COSE362 Machine Learning Final Report

2017320108 고재영

2017320118 임규정

2017320121 이준구

➢ *풀고자 하는 문제 정의*

저희 조의 프로젝트 주제는 교통 환경을 고려한 교통사고 피해 강도의 예측입니다. 교통 환경에 영향을 끼치는 날씨, 시간, 도로 상태, 조명 강도 등 요인들끼리의 상관관계(correlation)에 대하여 기계학습을 통해 확인하고, 어느 환경에서 교통사고 시 예상되는 피해가 가장 큰지를 예측하는 모델을 만드는 것입니다.

학습 모델을 통해 답하고자 하는 질문은 크게 두 가지입니다.

- 교통사고가 발생한 주변 환경을 나타내는 지표만 사용해서 교통사고 피해 강도를 정확하게 예측할 수 있을까?
- 어떤 요인(feature)들이 교통사고 피해의 강도와 가장 큰 연관성이 있을까?

이하는 학습 모델에 사용된 Kaggle dataset입니다.
https://www.kaggle.com/datasets/dorianvoydie/2019-database-of-road-traffic-injuries

➢ *기계학습 방법론*

프로젝트 진행 중 사용한 방법론은 총 6가지입니다. One-hot encoding, upsampling, feature selection, random forest, k-fold cross validation, multi-output logistic regression 순으로 사용되었습니다.

➢ *중간제출 레포트 내용 요약 및 보완 방법*

중간 레포트 제출까지 저희가 진행한 내용은 다음과 같습니다.

1. 데이터 전처리 (data munging, 사용할 feature들 종합, label 표기 변경)
2. 데이터 train-test split
3. One-hot encoding
4. Dimensionality reduction - Chi-square feature selection
5. Decision tree 모델 구현

이하는 중간 레포트 기점으로 학습에 사용된 8개의 feature들입니다.

날씨('atm', atmospheric condition)
시간('hrmn', hour and minutes)
조명('lum', light)
도로 표면 상태 ('surf', surface condition)
도로 종류('catr', road category)
교차로 종류('int', intersection)
교통 체제('circ', traffic regime)
시설('infra', infrastructure)

위 feature들을 사용하여 교통사고 피해 강도 ('grav', severity of user injury)를 예측하는 것이 프로젝트의 목적입니다.

첫 모델 구현 결과를 보고, 저희가 보완해야 하는 부분은 크게 두 가지였습니다. 첫 번째는 모델의 성능에 대한 지표입니다. 구현 과정을 거쳐 모델의 accuracy를 약 50~60%가 나오게끔 했지만, 성능을 더 높일 필요성을 느꼈습니다. 더군다나, f1-score는 여러 과정을 거치고서도 0.1를 넘기지 못하는 모습을 보아 아직까지 학습 모델의 완성도가 많이 떨어진다는 결론을 내렸습니다. 따라서 중간 레포트에 작성한 대로, voting 과정을 도입하거나 feature를 추가함으로써 모델의 성능을 향상시키고자 했습니다. 자세한 구현 방법은 이하 페이지에 설명하겠습니다.

두 번째는 data skewing 현상입니다. 학습 모델이 예측하고자 하는, 교통사고 피해 강도를 나타내는 'grav' feature는 'unharmed'(피해 없음), 'killed'(사망), 'injured hospitalized'(중상), 'slightly injured'(경상)으로 나뉘어져 있습니다. 중간 레포트 제출 당시, 저희는 모델 구현을 간소화하기 위해 저희는 4개의 출력값을 'killed'와 'survived', 2개로 줄여 사용했습니다. 이 과정에서 'killed' 데이터의 양이 나머지 3개의 데이터를 합친 'survived'의 양보다 압도적으로 적다 보니 'killed' 데이터가 outlier처럼 처리되는 것을 확인할 수 있었습니다. 이를 보완하기 위해 4개의 출력값을 모두 사용하는 multi-output 모델을 구현하는 방법을 택했습니다.

➢ *구현 과정 - 전처리 및 Feature Selection*

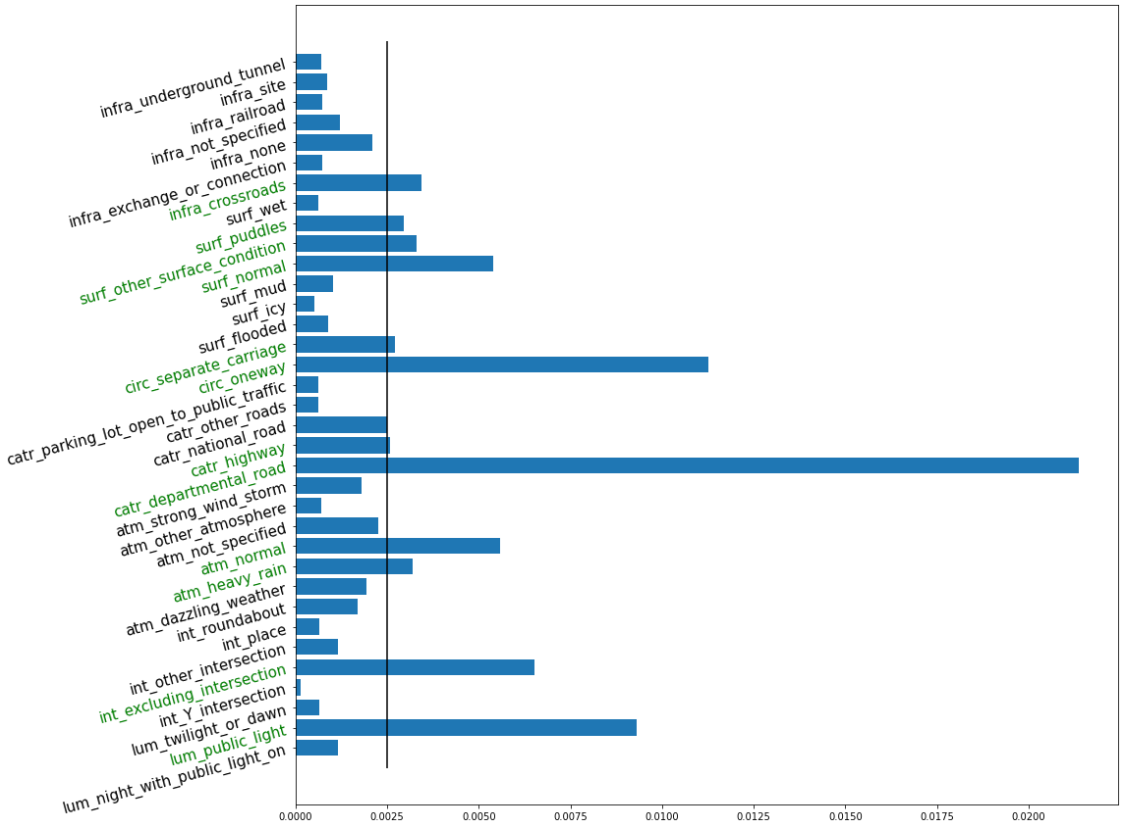
데이터 전처리 과정은 중간 레포트 내용과 유사하게 진행했습니다. 사용한 dataset이 4개의 파일로 나눠져 있었기 때문에, 필요한 feature들만 종합하여 dataframe을 생성했습니다. Categorical data의 경우, 직관성을 위해 출력되는 값을 numerical에서 descriptive하게 변경했습니다. 예를 들어, 앞서 소개한 'grav' feature가 원래 [1, 2, 3, 4] 중 하나를 출력했다면, 수정을 거쳐 ['unharmed', 'killed', 'injured hospitalized', 'slightly injured'] 중 하나를 출력하도록 하는 것입니다.

데이터 정리 후에는 one-hot encoding, upsampling, 그리고 train-test split을 순서대로 실시했습니다. One-hot encoding은 중간 레포트에 작성했듯이 categorical data를 학습에 용이하도록 조정하려는 목적도 있지만, categorical data인 만큼 데이터의 연속성을 제거하기 위해서도 중요한 과정입니다. 중간 레포트 제출 때는 data skewing을 방지하기 위해 downsampling을 진행했지만, 이후 upsampling을 사용하는 방안으로 선회했습니다. Downsampling을 하면 데이터의 크기가 양이 가장 적은 'killed' 데이터의 크기에 맞춰지는데, 이 경우 학습에 사용되는 데이터의 양이 과하게 적어지는 문제가 생기기 때문입니다.

이하 보고서에 사용된 모든 이미지들은 ipynb 코드에서 첫 코드 블럭에 속한 randomness 변수를 40으로 설정하면 재현 가능합니다. randomness 변수를 다른 값으로 설정하거나 randomness=None (모든 과정을 랜덤하게 처리) 로 설정할 경우 밑 이미지와 조금 다른 결과들이 생성되지만, 모델의 성능에는 큰 차이가 없습니다.

➢ *구현 과정 - Feature Selection*

학습 과정에 사용되는 feature의 개수는 원래 8개로, 많지 않은 개수입니다. 그러나, categorical feature들에 one-hot encoding을 실시하면 feature의 개수가 52개까지 증가하므로, feature selection이 불가피했습니다. 중간 레포트 제출 시 사용됐던 chi-square feature selection은 p값이 feature들의 dependency가 높을수록 0에 가까워지는 성질이 있지만, 샘플의 크기가 커져도 0에 수렴한다는 단점이 존재합니다. 즉, p값이 0에 근접한 값일 경우, 이 feature가 dependency가 높아서 생긴 값인지 아니면 샘플 크기가 커서 일어난 현상인지 구분하기가 어렵습니다. Upsampling을 하면 데이터의 양이 자연스럽게 많아질 수밖에 없기 때문에 위 방법은 사용이 부적합하다는 결론을 내렸습니다.



대신, mutual_info_classif 함수를 사용하여 각 feature가 교통사고 피해 강도 'grav' feature와 가지는 dependency를 수치화했습니다. 그 후, threshold를 정해서 'grav'와 가장 강한 연관성을 가진 feature들을 선정했습니다. 위 이미지의 경우, threshold 값을 0.0025로 설정했을 때 52개의 feature 중 12개가 선택된 것을 볼 수 있습니다. 그래프의 총 feature 개수가 52개보다 적은 이유는 'grav' feature와의 dependency가 0인 feature들은 그래프에 포함하지 않았기 때문입니다.

➢ *구현 과정 - Random Forest*

저희가 처음으로 고안한 모델은 training size에만 변동을 주는 단순한 decision tree 모델이었습니다. 첫 모델에서 발생한 문제인 overfitting, data skewing 등을 해결하기 위해 여러 개의 decision tree를 생성한 후, voting 과정을 거치는 것이 이상적이라고 판단했습니다. 이에 따라, 앞서 선별된 dependency가 높은 feature들로 101개의 decision tree를 생성하여 random forest classifier를 사용했습니다.

Random Forest

```
[ ]  #unraveling to avoid warning message
     Y_train_forest = Y_train.values.ravel()

     #create random forest with 101 trees
     clf = RandomForestClassifier(n_estimators = 101, random_state = randomness)

     #print out accuracy and f1 score
     clf.fit(X_train, Y_train_forest)
     accuracy = clf.score(X_test, Y_test)
     f1 = f1_score(clf.predict(X_test), Y_test, average='weighted')
     print("accuracy: %2.6f" % (accuracy))
     print("f1 score: %2.6f" % (f1))
     print("₩n")

     accuracy: 0.357774
     f1 score: 0.375476
```

이 과정의 의의는 두 가지가 있습니다. 우선, 기존에는 0.1도 넘지 못했던 f1-score를 accuracy와 비슷한 수준으로 증가시켰습니다. 또한, feature selection 과정이 효과적임을 알 수 있었습니다. 이전에 모델 구현 과정에서 feature selection 없이 모든 52개 feature들을 사용해서 random forest를 사용했을 때, 약 0.36의 accuracy를 얻었습니다. 전체 feature 개수 중 ¼ 이하의 feature들만 사용했음에도 불구하고, accuracy loss가 매우 적은 것을 확인할 수 있습니다.

그러나, 여전히 두 지표 모두 이상적이라고 보기에는 너무 낮은 수치입니다. 모델의 낮은 성능에 대해서는 두 가지의 가능한 원인을 생각 해 보았습니다. 첫 번째 가능성은 학습에 사용된 feature들의 한계입니다. 저희는 지금까지 사용된 8개의 feature만으로는 교통사고 피해 강도를 예측하는 것이 어려울 수도 있겠다는 의문이 들었습니다. 따라서, dataset의 다른 feature들을 추가하여 모델의 성능을 증가시키는 방향을 잡았습니다.

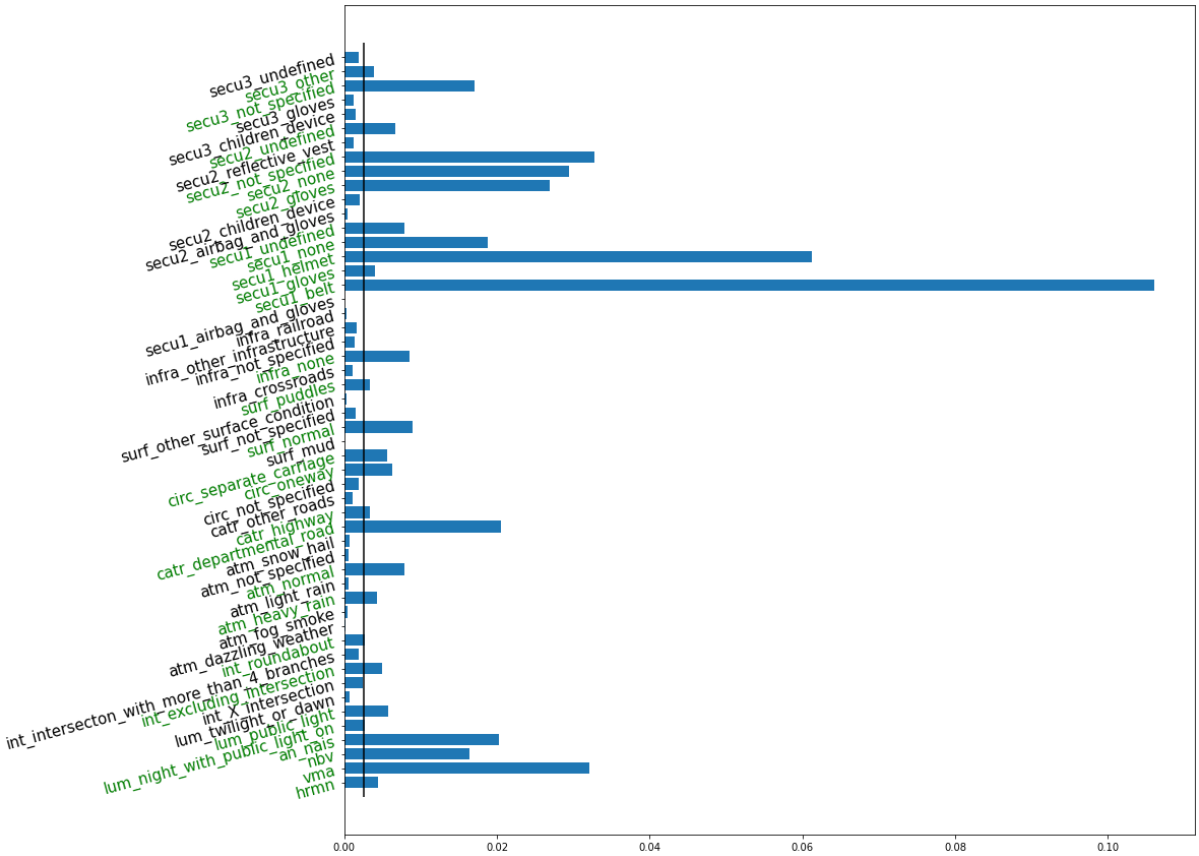
두 번째는 모델 자체의 문제입니다. Random forest를 바탕으로 한 모델을 구현하면서 특별히 의아했던 점이나 문제 삼을 부분은 찾지 못했습니다. 그러나, 다른 모델을 사용했을 때 random forest보다 더 좋은 성능을 보여주는 것도 충분히 가능하다고 느꼈습니다. 따라서, 다른 방법론인 K-fold cross validation과 logistic regression을 사용했을 때, random forest와 비교해서 어떤 결과가 나오는지 확인했습니다.

➢ *구현 과정 - Feature 추가*

추가적으로 학습에 사용된 feature들은 다음과 같습니다.

사고 차량 속도('vma')	안전장치 착용여부('secu')
도로 위 차선 수('nbv')	운전자의 출생년도('an_nais')

'nbv' feature를 제외하면, 이전에 사용된 8개 feature에 비해서는 교통사고 발생 장소보다는 운전자 또는 탑승자, 즉 사람에 초점이 더 맞춰진 feature들입니다. 교통 환경을 기준으로 한 프로젝트의 원래 방향성과는 거리가 있을 수 있지만, feature의 개수에 따라 모델의 성능이 어떻게 변화하는지 확인하는 것이 중요하다고 여겼습니다. 따라서, 교통사고 피해 정도와 가장 밀접한 연관이 있을 것으로 여겨지는 feature들을 선정하여 추가했습니다. 정확한 비교를 위해 모델 구현은 이전과 동일한 random forest으로 진행했습니다.



위 이미지의 경우, threshold 값은 전과 동일하게 0.025를 사용했습니다. 총 84개의 feature 중 28개가 선택되었습니다. 특이점으로는 새로 추가된 'vma', 'nbv', 'an_nais', 'secu' feature들이 대거 포함된 것을 확인할 수 있었습니다.

Random Forest

```
[ ]  #unraveling to avoid warning message
     Y_train_forest = Y_train.values.ravel()

     #create random forest with 101 trees
     clf = RandomForestClassifier(n_estimators = 101, random_state = randomness)

     #print out accuracy and f1 score
     clf.fit(X_train, Y_train_forest)
     accuracy = clf.score(X_test, Y_test)
     f1 = f1_score(clf.predict(X_test), Y_test, average='weighted')
     print("accuracy: %2.6f" % (accuracy))
     print("f1 score: %2.6f" % (f1))
     print("₩n")
```

```
accuracy: 0.689809
f1 score: 0.692542
```

Feature의 개수를 늘린 것 외에는 모델에 전혀 변화를 주지 않았음에도 불구하고, 모델의 성능을 나타내는 지표가 두 배 가까이 증가한 것을 볼 수 있습니다. 추가한 feature들이 학습에 모두 사용된 만큼, 이 feature들이 지표에 상당한 영향을 끼쳤다는 결론을 내릴 수 있습니다. 반면, 처음부터 사용된 교통 환경 관련의 8개 feature들 중에서도 학습에 사용된 feature들이 존재하지만, 교통사고 피해 강도 feature와의 dependency가 상대적으로 확연히 낮다는 것을 확인했습니다. 교통 환경 관련 정보만으로 교통사고 피해를 높은 정확도로 예측하는 것이 어렵다는 것을 이 프로젝트를 통해 느낄 수 있었습니다.

➢ *구현 과정 - 다른 모델들*

프로젝트의 초창기에 저희가 학습 모델을 decision tree 및 random forest 기반으로 결정한 이유는, 데이터 중 categorical data가 대다수였기 때문에 다른 방법론을 사용하기 어려웠기 때문입니다. 그러나 one-hot encoding을 성공적으로 적용한 후에는 numerical data를 사용하는 방법론도 고려할 수 있었습니다. 이에 따라, 같은 feature들을 사용한 K-fold cross validation, 그리고 multi-output logistic regression 모델이 random forest 기반의 모델과 성능 차이가 존재하는지 확인하는 과정을 거쳤습니다.

Stratified K Fold Cross Validation WITH Upsampling

```
[ ]  #choose model as a decision tree and train it
     decision_tree = tree.DecisionTreeClassifier()
     decision_tree.fit(X_train, Y_train)

     #stratified K-fold
     cv = StratifiedKFold(n_splits = 5, shuffle=True, random_state = randomness)


     print((cross_val_score(decision_tree, X_test, Y_test, cv = cv, scoring='accuracy')).mean())
```

```
0.5746615654967677
```

Cross validation 모델은 k=5를 사용했습니다. k가 hyperparameter인 만큼 임의로 5라는 값을 선택했지만, 확인삼아 k를 5에서 15까지 1씩 증가시키며 코드를 순차적으로 실행해 보았습니다. 그러나 k가 증가해도 모델의 성능이 크게 좋아지지 않는 것을 보고 코드에 포함하지는 않았습니다.

위 코드를 보면 첫 모델은 stratified k-fold cross validation인 것을 확인할 수 있습니다. 기본적으로 데이터가 편향성을 보였기 때문에, stratified한 형태를 사용하는 것이 바람직하다고 생각했습니다. 또한, 모델에 사용된 dataset으로는 이전 random forest때 사용했던 train/test dataset을 그대로 사용했습니다. 즉, upsampling과 feature selection이 이미 진행된 상태로 학습을 진행한 것입니다. 이 경우, stratified k-fold와 upsampling의 목적성이 겹치기 때문에, cross validation을 upsampling 없이 진행해도 모델의 성능이 유지될 지 확인해보고자 했습니다.

Cross Validation no Oversampling

```
     #choose model as a decision tree and train it
     decision_tree = tree.DecisionTreeClassifier()
     decision_tree.fit(X_train, Y_train)

     #stratified K-fold
     cv = StratifiedKFold(n_splits = 5, shuffle=True, random_state = randomness)


     print((cross_val_score(decision_tree, X_test, Y_test, cv = cv, scoring='accuracy')).mean())
```

```
0.47490689101596384
```

Upsampling을 제외한 두 번째 모델은 첫 번째 모델에 비해 성능이 많이 낮아진 것을 볼 수 있습니다. Stratified k-fold cross validation만으로는 데이터의 편향성을 온전히 보완할 수 없다는 증거로 판단했습니다.

Multi-Output Logistic Regression With Only Numeric

```
[ ]  MAX_ITER = X_train.shape[0] + 1

     #get only the numeric values
     X_numeric_train = X_train[['an_nais', 'hrmn', 'vma', 'nbv']]
     X_numeric_test = X_test[['an_nais', 'hrmn', 'vma', 'nbv']]

     #Logistic Regression
     clf = MultiOutputClassifier(LogisticRegression(max_iter=MAX_ITER)).fit(X_numeric_train, Y_train)
     print("accuracy: %2.6f" % (clf.score(X_numeric_test, Y_test)))
     print("f1 score: %2.6f" % (f1_score(clf.predict(X_numeric_test), Y_test, average='weighted')))

     accuracy: 0.451981
     f1 score: 0.497085
```

Logistic regression 모델의 경우, 모델이 예측하고자 하는 feature인 'grav'이 4개의 가능한 output이 존재하기 때문에 multi-output 형태로 구현했습니다. 기본적으로 logistic regression은 numerical data를 사용하기 때문에, 두 가지 방법을 사용하기로 했습니다. 첫 번째 방법은 기존에 사용된 feature들 중 numerical한 feature들만 따로 logistic regression에 사용하고, 그 결과가 좋을 경우 categorical data의 random forest 모델과 합치는 방안입니다. 만약 위 코드를 실행했을 때, 지표가 random forest 모델보다 좋을 경우 구현을 이어나갈 수 있었지만, 성능이 더 낮았기 때문에 다른 방법을 모색했습니다.

Multi-Output Logistic Regression

```
     MAX_ITER = X_train.shape[0] + 1

     #Logistic Regression
     clf = MultiOutputClassifier(LogisticRegression(max_iter=MAX_ITER)).fit(X_train, Y_train)
     print("accuracy: %2.6f" % (clf.score(X_test, Y_test)))
     print("f1 score: %2.6f" % (f1_score(clf.predict(X_test), Y_test, average='weighted')))

     accuracy: 0.582981
     f1 score: 0.608852
```

두 번째 방법은 모든 feature들을 모두 학습에 사용하는 것입니다. 이미 프로젝트 초반에 one-hot encoding을 실시했기 때문에, categorical data라도 logistic regression에 사용할 수 있었습니다. 학습 결과는 이전 모델보다는 좋은 성능을 보였지만, 마찬가지로 random forest 모델보다는 낮은 지표가 나왔습니다. Upsampling을 제외했을 때 어떤 변화가 생기는지도 확인해 보았지만, 성능이 더 낮아지는 것만 확인할 수 있었습니다.

➢ *결과와 해석*

프로젝트의 원래 목표였던, 교통 환경에 대한 정보만 학습하여 교통사고 피해 강도를 예측하는 모델은 좋지 않은 성능을 보였습니다. 이에 대한 대책으로 교통사고와 밀접한 관계가 있는 다른 feature들을 추가함으로써 모델의 성능을 향상시킬 수 있었습니다. '차량 속도가 높을수록 교통사고 피해가 크다', '탑승자의 안전장치 착용 여부에 따라 교통사고 피해를 감소시킬 수 있다' 등의 명제들은 많은 사람들에게 직관적일 수 있지만, 이를 데이터로 분석하여 증명했다는 점이 유의미하다고 느꼈습니다. 특히, 안전벨트 착용이 교통사고 피해 강도와 가장 강한 연관성을 보인다는 것은 안전벨트의 중요성에 대한 경각심을 주기 아주 적합한 내용이라고 보였습니다.

	Num_Acc	hrmn	lum	int	atm	catr	circ	surf	infra	grav
0	201900000001	1.500000	night_with_public_light_not_on	excluding_intersection	normal	highway	separate_carriage	normal	bridge_flyover	very
1	201900000001	1.500000	night_with_public_light_not_on	excluding_intersection	normal	highway	separate_carriage	normal	bridge_flyover	noharm

그러나 위 모델도 accuracy가 0.7 이상이 나오게끔 성능을 높이는 것은 어려워 보였습니다. 저희는 이 문제가 교통사고 관련 데이터의 구조에 존재한다고 느꼈습니다. 프로젝트에 사용된 dataframe을 보면, 다른 feature들이 모두 동일함에도 불구하고 grav 값이 다른 것을 확인할 수 있습니다. 여러 인원이 같은 차량 내에서 같은 교통사고를 당할 경우, 인원마다 받는 피해가 다른 경우가 많기 때문입니다.

그럼에도 학습 모델의 결과를 볼 때, 해석의 여지가 있는 내용이 많았습니다. 학습 과정에서 사용된 교통 환경 feature들 중에서 피해 강도와 강한 연관성을 보이는 것들이 존재했습니다. 이 feature들 중 일부를 나열하여, 학습 결과를 해석해보고자 합니다.

- catr: departmental road (도로 종류: 지방 고속도로)

이 feature는 교통사고가 발생한 도로가 지방에 위치한 고속도로임을 뜻합니다. 이 feature는 grav=4, 즉 교통사고 피해자가 중상을 입은 경우와 강하게 연관되었습니다. 기본적으로 고속도로인 만큼, 높은 차량 속도가 더 큰 교통사고 피해로 이어졌을 것입니다. 사고 차량의 속도를 나타내는 feature인 'vma'도 강한 연관성을 보인다는 것이 이 가설에 힘을 실었습니다. 또한, 도심에 비해 지방에 있는 고속도로는 도로의 상태나 조명 등 시설들이 상대적으로 관리가 덜 되어있을 것이라는 점도 교통사고 피해 강도와의 연관성에 영향을 줬다는 결론을 내렸습니다.

- nbv (도로 위 차선 수)

이 feature는 random forest를 사용한 두 번째 모델에서 추가한 feature이지만, 교통 환경과도 연관성이 강하다고 판단했습니다. 안전장치 관련 feature인 'secu'를 제외하면 교통사고 피해 강도와 두 번째로 높은 연관성을 보였고, 기본적으로 'nbv'가 높은 값일 수록 더 큰 교통사고 피해로 이어진다는 패턴이 있었습니다. 도로에 차선 수가 많으면 그만큼 평소에 차량의 수도 높을 것이며, 교통사고가 발생했을 경우 2차, 3차 피해도 그만큼 많아질 확률이 높기 때문에 이 관계는 이해하기 쉽다고 느껴졌습니다. 이 결과를 현실과 접목해 보자면, 교통 관련 기관에서 교통 안전을 위한 시스템을 새로 도입할 때 차선 수가 많은 도로를 우선적으로 관리하는 방법을 고려할 수 있겠습니다.

- surf: normal (도로 표면 상태: 일반)

'surf' feature의 경우, 대부분의 데이터가 'normal'이나 'wet' 중 하나로 기록되어 있었습니다. 직관적으로 생각했을 때, 평균적으로 일반적인 도로보다 물기가 있는 젖은 도로가 더 큰 교통사고 피해로 이어질 것이라고 생각할 수 있습니다. 실제로 데이터도 이런 경향을 조금은 보여줬습니다. 그러나 다른 feature들에 비해 연관성이 강하지는 않은 만큼, 많은 데이터가 'normal'로 편향되어 있었기 때문에 학습 과정에서 연관성이 강하게 나타난 가능성도 있을 것이라고 생각했습니다.

➢ *향후 가능한 추가 연구 개발 방향*

저희가 처음에 설정했던 방향은 교통사고 피해 강도 예측에 있어서 교통 환경을 고려하는 것에 초점을 맞춘 점이었습니다. 그러나 프로젝트를 진행하며 시행착오를 겪으면서, 최초에 설정했던 교통 환경에 관련한 feature만으로는 accuracy와 f1-score의 기준에서 어느 한쪽의 방향으로도 만족할 만한 성능을 보이지 못했습니다. 이로 인해 처음에 잡았던 방향에 변화를 주어 예측에 있어서 더 유의미한 성과를 보일 수 있도록 기존 교통 환경에 국한되어 있던 feature 뿐만 아니라 운전자 및 탑승자와 관련한 인적 속성과 관련한 feature를 추가하여 좀 더 일반화시킬 수 있는 방향으로 모델을 설계하게 되었습니다.

따라서 저희가 접근한 방식을 토대로, 이를 추가적으로 더 발전시킨다면 생각해 볼 수 있는 점은 추가적인 feature를 고려하는 방향으로 볼 수 있습니다. 현재 주어진 데이터셋으로는 가능하지 않지만, 데이터가 주어진다면, 마치 자연어 처리 분야에 있어서 center word에 대해 문맥을 반영하기 위해 앞뒤 주변 단어를 고려하듯이, 사고 차량 당시 주변에 존재했던 차량에 관련한 정보를 같은 맥락으로 반영해 보는 접근법도 유의미할 것으로 생각됩니다. 현재 저희 프로젝트에서 반영한 feature set의 특징을 살펴본다면, 사고 발생 당시의 교통 환경이나 운전자 및 동승자 정보에 대해서 사고에 직접적으로 연루된 피해차량에 제한되어 있으며, 다른 무엇보다 "사고 발생" 시점에 국한되어 있는 점이 한계라고 꼽아볼 수 있습니다. 그렇기 때문에 좀 더 실제 상황에 관련하여 통찰력을 더해 줄 수 있는 주변부와 관련한 데이터가 필요할 것이라는 믿음에 기반하여, 간접적으로 연루된 차량에 대한 정보 뿐만 아니라 주목할 시점에 대해서도 주변부를 포괄하여 범위를 넓히도록 접근해야 한다고 생각합니다. 일례로, 운전자 및 동승자의 인적 속성에 관련해서도 사고 당시의 속성 뿐만 아니라 평소 운전과 관련한 이력 등을 반영하여 이에 관련한 feature를 추가하는 방향도 고려해 볼 수 있을 것 같습니다.

위와 같은 방향성으로 추가적인 데이터를 수집하여 반영한다면, 기존의 방식보다 좀 더 자연스럽고 개연성이 있는 사고 예측이 가능할 것이라 생각합니다. 따라서 단순한 예측에 머무는 것이 아니라, 사고 발생이라는 센터 포인트에 벗어나서 사고를 발생시킬만한 가능성이 있는 주변부 요소들에 주목할 수 있어 그 결과를 현실에 접목시킨다면 운전자 및 동승자들의 행동 패턴 개선을 유도하는 근거로 활용할 수 있을 거라 생각합니다.