Machine Learning: NJ Accidents

August 8, 2023

Anika Dinger, Lauren Golden, Justine Owsik, MJ Teng, Ricardo Uspango-Hormiga

Data Source

- 49 states of car accident data
- Collected from 2/2016 - 3/2023
- Contains 7.7M rows of accident data
- Pulls real time data with multiple Traffic APIs
- 140K rows for NJ specific accidents

Source: https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents

Analysis Goals

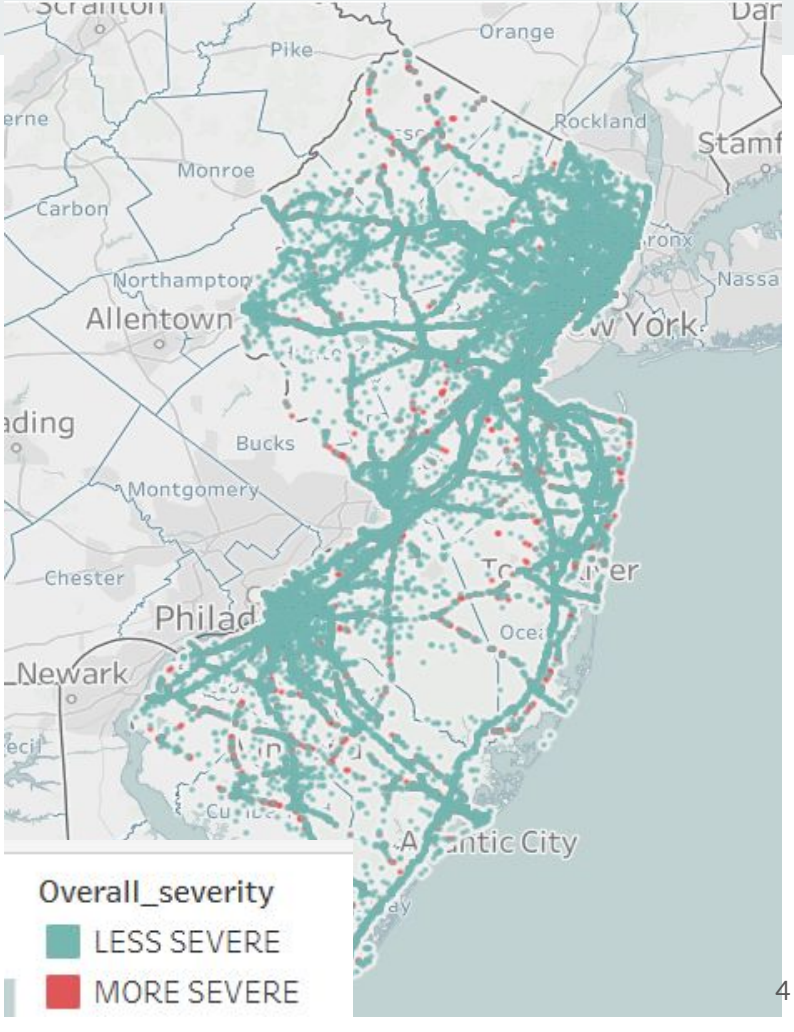
- Identify the most important features of the dataset
  - Environmental factors (weather conditions, temperature, humidity, etc)
  - Traffic controllers (stop signs, traffic signals, roundabouts)

- Determine accident severity accuracy through machine learning

Map of NJ accidents

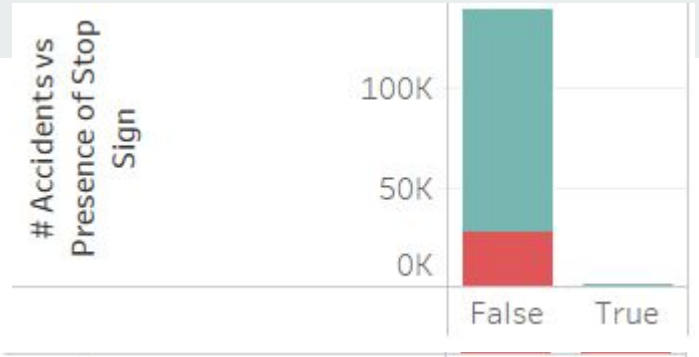
Initial Findings:

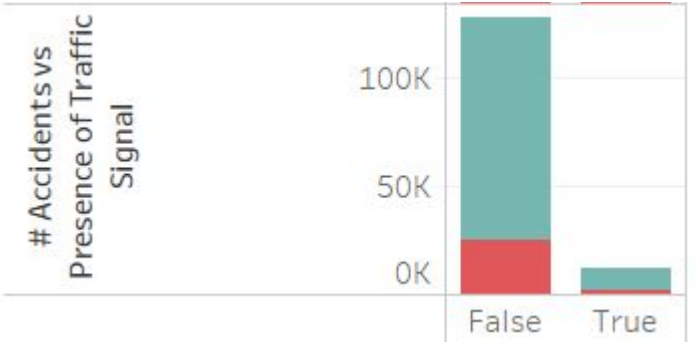
- More accidents occurred on major highways and cities
- Dataset has more observations of “less severe” accidents
- No apparent correlation between severity of accidents and location

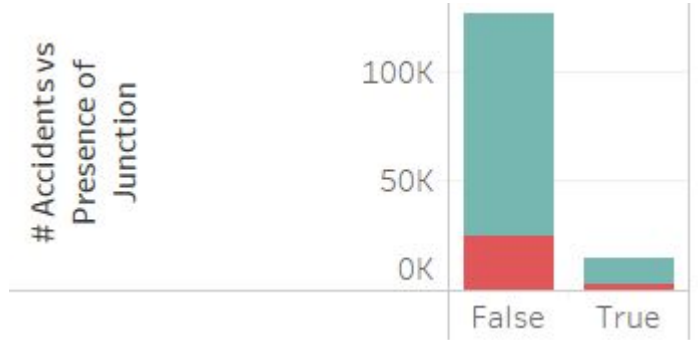


Traffic Controllers

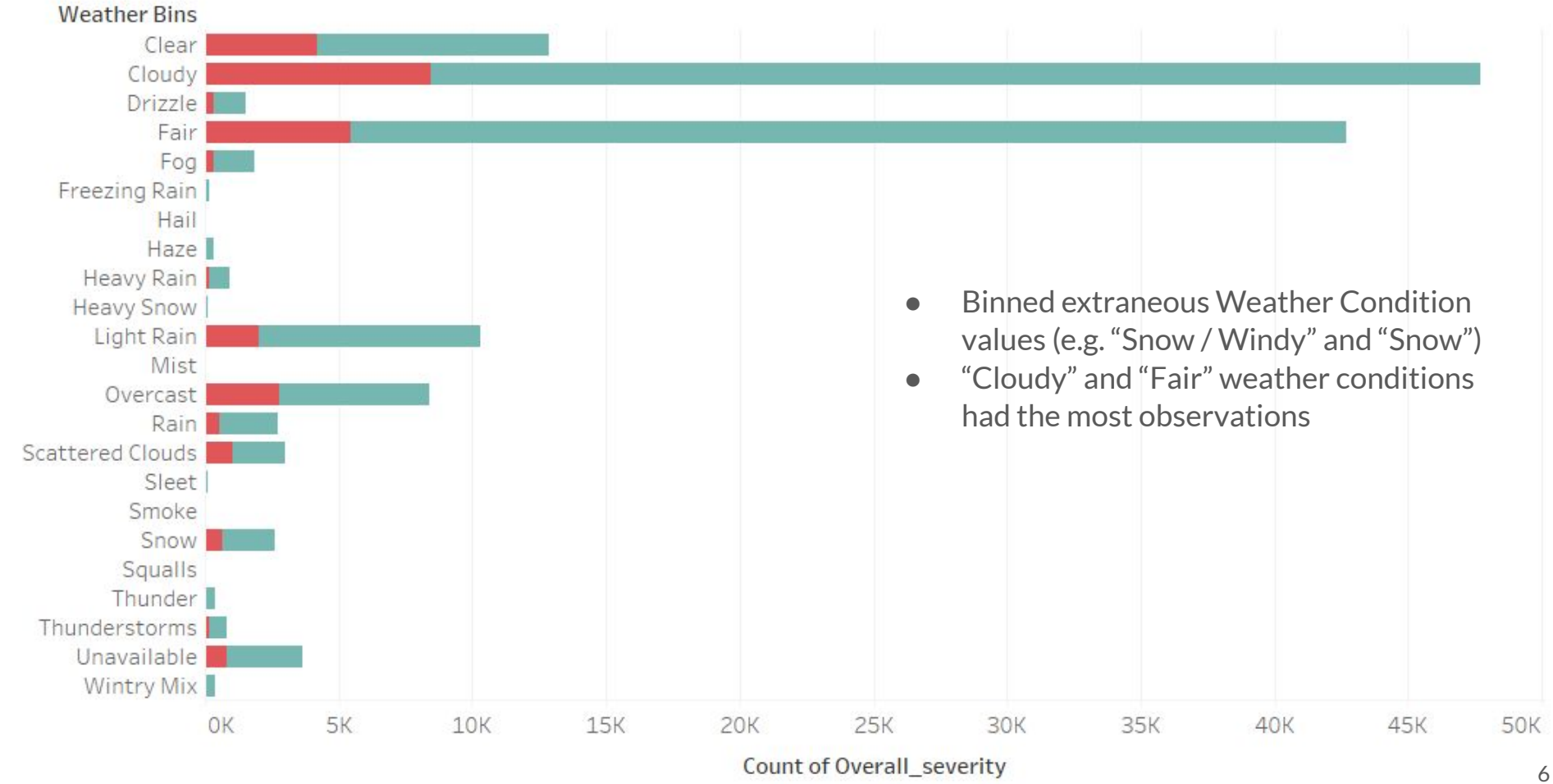
- Traffic controllers include stop signs, junctions, traffic lights
- Drivers have to be more cautious
- Significantly fewer accidents wherever a traffic controller was present







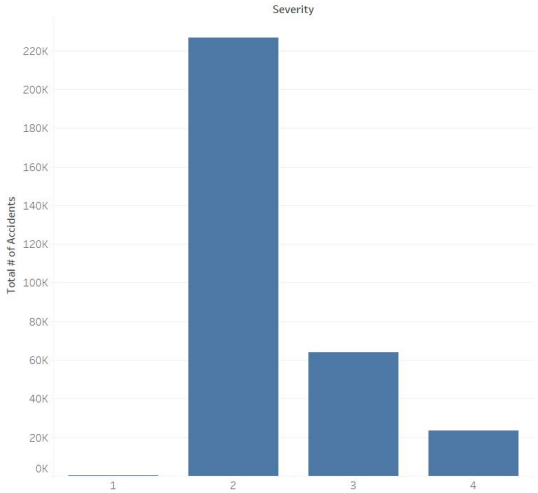
Weather Condition Breakdown



- Binned extraneous Weather Condition values (e.g. “Snow / Windy” and “Snow”)
- “Cloudy” and “Fair” weather conditions had the most observations

Data Cleaning, Preprocessing

- Severity Levels originally 1,2,3,4
- 1,2 → 0, “Less Severe”
  - 113K rows of data
- 3,4 → 1, “More Severe”
  - 27K rows of data



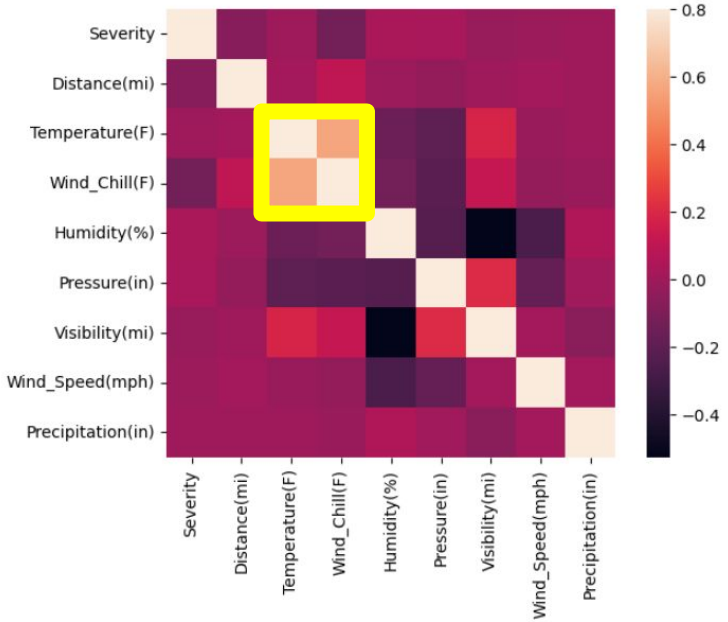
- Dropped columns that could not be used by the model (e.g. coordinates, verbal descriptions, Airport Code, etc.)
- Filled NaNs with average values by month and year
- Binning similar values (e.g. “N” vs “North”

```
# Create a summary table to use groupby to average all numerical columns for data imputation purposes
summary = nj_rows.groupby(["Month", "Year"]).mean()
summary
```

Month	Year	Severity	Distance(mi)	Temperature(F)	Wind_Chill(F)	Humidity(%)	Pressure(in)	Visibility(mi)	Wind_Speed(mph)	Precipitation(in)
	2017	0.321127	0.188084	36.757183	29.552778	70.605634	30.058554	8.468649	9.706264	0.472385
	2018	0.357886	0.352946	30.579791	17.532300	67.705694	30.127927	8.082095	11.292929	0.141541
1	2019	0.308891	0.269234	32.157117	21.369162	63.343434	30.071899	8.860890	10.463382	0.050876
	2020	0.306701	0.295952	38.574266	35.004464	61.849957	30.090516	9.191101	6.542705	0.001873
	2021	0.093902	0.722234	34.477439	28.852349	64.620732	29.925543	9.045009	8.176327	0.003591
...	...	...	...	...	...	...	...	...	...	...
	2018	0.328583	0.312276	40.796815	31.058372	72.883487	30.070142	8.318182	9.377968	0.054322
	2019	0.353059	0.375054	37.707194	33.215805	72.076220	29.941647	7.933972	7.799197	0.008887
12	2020	0.094168	0.753235	37.625918	32.837733	68.512532	29.942147	8.792910	8.522285	0.004721
	2021	0.139908	0.924514	43.638379	41.258324	61.584480	29.970409	9.195340	6.436280	0.001139
	2022	0.026053	0.854321	39.661005	35.120583	66.204364	30.064433	8.467275	8.169784	0.007716

Data Preprocessing

- ➔ Both temperature and wind chill have a strong positive correlation to each other
- ➔ Using both would produce negligible results



Model Selection and Training

- Random Forest: high reliability, moderate interpretability
- Hot one encoding
- Undersampling with 


```
In [6]: # Balance out two classes
        from collections import Counter
        from imblearn.under_sampling import RandomUnderSampler
        rus = RandomUnderSampler(random_state=0)
        X_resampled, y_resampled = rus.fit_resample(X, y)
        print(sorted(Counter(y_resampled).items()))

        [(0, 26906), (1, 26906)]
```

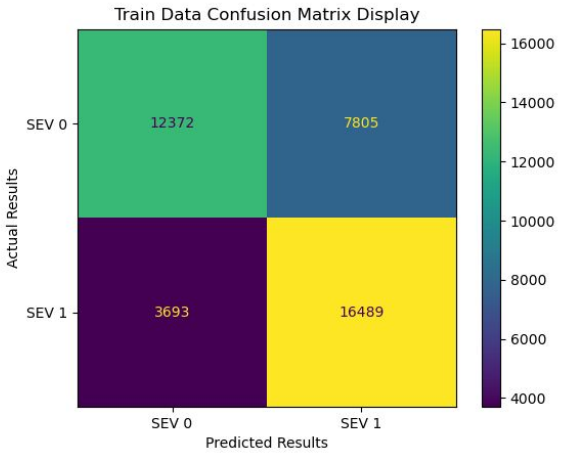
Model Selection and Training

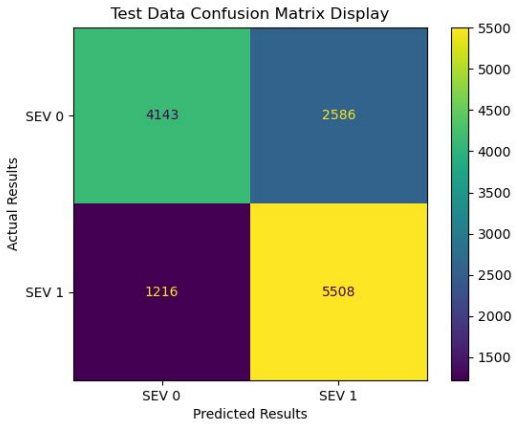
- Hyperparameter Tuning (number of estimators, max depth and max number of leaves)

In [9]:
```
# Establish function to create a model
def tune_random_forest(X_train, X_test, y_train, y_test, n_estimators_val, max_depth_val, min_samples_val):

    rf_model = RandomForestClassifier(n_estimators=n_estimators_val, max_depth = max_depth_val, min_samples_leaf = min_samples_va

    start_time = time.time()

    rf_model.fit(X_train, y_train)

    end_time = time.time()

    total_time = end_time - start_time

    predictions = rf_model.predict(X_train)
    accuracy_score_train = accuracy_score(y_train, predictions)

    predictions_test = rf_model.predict(X_test)
    accuracy_score_test = accuracy_score(y_test, predictions_test)

    return total_time, accuracy_score_train, accuracy_score_test
```

- Compile, fit and train the optimal model

Confusion Matrix Display



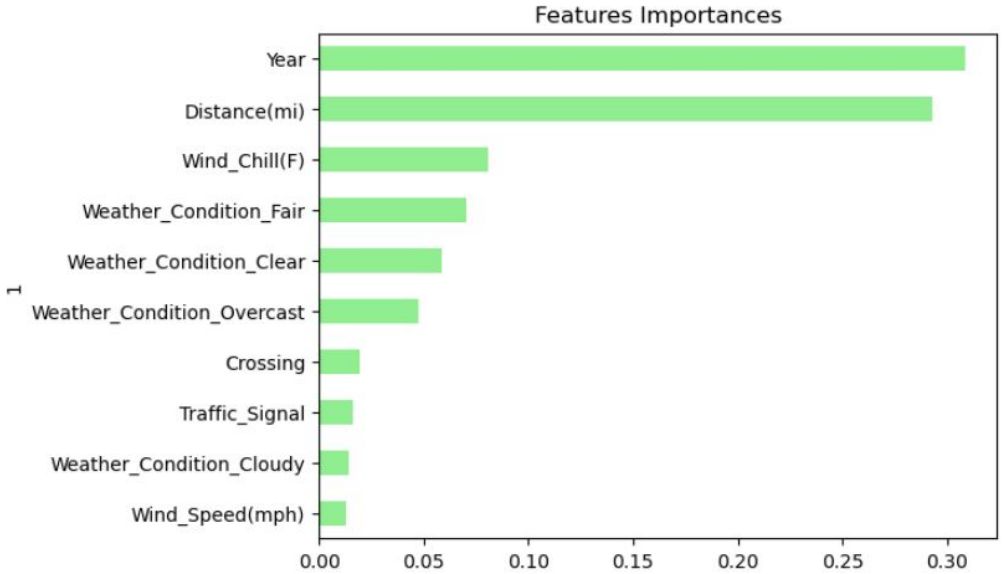


Findings

- ➔ Model predicts less severe accidents as less severe (0) and severe accidents as severe (1) more times correctly than incorrectly.
- ➔ The train data and test data show similar accuracy with the trained model

Feature Importance

- Significant features that influence an accident's severity and its impact on traffic were:
  - Year
  - Distance
  - Weather Conditions
  - Wind Chill
  - Traffic Signal
  - Traffic Crossing



Final Findings

- Confusion Matrix Results
  - 71% accuracy
- Top 3 features were year, distance and wind chill. Due to COVID, we noted year may hold importance due to the lack of drivers/accidents on the road during lockdown. This may hold higher importance in these years but may not have much importance in predicting accidents in future years
- Test Data Results

Accuracy Score : 0.7173864565524418
Classification Report

	precision	recall	f1-score	support
0	0.77	0.62	0.69	6729
1	0.68	0.82	0.74	6724
accuracy			0.72	13453
macro avg	0.73	0.72	0.71	13453
weighted avg	0.73	0.72	0.71	13453

Training Data Results

Accuracy Score : 0.7151069154339801
Classification Report

	precision	recall	f1-score	support
0	0.77	0.61	0.68	20177
1	0.68	0.82	0.74	20182
accuracy			0.72	40359
macro avg	0.72	0.72	0.71	40359
weighted avg	0.72	0.72	0.71	40359