Predicting accident severity: Analysis of Seattle collisions data using supervised machine learning (SML)

Sean Moffat

September 20, 2020

Case study – predicting accident severity in Seattle using supervised machine learning (SML) classification models

Business understanding / project background and objectives

- Seattle, WA city managers want to predict severity of traffic accidents. Understanding factors that contribute to accidents and making accurate predictions could be used for data-driven policies to mitigate accidents

Approach and methodology

- Used 16 years (2004 – 2019) of Seattle collisions data to train and validate SML logistic regression and k-NN classification models
- Across time, the general trend was a year-over-year decline in overall collisions; however, the rate of collisions with injuries held around 30%
- Used out-of-sample (OOS) test set to evaluate SML classification models

Findings and next steps

- The association between accident severity and weather at time of collision is not statistically significant
- SML logistic regression was the best classifier, with Jaccard similarity and F1 score at ~0.34 & ~0.51 respectively, with true positives ~93%
- In light of the SML logistic results, plan to initiate additional work related to addressing overrepresentation of outcomes & trying additional estimation solvers

Project snapshot

Intended audience / stakeholders

City managers and policy experts

Project type

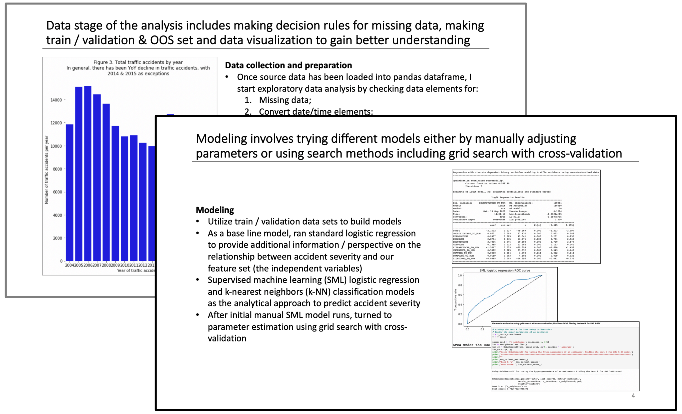
Predictions using SML classification models

Delivery date

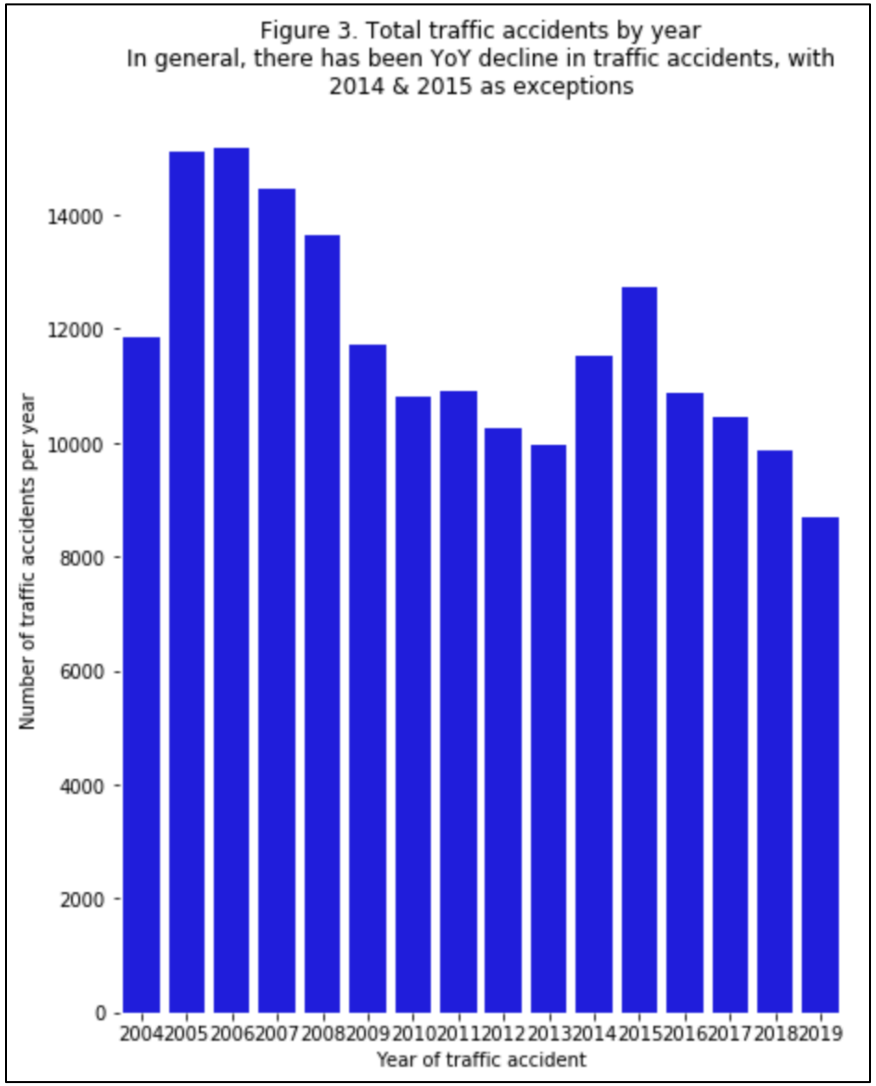
September 2020

Key outcome

Building shared understanding regarding what features are statistically significant in predicting accident severity. Also have a first SML classification model to predict accident severity



Data stage of the analysis includes making decision rules for missing data, making train / validation & OOS set and data visualization to gain better understanding



Data collection and preparation

- Once source data has been loaded into pandas dataframe, I start exploratory data analysis by checking data elements for:
    1. Missing data;
    2. Convert date/time elements;
    3. Data transformation / feature engineering; and
    4. Create a) train / validate and b) out-of-sample (OOS) test sets

Data understanding

- After data collection and preparation, I created a few time series charts to illustrate accidents across time
- Figure 3 show a general year-over-year decline in traffic accidents, with 2014 and 2015 as exceptions
- It’s possible that the decline in accidents could be driven by increased access to public transportation; would need additional data to explore this hypothesis

Modeling involves trying different models either by manually adjusting parameters or using search methods including grid search with cross-validation

Modeling

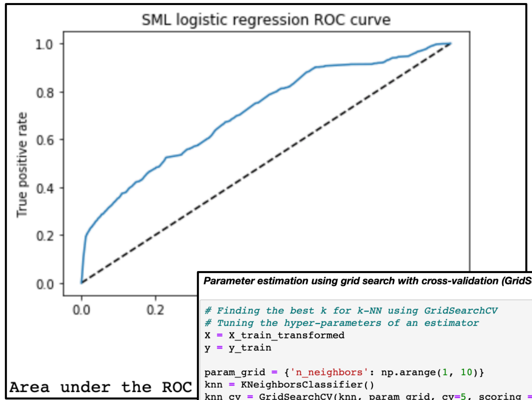
- Utilize train / validation data sets to build models
- As a base line model, ran standard logistic regression to provide additional information / perspective on the relationship between accident severity and our feature set (the independent variables)
- Supervised machine learning (SML) logistic regression and k-nearest neighbors (k-NN) classification models as the analytical approach to predict accident severity
- After initial manual SML model runs, turned to parameter estimation using grid search with cross-validation

```
Regression with discrete dependent binary variable: modeling traffic accidents using non-standardized data
-----------------------------------------------------------------------------------------------------------

Optimization terminated successfully.
         Current function value: 0.538190
         Iterations 7

Estimate of Logit model, re: estimated coefficients and standard errors

                           Logit Regression Results
==============================================================================
Dep. Variable:     SEVERITYCODE_TO_NUM   No. Observations:               188061
Model:                           Logit   Df Residuals:                   188050
Method:                            MLE   Df Model:                           10
Date:                 Sat, 19 Sep 2020   Pseudo R-squ.:                  0.1204
Time:                         14:34:14   Log-Likelihood:            -1.0121e+05
converged:                        True   LL-Null:                   -1.1507e+05
Covariance Type:             nonrobust   LLR p-value:                     0.000
=======================================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------------
const                  -2.1502      0.027    -79.569      0.000      -2.203      -2.097
COLLISIONTYPE_TO_NUM    0.0771      0.003     27.430      0.000       0.072       0.083
PERSONCOUNT             0.2407      0.005     49.041      0.000       0.231       0.250
PEDCOUNT                2.8794      0.045     63.971      0.000       2.791       2.968
PEDCYLCOUNT             2.7896      0.046     60.888      0.000       2.700       2.879
VEHCOUNT                0.1365      0.012     11.282      0.000       0.113       0.160
HITPARKEDCAR_TO_NUM    -1.5567      0.053    -29.399      0.000      -1.660      -1.453
UNDERINFL_TO_NUM        0.5912      0.025     23.892      0.000       0.543       0.640
WEATHER_TO_NUM          0.0060      0.004      1.393      0.164      -0.002       0.014
ROADCOND_TO_NUM         0.0159      0.003      4.802      0.000       0.009       0.022
LIGHTCOND_TO_NUM       -0.0365      0.003    -14.296      0.000      -0.041      -0.031
=======================================================================================
```



Parameter estimation using grid search with cross-validation (GridSearchCV): Finding the best k for SML k-NN

```
# Finding the best k for k-NN using GridSearchCV
# Tuning the hyper-parameters of an estimator
X = X_train_transformed
y = y_train

param_grid = {'n_neighbors': np.arange(1, 10)}
knn = KNeighborsClassifier()
knn_cv = GridSearchCV(knn, param_grid, cv=5, scoring = 'accuracy')
knn_cv.fit(X, y)
print('Using GridSearchCV for tuning the hyper-parameters of an estimator: Finding the best k for SML k-NN model')
print('------------------------------------------------------------------------------------------------------------')
print('')
print(knn_cv.best_estimator_)
print("Best k =:", knn_cv.best_params_)
print("Best score:", knn_cv.best_score_)
```

```
Using GridSearchCV for tuning the hyper-parameters of an estimator: Finding the best k for SML k-NN model
---------------------------------------------------------------------------------------------------------

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=8, p=2,
                     weights='uniform')
Best k =: {'n_neighbors': 8}
Best score: 0.742673310949393
```

With models out of train/validation stage, it's time to really test their accuracy by running OOS test data through them to evaluate and id best model

Evaluation

- Applied SML models (with output from grid search) built in modeling phase to out-of-sample (OOS) test data
- Used Jaccard similarity and F1 scores to compare and identify the best model
- Normalized confusion matrix was used for a quick visualization of true positives and true negatives

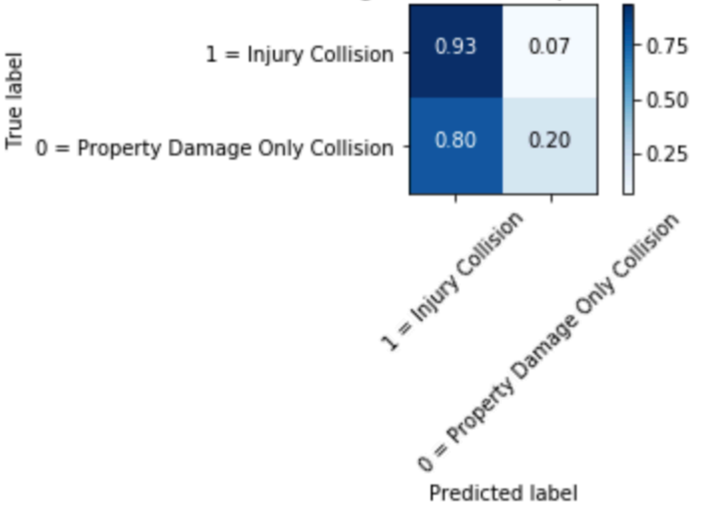
Results

- Evaluation data suggests SML logistic regression as the best classifier
- However, I have a lower Jaccard similarity and F1 score than what I'd like, as a result, plan on conducting additional analyses to see if I can increase these metrics

Table 4. Evaluation of SML models on out-of-sample (OOS) test set

Evaluation metirc	Model 1: Logistic regression	Model 2: k-nearest neighbors (k-NN)
Jaccard similarity coefficient score	0.343808	0.198276
F1 score	0.511692	0.330935
Log loss	0.859434	Not applicable
Confusion matrix from OOS test set		
True positives	93.47%	34.67%
False negatives	6.53%	65.33%
False positive	79.81%	34.77%
True negatives	20.19%	65.23%

Figure 4. Normalized confusion matrix for SML logistic regression Using GridSearchCV parameters



The analysis is rarely ever done. A Data Scientist evaluates own analysis and identifies possibilities for improvement

Discussion

- Possible future enhancements include:
  1. Use feature engineering to transform the label from a binary outcome into a multinomial, using an ***ordinal*** scale; this would allow me to get more severity detail
  2. Consider additional SML classifier models such as: Support vector machine (SVM), decision tree; and neighborhood components analysis (NCA)
  3. Could collect additional feature data elements and determine correlation between variables or run principal component analysis (PCA) to reduce the number of features to the most impactful set

Conclusion

- In this analysis, I started statistical modeling with a standard logistic regression to provide additional information / perspective on the relationship between accident severity (the dependent variable) and our feature set (the independent variables). I then moved to SML without and with hyper-parameter tuning to predict accident severity.
- Comparing SML models, the logistic regression was the best performing classifier based on Jaccard and F1 scores; the confusion matrix reported true positives at ~93%. However, this performance appears to come at a price, with false positives ~80%.
- In light of the SML logistic results, plan to initiate additional work related to addressing overrepresentation of outcomes & trying additional estimation solvers