Appendix_D_NB_and_RF_modeling

August 10, 2024

1 Na ve Bayes and Random Forest Modeling

This notebook contains the code and details for Naïve Bayes and Random Forest modeling.

[ ]:
```
import warnings
warnings.filterwarnings('ignore')
```

[ ]:
```
# Import libraries
import pandas as pd
import numpy as np

from pathlib import Path
from imblearn.over_sampling import RandomOverSampler, SMOTE
from imblearn.under_sampling import RandomUnderSampler
from sklearn.dummy import DummyClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score,␣
 ↪confusion_matrix
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.naive_bayes import CategoricalNB
```

[ ]:
```
%run ../custom/jc-functions.ipynb
```

1.1 Prepare Dataset

[ ]:
```
# Import training dataset
dataset = Path('../dataset')
df = pd.read_csv(dataset/'accidents_clean_train.csv')
df.head()
```

[ ]:
```
  Area_accident_occured Types_of_Junction       Light_conditions  \
0     Residential areas       No junction              Daylight
1          Office areas       No junction              Daylight
2    Recreational areas       No junction              Daylight
3          Office areas           Y Shape  Darkness - lights lit
4      Industrial areas           Y Shape  Darkness - lights lit

   Number_of_vehicles_involved  Number_of_casualties  \
```

```
0                            2                    2
1                            2                    2
2                            2                    2
3                            2                    2
4                            2                    2

             Cause_of_accident Day_of_week Sex_of_driver Age_band_of_driver  \
0              Moving Backward      Monday          Male               18-30
1                   Overtaking      Monday          Male               31-50
2   Changing lane to the left      Monday          Male               18-30
3  Changing lane to the right      Sunday          Male               18-30
4                   Overtaking      Sunday          Male               18-30

  Accident_severity
0     Slight Injury
1     Slight Injury
2    Serious Injury
3     Slight Injury
4     Slight Injury
```

[ ]:
```
df.shape
```

[ ]:
```
(8210, 10)
```

[ ]:
```
# Get feature columns
columns = df.columns.tolist()
print(columns)

features = ['Area_accident_occured', 'Types_of_Junction', 'Light_conditions',␣
 ↪'Number_of_vehicles_involved', 'Number_of_casualties', 'Cause_of_accident',␣
 ↪'Day_of_week', 'Sex_of_driver', 'Age_band_of_driver']

target = 'Accident_severity'
```

```
['Area_accident_occured', 'Types_of_Junction', 'Light_conditions',
'Number_of_vehicles_involved', 'Number_of_casualties', 'Cause_of_accident',
'Day_of_week', 'Sex_of_driver', 'Age_band_of_driver', 'Accident_severity']
```

1.1.1 Training dataset

[ ]:
```
# Convert to categorical
X = df[features]
X = pd.get_dummies(X, drop_first=True)
X.head()
```

[ ]:
```
   Number_of_vehicles_involved  Number_of_casualties  \
0                            2                    2
1                            2                    2
```

```
2                           2                    2
3                           2                    2
4                           2                    2

   Area_accident_occured_  Recreational areas  \
0                                       False
1                                       False
2                                        True
3                                       False
4                                       False

   Area_accident_occured_ Church areas  Area_accident_occured_ Hospital areas  \
0                                False                                   False
1                                False                                   False
2                                False                                   False
3                                False                                   False
4                                False                                   False

   Area_accident_occured_ Industrial areas  \
0                                    False
1                                    False
2                                    False
3                                    False
4                                     True

   Area_accident_occured_ Outside rural areas  \
0                                       False
1                                       False
2                                       False
3                                       False
4                                       False

   Area_accident_occured_Office areas  Area_accident_occured_Other  \
0                               False                         False
1                                True                         False
2                               False                         False
3                                True                         False
4                               False                         False

   Area_accident_occured_Recreational areas  …  Day_of_week_Sunday  \
0                                    False  …               False
1                                    False  …               False
2                                    False  …               False
3                                    False  …                True
4                                    False  …                True

   Day_of_week_Thursday  Day_of_week_Tuesday  Day_of_week_Wednesday  \
```

```
0                  False                False                    False
1                  False                False                    False
2                  False                False                    False
3                  False                False                    False
4                  False                False                    False

   Sex_of_driver_Male  Sex_of_driver_Unknown  Age_band_of_driver_31-50  \
0                True                  False                     False
1                True                  False                      True
2                True                  False                     False
3                True                  False                     False
4                True                  False                     False

   Age_band_of_driver_Over 51  Age_band_of_driver_Under 18  \
0                       False                        False
1                       False                        False
2                       False                        False
3                       False                        False
4                       False                        False

   Age_band_of_driver_Unknown
0                       False
1                       False
2                       False
3                       False
4                       False

[5 rows x 56 columns]
```

[ ]:
```python
# Create mapped columns of target
df['Accident_slight'] = df[target].map(
    {'Slight Injury': 0}
    ).fillna(1).astype(int)
df['Accident_serious'] = df[target].map(
    {'Serious Injury': 0}
    ).fillna(1).astype(int)
df['Accident_severity_mapped'] = df[target].map({
    'Serious Injury': 0,
    'Slight Injury': 1,
    'Fatal injury': 2
    })

df.head()
```

[ ]:
```
  Area_accident_occured Types_of_Junction       Light_conditions  \
0     Residential areas       No junction               Daylight
1          Office areas       No junction               Daylight
```

```
2    Recreational areas       No junction              Daylight
3          Office areas           Y Shape  Darkness - lights lit
4      Industrial areas           Y Shape  Darkness - lights lit

   Number_of_vehicles_involved  Number_of_casualties  \
0                            2                     2
1                            2                     2
2                            2                     2
3                            2                     2
4                            2                     2

            Cause_of_accident Day_of_week Sex_of_driver Age_band_of_driver  \
0             Moving Backward      Monday          Male               18-30
1                   Overtaking      Monday          Male               31-50
2   Changing lane to the left      Monday          Male               18-30
3  Changing lane to the right      Sunday          Male               18-30
4                   Overtaking      Sunday          Male               18-30

  Accident_severity  Accident_slight  Accident_serious  \
0     Slight Injury                0                 1
1     Slight Injury                0                 1
2    Serious Injury                1                 0
3     Slight Injury                0                 1
4     Slight Injury                0                 1

   Accident_severity_mapped
0                         1
1                         1
2                         0
3                         1
4                         1
```

[ ]:
```
df.shape
```

[ ]:
```
(8210, 13)
```

[ ]:
```
y = df[target]
y_mapped = df['Accident_severity_mapped']
y_slight = df['Accident_slight']
y_serious = df['Accident_serious']
```

1.2 Modeling using Na ve Bayes

1.2.1 Accident Severity: Slight Injury (1) vs. Serious Injury (0) vs. Fatal Injury (2)

[ ]:
```
def nb_report(test, pred):
    print("Accuracy: ", accuracy_score(test, pred))
    print("Confusion Matrix:\n", confusion_matrix(test, pred))
    print("Classification Report:\n", classification_report(test, pred))
```

[ ]:
```
def cross_scores(score):
    print("Cross-validation scores: ", score)
    print("Average score: ", score.mean())
```

[ ]:
```
# Split testing data
X_train, X_test, y_train, y_test = train_test_split(X, y_mapped,
                                                    test_size=0.3,
                                                    random_state=42)
# Train model
nb_model = CategoricalNB()
nb_model.fit(X_train, y_train)
# Create predictions
y_pred = nb_model.predict(X_test)

nb_report(y_test, y_pred)

score = cross_val_score(nb_model, X_test, y_test, cv=5)
cross_scores(score)
```

```
Accuracy:  0.857896873731222
Confusion Matrix:
 [[   2  327    1]
 [   0 2110    1]
 [   0   21    1]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.01      0.01       330
           1       0.86      1.00      0.92      2111
           2       0.33      0.05      0.08        22

    accuracy                           0.86      2463
   macro avg       0.73      0.35      0.34      2463
weighted avg       0.87      0.86      0.79      2463

Cross-validation scores:  [0.85801217 0.85801217 0.85395538 0.8597561
0.85772358]
Average score:  0.8574918781642177
```

1.2.2 Accident Severity: Slight Injury (0) vs Serious Injury/Fatal Injury (1)

[ ]:
```
# Split testing data
X_train, X_test, y_train, y_test = train_test_split(X, y_slight,
                                                    test_size=0.3,
                                                    random_state=42)
# Train model
nb_model_slight = CategoricalNB()
nb_model_slight.fit(X_train, y_train)

# Create prediction
y_pred_slight = nb_model_slight.predict(X_test)
nb_report(y_test, y_pred_slight)

score_slight = cross_val_score(nb_model_slight, X_test, y_test, cv=10)
cross_scores(score_slight)
```

```
Accuracy:  0.8591149005278116
Confusion Matrix:
 [[2111    0]
 [ 347    5]]
Classification Report:
              precision    recall  f1-score   support

           0       0.86      1.00      0.92      2111
           1       1.00      0.01      0.03       352

    accuracy                           0.86      2463
   macro avg       0.93      0.51      0.48      2463
weighted avg       0.88      0.86      0.80      2463

Cross-validation scores:  [0.85425101 0.86234818 0.8582996  0.85365854
0.86178862 0.85365854
 0.85772358 0.86178862 0.85772358 0.85772358]
Average score:  0.8578963826075506
```

1.2.3 Accident Severity: Slight/Fatal Injury (1) vs Serious Injury (0)

[ ]:
```
# Split testing data
X_train, X_test, y_train, y_test = train_test_split(X, y_serious,
                                                    test_size=0.3,
                                                    random_state=42)
# Train model
nb_model_serious = CategoricalNB()
nb_model_serious.fit(X_train, y_train)

# Create prediction
y_pred_serious = nb_model_serious.predict(X_test)
```

```
nb_report(y_test, y_pred_serious)

score_serious = cross_val_score(nb_model_serious, X_test, y_test, cv=5)
cross_scores(score_serious)
```

```
Accuracy:  0.8672350791717418
Confusion Matrix:
 [[   3  327]
 [   0 2133]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.01      0.02       330
           1       0.87      1.00      0.93      2133

    accuracy                           0.87      2463
   macro avg       0.93      0.50      0.47      2463
weighted avg       0.88      0.87      0.81      2463

Cross-validation scores:  [0.86612576 0.86815416 0.86409736 0.86788618
0.86585366]
Average score:  0.8664234238691272
```

1.3 Resample Target Data

1.3.1 Accident Severity: Slight Injury (1) vs. Serious Injury (0) vs. Fatal Injury (2)

[ ]:
```
oversample = RandomOverSampler(random_state=42)

X_resampled, y_resampled = oversample.fit_resample(X, y_mapped)
# Check distribution
print(y_resampled.value_counts())

# Split testing data
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
                                                    test_size=0.3,
                                                    random_state=42)
# Train model
nb_model_serious = CategoricalNB()
nb_model_serious.fit(X_train, y_train)

# Create prediction
y_pred_serious = nb_model_serious.predict(X_test)
nb_report(y_test, y_pred_serious)

score_serious = cross_val_score(nb_model_serious, X_test, y_test, cv=5)
cross_scores(score_serious)
```

```
Accident_severity_mapped
1    7082
0    7082
2    7082
Name: count, dtype: int64
Accuracy:  0.5258864135550675
Confusion Matrix:
 [[ 831  865  471]
 [ 578 1118  420]
 [ 352  336 1403]]
Classification Report:
              precision    recall  f1-score   support

           0       0.47      0.38      0.42      2167
           1       0.48      0.53      0.50      2116
           2       0.61      0.67      0.64      2091

    accuracy                           0.53      6374
   macro avg       0.52      0.53      0.52      6374
weighted avg       0.52      0.53      0.52      6374

Cross-validation scores:  [       nan 0.54117647 0.51921569        nan
0.50784929]
Average score:  nan
```

1.3.2 Accident Severity: Slight Injury (0) vs Serious Injury/Fatal Injury (1)

[ ]:
```
X_resampled, y_resampled = oversample.fit_resample(X, y_slight)
# Check distribution
print(y_resampled.value_counts())

# Split testing data
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
                                                    test_size=0.3,
                                                    random_state=42)
# Train model
nb_model_serious = CategoricalNB()
nb_model_serious.fit(X_train, y_train)

# Create prediction
y_pred_serious = nb_model_serious.predict(X_test)
nb_report(y_test, y_pred_serious)

score_serious = cross_val_score(nb_model_serious, X_test, y_test, cv=5)
cross_scores(score_serious)
```

```
Accident_slight
0    7082
```

```
1    7082
Name: count, dtype: int64
Accuracy:  0.576
Confusion Matrix:
 [[1306  825]
 [ 977 1142]]
Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.61      0.59      2131
           1       0.58      0.54      0.56      2119

    accuracy                           0.58      4250
   macro avg       0.58      0.58      0.58      4250
weighted avg       0.58      0.58      0.58      4250

Cross-validation scores:  [0.59294118        nan        nan 0.59764706
0.55176471]
Average score:  nan
```

1.3.3 Accident Severity: Slight/Fatal Injury (1) vs Serious Injury (0)

[ ]:
```
X_resampled, y_resampled = oversample.fit_resample(X, y_serious)
# Check distribution
print(y_resampled.value_counts())

# Split testing data
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
                                                    test_size=0.3,
                                                    random_state=42)
# Train model
nb_model_serious = CategoricalNB()
nb_model_serious.fit(X_train, y_train)

# Create prediction
y_pred_serious = nb_model_serious.predict(X_test)
nb_report(y_test, y_pred_serious)

score_serious = cross_val_score(nb_model_serious, X_test, y_test, cv=5)
cross_scores(score_serious)
```

```
Accident_serious
1    7164
0    7164
Name: count, dtype: int64
Accuracy:  0.5796696906257269
Confusion Matrix:
 [[1092 1073]
```

```
 [ 734 1400]]
Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.50      0.55      2165
           1       0.57      0.66      0.61      2134

    accuracy                           0.58      4299
   macro avg       0.58      0.58      0.58      4299
weighted avg       0.58      0.58      0.58      4299

Cross-validation scores:  [       nan 0.59767442        nan 0.55930233
0.56111758]
Average score:  nan
```

1.3.4 Analysis of Results

Target: Slight Injury (1) vs Serious Injury (0) vs. Fatal Injury (2) Accuracy: 86% Precision Serious Injury: 100% Recall: 1%

Resampled Accuracy: 53% Precision Serious Injury: 47% Recall: 38%

Target: Slight Injury (0) vs Serious Injury/Fatal Injury (1) Accuracy: 86% Precision: 100% Recall: 1%

Resampled Accuracy: 58% Precision Serious Injury: 58% Recall: 61%

Target: Slight/Fatal Injury (1) vs Serious Injury (0) Accuracy: 87% Precision: 100% Recall: 1%

Resampled Accuracy: 58% Precision Serious Injury: 60% Recall: 50%

A successful model would have 85% accuracy and precision of 90%. Unfortunately, these models do not fit the criteria.

1.4 Modeling Using Random Forest

1.4.1 Accident Severity: Slight Injury (1) vs. Serious Injury (0) vs. Fatal Injury (2)

[ ]:
```
# Split model
X_train, X_test, y_train, y_test = train_test_split(X, y_mapped, test_size=0.3,
                                                    random_state=42)

# Initialize model
rf_model = RandomForestClassifier(n_estimators=100, random_state=84)

# Train
rf_model.fit(X_train, y_train)
```

```
# Predict
y_pred = rf_model.predict(X_test)
nb_report(y_test, y_pred)
```

```
Accuracy:  0.8501827040194885
Confusion Matrix:
 [[  32  297    1]
 [  46 2062    3]
 [   1   21    0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.41      0.10      0.16       330
           1       0.87      0.98      0.92      2111
           2       0.00      0.00      0.00        22

    accuracy                           0.85      2463
   macro avg       0.42      0.36      0.36      2463
weighted avg       0.80      0.85      0.81      2463
```

[ ]:
```
# Split model / Unmapped
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=42)

# Initialize model
rf_model = RandomForestClassifier(n_estimators=100, random_state=84)

# Train
rf_model.fit(X_train, y_train)
# Predict
y_pred = rf_model.predict(X_test)
nb_report(y_test, y_pred)
```

```
Accuracy:  0.8505887129516849
Confusion Matrix:
 [[   1    1   20]
 [   1   32  297]
 [   3   46 2062]]
Classification Report:
                precision    recall  f1-score   support

  Fatal injury       0.20      0.05      0.07        22
Serious Injury       0.41      0.10      0.16       330
 Slight Injury       0.87      0.98      0.92      2111

      accuracy                           0.85      2463
     macro avg       0.49      0.37      0.38      2463
```

```
weighted avg       0.80      0.85      0.81      2463
```

1.4.2 Accident Severity: Slight Injury (0) vs Serious Injury/Fatal Injury (1)

[ ]:
```
# Split model
X_train, X_test, y_train, y_test = train_test_split(X, y_slight, test_size=0.3,
                                                    random_state=42)

# Initialize model
rf_model_slight = RandomForestClassifier(n_estimators=100, random_state=84)

# Train
rf_model_slight.fit(X_train, y_train)
# Predict
y_pred = rf_model_slight.predict(X_test)
nb_report(y_test, y_pred)
```

```
Accuracy:  0.8518067397482745
Confusion Matrix:
 [[2058   53]
 [ 312   40]]
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.97      0.92      2111
           1       0.43      0.11      0.18       352

    accuracy                           0.85      2463
   macro avg       0.65      0.54      0.55      2463
weighted avg       0.81      0.85      0.81      2463
```

1.4.3 Accident Severity: Slight/Fatal Injury (1) vs Serious Injury (0)

[ ]:
```
# Split model
X_train, X_test, y_train, y_test = train_test_split(X, y_serious, test_size=0.3,
                                                    random_state=42)

# Initialize model
rf_model_serious = RandomForestClassifier(n_estimators=100, random_state=84)

# Train
rf_model_serious.fit(X_train, y_train)
# Predict
y_pred = rf_model_serious.predict(X_test)
nb_report(y_test, y_pred)
```

```
Accuracy:  0.8587088915956151
Confusion Matrix:
 [[  28  302]
 [  46 2087]]
Classification Report:
              precision    recall  f1-score   support

           0       0.38      0.08      0.14       330
           1       0.87      0.98      0.92      2133

    accuracy                           0.86      2463
   macro avg       0.63      0.53      0.53      2463
weighted avg       0.81      0.86      0.82      2463
```

1.5 Resample Target Data

1.5.1 Accident Severity: Slight Injury (1) vs. Serious Injury (0) vs. Fatal Injury (2)

[ ]:
```
oversample = RandomOverSampler(random_state=42)

X_resampled, y_resampled = oversample.fit_resample(X, y_mapped)
# Check distribution
print(y_resampled.value_counts())

# Split testing data
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
                                                    test_size=0.3,
                                                    random_state=42)
# Train model
rf_model_rs = RandomForestClassifier()
rf_model_rs.fit(X_train, y_train)

# Create prediction
y_pred_rs = rf_model_rs.predict(X_test)
nb_report(y_test, y_pred_rs)

score_serious = cross_val_score(rf_model_rs, X_test, y_test, cv=5)
cross_scores(score_serious)
```

```
Accident_severity_mapped
1    7082
0    7082
2    7082
Name: count, dtype: int64
Accuracy:  0.9590524003765296
Confusion Matrix:
 [[2120   40    7]
```

```
 [ 206 1902    8]
 [   0    0 2091]]
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.98      0.94      2167
           1       0.98      0.90      0.94      2116
           2       0.99      1.00      1.00      2091

    accuracy                           0.96      6374
   macro avg       0.96      0.96      0.96      6374
weighted avg       0.96      0.96      0.96      6374

Cross-validation scores:  [0.88941176 0.86588235 0.88       0.86509804
0.88304553]
Average score:  0.8766875365530827
```

[ ]:
```
# Confusion matrix metrics
cm = confusion_matrix(y_test, y_pred_rs)
multiclass_cm_metrics(cm)
```

```
Confusion Matrix:
[[2120   40    7]
 [ 206 1902    8]
 [   0    0 2091]]
```

[ ]:

	Class 0	Class 1	Class 2
Accuracy	0.96031	0.96015	0.99765
Error rate	0.03969	0.03985	0.00235
Sensitivity (Recall)	0.97831	0.89887	1.00000
Specificity	0.95103	0.99061	0.99650
Precision	0.91144	0.97940	0.99288
F1	0.94369	0.93741	0.99643
F2	0.96416	0.91390	0.99857
F0.5	0.92407	0.96216	0.99429

1.5.2 Accident Severity: Slight Injury (0) vs Serious Injury/Fatal Injury (1)

[ ]:
```
X_resampled, y_resampled = oversample.fit_resample(X, y_slight)
# Check distribution
print(y_resampled.value_counts())

# Split testing data
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
                                                    test_size=0.3,
                                                    random_state=42)
# Train model
rf_model_rs = RandomForestClassifier()
```

```
rf_model_rs.fit(X_train, y_train)

# Create prediction
y_pred_rs = rf_model_rs.predict(X_test)
nb_report(y_test, y_pred_rs)

score_serious = cross_val_score(rf_model_rs, X_test, y_test, cv=5)
cross_scores(score_serious)
```

```
Accident_slight
0    7082
1    7082
Name: count, dtype: int64
Accuracy:  0.9291764705882353
Confusion Matrix:
 [[1876  255]
 [  46 2073]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.88      0.93      2131
           1       0.89      0.98      0.93      2119

    accuracy                           0.93      4250
   macro avg       0.93      0.93      0.93      4250
weighted avg       0.93      0.93      0.93      4250

Cross-validation scores:  [0.84117647 0.82588235 0.81882353 0.82       0.82
]
Average score:  0.8251764705882353
```

1.5.3 Accident Severity: Slight/Fatal Injury (1) vs Serious Injury (0)

[ ]:
```
X_resampled, y_resampled = oversample.fit_resample(X, y_serious)
# Check distribution
print(y_resampled.value_counts())

# Split testing data
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
                                                    test_size=0.3,
                                                    random_state=42)
# Train model
rf_model_rs = RandomForestClassifier()
rf_model_rs.fit(X_train, y_train)

# Create prediction
y_pred_rs = rf_model_rs.predict(X_test)
```

```
nb_report(y_test, y_pred_rs)

score_serious = cross_val_score(rf_model_rs, X_test, y_test, cv=5)
cross_scores(score_serious)
```

```
Accident_serious
1    7164
0    7164
Name: count, dtype: int64
Accuracy:  0.9418469411491044
Confusion Matrix:
 [[2129   36]
 [ 214 1920]]
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.98      0.94      2165
           1       0.98      0.90      0.94      2134

    accuracy                           0.94      4299
   macro avg       0.95      0.94      0.94      4299
weighted avg       0.94      0.94      0.94      4299

Cross-validation scores:  [0.82093023 0.81860465 0.84767442 0.81976744
0.82887078]
Average score:  0.8271695048325528
```

[ ]:
```
smote = SMOTE(random_state=42)

X_resampled, y_resampled = smote.fit_resample(X, y_mapped)
# Check distribution
print(y_resampled.value_counts())

# Split testing data
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
                                                    test_size=0.3,
                                                    random_state=42)
# Train model
rf_model_rs = RandomForestClassifier()
rf_model_rs.fit(X_train, y_train)

# Create prediction
y_pred_rs = rf_model_rs.predict(X_test)
nb_report(y_test, y_pred_rs)

score_serious = cross_val_score(rf_model_rs, X_test, y_test, cv=5)
cross_scores(score_serious)
```

```
Accident_severity_mapped
1    7082
0    7082
2    7082
Name: count, dtype: int64
Accuracy:  0.8870411044869784
Confusion Matrix:
 [[1879  211   77]
 [ 379 1707   30]
 [  13   10 2068]]
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.87      0.85      2167
           1       0.89      0.81      0.84      2116
           2       0.95      0.99      0.97      2091

    accuracy                           0.89      6374
   macro avg       0.89      0.89      0.89      6374
weighted avg       0.89      0.89      0.89      6374

Cross-validation scores:  [0.83137255 0.82823529 0.83372549 0.82196078 0.8422292
]
Average score:  0.831504663403823
```

[ ]:
```python
rus = RandomUnderSampler(random_state=42)

X_resampled, y_resampled = rus.fit_resample(X, y_mapped)
# Check distribution
print(y_resampled.value_counts())

# Split testing data
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
                                                    test_size=0.3,
                                                    random_state=42)
# Train model
rf_model_rs = RandomForestClassifier()
rf_model_rs.fit(X_train, y_train)

# Create prediction
y_pred_rs = rf_model_rs.predict(X_test)
nb_report(y_test, y_pred_rs)

score_serious = cross_val_score(rf_model_rs, X_test, y_test, cv=5)
cross_scores(score_serious)
```

```
Accident_severity_mapped
0    82
```

```
1    82
2    82
Name: count, dtype: int64
Accuracy:  0.5405405405405406
Confusion Matrix:
 [[13  6  3]
 [ 8 14  2]
 [10  5 13]]
Classification Report:
              precision    recall  f1-score   support

           0       0.42      0.59      0.49        22
           1       0.56      0.58      0.57        24
           2       0.72      0.46      0.57        28

    accuracy                           0.54        74
   macro avg       0.57      0.55      0.54        74
weighted avg       0.58      0.54      0.55        74

Cross-validation scores:  [0.6        0.4        0.4        0.66666667
0.35714286]
Average score:  0.4847619047619047
```

[ ]:
```
# Create baseline model
baseline_model = DummyClassifier(strategy='most_frequent')
baseline_model.fit(X_train, y_train)

y_pred_baseline = baseline_model. predict(X_test)

nb_report(y_test, y_pred_baseline)
```

```
Accuracy:  0.2972972972972973
Confusion Matrix:
 [[22  0  0]
 [24  0  0]
 [28  0  0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.30      1.00      0.46        22
           1       0.00      0.00      0.00        24
           2       0.00      0.00      0.00        28

    accuracy                           0.30        74
   macro avg       0.10      0.33      0.15        74
weighted avg       0.09      0.30      0.14        74
```

1.6 Conclusion

Our target for a successful model is >=85% accuracy and >=90% precision for the target classification of "Serious Injury". Neither model was able to acheive this requirement due to the class imbalance amongst "Slight injury", "Serious injury", and "Fatal injury". Therefore, resampling was performed. Modeling was performed on the resampled data and "Random Forest Classifier" was able to achieve our success criteria.