Data Mining Final Project Paper - Group #5
Exploring Factors that Impact Car Accident Severity

Abdulaziz Gebril, Jenny Tsai, & Mojahid Osman

April 28, 2020

I. Introduction

Car accidents happen everyday in the United States, but only the fatal ones take away people's lives. According to the U.S. Department of TransportationIn, there were 36,560 deaths and 33,654 fatal crashes in 2018. It is worthwhile to explore factors that could possibly impact car accident severity, which in turn, help to inform strategies on how to lower the occurrence of the fatal crashes.

A number of prior studies have investigated the relationship between traffic accidents and environmental stimuli (e.g., Jaroszweski & McNamara, 2014), or making predictions of accident frequency by geographic location (e.g., Chang & Chen, 2005). However, little research has been done to predict fatal traffic accidents based on weather and road conditions combined. Therefore, we formulated our problem statement as follows: "What weather conditions and road conditions would impact car accident severity?"

II. Description of Dataset

The U.S. accident dataset was created by Moosavi et. all (2019) who compiled the data from various publicly available sources, including two APIs that provide streaming traffic incident data. These APIs broadcast traffic data were captured by a variety of entities, such as the U.S. State Departments of Transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks.

The dataset contains about three million instances of traffic accidents that took place within the 49 states of the United States from February 2016 to December 2019. A total of 49 variables are in this dataset, including intrinsic and contextual attributes such as location, time, weather, and points-of-interest (see Appendix A for the full list of variables). Though the preliminary data merging and cleaning was done, the dataset still has a lot of missing values and attributes that are not relevant to our problem statement. The next section will describe the data mining techniques we used for pre-processing.

III. Data Mining Techniques: Preprocessing

a) Date Column Creation

Since time-series modeling was not of our research interest, we decided to only focus on the year of 2019 (the most recent year) in our study. To do that, we used columns "Start_Time" and "End_Time" to extract "Year" and other date/time variables (e.g., Date, Month, Day, Hour, WeekDay, Time Duration) for the purpose of subsetting and missing value imputation.

b) Dependent Variable Discretization: Accident Severity

The "Severity" column shows the severity of the accident ranged from 1 and 4, where 1 indicates the least impact on traffic (i.e., short delay as a result of the accident) and 4 indicates a significant impact on traffic (i.e., long delay). For the ease of data interpretation, accidents with severity of 1 or 2 were classified as "Low" and accidents with severity of 3 or 4 were classified as "High".

c) Missing Data Imputation & Data Cleaning

The 2019 dataset contains 953,630 records. The missing values per column is shown in *Figure 1*.

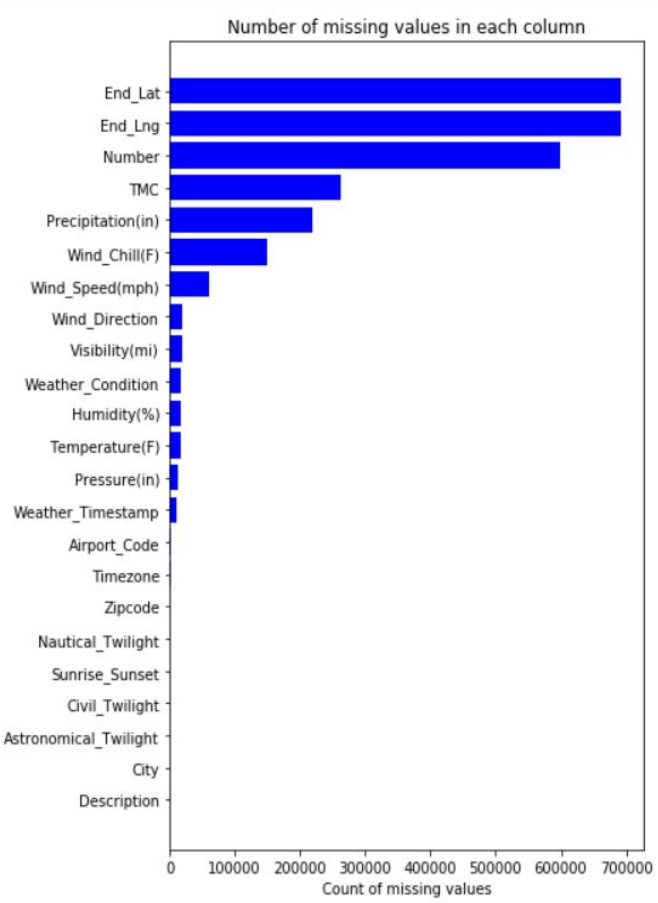

Figure 1. Number of Missing Values in Each Variable

c.1. Continuous Variables Imputation

In this dataset, all continuous variables are related to weather conditions. Upon visualizing the missing value patterns, a random pattern of the occurrence of the missing data for the weather condition variables was noticed. Therefore, imputing these values using different methods is achievable.

The imputation was done in two stages. The 1st imputation was to fill missing values with the average value of accidents that occurred on the same *Date* and *City*. For the remaining missing values, the 2nd imputation was used to fill the missing values with the average value of accidents that occurred on the same *Date* and *State*. The results of imputation are shown in *Table 1*.

Although the 2nd imputation significantly cuts down the total number of missing values, we still needed to assess whether the 2nd imputation was appropriate as there could be a great variance within a large state. Therefore, a distance function was created to calculate the Euclidean distances among cities in California where we have the most data points. It was found that some cities are pretty far from each other, thus averaging values among these cities might not be a reasonable approach. For example, the distance between San Diego and Riverside is 140 km and both have accidents that occurred on the same day. As a result, we decided to go with the 1st imputation method.

Additionally, we noticed that "Precipitation" and "Wind Chill" still contained a great number of missing values even after the 1st imputation had imputed a significant amount of data points, so we tried the *regression imputation approach* using other continuous weather variables as predictors. The regression model on Precipitation resulted in a very low R^2 score of 0.07, so we decided to drop Precipitation. As for "Wind Chill", the regression resulted in a high R^2 score of 0.98 (adjusted R^2 = 0.97). However, we compared the model with vs. without imputed Wind Chill and found that the model performance was almost the same. It is possible that Wind Chill has a high correlation with other weather variables and therefore did not add extra value to the model. Therefore, we decided to drop Wind Chill in the modeling stage as well.

Table 1. Number of Missing Values Before and After Imputation in 2019 Dataset

Feature	Missing Value (MV) Count	# of MV after 1st Imputation	# of MV after 2nd Imputation
Temperature	17,184	12,515	28
Wind Chill	149,234	112,095	14,166
Humidity	18,266	12,961	31
Pressure	13,795	9,944	28
Visibility	18,481	13,965	49
Wind Speed	60,083	30,526	170
Precipitation	219,122	177,214	51,652
Total	**496,165**	**369,220**	**66,124**

c.2. Categorical Variable Cleaning

Since we barely found missing values in the categorical variables, we turned our focus to cleaning and discretization. For multi-class variables such as "Weather Condition" and "Wind Direction", we noticed that some of those categories can be reduced to one category. For example, "Rains", "Heavy Rain", "Light Rain" can all be reduced to "Rain". This process reduced "Weather Condition" to 10 categories (Cloudy, Windy, Rain, Snow, Thunderstorm, Fog, Sleet, Haze, Wintry-Mix and Dust). Same discretization and cleaning approach was applied to "Wind Direction", where some values were found to be abbreviations of other categories thus they were combined into the same category. For example, "E" and "East" were combined to "East".

Upon EDA, we observed that "Weather Condition" and "Wind Direction" still included too many levels that are not fitting well to our models. Therefore, we decided to drop these two variables in the modeling stage.

d) Imbalance Data Treatment

Upon classifying our dependent variable Severity as "high" and "low", we noticed that the 2019 data was extremely imbalanced where around 72% are low-severity cases (see *Figure 2*). As imbalanced data tend to bias model predictions towards the majority class and impact the model performance, firstly we tried resampling to address this issue (e.g., oversample the minority class to match the number of the majority class). However, the results were either underfitting (when undersampled) or overfitting (when oversampled). Therefore, we decided to pull the high severity cases from 2018 data and added to the 2019 data (Note: all prior year data have been pre-processed the same way as the 2019 dataset was; see Appendix B for more details). The percentage of low-severity cases came down to 55% in this combined dataset, so we decided to use this combined dataset for modeling.

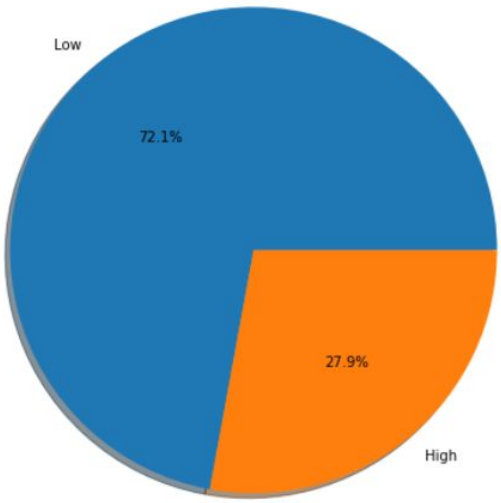

Figure 2. Severity Distribution for the year 2019

e) Drop Irrelevant Variables and Label Recoding

To prep the data for modeling, we dropped the attributes that are irrelevant to our study, such as Location and Time variables. Next, we dropped the remaining NaNs in each variable after 1st imputation was done. Finally, we recoded the binary variables into 0 and 1 (e.g., High Severity = 1, Low Severity = 0). The total sample size of the final cleaned dataset came to 1,203,564 with 18 attributes (6 weather conditions and 12 road conditions) and one target (Severity) (see *Table 2* for the list of variables remained after preprocessing).

Table 2. List of Variables after Preprocessing

Label	Variable Name	
Y: Target	• Severity	
X: Weather Condition	• Temperature • Humidity • Pressure	• Visibility • Wind Speed • Sunrise/Sunset
X: Road Condition	• Amenity • Bump • Crossing • Give Way • Junction • No Exit	• Railway • Roundabout • Station • Stop • Traffic Calming • Traffic Signal

IV. Experimental Setup

a) Theoretical Framework

To obtain robust and reliable results, we chose two ensemble models as our analytic framework: Random Forest and Adaptive Boosting Decision Trees. Random Forest uses the bagging technique, where it generates randomly sampled subsets to train a bunch of trees in a parallel way. The final prediction is based on the majority vote across all trees.

On the other hand, Adaptive Boosting uses the boosting technique, where it trains a bunch of trees in a sequential way. The model *adapts* as it increases the weight of misclassified datapoint for the next tree training; each tree starts out with equal weight, but the decision power of each tree is re-assigned based on its weighted error rate (see formula below). The final prediction is based on the weighted vote across all trees.

$$\text{the weight of this tree} = \text{learning rate} * \log(\ (1 - e) / e)$$

Additionally, we use *Grid Search* to find the best hyper-parameters for each model. That is, an exhaustive search is conducted to try all parameter combinations we pass on and find the best parameters with the highest performance score (e.g., accuracy). We also use *K-fold Cross Validation* in the grid search to increase reliability of the results (that is, our results are not just obtained through a particular train-test set, but cross-validated through k combinations).

b) Algorithms with Actual Data

We used the *sklearn* package in Python for modeling and training. No customization is made except for the hyper-parameters we pass onto the models based on the grid search results. Our modeling steps are listed below:

(1) Perform Grid Search and Cross Validation for Random Forest and AdaBoost Classifier, respectively. Please note that due to computational power constraint, we had to use a subset of 2,000 samples (stratified random sampling: 1,000 high severity and 1,000 low severity) rather than the full dataset to run these codes. See below for the sample codes of grid search for Random Forest:

```
#------------------------- Part II: Grid Search for Random Forest ----------------
# random forest classifier
clf = RandomForestClassifier(n_estimators=100, random_state=200)
```

```
# set grid parameters
grid_param ={'n_estimators': [50, 75, 100, 200],'criterion' :['gini', 'entropy']}
```

```
# specify grid search
gd_sr = GridSearchCV(estimator=clf,
                     param_grid=grid_param,
                     scoring='accuracy',
                     cv=5)
```

```
# perform grid search
gd_sr.fit(X_train, y_train)
```

```
# find best parameters
best_parameters = gd_sr.best_params_
print(best_parameters)
```

(2) Split the full dataset of 1.2 million samples into train and test in a ratio of 7:3

```
# split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,
```

(3) Build Random Forest and AdaBoost models with the best parameters from grid search

```
# perform training with random forest using all features
clf = RandomForestClassifier(n_estimators=100,criterion='gini')
clf.fit(X_train, y_train)
```

```
# AdaBoost - using all features
clf_ab = AdaBoostClassifier(n_estimators=100, learning_rate = 0.1)
clf_ab.fit(X_train, y_train)
```

(4) Select Top 10 important features and re-run each model. Below are the sample codes for Random Forest.

```
#------------------------- Part III: Random Forest with Top 10 Features -----
# select the training and test set using top 10 features
newX_train = X_train[:, clf.feature_importances_.argsort()[::-1][:10]]
newX_test = X_test[:, clf.feature_importances_.argsort()[::-1][:10]]
```

```
# perform training with random forest with top 10 features
clf_k_features = RandomForestClassifier(n_estimators=100,criterion='gini')
clf_k_features.fit(newX_train, y_train)
```

(5) Predict against the test set and evaluate performance. Below are the sample codes for the Random Forest model.

```
# prediction on test using top 10 features
y_pred_k_features = clf_k_features.predict(newX_test)
y_pred_k_features_score = clf_k_features.predict_proba(newX_test)
```

```
# classification report - top 10 features
print("\n")
print("Results Using Top 10 features: \n")
print("Classification Report: ")
print(classification_report(y_test,y_pred_k_features))
print("\n")
print("Accuracy : ", accuracy_score(y_test, y_pred_k_features) * 100)
print("\n")
print("ROC_AUC : ", roc_auc_score(y_test,y_pred_k_features_score[:,1]) * 100)
```

V. Results

a) Best Parameters from Grid Search and Cross-Validation

For Random Forest, we performed grid search for n_estimators (i.e., number of trees in the ensemble: 50, 75, 100, 200) and for criterion (entropy vs. gini). The best parameters found for RF were 100 trees with gini. For Adaptive Boosting, we performed grid search for n_estimators (50, 75, 100) and for learning rate (0.1, 0.25, 0.5, 1), and the best parameters were 100 trees at the learning rate of 0.1. Both searches were based on a cross validation of 5 folds and accuracy score.

b) Feature Selection

The top 10 important features from Random Forest and Adaptive Boosting are shown in *Figure 3* and *Figure 4*. In Random Forest, weather variables were ranked higher than road conditions. Though the top 2 important variables were also related to weather in AdaBoost, the weather variables were more spread out throughout the top 10, where some road condition variables actually had higher ranking (Traffic Signal, Stop, Crossing). Overall, the two models had eight important features in common as shown in *Table 3*.

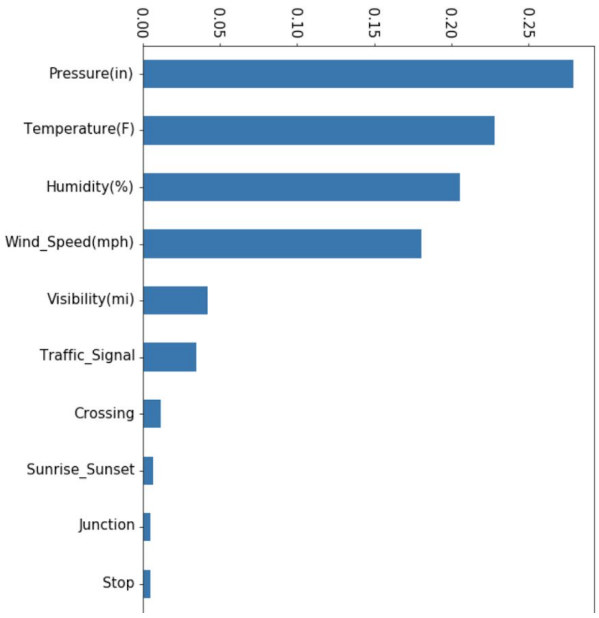

Figure 3. Top 10 Important Features from Random Forest Model

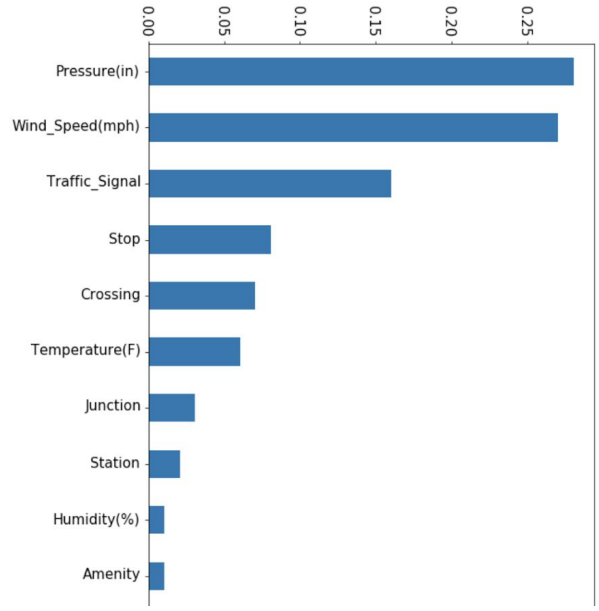

Figure 4. Top 10 Important Feature from AdaBoost

Table 3. Common Important Features in Random Forest and AdaBoost

Weather Conditions	Road Conditions
• Pressure • Temperature • Humidity • Wind Speed	• Traffic Signal • Crossing • Junction • Stop

c) Model Performance

The performance of the two models are shown in *Table 4* and *Table 5.* Random Forest had an accuracy rate of 0.74, an AUC over 0.81, and an average F-1 score of 0.74. Except for a slightly lower score of 0.68 on recall predicting High Severity class, which makes sense as we had less high severity cases in our dataset and the prediction would be biased towards the majority class. Overall, the Random Forest model had a good performance scoring between 0.70 and 0.79.

Compared to RF, the AdaBoost model had a lower accuracy rate of 0.68, an AUC of 0.73, and a F-1 score of 0.68. The performance pattern was similar to that of RF, scoring between 0.62 to 0.73, which fell within an acceptable range to be considered as a good model.

Simply put, our data seemed to fit better with Random Forest than the Adaptive Boosting Trees. One possible reason is that the Boosting models tend to outperform other models when the dataset is extremely imbalanced because higher weight is given to the minority class when misclassified. Since we had already addressed the imbalance data issue during the pre-processing stage, Random Forest was no longer affected so much by this issue and therefore had better performance than AdaBoost Trees. We actually tested this assumption by running both models with an imbalanced dataset, and AdaBoost turned out to have a better accuracy rate than the Random Forest model.

Another possible explanation that AdaBoost didn't perform as well is that AdaBoost is sensitive to noisy data and outliers. Despite our efforts in data cleaning, this huge dataset might still contain noisy data that can only be identified by domain experts.

Table 4. Performance of Random Forest

```
Results Using Top 10 features:

Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.79      0.77    199174
           1       0.73      0.68      0.70    161896

    accuracy                           0.74    361070
   macro avg       0.74      0.74      0.74    361070
weighted avg       0.74      0.74      0.74    361070



Accuracy :  74.2185725759548


ROC_AUC :  81.16041778890892
```

Table 5. Performance of Adaptive Boosting

```
Results Using Top 10 Features:

Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.73      0.72    199174
           1       0.65      0.62      0.64    161896

    accuracy                           0.68    361070
   macro avg       0.68      0.68      0.68    361070
weighted avg       0.68      0.68      0.68    361070



Accuracy :  68.25546292962584


ROC_AUC :  73.30263509189082
```

VII. Summary and Conclusion

According to the results from the Random Forest and AdaBoost model, eight features were found that impact the car accident severity the most: Pressure, Humidity, Temperature, Wind Speed, Traffic Signal, Crossing, Junction, and Stop Sign. The Random Forest model predicts better when the data is balanced, while the AdaBoost model would yield more accurate results when data is imbalanced.

When it comes to application, our models can be used to predict accident severity based on the weather conditions and road conditions at a particular location. For example, the US Department of Transportations can craft strategies to prevent fatal accidents in high-severity areas. Auto-insurance companies can also reference the models and set area premium rates accordingly.

Nevertheless, data scientists still have to exercise caution when generalizing our findings as our data was collected in the United States and may not be applicable to other countries. Future research should explore more factors to refine the models besides weather and road conditions, such as driver behaviours, population density, availability of public transportation, etc.

IX. References

Chang, L. Y. & Chen, W. C. (2005). Data mining of tree-based models to analyze freeway accident frequency. *Journal of Safety Research, (36)*4, 365–375.

Eisenberg, D. (2004). The mixed effects of precipitation on traffic crashes. *Accident Analysis & Prevention*, *(36)*4, 637–647.

García, S., Luengo, J., & Herrera, F. (2015). *Data preprocessing in data mining*. Switzerland: Springer.

Jaroszweski, D. & McNamara, T. (2014). The influence of rainfall on road accidents in urban areas: A weather radar approach. *Travel Behaviour and Society*, *(1)*1, 15–21.

Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, & Rajiv Ramnath. (2019). *A Countrywide traffic accident dataset.* Retrieved from https://smoosavi.org/datasets/us_accidents.

Moosavi, S., Samavatian, M. H., Parthasarathy, S., Teodorescu, R., & Ramnath, R. (2019). Accident risk prediction based on heterogeneous sparse data: New dataset and insights. In proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2019.

Appendix A: The Full List of Variables in the Original Dataset

Feature	Description
ID	Unique Identifier of an accident
Source	Source of accident report
TMC	Traffic Message Channel code
Severity	Shows Severity of accident
Start_Time	Start time of accident in local time zone
End_Time	End time of accident in local time zone
Start_Lat	Shows latitude in GPS coordinate of the start point
Start_Lng	Shows longitude in GPS coordinate of the start point
End_Lat	Shows latitude in GPS coordinate of the end point
End_Lng	Shows longitude in GPS coordinate of the end point
Distance(mi)	The length of road extent affected by accident
Description	Description of accident
Number	Shows the street number in address field
Street	Shows the street number in address field
Side	Shows the relative side of street (Right/Left) in address field
City	Shows the city in address field
County	Shows the county in address field
State	Shows the state in address field
Zipcode	Shows the zip code in address field
Country	Shows the country in address field
Time zone	Shows the timezone based on location of accident
Airport Code	Denotes an airport-weather station which is the closest one to the location of accident
Weather Stamp	Shows the Time-stamp of weather observation of the accident record

Temperature (F)	Shows the temperature in Fahrenheit
Wind_Chill(F)	Shows the wind chill in Fahrenheit
Humidity(%)	Shows the humidity in percentage
Pressure(in)	Shows the pressure in inches
Visibility(mi)	Shows the visibility in miles
Wind_Direction	Shows the wind direction
Wind_Speed(mph)	Shows the wind speed in mph
Precipitation(in)	Shows the precipitation in inches
Weather_Condition	Shows the weather condition (Rain,snow,fog,...etc)
Amenity	A POI annotation which indicates the presence of an amenity in a nearby location.
Bump	A POI annotation which indicates the presence of a bump in a nearby location
Crossing	A POI annotation which indicates the presence of a crossing in a nearby location
Give Way	A POI annotation which indicates the presence of a give way in a nearby location
Junction	A POI annotation which indicates the presence of a junction in a nearby location
No Exit	A POI annotation which indicates the presence of a No Exit in a nearby location
Railway	A POI annotation which indicates the presence of a railway in a nearby location
Roundabout	A POI annotation which indicates the presence of a roundabout in a nearby location
Station	A POI annotation which indicates the presence of a station in a nearby location
Stop	A POI annotation which indicates the presence of a stop in a nearby location
Traffic Claiming	A POI annotation which indicates the presence of a traffic claiming in a nearby location
Traffic Signal	A POI annotation which indicates the presence of a traffic signal in a nearby location
Turning Loop	A POI annotation which indicates the presence of a turning loop in a nearby location
Sunrise Sunset	Shows a period of day based on Sunrise/Sunset
Civil Twilight	Shows a period of day based on civil twilight
Nautical Twilight	Shows a period of day based on nautical twilight

Astronomical Twilight	Shows a period of day based on astronomical twilight

Appendix B: Imputation Results for Prior Year Dataset

Year 2018: 318,340 records

Feature	Missing Value (MV) Count	# of MV after 1st Imputation	# of MVs after 2nd Imputation
Temperature(F)	7,832	7,832	1,204
Wind_Chill(F)	254,905	239,833	126,093
Humidity(%)	12,961	6,204	1,205
Pressure(in)	7,191	5,612	1,198
Visibility(mi)	9,315	7,464	1,213
Wind_Speed(mph)	60,459	32,193	1,641
Precipitation(in)	271,365	241,650	104,913
Total	**624,028**	**540,788**	**237,467**

Year 2017: 255,230 records

Feature	Missing Value (MV) Count	# of MV after 1st Imputation	# of MVs after 2nd Imputation
Temperature(F)	5,696	3,882	236
Wind_Chill(F)	213,692	203,573	108,128
Humidity(%)	5,872	3,991	237
Pressure(in)	5,135	3,543	232
Visibility(mi)	7,728	5,411	248
Wind_Speed(mph)	49,702	26,078	793
Precipitation(in)	222,591	200,953	88,274
Total	**510,416**	**447,431**	**198,148**

Year 2016: 410,600 records

Feature	Missing Value (MV) Count	# of MV after 1st Imputation	# of MVs after 2nd Imputation
Temperature(F)	2,616	1,964	135
Wind_Chill(F)	126,303	122,909	57,314
Humidity(%)	2,768	2,064	136
Pressure(in)	2,073	1,625	127
Visibility(mi)	3,613	2,950	147
Wind_Speed(mph)	28,178	14,324	677
Precipitation(in)	127,368	118,552	45,128
Total	**292,919**	**264,388**	**103,664**