에 대해 적용될 때 높은 성능을 보일 수 있음을 시사합니다.

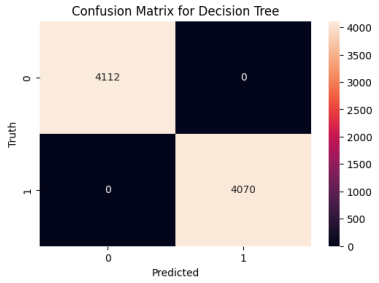

Figure 4: Confusion Matrix for Decision Tree

7 결론

본 연구에서는 뇌졸중 예측에 대한 머신러닝 모델의 성능을 평가하였습니다. 특히, Decision Tree, Random Forest, 그리고 K-Nearest Neighbors (Knn) 모델을 활용하였고, 이들 모델의 성능 비교를 통해 뇌졸중 예측에 가장 효과적인 모델을 찾아냈습니다.

결과적으로, Decision Tree 모델이 가장 뛰어난 예측 성능을 보였으며, 그 다음으로 Random Forest와 Knn 모델이 우수한 성능을 보였습니다. 더불어, 세 모델 간의 성능 차이는 통계적으로 유의미하였으며, 모든 모델들이 뇌졸중 예측에 필요한 중요한 요소인 'False Positive' 예측을 적절하게 처리하였습니다.

이번 연구의 결과는 뇌졸중의 미리 예측하고 예방하는 데 중요한 단서를 제공하며, 향후 뇌졸중 예측 모델 개발에 있어서 중요한 참고가 될 수 있습니다. 또한, 다양한 머신러닝 모델을 실제 문제에 적용하고, 그 성능을 비교하는 과정은 머신러닝을 활용한 문제 해결에 일반적으로 중요한 과정이며, 본 연구를 통해 그 중요성을 한층 더 확인하였습니다.

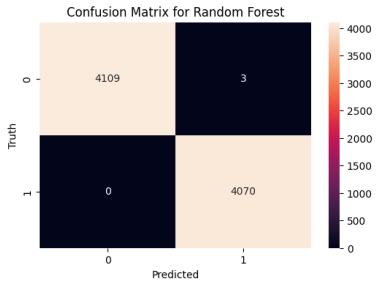

Figure 5: Confusion Matrix for Random Forest

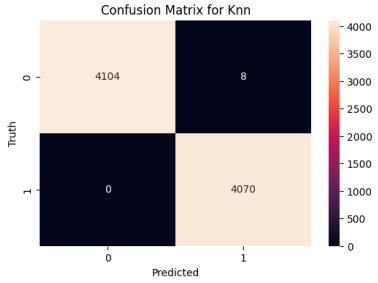

Figure 6: Confusion Matrix for Knn

모든 모델의 적용과 성능 평가는 해당 분야의 문맥과 요구 사항을 충분히 고려해야 함을 인지해야 합니다. 또한, 모델의 성능을 평가할 때에는 다양한 지표를 고려하여 종합적인 성능을 이해하는 것이 중요합니다. 이러한 점을 반영하여 본 연구를 진행한 결과, 의미있는 모델 선정과 성능 평가를 할 수 있었으며, 이는 뇌졸중 예측 뿐만 아니라 다양한 머신러닝 문제에 적용될 수 있을 것입니다.

References

[1] 서울성모병원 평생건강증진센터. 뇌졸중이란?

[2] Diabetes, hypertension and stroke prediction. kaggle. `https://www.kaggle.com/datasets/prosperchuks/health-dataset`.

[3] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.