CS136 Final Report

Pengcheng Xu

Dexter Wu

Abstract

> In this experiment, we used the Algerian Forest Fires Dataset to measure the binary classfication performance using regression model. Specifically, we used the logistic regression model function as the baseline model for comparison. Furthermore, we applied feature rescaling, L-1 regularization, and feature transformation methodologies as our upgrade methods. The dataset that we used can be found here: https://archive.ics.uci.edu/ml/datasets/Algerian+Forest+Fires+Dataset++.

Sec. 1: Introduction

The wild fire has always been one of the catastrophic disasters to human society, the impact including endangering people and animals' lives, exacerbating air pollution, shutting down communication or transporation infrastructures, and many other detrimental affects. According to the tabular dataset[2] provided by National Interagency Fire Center, from 2017 through 2021, the average total cost that the wildfire has done to North American soceity is 2,862,884,200 USD per year. Notifying the affected area prior to the occurrence of a tragedy would enable the local government to be better equipped to proactively mitigate and extinguish the fire, thereby reducing the financial burden associated with the event.

This paper's main task is to devise a model that would predict whether there would be a fire or not based on weather attributes of the Bejaia region. The paper will utilize logistic model as the baseline model. After tested the baseline model, we would apply our proposed upgrades to the basline, namely, feature rescaling, L-1 regularization, and feature transformation methodologies.

The logistic regression model works well in classifying unkown features with the benefits of easy

implementation and requering minimum computation power to train. Hence we utilize this model for the Algerian forest fires dataset since it is a relatively small dataset with less complicate feature dimensions. The propose models are also a good fit for our task since the output label is binary(0 or 1) indicating fire or not fire. Overall, the fact that multiple predictors and binary respond labels naturally fit to the GML fields.

Sec. 2: Data and Analysis Plan

The original dataset can be distributed into two sub-dataset that including the weather statistics of two different region, namely Bejaia and Sidi-Bel Abbes. After dropping the missing values rows and matching data types, each dataset has 244 number of data entry, where each entry includes 14 features. The table below contains the specific data type for the feature attributes. The specification for each data observation has also been recorded from the table.

Feature	Specification(data type)
Date	(DD/MM/YYYY)(int64)
Temp	temperature noon (temperature max) in Celsius degrees: 22 to 42(int64)
RH	Relative Humidity in %: 21 to 90(int64)
Ws	Wind speed in km/h: 6 to 29(int64)
Rain	total day in mm: 0 to 16.8(float64)
Fine Fuel Moisture Code(FFMC)	index from the FWI system: 28.6 to 92.5(float64)
Duff Moisture Code(DMC)	index from the FWI system: 1.1 to 65.9(float64)
Drought Code(DC)	index from the FWI system: 7 to 220.4(float64)
Initial Spread Index(ISI)	index from the FWI system: 0 to 18.5(float64)
Buildup Index(BUI)	index from the FWI system: 1.1 to 68(float64)
Fire Weather Index(FWI)	index from 0 to 31.1(float64)
Classes	Fire, Not Fire(int)

After examining the raw dataset, we decided to the following processing:

- Treat the last column (i.e. "Classes") as the output label (i.e. binary 0 or 1),
- Drop the data columns (i.e. "year", "month", and "day"), since the information they contained could also be embodied in other features (e.g. a chilly day in November could also be represented by other features like "Temp", "Rain", etc)

After doing these, we have 10 input features, and 1 output label.

The performance metric we're using for this project is average predicatioin accruate rate, which is defined as follow:

$$avg\_acc = \frac{\#\ of\ correct\ predictions}{\#\ of\ predictions}$$

We think this is appropriate for our project, because we would like our model to perfrom (i.e. predict fires) as accurate as possible.

Alongside the process (Baseline model, Upgrade 1, and Upgrade 2), we also use K-fold to do the cross-validation and spliting train/test data.

Sec. 3: Baseline Method

Since this is a typical classification problem, and a "S-shaped" curve would probably fit into it based on our common sense and intuition (e.g. low temperature normally wouldn't cause fire, while high temperature more likely, and there's probably a threshold somewhere in the middle), we'd choose "Logistic Regression" as our base model.

Next, we're going to talk about some mathmatical aspects of our baseline mode.

- **Model Assumption**

  > Logistic Regression assume a "S-shaped" curve would fit the data

- **Random Variables**

  > Each raw observation is one data point (i.e. X), which has 10 features after dropping some trivial columns (e.g. 'Year', since all the data collected from the same year).
  > The features are: [Temperature, RH, Ws, Rain, FFMC, DMC, DC, ISI, BUI, FWI]

- **Objective Function**

  > Since we don't have any prior here, this is a ML problem.
  > In some sense, it borrows the concept from Linear regression (assume input X is one dimensional):
  >
  > $$y = w_0 + w_1 x$$
  >
  > Then adds exponential to make it S-curved (p means probability):
  >
  > $$p = \frac{1}{1 + e^{-w_0 + w_1 x}}$$
  >
  > After computing the probability of a given observation (i.e. X), we compares this probability with a threshold (usually it's 0.5), and assign it the label '1' if $p \geq threshold$ is true, 0 otherwise.
  >
  > Finally, we want to maximize our model's prediction accuracy as much as possible, so we define objetive function as follow (and want to maximize it):

> $$avg\_acc = \frac{\#\ of\ correct\ predictions}{\#\ of\ predictions}$$

- **Algorithm**

  > The algorithm we used here for Logistic Regession is "Gradient Descent". Depending on the solver we choose, it also has many variants, like "stochastic gradient descent", "stochastic average gradient descent", etc.
  >
  > The basic idea behind this is that via each iteration, we want to get a better parameter w by going to the opposite direction of parameter's gradient (supposing there's min objective function):
  >
  > $$w_i \leftarrow w_i - \frac{\partial L_i}{\partial w_i} \qquad\qquad (L\ is\ the\ loss\ function)$$

- **Implementation**

  > Since Sklearn has already provided us "Logistic Regression" Module, we don't want to re-invent the wheel, and just import this module.
  >
  > We also set the random*state to "42" for reproducible purpose, and set the "max*iter" to 2000 because the default value (i.e. 100) is not enough for convergence after some trials.

Sec. 4: Proposed Upgrade Method [~0.5-1 pages]

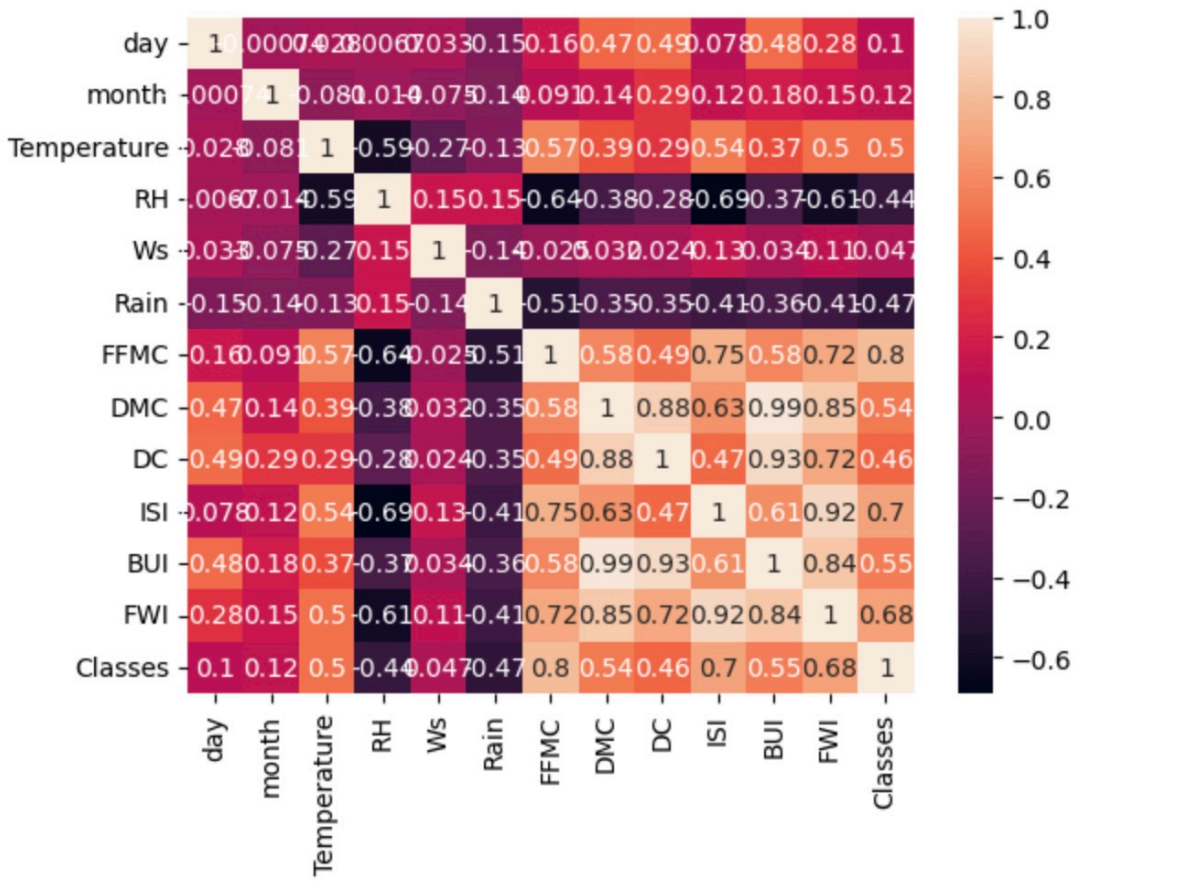
After going over our dataset, we come up with a two-stage upgrade plan:

1. Rescale and eliminate unnecessary features

- **Motivation**
  This comes from the fact that our features are in different ranges (e.g. Temp is from 22 to 42, while DC index is from 7 to 220.4, as in the figure below), which will cause unfair update to our parameters due to the different magnitude of unit measurements. Therefore, we'd like to convert them into the same range.

1. Date : (DD/MM/YYYY) Day, month ('june' to 'september'), year (2012) Weather data observations
2. Temp : temperature noon (temperature max) in Celsius degrees: 22 to 42
3. RH : Relative Humidity in %: 21 to 90
4. Ws :Wind speed in km/h: 6 to 29
5. Rain: total day in mm: 0 to 16.8 FWI Components
6. Fine Fuel Moisture Code (FFMC) index from the FWI system: 28.6 to 92.5
7. Duff Moisture Code (DMC) index from the FWI system: 1.1 to 65.9
8. Drought Code (DC) index from the FWI system: 7 to 220.4
9. Initial Spread Index (ISI) index from the FWI system: 0 to 18.5
10. Buildup Index (BUI) index from the FWI system: 1.1 to 68
11. Fire Weather Index (FWI) Index: 0 to 31.1
12. Classes: two classes, namely Fire and not Fire

The other comes from the correlation matrix of the raw data, and relatively many features (i.e. more than 10). Some of them have a high correlation value (e.g. corr between "DC" and "BUI" is 0.93, while corr between "DMC" and "BUI" is 0.99, as shown below), which means there're probably some redundant features (i.e. one feature could potentially represented by other features). Thus, we want to eliminate those redundant ones.



- **Scheme**
  For rescaling, we want to convert them into the range [0, 1], the way we do it is to use the following conversion formula:

$$scaled\_x = \frac{x - min\_x}{max\_x - min\_x}$$

  For eliminating redundance, we want to add a penalty term (i.e. "L1" penalty), which will force some coeffiecents to zero (and thus the corresponding features could be dropped since they don't contribute any to the objective function):

$$new\_obj = old\_obj + L1\_penalty$$

- **Implementation**
  For rescaling, we implement it on our own since it's relatively simple (i.e. just following the fomula listed above), even though we realize that sklearn has already provide us an implementation later on.

  For eliminating redundance, after referencing to the sklearn "Logistic Regression" doc, we figure out we could set up construtor parameters to achieve this (i.e. set up "penalty=l1",

and choose an appropriate solver that supports "L1 penalty").

2. Feature transformation

- **Motivation**
  This comes from the daily observation that when a feature is in certain range, a relative big change won't cause much difference. However, once it passes by a certain threshold, even a small change will result in a big difference.
  For example, When the temperature is very low (e.g. [-50˚C, -30˚C]), it has little chance to get fired even there's a big change (e.g. from -50˚C to -40˚C). When it's very hot (e.g. [30˚C, 40˚C]), even a small increasement in temp would make it more likely to get fired (cuz it's very dry).

- **Scheme**
  Based on our observation mentioned above, we'd like to do a feature transformation (i.e. order 2, since we believe the pattern (i.e. after a threshold, a small change make more effect) is like a polynomial pattern). The underlying math could be presented as below (suppose X_1 and X_2 are all order one):

$$X_1, X_2 \rightarrow 1, X_1, X_2, X_1X_2, X_1^2, X_2^2$$

- **Implementation**

  The order 2 feature transformation is not that hard, and we could actually implemenet by ourselves. However, after looking at the sklearn doc, we find out there's already an implementation in Sklearn (so we don't need to reinvent the wheel):
  `sklearn.preprocessing.PolynomialFeatures`

Sec. 5: Results

- **Results**

  - **Baseline model**
    After applying the baseline logistic regression and use K-fold cross validation (i.e. k is set to 5 in our case), we get the average accurate rate over training set and testing set as below (i.e average value over 5 iterations, since k = 5):

item	value
Average training accurate	0.926203
Average testing accurate	0.913127

- **Upgrade 1**
  After applying "L1" penaly to our objective function (i.e. eliminating unnecessary features), and rescaling all features into the same range (i.e. [0, 1], seeing the details in Sec. 4), redo the cross validation and get the table as below:

item	value
Average training accurate	0.951186
Average testing accurate	0.946513

- **Upgrade 2**
  After upgrade 1, we get rid of all features except two (i.e. ISI and FWI), then applying feature transformation (i.e. order two), we get 6 features. Redoing the cross validation (in order to avoid overfitting, we use L2 penalty this time), we get the table as below:

item	value
Average training accurate	0.934098
Average testing accurate	0.923127

- **Result analysis**

  From the tables above we could see, Upgrade 1 (i.e. Rescaling and Eliminating rudundance) indeed improve the model performance. This is probably because rescaling remove the potential unfair update to parameters caused by different unit magnitude (i.e. after rescaling, all features are converted to the range [0, 1]).

  The further upgrade 2 (i.e. order 2 feature transformation) doesn't improve much based on the first upgrade (but it's still a little better than the baseline model). This is probably because our dataset is relatively small (i.e. only 244 instances), only two features (i.e. ISI and FWI) left after L1 penalty. And order-2 poly transformation also brings in cross product (e.g. we may only want the terms X_1^2 and X_2^2, but feature transformation also brings in $X_1 X_2$, which we may not want and doesn't make any sense).

Whether upgrade 2 is valid or not is still to be seen, we probably need more data points to verify it.

Sec. 6: References

1. **Logistic Regression**: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
2. **Stochastic Gradient Desent**: https://scikit-learn.org/stable/modules/sgd.html
3. **L1 penalty in Logistic Regression**: https://scikit-learn.org/stable/auto*examples/linear*model/plot*logistic*1*l2*sparsity.html
4. **Polynomial features**: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html