Chicago Insurance Agency

Matt Gracer

Business Context

- ➢ I’ve been hired by this insurance company to help them save money
- ➢ I’ve been asked to study data on car crashes with a specific interest in what is explaining the breakdown between Injuries and No Injuries for accidents
- ➢ Accidents that result in no injuries or significant damage do not need as much financial reimbursement so it’s an area where the company can save.

Data

- The data for this project is from Chicago Data Portal
- It contains data on car crashes in the Chicago region
- Crash Data
  - Crash Type(No injury(0), Injury(1)) - Target variable
  - Weather Condition - Predictor variables
  - Lighting Condition
  - Trafficway Type
  - First Crash Type
  - Device Condition
  - Damage

Baseline Logistic Regression Model - Train Data

- ➢ Train results from 300,000 rows and 116 columns
- ➢ From these results there is a 50.3% accuracy

```
0    153134
1    151244
Name: CRASH_TYPE, dtype: int64
0    0.503105
1    0.496895
Name: CRASH_TYPE, dtype: float64
```

Baseline Logistic Regression Model - Test data

- ➢ Test results from 100,000 rows and 116 columns
- ➢ From these results there is a 51.1 % accuracy

```
1    51884
0    49576
Name: CRASH_TYPE, dtype: int64
------------------------------------
1    0.511374
0    0.488626
Name: CRASH_TYPE, dtype: float64
```

Hyperparameter Logistic Regression Model

```
LogisticRegression(C=0.1, fit_intercept=False, solver='liblinear')
AUC for 0.1: 0.46408750427065004
LogisticRegression(C=1, fit_intercept=False, solver='liblinear')
AUC for 1: 0.4646881110654265
LogisticRegression(C=10, fit_intercept=False, solver='liblinear')
AUC for 10: 0.454669504236654
LogisticRegression(C=100, fit_intercept=False, solver='liblinear')
AUC for 100: 0.45841638571086374
LogisticRegression(C=1000, fit_intercept=False, solver='liblinear')
AUC for 1000: 0.45493239534215263
LogisticRegression(C=10000, fit_intercept=False, solver='liblinear')
AUC for 10000: 0.46012411260442726
```

- ➢ This hyperparameter model is the result of a for loop with 6 different C values
- ➢ All of the Hyperparameter models resulted in low accuracy values and significant changes did not result
- ➢ The same held when the C values were changed to a scale of 0.01, 0.1, 1, 10, 100 instead of starting at 0.1

Decision Tree Baseline

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=10)
```

- Accuracy for the Decision Tree was also near 50%
- Confusion Matrix is a visualization of the breakdown of the breakdown of the numbers

```
Accuracy is :50.00985610092648

AUC is :0.5

Confusion Matrix
----------------
```

Predicted	0	1	All
True			
0	25290	25194	50484
1	25526	25450	50976
All	50816	50644	101460

Cross Validation- Train & Test Data

```
cv_results = cross_validate(
                    estimator=logreg,
                    X=X_train_,
                    y=y_train,
                    cv=5,
                    return_train_score=True)
cv_results
```

```
{'fit_time': array([0.98369002, 1.41792107, 1.41900587, 1.50296974, 2.11237001]),
 'score_time': array([0.07455707, 0.07888269, 0.07184005, 0.07313728, 0.07560086]),
 'test_score': array([0.50080491, 0.49921151, 0.49870228, 0.50022177, 0.50092813]),
 'train_score': array([0.50080903, 0.50032443, 0.50010267, 0.50081929, 0.50363651])}
```

- Train and test results are both near the 50% level

Recommendations

- The primary recommendations would be that this study requires further analysis of the data to see if the accuracy score can be raised above the minimum level.
- A specific approach would be to sort through more of the columns with a high number of NAs. The challenge would be deciding how to filter those NAs that are object variables and then One Hot encoding those variables for a train test split.
- You could also change the target variable so the effect is stronger and accuracy levels are higher
- One additional recommendation would be to merge different crash datasets together and then filter NAs