- **Solutions & Results**
    - **Solution 1:**
        - **idea:**
            - The assumption here is that labels are mutually exclusive. Therefore, this multi-label classification problem can be decomposed into multiple independent binary classification problems. Based on “One-vs-Rest” strategy, we build 5 independent binary classifier using logistic regression.
        - **data preprocessing**
            - Both damages and features are encoded with one-hot encoding. The damage example for one visit is a 5-element vector, with 1 indicating the apperance of such damage type and 0 for no apperance. The feature example is in the same style but with 500-element length.
            - After checking the distribution of all data, I select the first 9,000 visits as training set and the last 1,000 visits as test set.
            - The input to the classifiers is the featuers and the ground truth is the damages.
        - **performance**
            - top 1 precision: 86.0%
            - exact matching accracy: 39.5%

    - **Solution 2:**
        - **idea:**
            - Build a neural network to solve this multi-label classification problem. The network has one one hidden layer with 100/200 neurons and it's a fully-connected layer. The input layer has 500 neurons to accept the features. The output layer consists of 5 sigmoid neurons correponding to the damage types.
        - **data preprocessing**
            - Same as solution 1

        - **performance**
            - 50-neuron version
                - top 1 precision: 85.6%
                - exact matching accracy: 40.7%
            - 100-neuron version
                - top 1 precision: 85.9%
                - exact matching accracy: 40.1%
            - 200-neuron version
                - top 1 precision: 86.4%
                - exact matching accracy: 39.6%

- **Project structure**

--root

-- *data* : to contain the data files

-- *logs* : to store the TensorBoard logs

-- *model*: to store the trained weights for neural network model

-- *uitls* :to store the funcational scripts

-- *merge_csv_files.py*: to merge two csv files into one where each visit is one row and the type of damges and features is encoded with numbers

-- *EDA.py*: to perform a Exploratory Data Analysis,e.g. plotting the distribution of damages and features

-- *metrics.py*: to define two funcations to calculate top 1 precision and exact matching accuracy

-- *parser.py*: to parse the merged csv files, perfrom one-hot-encoding, and return the damages and features as numpy arrays. It also save this two arrays in a pickle files for easy access.

-- *neural_network_evaluation.py*: to help evaluate the neural network models independently with the saved weights.

--*neural_network_solution.py*: to realize the neural network solution. The main part is training but evalution also can be enabled directly after training

-- *one_vs_all_solution.py* to realize and evaluate the one vs all solution.

- **Steps to to walk through the project**
    - **install packages**
        - python 3.6
        - keras
        - tensorflow
        - numpy
        - pandas
        - seaborn
        - matplotlib
        - pickle

    - **data processing**
        1. run *merge_csv_files.py* to generate the ./data/all_data.csv
        2. run *parser.py* to encode the data and save the it into ./data/all_data.pickel
        3. run *EDA.py* to check the distributions

    - **check solutions**
        - run *one_vs_all_solution.py* to check the performance of solution 1
        - run *neural_network_solution.py* to train the neural network model and check the performance if the evaluation is enabled.
        - run *neural_network_evaluation.py* to check the performance offline.