안전벨트의 중요성

2018020
박경민
https://github.com/2018020Park/Seatbelt

1. 안전 관련 머신러닝 모델 개발의 목적

매년 일어나는 사고 중 단연 1위는 교통사고라고 할 수 있다. 교통사고에서 안전벨트의 중요성을 부각시키고자 개발하였다.

교통사고의 위험에 대한 교육이나 운전을 배울 때 보면 좋다.

안전벨트의 착용 유무를 독립변수로 설정하여 어떨 때 죽을 확률이 높은지 구하였다.

2. 안전관련 머신러닝 모델의 네이밍의 의미

안전벨트의 중요성은 이름 그대로 안전벨트를 잘 착용하자는 의미에서 지었다.

3. 개발 계획

3.1 데이터에 대한 요약 정리 및 시각화

dvcat	weight	dead	airbag	seatbelt	frontal	sex	ageOFocc	yearacc	yearVeh	abcat	occRole	deploy	caseid	injSeverity
55+	53.342	dead	airbag	belted	1	f	48	2002	1997	deploy	driver	1	45:150:1	3
25-39	154.96	alive	none	none	1	m	26	2001	1968	unavail	driver	0	76:40:01	3
55+	38.994	alive	none	none	1	f	51	2002	1994	unavail	driver	0	11:184:1	3
25-39	168.568	alive	airbag	belted	1	m	27	1998	1996	deploy	pass	1	9:17:01	3
10-24	27.751	alive	airbag	belted	0	m	26	2002	1997	nodeploy	pass	0	2:02:02	0
10-24	720.453	alive	airbag	belted	1	m	47	1998	1997	nodeploy	driver	0	11:32:01	0
10-24	1	alive	airbag	belted	1	m	17	2002	1991	deploy	driver	1	50:03:01	2
40-54	20.862	alive	none	none	1	m	39	2001	1994	unavail	driver	0	73:95:1	3
10-24	115.576	alive	none	belted	1	f	74	2002	1987	unavail	driver	0	78:107:1	3

3.2 데이터 전처리 계획

- 범주형 데이터를 변환한다.
  데이터에 보면 'dead' 가 'dead', 'alive' 둘 중에 하나인 것을 알 수 있다. 0또는 1로 변환한다.
- 결측치를 처리한다.
  weight같은 경우 평균값으로 대체한다.
- 이상치를 제거한다.
  weight의 값의 변동이 크기 때문에 500이하로 잡는다.

3.3 머신러닝 모델 선택

- 로지스틱 회귀를 사용한다.

※로지스틱 회귀란.

- 주로 이진 분류 문제를 해결하는 데 사용함.(종속 변수가 0또는 1인 경우)

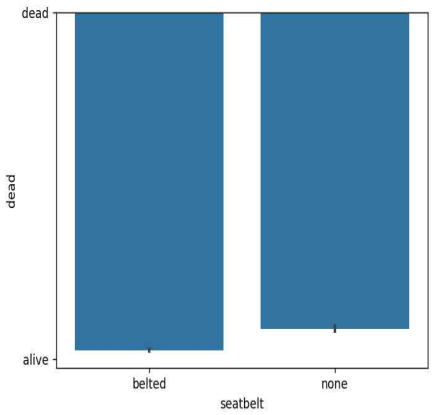
3.4 성능 검증 방법

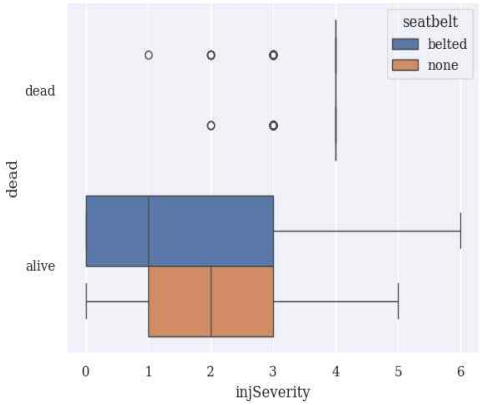
- accuracy로 정확도를 평가할 계획이다.

4. 개발 과정

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data = pd.read_csv('./data/train-new.csv')

sns.barplot(data=data,x='dead',y='injSeverity')
sns.barplot(data=data,x='seatbelt',y='injSeverity')
sns.boxplot(x='injSeverity',y='dead',data=data,hue='seatbelt')
```

- 안전벨트와 죽음의 상관관계를 확인하기 위해 시각화하였다.






```
# 데이터 불러오기
data = pd.read_csv("./data/train-new.csv")

# 전처리 단계
data['seatbelt'] = data['seatbelt'].map({'none': 0, 'belted': 1})
data['weight'].fillna(data['weight'].mean(), inplace=True)
data = data[data['weight'] < 500]
data['dead'] = data['dead'].map({'dead': 0, 'alive': 1})
```

- pandas를 이용하여 데이터를 불러왔고,
- 전처리 단계로서 안전벨트와 죽음에 대해서 0또는 1로 변환하였다.
- 무게는 사용하지 않지만 오류발생방지를 위해 전처리를 실행했다.

```
# Feature와Target 설정
X = data[['seatbelt']]
y = data['dead']
```

- 독립변수와 종속변수를 설정했다.

```
# 학습 데이터와 테스트 데이터로 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# 로지스틱 회귀 모델 학습
logreg_model = LogisticRegression()
logreg_model.fit(X_train, y_train)

# 예측
y_pred_logreg = logreg_model.predict(X_test)
```

- 로지스틱 회귀 모델로 학습시키기 위해 학습 데이터와 테스트 데이터로 분리했다.

```
# 정확도 평가
accuracy_logreg = accuracy_score(y_test, y_pred_logreg)
print(f'Accuracy (Logistic Regression): {accuracy_logreg:.2f}')
```

- 코드가 실행되고 정확도를 구하기 위해 accuracy함수를 사용했다.

```
# ROC 곡선 생성
fpr, tpr, thresholds = roc_curve(y_test, y_prob_logreg)

# AUC 계산
auc = roc_auc_score(y_test, y_prob_logreg)

# ROC 곡선 시각화
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc:.2f})', color='blue')
plt.plot([0, 1], [0, 1], '--', color='gray', label='Random')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.title('ROC Curve - Logistic Regression')
plt.legend()
plt.show()
```

- 한 눈에 알아보기 쉽게 ROC곡선으로 나타내었다.

● 에러 발생지점

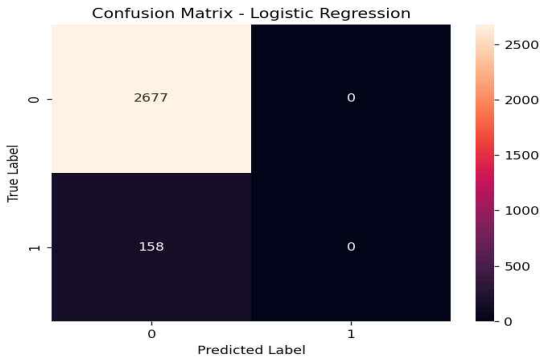
```
# 전처리 단계
data['seatbelt'] = data['seatbelt'].map({'none': 0, 'belted': 1})
data['weight'].fillna(data['weight'].mean(), inplace=True)
data = data[data['weight'] < 500]
data['dead'] = data['dead'].map({'dead': 0, 'alive': 1})
```

- 전처리 단계에서 0과 1로 변환하지 않아 코드가 실행되지 않았었다.

● 다른 시도

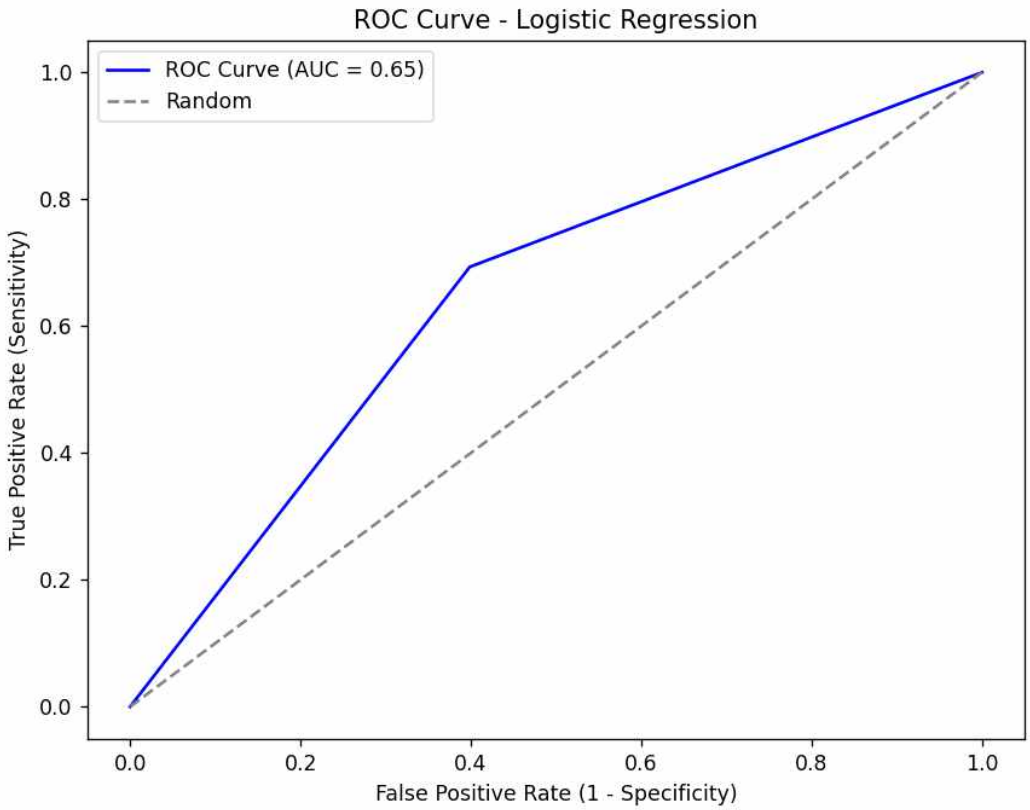
```
sns.heatmap(confusion_matrix(y_test, y_pred_logreg), annot=True, fmt='d')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix - Logistic Regression')
plt.show()
```

- 성능을 평가하기 위해 오차 행렬을 사용해 보았는데, alive와 dead의 차이가 너무 커서 잘 나오지 않았다.(잘 모르겠습니다..)



결과

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score

# 데이터 불러오기
data = pd.read_csv("./data/train-new.csv")

# 전처리 단계
data['seatbelt'] = data['seatbelt'].map({'none': 0, 'belted': 1})
data['weight'].fillna(data['weight'].mean(), inplace=True)
data = data[data['weight'] < 500]
data['dead'] = data['dead'].map({'dead': 0, 'alive': 1})

# Feature와Target 설정
X = data[['seatbelt']]
y = data['dead']

# 학습 데이터와 테스트 데이터로 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# 로지스틱 회귀 모델 학습
logreg_model = LogisticRegression()
logreg_model.fit(X_train, y_train)

# 예측
y_pred_logreg = logreg_model.predict(X_test)

# 정확도 평가
accuracy_logreg = accuracy_score(y_test, y_pred_logreg)
print(f'Accuracy (Logistic Regression): {accuracy_logreg:.2f}')

# 로지스틱 회귀에 대한 예측 확률 얻기
y_prob_logreg = logreg_model.predict_proba(X_test)[:, 1]

# ROC 곡선 생성
fpr, tpr, thresholds = roc_curve(y_test, y_prob_logreg)

# AUC 계산
auc = roc_auc_score(y_test, y_prob_logreg)

# ROC 곡선 시각화
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc:.2f})', color='blue')
plt.plot([0, 1], [0, 1], '--', color='gray', label='Random')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.title('ROC Curve - Logistic Regression')
plt.legend()
plt.show()

# 분류 보고서 출력
print(classification_report(y_test, y_pred_logreg))
```



	precision	recall	f1-score	support
0	0.00	0.00	0.00	158
1	0.94	1.00	0.97	2677
accuracy			0.94	2835
macro avg	0.47	0.50	0.49	2835
weighted avg	0.89	0.94	0.92	2835

5. 개발 후기

만들고 난 지금도 느끼는 것이지만, 잘 모르겠다. 정말 어렵다. 사실 벼락치기로 이해하려고 해서 그런 것인지도 모른다. 파이썬이 정말 재미있지만, 이번엔 대충 공부한 느낌이 난다. 사실 내가 개발했다기보다 kaggle과 gpt의 힘을 많이 빌렸다. 중간고사때 했던 기본 파이썬은 어린애들 장난이었다는게 느껴질 정도로 머신러닝은 난이도가 높았다. 또 기회가 된다면 그 땐 확실하게 공부하고 코딩을 짜보고 싶다. 그렇다고 이번에 공부를 안 한 것은 아니지만 너무 어렵다..

담번엔 더 열심히 해보겠습니다.

한 학기 동안 고생 많으셨습니다. 교수님. (교수님 안재현 닮았어요..!)