Transportation and Accident Prediction

Alan Meyer

December 2025

1 Abstract

For this project, I applied two different types of supervised machine learning techniques: Logistic Regression, and Random Forest. To classify whether or not given certain conditions accidents will happen. The results indicate that it is difficult to predict whether or not an accident will occur meaning there must exist a better machine learning model such as extreme gradient boosting or neural networks in order to produce more accurate results. We experiment with preprocessing methods with pipeline while adjusting model parameters to yield better results. In the end, it can be concluded that traffic density and low visibility produced the highest result and that both models that were tested produced inaccurate and insignificant results.

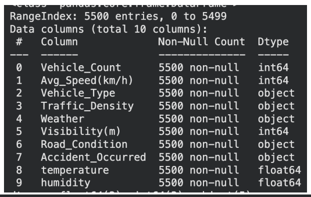
2 Introduction

In Los Angeles County, there were 41,029 crashes reported to resulting in either fatality or injury according to UC Berkeley's Transportation and Injury Mapping system. Crashes happen too frequently and it often takes emergency services a while to be notified and show up promptly. Understanding the conditions of what led to these crashes can help prepare emergency services know when more help may be needed using machine learning models.

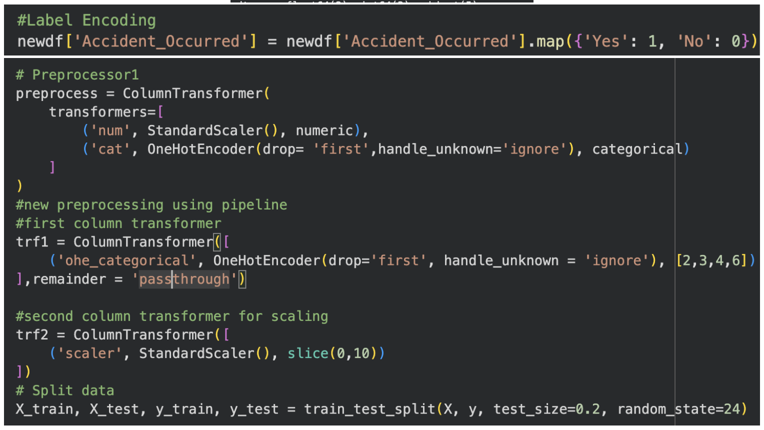
The intention of this project is to test and understand the effectiveness of two basic supervised machine learning models to see if it can accurately predict crashes given certain conditions. By analyzing our outcomes, we may be able to prepare emergency services where accidents are most likely to happen and act accordingly.

3 Data

This dataset contains synthetic transportation data representing vehicle movement, traffic density, road and weather conditions, and accident events on Kaggle. This data set originally contained 5500 entries with 17 columns which was reduced down to 10 columns.



```
RangeIndex: 5500 entries, 0 to 5499
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Vehicle_Count      5500 non-null   int64  
 1   Avg_Speed(km/h)    5500 non-null   int64  
 2   Vehicle_Type       5500 non-null   object 
 3   Traffic_Density    5500 non-null   object 
 4   Weather            5500 non-null   object 
 5   Visibility(m)      5500 non-null   int64  
 6   Road_Condition     5500 non-null   object 
 7   Accident_Occurred  5500 non-null   object 
 8   temperature        5500 non-null   float64
 9   humidity           5500 non-null   float64
```



```python
#Label Encoding
newdf['Accident_Occurred'] = newdf['Accident_Occurred'].map({'Yes': 1, 'No': 0})
```

```python
# Preprocessor1
preprocess = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric),
        ('cat', OneHotEncoder(drop= 'first',handle_unknown='ignore'), categorical)
    ]
)
#new preprocessing using pipeline
#first column transformer
trf1 = ColumnTransformer([
    ('ohe_categorical', OneHotEncoder(drop='first', handle_unknown = 'ignore'), [2,3,4,6])
],remainder = 'passthrough')

#second column transformer for scaling
trf2 = ColumnTransformer([
    ('scaler', StandardScaler(), slice(0,10))
])
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=24)
```

Then, I label encoded whether or not an accident occured 0 or 1. Next, I preprocessed the data and compared sequential transformations and parallel transformations. The main difference of these two transformations is that for the parallel scaling I only scaled the numeric columns while the sequential scaling scaled all columns. To optimize see if there were any optimizations in the order of scaling. Finally, I used train_test_split from SKLEARN.MODEL_SELECTION to split our data into a training set and a testing set. For this, we used 80% of the data for the training and 20% for the test.

4 Modeling

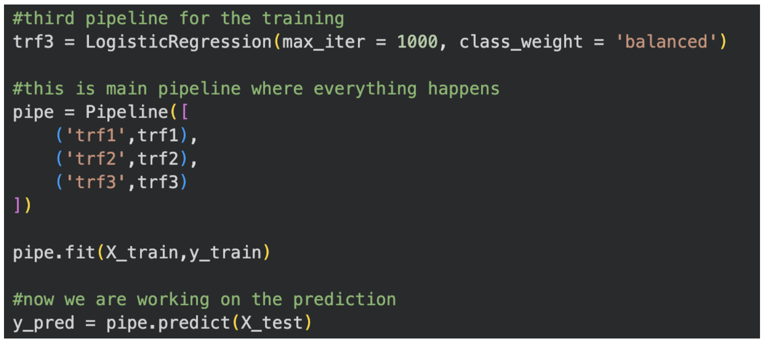
In order to predict whether an accident would happen based on the data, we decide to use supervised classification methods. Because of the nature of whether or not we determine the likelihood of a crash, we can binarily classify the target variable Accident_occurred and relevant features using Logistic Regression and Random Forest.

4.1 Logistic Regression

Since our model had a target variable that had two classes (0/1,Yes/No) we used Binary Logistic Regression. Which utilizes the formula

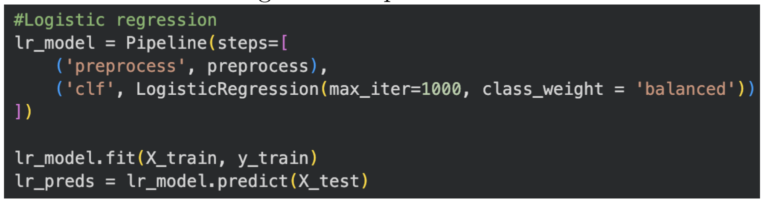
$$P(Y = 1) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_n x_n)}}$$

The difference in accuracy was negligible as the sequential pipeline had an



```
#third pipeline for the training
trf3 = LogisticRegression(max_iter = 1000, class_weight = 'balanced')

#this is main pipeline where everything happens
pipe = Pipeline([
    ('trf1',trf1),
    ('trf2',trf2),
    ('trf3',trf3)
])

pipe.fit(X_train,y_train)

#now we are working on the prediction
y_pred = pipe.predict(X_test)
```

Figure 1: Sequential code



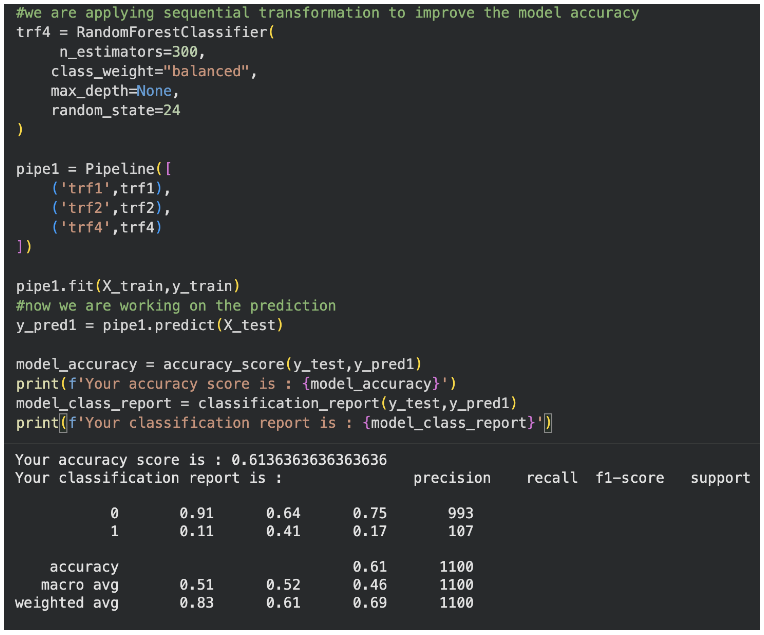
```
#Logistic regression
lr_model = Pipeline(steps=[
    ('preprocess', preprocess),
    ('clf', LogisticRegression(max_iter=1000, class_weight = 'balanced'))
])

lr_model.fit(X_train, y_train)
lr_preds = lr_model.predict(X_test)
```

Figure 2: Parallel code

accuracy of 50% while the parallel pipeline had a accuracy of 51%. This deviates from the norm as the scaling should've had a more pronounced effect due to its sensitivity to scaling.

4.2 Random Forest

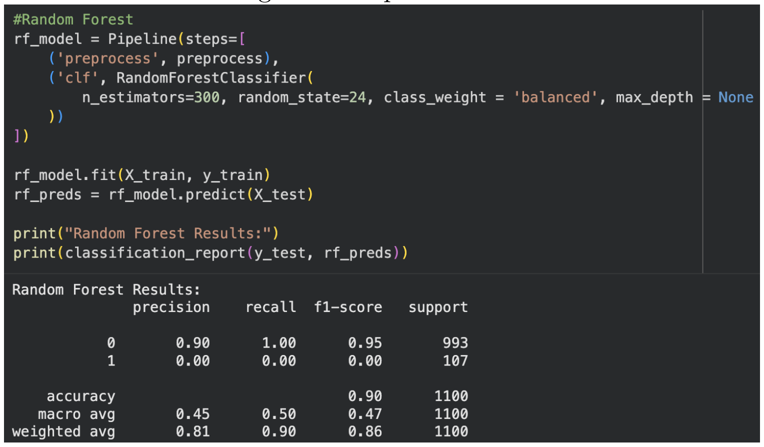
Random Forests are an ensemble learning method that utilizes a technique calle bagging which is also known as bootstrap aggregation. This technique trains multiple decision trees on samples of random subsets of data (with replacement) and then averages the results for our final model. For this project, the model was built using scikit-learn's RANDOMFORESTCLASSFIER. An interesting find here was that with the although the parallel code yielded a higher accuracy of 90%, it is not predicting if an accident is happening at all just that there is a high probabilty of no crash. The sequential transformation at least was able to predict the accidents 17% of the time but it is difficult due to the imbalance of the amount of noncrash data compared to crash data. (despite using class_weight = 'balance').



```
#we are applying sequential transformation to improve the model accuracy
trf4 = RandomForestClassifier(
     n_estimators=300,
    class_weight="balanced",
    max_depth=None,
    random_state=24
)

pipe1 = Pipeline([
    ('trf1',trf1),
    ('trf2',trf2),
    ('trf4',trf4)
])

pipe1.fit(X_train,y_train)
#now we are working on the prediction
y_pred1 = pipe1.predict(X_test)

model_accuracy = accuracy_score(y_test,y_pred1)
print(f'Your accuracy score is : {model_accuracy}')
model_class_report = classification_report(y_test,y_pred1)
print(f'Your classification report is : {model_class_report}')
```

```
Your accuracy score is : 0.6136363636363636
Your classification report is :               precision    recall  f1-score   support

           0       0.91      0.64      0.75       993
           1       0.11      0.41      0.17       107

    accuracy                           0.61      1100
   macro avg       0.51      0.52      0.46      1100
weighted avg       0.83      0.61      0.69      1100
```

Figure 3: Sequential code



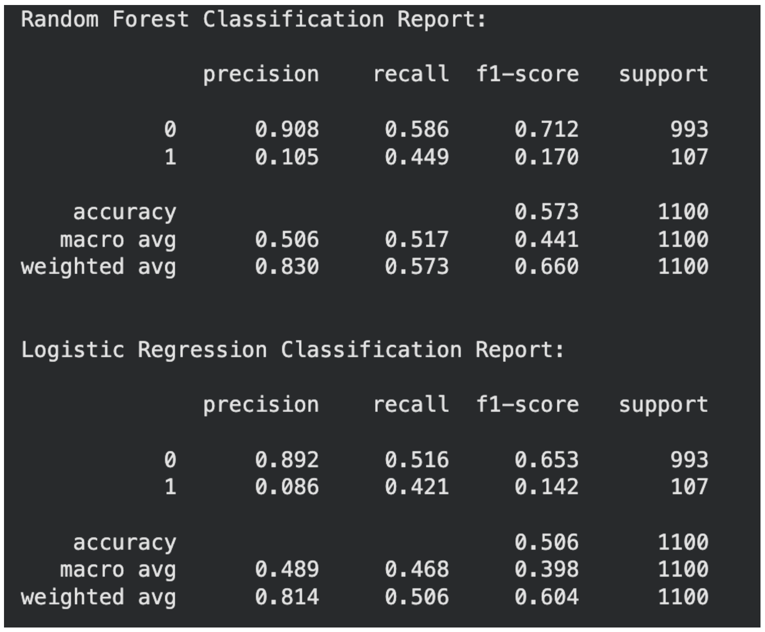
```
#Random Forest
rf_model = Pipeline(steps=[
    ('preprocess', preprocess),
    ('clf', RandomForestClassifier(
        n_estimators=300, random_state=24, class_weight = 'balanced', max_depth = None
    ))
])

rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_test)

print("Random Forest Results:")
print(classification_report(y_test, rf_preds))
```

```
Random Forest Results:
              precision    recall  f1-score   support

           0       0.90      1.00      0.95       993
           1       0.00      0.00      0.00       107

    accuracy                           0.90      1100
   macro avg       0.45      0.50      0.47      1100
weighted avg       0.81      0.90      0.86      1100
```

Figure 4: Parallel code

5 Results

As stated above the two scaling methods produced different results with the most prominent being in the Random Forest where the parallel transformation technique didn't try to predict whether or not a crash would happen while the sequential transformation did. For simplicity sakes, we only compare those models. We note that the accuracy of both of these methods are insignificant with an



```
Random Forest Classification Report:

              precision    recall  f1-score   support

           0      0.908     0.586     0.712       993
           1      0.105     0.449     0.170       107

    accuracy                          0.573      1100
   macro avg      0.506     0.517     0.441      1100
weighted avg      0.830     0.573     0.660      1100


Logistic Regression Classification Report:

              precision    recall  f1-score   support

           0      0.892     0.516     0.653       993
           1      0.086     0.421     0.142       107

    accuracy                          0.506      1100
   macro avg      0.489     0.468     0.398      1100
weighted avg      0.814     0.506     0.604      1100
```

Figure 5: Accuracy results of sequential comparison

overall accuracy of 57% for Random Forest and 50% for the Logistic Regression. Additionally, my results only show that I was able to predict crashes about 10% of the time on Random Forest and 8% for Logistic Regression. Therefore, there must be a better way to better predict whether or not an accident can occur.

Next is the ROC curve which best represents how well my model can separate my two target classes (Accident and No-Accident). Both reveal poor results of about 50% which is closer to random guessing. Below that is a precision recall curve which shows how well I can predict whether or not an accident occurs which is about 10%.

Finally, shows what the coefficients are used for the Logistic Regression while and what features were most important for the Random Forest model.

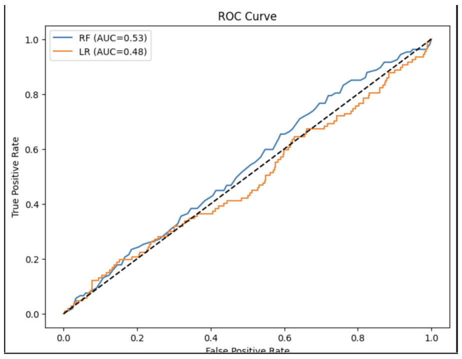


Figure 6: ROC curve

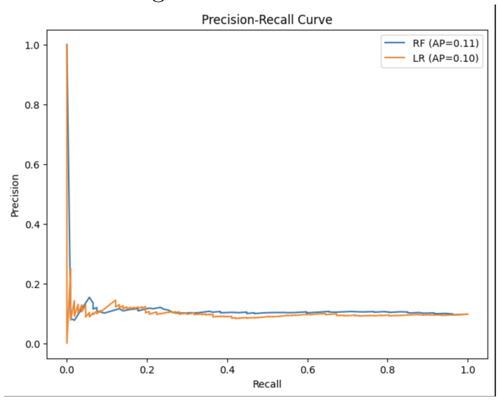
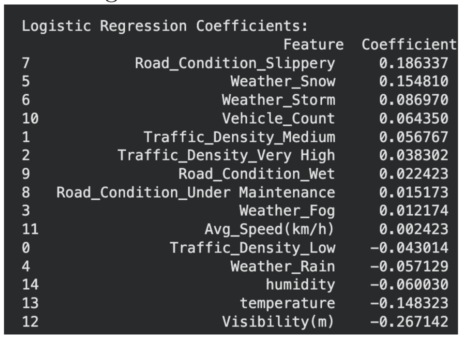
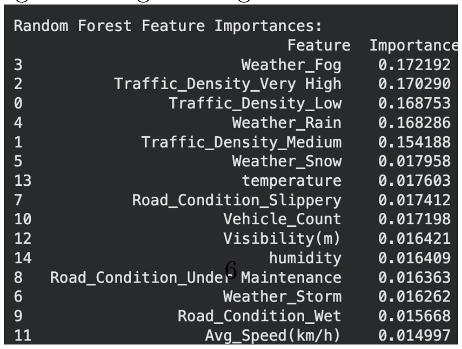


Figure 7: Precision Recall



```
Logistic Regression Coefficients:
                         Feature  Coefficient
7          Road_Condition_Slippery     0.186337
5                     Weather_Snow     0.154810
6                    Weather_Storm     0.086970
10                   Vehicle_Count     0.064350
1           Traffic_Density_Medium     0.056767
2        Traffic_Density_Very High     0.038302
9              Road_Condition_Wet     0.022423
8   Road_Condition_Under Maintenance     0.015173
3                      Weather_Fog     0.012174
11                  Avg_Speed(km/h)     0.002423
0              Traffic_Density_Low    -0.043014
4                     Weather_Rain    -0.057129
14                        humidity    -0.060030
13                     temperature    -0.148323
12                   Visibility(m)    -0.267142
```

Figure 8: Logistic Regression Coefficients



```
Random Forest Feature Importances:
                          Feature  Importance
3                      Weather_Fog    0.172192
2        Traffic_Density_Very High    0.170290
0              Traffic_Density_Low    0.168753
4                     Weather_Rain    0.168286
1           Traffic_Density_Medium    0.154188
5                     Weather_Snow    0.017958
13                     temperature    0.017603
7          Road_Condition_Slippery    0.017412
10                   Vehicle_Count    0.017198
12                   Visibility(m)    0.016421
14                        humidity    0.016409
8   Road_Condition_Under Maintenance    0.016363
6                    Weather_Storm    0.016262
9              Road_Condition_Wet    0.015668
11                 Avg_Speed(km/h)    0.014997
```

Figure 9: Random Forest Feature Importances

6 Discussion

The results yielded were unfortunately insignificant with my models both scoring around 50% accuracy and only predicting an accident about 10% of the time which leads to the conclusion that there must exist a better supervised learning model. My guess is that a neural network or a Extreme gradient descent would improve performance but more research must be done.

When modeling, the most surprising result that occurred would be the difference of how I utilizing scaling columns in my preprocessing phase. With the parallel scaling I only scaled the numeric values, while the sequential scaling of all values yielded different results. The numeric (parallel) scaling found that the best model was one that never predict an accident while sequential transformation method at least attempted. In the end both models predicted about 10% which was disappointing. A key reason for this may be the disproportionate amount of non-accidents compared to the amount of accidents. Despite balancing the target variables in the parameters, our model still couldn't predict accidents resulting in super poor performance. More methods need to be tested. Additionally, maybe some form of regularization or checking of overfitting may have been beneficial.

7 Conclusion

In summary, the purpose of this project was to see if Logistic Regression or Random Forest can better accurately predict if an accident is likely to occur to this classification problem.

Unfortunately, the two models tested didn't yield significant results or remotely accurate results for the main thing we are predicting which was whether or not an accident would occur. However, it did show that visibility, weather, and traffic density had part in predicting whether or not an accident occurred but my model was still guessing.

In conclusion, Logistic Regression and Random Forests are not the best supervised learning models for this type of problem and there must exist a better method to better model whether or not an accident may occur given specific conditions.

8 References

University of California, Berkeley SafeTREC. (n.d.). *Transportation Injury Mapping System (TIMS) — Crash Summaries.* Retrieved [12-4-25], from TIMS Crash Summary Tool.

kundanbedmutha. (n.d.). *Transportation and Accident Prediction Dataset* [Data set]. Kaggle. Retrieved [12-4-25], from https://www.kaggle.com/datasets/kundanbedmutha/transportation-and-accident-prediction-dataset