Machine Learning Engineer Nanodegree

Capstone Proposal

Proposal

Chieko Natori

November 5rd, 2018

Domain Background

Information and Communications Technology (ICT), including the Internet and smartphones, is playing an important role in our modern society. Most of our daily activities are supported by the swift and diverse communication through the networks. Therefore, reliable network management is crucial. When the network suffers any disruption or fault, the company providing the network access needs to detect the detail of it as soon as possible, judge the urgency based on the level of the impact on users, and solve the problem accordingly.

In the past, fault management used to require human involvement. However, it became troublesome as our communication environment got more complex and extensive after the internet and mobile phones gained popularity in the 1990s. Since then, researchers have been studying how to automate fault management using advanced technology such as Artificial Intelligence (Gurer, Khan, Ogier, and Keffer, 1996[1]). Today, networks are growing even larger and more complicated than ever, and companies are continuously seeking a way to make use of the latest technology to achieve better network management.

Problem Statement

As I discussed above, solving network faults instantly is one of the top priorities for telecommunication companies to meet customers' expectation. Because of the recent evolution of big data and machine learning techniques, however, companies also started to think of predicting future faults using data from their networks. If they can foresee faults and disruptions, they will be able to take a proactive action before the incident actually affects customers[2]. Telstra, Australia's largest telecommunications company, wants to predict when and where the disruption is likely to happen, and whether it is a momentary fault or a serious disruption that loses connectivity. This can be achieved by building a predictive model that employs machine learning techniques. In this problem, the model takes in a certain location and time set and outputs a predicted class of fault severity. That is, this is a multi-class classification problem.

Dataset and Inputs

I obtained a dataset for this project from Kaggle's past competition "Telstra Network Disruptions" [3].
To build a predictive model, I use 5 data files shown below:

- `train.csv`: the main dataset for fault severity with respect to a location and time
- `event_type.csv`: event type number related to the main dataset
- `log_feature.csv`: feature number and volume extracted from log files
- `resource_type.csv`: type of resource related to the main dataset
- `severity_type.csv`: severity type of a warning message coming from the log

All files are recorded in CSV format. In "train.csv", each row represents a location and a time point. It consists of "id", "location", and "fault_severity", and "id" is unique to the row. "fault_severity" (0, 1, or 2) is a measurement of reported faults, where 0 means no fault, 1 means only a few, and 2 means many. There are

7381 data points in this file. Approximately, 65% of the dataset belong to fault severity 0, 25% to fault severity 1, and 10% to fault severity 2, which means the class distribution is imbalanced. There is no missing data observed.

Other 4 files contain additional information related to “id”. The information was extracted from Telstra’s log files and other sources. In these files, there are more rows than “train_csv”. This is because there are multiple rows that share the same “id”, and also because there are “id”s that are not present in “train.csv”. There are rows that have an extra field which is not specified in the header.

I have one more file called “test.csv”, which has the same columns as “train.csv” except it does not have “fault_severity”. I do not use this file to train and test my model, but could use to see the final performance at the end of the project.

Solution Statement

As a solution, I propose a supervised learning model that predicts fault severity at a location and a time given. To train my model, I use "fault_severity" in "train.csv" as a label, and information in other files as features for a specific location and time. "fault_severity" and other information are connected by "id". This is a multi-class classification task that has three categories. My candidates of algorithms to solve this problem are Decision tree, SVMs, Nearest Neighbor, and ensemble methods such as Random Forests, XGBoost, and LightGMB. Neural networks can be a possible choice, too. However, the final decision will be made after data exploration.

Benchmark Model

As a benchmark model, I use Random Forest model. Random Forest creates multiple decision trees, and combines them into one strong model which is better than any single tree. I simply train this model without tuning, which I assume is able to perform the classification to some extent. Then I aim to build my final model so that it outweighs the Random Forest model by choosing a better algorithm and tuning parameters.

Evaluation Metrics

I plan to evaluate my model using multi-class log loss [4]. Having a quick look at the data, I noticed that around 65% of data points in “train.csv” are fault severity 0, whereas there are only 10% of samples labeled as fault severity 2. This is reasonable because being without a fault is a normal status for the network. Although one of the common metrics for classification is accuracy, it is inappropriate to use it in this case because it would still obtain 65% accuracy if the model predicted all samples as fault severity 0. I suppose detecting a fault in higher severity is rather important in this problem.

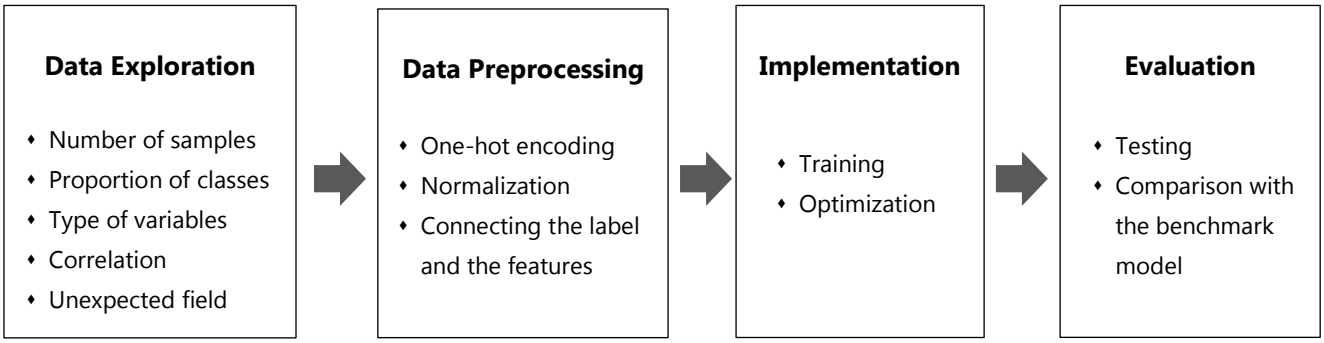
While accuracy only counts the number of samples that are correctly classified, multi-class log loss takes into account the predicted probability of how much likely the data point belongs to the category. Multi-class log loss works in multi-class classification problems, especially when the number of samples is imbalanced between classes like this case. Multi-class log loss is defined as below:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

where N is the number of data points, M is the number of fault severity classes, log is the natural logarithm, y_{ij} is 1 if observation i belongs to class j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to class j(This formula is taken from Kaggle's competition page [3]). Smaller log loss indicates that the model is performing better.

Project Design

The workflow of this project is shown below:



1. Data Exploration

Firstly, I check the precise number of samples and the proportion of each fault severity in the main dataset. I also inspect other files that contain features, and clarify whether it is a numerical data or categorical data for each variable. Then I see roughly whether or not there is any correlation between fault severity and other features, and observe correlations among features as well. I might detect outliers if applicable. There is no missing data, but some rows with an unexpected field are found in feature files. I need to decide how to treat them, whether to ignore the filed or use it to analyze the data, from the relationship among features.

2. Data Preprocessing

Next, I perform one-hot encoding for categorical variables. For a numerical variable, normalization can be necessary depending on the distribution of the data. I also deal with the unexpected columns in feature files by the way I decided in Data Exploration. Then I combine the main dataset and other features by using "id" as a key, and make one table internally. Each row represents a location and time, which contains its fault severity and corresponding features. At this point, I split the data into training data and testing data.

3. Implementation

In this step, I implement my supervised learning model using data I prepared in the previous step. My potential choice of the algorithm is Decision tree, SVMs, Nearest Neighbor, Random Forest, XGBoost, or LightGBM. Neural networks are also possible. It is necessary to tune parameters for the algorithm to optimize the model.

4. Evaluation

Finally, I evaluate my model with testing data I prepare in Data Preprocessing. Its performance is compared to the benchmark model. The evaluation metric used here is multi-class log loss. I expect that the value of multi-class log loss of my final model is obviously less than that of the benchmark model.

Reference

[1] Gurer, D. W., Khan, I., Ogier, R. and Keffer, R. (1996) *An Artificial Intelligence Approach to Network Fault Management.* from https://www.researchgate.net/publication/2731474_An_Artificial_Intelligence_Approach_to_Network_Fault_Management

[2] Pearc, R. (2018, May 15) *NBN using machine learning to proactively tackle network faults.* from https://www.computerworld.com.au/article/641183/nbn-using-machine-learning-proactively-tackle-network-faults/

[3] Kaggle (2016) *Telstra Network Disruptions.* from https://www.kaggle.com/c/telstra-recruiting-network

[4] scikit-learn *3.3. Model evaluation.*from https://www.kaggle.com/c/telstra-recruiting-network#evaluation