**Can we predict if a person will be killed by a train?**

# Background

In the year 2017, accidents was the 4th leading cause of death in the United States. Out of these 136,053 deaths, 889 were caused from railroad accidents. Yes, a seemingly small number, but certainly 889 too high. We are living in the 21st century, but yet people are still getting killed by trains.

Our goal, is to use the dataset provided by the Federal Railroad Administration, and predict if a person will be killed in a train-related accident. There were 9,385 reported railroad accidents in 2017, and 889 people were killed.

# Dataset

Our dataset was generously provided to us by the Federal Railroad Administration’s Office of Safety Analysis. In terms of answering our question, the dataset is quite rich with a total of 75,485 entries and 50 different variables. The dataset is from 2010 to 2017.

The dataset is not very clean with missing values and containing extraneous columns. The process for cleaning our data is as follows.

1. We got rid of the following columns because they didn’t contain any values:
   1. Old Casual Occurrence Code
   2. Old Equipment movement Indicator
   3. Dummy
   4. Dummy1
   5. Dummy2
   6. Dummy3
2. We removed the following columns because they would not add much to our analysis:
   1. Narrative1

Although these narrations would have provided context and would have been interesting to mine the sentiment, most of the entries had no narrations provided.

* 1. Narrative2
  2. Narrative3
  3. Narrative Length
  4. Accident/Incident Number

1. The following columns had missing values:
   1. Job Code
   2. Nature of Injury
   3. Location of Injury on Body
   4. Indicator of Death within a Year
   5. Number of Positive Alcohol Tests
   6. Number of Positive Drug Tests
   7. Employee Termination or Transfer?
   8. Covered Data (A, R, or P)
   9. Age of Person Reported
   10. Railroad Class
   11. Hazmat Exposure?
2. We removed the following columns because they were providing the same information as other columns.
   1. Year of Incident
   2. FIPS & County Code
   3. County Code
3. We removed the following columns due to exceptions in how we would process the data.
   1. Longitude & Latitude
      1. We wanted to treat these value pairs as categorical data, but the encoder function we would use did not allow for negative values.
      2. Since we already had the state and county of the accident, we felt as though it was not necessary to include the specific location of the accident. Also, more than 40% of the entries were missing the specific coordinate location anyways.

Since most of our data is categorical due to a nature of their values, we will be imputing the missing values by assigning a new categorical value of missing to them.

1. Imputing Values
   1. Job Code
      1. Assigned new categorical value
   2. Location of Injury on Body
      1. Assigned new categorical value
   3. Indicator of Death within a year
      1. Assigned new categorical value
   4. Number of Positive Alcohol Tests
      1. Assigned new categorical value
   5. Number of positive drug tests
      1. Assigned new categorical value
   6. Employee Termination or Transfer?
      1. Assigned new categorical value
   7. Covered Data
      1. Assigned new categorical value
   8. Age of person reported
      1. Assigned new categorical value
   9. Railroad Class
      1. Assigned new categorical value
   10. Hazmat exposure?
       1. Assigned new categorical value
   11. Employee termination or transfer?
       1. Assigned new categorical value
   12. Covered Data
       1. Assigned new categorical value
2. Encoding of the categorical features

We decided that we would treat all of our features as categorical features. Any missing values would be given the value of NaN and every variable with missing values will have a NaN categorical value.

We used the get\_dummies function from the Scikit learn package to encode our dataset.

The resulted in a dataframe with 75,485 entries and 3,488 features.

## Outlier Detection

Before we could perform PCA on the data to reduce the dimensionality of it, we have to detect and remove outliers from the data. This is because Principal Components Analysis is sensitive to outliers.

Because we consider our data to be high dimensional, there were two methods that we compared: Isolation Forests and Local Outlier Factors.

However, the problem we faced was a lack of RAM. We were unable to use Isolation Forests and Local Outlier Factors without running into this problem. Even after turning the dataset into a sparse dataframe, we were still running into the same problem.

So we had to think about the question that we wanted to answer. Could we predict if a person would be killed in a railroad-related accident?

Keeping that question in mind, we were able to reduce the number of categorical values to the following:

* Month of Incident
* Railroad Reporting
* Type of Person
* Job Code
* Age of Person Reported
* FIPS State Code
* Railroad Class
* FRA Designated Region
* Form F6180-54 Filled?
* Form F6180-57 Filled?
* Day of Incident
* Year of Incident – 4 Digits
* Hour of Incident
* Minute of Incident
* AM or PM Indicator
* County
* Number of Positive Alcohol Tests
* Number of Positive Drug Tests
* Physical Act Circumstance Code
* General Location of Person at Time of Injury
* On-track Equipment Involved
* Specific Location of Person at Time of Injury
* Event Code
* Cause Code
* Hazmat Exposure?
* Fatality?

This reduced the resulting dataframe from 3,488 features to 2,797 features. Again, we run into the memory error.

In order to reduce the number of features, we had to determine which features we could avoid creating dummies for.

* Age of Person Reported
* Day of Incident
* Year of Incident – 4 Digits
* Hour of Incident
* Minute of Incident
* Month of Incident

Since time can be considered both a categorical or quantitative value, we decided to treat it as a quantitative value. This number of features drops to 2,580 but we still encounter a memory error.

Thus, we will have to perform feature selection before using our outlier detection methods.

### Feature Selection

Before we choose which features we wanted, we had to investigate why our initial number of 50 variables would balloon to such a significantly larger amount. Thus, we counted the number of unique values for each column.

|  |  |
| --- | --- |
| County | 1460 |
| Railroad Reporting | 562 |
| Job Code | 119 |
| Age of Person Reported | 99 |
| Physical Act Circumstance Code | 95 |
| Event Code | 82 |
| Minute of Incident | 60 |
| Cause Code | 54 |
| FIPS State Code | 51 |
| Specific Location of Person at Time of Injury | 50 |
| On-track Equipment Involved | 39 |
| Day of Incident | 31 |
| General Location of Person at Time of Injury | 23 |
| hour of Incident | 12 |
| Hour of Incident | 12 |
| Type of Person | 10 |
| FRA Designated Region | 8 |
| Year of Incident - 4 Digits | 8 |
| Railroad Class | 7 |
| Number of Positive Drug Tests | 5 |
| Number of Positive Alcohol Tests | 4 |
| Hazmat Exposure? | 3 |
| Form F6180-54 Filled? | 2 |
| Form F6180-57 Filled? | 2 |
| AM or PM Indicator | 2 |
| Accident? | 1 |

We see the county variable, with 1460 unique values, was the reason for the explosion in dimensionality. Since we do have a FRA Designated Region, we decided to forgo the granularity that the county variable region would provide and drop it. This would drop the dummified number of features to 1,120. But yet again, another memory error.

We can trim the dataset some more by dropping the Railroad Reporting since we have the Railroad Class variable. Again, we will be making a trade off in the granularity of the data. The number of dummified features drops to 558. Same error again.

We will also drop the job code variable. This variable describes the specific type of employee involved in the accident. But we have the Type of Person feature in our dataset. This type of person variable would indicate if a the person involved was an employee or non-employee. The # of features drops to 440. Finally, we don’t get an out of memory error.

### Outlier Detection Results

After finally reducing the number of features to a manageable result, we were able to apply the isolation forests and local outlier factor methods to detect outliers.

|  |  |
| --- | --- |
| **Outlier Detection Method** | **# Of Outliers** |
| Local Outlier Factor | 7550 |
| Isolation Forests | 7550 |

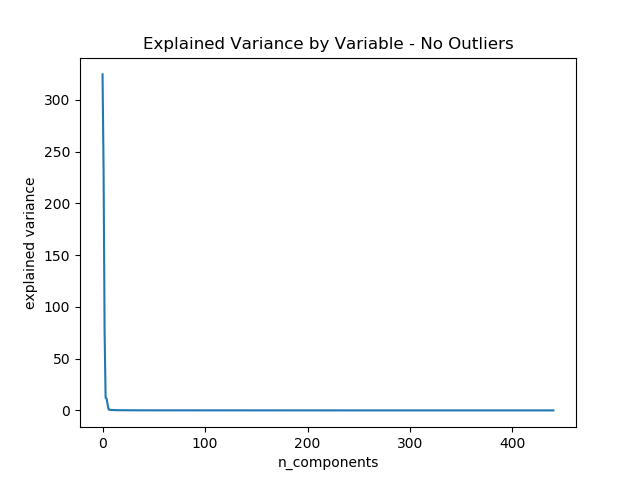
We see that the two methods agree upon the number of outliers. However, upon further investigation, only 1314 data points are flagged as outliers by both methods. Since both methods agree upon the number of outliers, but not quite which points should be outliers, we decided that we would just pick one method. In this case, we picked the isolation forests method. If the # of outliers differed, then we would have just picked the data points that both methods agreed were outliers.

## Dimensionality Reduction

### PCA

Now that we have removed the outliers from our data, we can finally perform our PCA analysis.

|  |  |
| --- | --- |
| **Variable** | **Explained Variance Ratio** |
| Month of Incident | 48% |
| Age of Person Reported | 35% |
| Day of Incident | 11% |
| Year of Incident - 4 Digits | 2% |
| Hour of Incident | 2% |
| Minute of Incident | 1% |
| Type of Person\_A | 0% |
| Type of Person\_B | 0% |
| Type of Person\_C | 0% |
| Type of Person\_D | 0% |
| Type of Person\_E | 0% |
| Type of Person\_F | 0% |
| Type of Person\_G | 0% |
| Type of Person\_H | 0% |
| Type of Person\_I | 0% |
| Type of Person\_J | 0% |
| FIPS State Code\_1 | 0% |
| FIPS State Code\_10 | 0% |
| FIPS State Code\_11 | 0% |
| FIPS State Code\_12 | 0% |
| FIPS State Code\_13 | 0% |
| FIPS State Code\_15 | 0% |
| FIPS State Code\_16 | 0% |
| FIPS State Code\_17 | 0% |
| FIPS State Code\_18 | 0% |
| FIPS State Code\_19 | 0% |
| FIPS State Code\_2 | 0% |
| FIPS State Code\_20 | 0% |
| FIPS State Code\_21 | 0% |
| FIPS State Code\_22 | 0% |
|  | ... |
| Cause Code\_44 | 0% |
| Cause Code\_45 | 0% |
| Cause Code\_46 | 0% |
| Cause Code\_47 | 0% |
| Cause Code\_48 | 0% |
| Cause Code\_49 | 0% |
| Cause Code\_5 | 0% |
| Cause Code\_50 | 0% |
| Cause Code\_59 | 0% |
| Cause Code\_6 | 0% |
| Cause Code\_7 | 0% |
| Cause Code\_8 | 0% |
| Cause Code\_9 | 0% |
| Cause Code\_99 | 0% |
| Cause Code\_R1 | 0% |
| Cause Code\_R2 | 0% |
| Cause Code\_R3 | 0% |
| Cause Code\_R4 | 0% |
| Cause Code\_R6 | 0% |
| Cause Code\_R8 | 0% |
| Cause Code\_U1 | 0% |
| Cause Code\_U2 | 0% |
| Cause Code\_U3 | 0% |
| Cause Code\_U4 | 0% |
| Cause Code\_U6 | 0% |
| Cause Code\_U7 | 0% |
| Cause Code\_U8 | 0% |
| Hazmat Exposure?\_N | 0% |
| Hazmat Exposure?\_Y | 0% |
| Accident | 0% |



The top 6 variables explain 99% of the variability in the data. So we slim our data from 441 features to 6.

### Manifold Learning

Unfortunately, PCA is a linear method and we’re not sure if this is the best way to reduce the dimensionality of the data. However, we were not able to perform any manifold learning techniques, due to the out of memory error. But this would be an interesting topic to explore in the future.

## Unbalanced Classes

Whenever we do classification, we need to make sure that our classes are not too unbalanced. In our situation, 5962 entries out of 75,486 resulted in a fatality. If our algorithms naively predicted that every event would not result in a death, then it would have a 92% accuracy! Clearly, we would have to pick an appropriate performance metric and we would have balance the classes.

We decided to attack this problem from two directions. One direction would be use machine learning algorithms that are resistant to imbalanced datasets. For example, we would use things like bagging or boosting. The other direction would be to balance out the data itself by using oversampling.

### Random Over-Sampling

Given the number of entries that we had (75,486), we were not comfortable under-sampling the data. We didn’t feel that we had enough data to discard potentially useful information.

However, we were aware of the fact that over-sampling could lead to over-sampling. That is why we decided to choose specific algorithms that could alleviate that issue.

Before we oversample our data, it’s important to note the timing of the application of this technique. If we oversample too early in the process, then we will think that a model will generalize better than it actually does. Thus, the proper way is to split our data into test & training datasets, and then oversample only our training dataset. We must remember that the point of model validation is to estimate how the model will generalize to new data. It’s critical that oversampling is done correctly.­‑

# Machine Learning

Our target variable is the fatality column and we are going to be predicting if an accident results in a death.

We used the following classification techniques:

1. Logistic Regression
2. Decision Trees
3. Gradient Boosting Classifier
4. Random Forest Classifier

With no tweaking of the hyperparameters and using Stratified 10-fold cross validation, we were able to achieve the following scores.

|  |  |
| --- | --- |
| **Method** | **Average Score** |
| Logistic Regression | 91.79% |
| Decision Tree | 83.07% |
| Gradient Boosting Classifier | 91.75% |
| Random Forest Classifier | 91.10% |

## Interpretation

Our best method using the default settings was the Logistic Regression with 91.79% of the test data correctly labeled.

The following are the coefficients of our Logistic Regression Model.

|  |  |
| --- | --- |
| **Logistic Regression** | |
| **Variable** | **Coefficient** |
| Month of Incident | 0.0138 |
| Age of Person Reported | -0.0178 |
| Day of Incident | 0.0033 |
| Year of Incident - 4 Digits | -0.0010 |
| Hour of Incident | -0.0212 |
| Minute of Incident | 0.0152 |

So we would interpret the coefficients as the following:

Pretend that we only have our month of the incident as January. Then the probability that a person would be killed by a train would be 1.3% and for every month after January, the probability would increase by 1.3%. The rest of the variables are similarly intuitive.

The only non-intuitive variable is the year of incident. The starting point for that variable is 2010. So every year after 2010 decreases the probability by .1%.

## Hyperparameter Tuning