Machine Learning with Accelerometer and GPS Data
Deliverable #2

1. **Exploratory Data Analysis**
   1.1. Findings:
   We found that when acceleration is measured in G-Force, it actually makes more sense when it comes to data visualization. For example, when the vehicle stops or is moving constantly on a flat road, the vertical acceleration read approximately -1 G-Force (normal reading). The normal reading should show relatively small fluctuations, the magnitude of which depends on the IRI of the current location. Then, if the vehicle hits some road hazards (i.e. potholes, speed bumps, etc.), the vertical acceleration will show a big anomaly reading.

   1.2. Recommendations:
   In the spirit of "make it work, then make it better", we plan to start very simple with a model using an average and standard deviation to find potholes and speedbumps. We then plan to build a logistic regression model for potholes and speed bumps. If time permits, we will evolve to recurrent neural networks (RNN). We haven't done any exploratory data analysis on the cases of curvatures or inclination. However, we included our educated guess for the approaches to these two cases in the *Kyrgyzstan Exploratory Analysis*.

   1.3. EDA Scripts and Documents: [LINK]
      1.3.1. *Kyrgyzstan Exploratory Analysis*
         1.3.1.1. kyrgyzstan.csv: data file
         1.3.1.2. kyrgyzstan.Rmd: R script
         1.3.1.3. kyrgyzstan.pdf: final document
      1.3.2. *Los Angeles Exploratory Analysis*
         1.3.2.1. los_angeles_5.csv: data file
         1.3.2.2. los_angeles_5.Rmd: R script
         1.3.2.3. los_angeles_5.pdf: final document
2. **Logistic Regression Model Specification**
   2.1. Measurements of Accuracy:
      2.1.1. F1 score (harmonic mean of precision and recall)

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

2.1.1.1. Precision (Positive Predictive Rate)

$$PPV = TP/(TP + FP)$$

2.1.1.2. Recall (True Positive Rate)

$$TPR = TP/P = TP/(TP + FN)$$

2.2. Machine Learning Setup
- 2.2.1. Dependencies:
  - 2.2.1.1. Numpy: matrix math
  - 2.2.1.2. Pandas: data manipulation
  - 2.2.1.3. Patsy: matrix data structure
  - 2.2.1.4. Warning: error logging
- 2.2.2. sigmoid() function: logistic regression curve
- 2.2.3. Set a seed: reproducibility (good for debugging)
- 2.2.4. Hyperparameter tuning
  - 2.2.4.1. Convergence tolerance: the minimum threshold between predicted and actual outputs
  - 2.2.4.2. Maximum allowed iteration
- 2.2.5. Regularization: L1 vs. L2 (we will try both of them)
- 2.2.6. Data creation setting (if to generate random data points)
  - 2.2.6.1. Covariance
  - 2.2.6.2. Number of data points
  - 2.2.6.3. Variance of noise: how spread out the data is
- 2.2.7. Model setting (if to generate random data points)
  - 2.2.7.1. True beta coefficients
  - 2.2.7.2. Variance of inputs
  - 2.2.7.3. Model specification
- 2.2.8. Relevant factors:
  - 2.2.8.1. Speed
  - 2.2.8.2. Vertical acceleration
  - 2.2.8.3. Forward acceleration
- 2.2.9. Training set & testing set
  - 2.2.9.1. Training: 80%
  - 2.2.9.2. Testing: 20%

2.3. Regularization: to prevent overfitting
- 2.3.1. L1 regularization:
  - 2.3.1.1. Computationally inefficient on non-sparse cases

2.3.1.2. Sparse outputs
2.3.1.3. Built-in feature selection
2.3.2. L2 regularization:
2.3.2.1. Computationally efficient due to analytical solutions
2.3.2.2. Non-sparse outputs
2.3.2.3. No feature selection

2.4. Cross-Validation
2.4.1. Use K-fold cross-validation
2.4.2. Examples: