Analysis Algerian Forest Fires with Machine Learning Techniques

Data Set : Algerian forest fires

Haolun Cheng, haolunch@usc.edu

4/29/2022

1. Abstract

Forest fire is a devastating disaster, which not only damages the ecological environment irretrievably, but also does great harm to the human survival. Since the last century, the global climate has been warming continuously, and forest fires are on the rise [1][2]. The annual forest fires have caused huge economic losses to all countries in the world, therefore, how to predict, control or reduce the harm of forest fire has become a common scientific task in many fields. Rapid detection is an effective way to predict forest fire. One way to do this is to use data collected from sensors combined with machine learning methods [3], such as those provided by weather stations. It is found that meteorological conditions (such as temperature and wind speed) are important factors affecting the occurrence of forest fires and some fire indexes. Therefore, we will explore several methods of machine learning to predict whether forest fires will occur. Using real data collected over a three month period from two different locations in Algeria, using a variety of machine learning techniques such as support vector machine [4] and logistic regression [5], and neural network methods [6], as well other two baselines methods, analysis of different features. In the experiment, we run several repeated experiments by cross-validation, compared some baseline methods, and finally found that the best results are obtained by using the support vector machine method. The prediction methods mentioned above are of great significance for forewarning the occurrence of forest fires and improving the management and deployment of fire control resources.

2. Introduction

2.1. Problem Statement and Goals

In this study, we selected data on forest fires in two regions of Algeria over a four-month period [7]. The data include training sets and test sets, where training set starts from June 1, 2012, ends to August 31,2012, and testing set lasts for September 2012. The data had 11 characteristic dimensions, with the first representing the time of the data record and the last representing

whether there was a fire. The remaining dimensions represent some information related to forest fires. The training set contains 185 pieces of data, and the test set contains 61 pieces of data. In this task, we hope to train the appropriate algorithm in the training set, and then test on the test set, to get better performance, help us to predict and prevent forest fires to make better decisions.

2.2. Literature Review

In recent years, a number of researchers have been studying the prediction of forest fires. The ARMA model is used to simulate and predict the affected area of forest fire [8]. Unlike this approach, which used forest fire data, the prediction model was based on the support vector machine's yard [9]. On this basis, ESDA technology is used to explore the spatial distribution of forest fires, and Kriging interpolation method is used to forecast the trend of forest fires [10]. In [11], they used the BP neural network model to predict the disaster area. From the research history, there have been many methods combined with machine learning technology for forest fire prediction and analysis.

3. Approach and Implementation

In this experiment, we compared five methods, Support vector machine, logistic regression and neural network, which we implemented through the scikit-learn [12] library, the trivial system and baseline method are implemented by Numpy. In this section we will introduce the principles of each method separately.

3.1. Support Vector Machine

The support vector machines (SVM) [4] is a binary classification model. The basic model is the Linear classifier defined in the feature space with the largest interval, which distinguishes it from the perceptron. The SVM also includes kernel technique which makes it non-linear classifier. The learning strategy of SVM is to maximize the interval, which can be formalized as a convex quadratic programming problem, and is equivalent to the minimization of the normalized hinge loss function. The learning algorithm of SVM is the optimal algorithm for solving convex quadratic programming.

3.2. Logistic Regression

Mathematically, a binary logic model [5] has a dependent variable with two possible val such as pass or fail, win or lose, live or die, or health or illness; these are represented by indicator variables, two of the variables marked as "0" and "1". In the logical model, the logarithmic ratio (in logarithmic probability) , used to mark the value "1" as a linear combination of one or

more independent variables ("prediction") ; An argument can be a binary variable (two classes, encoded by an indicator variable) or a continuous variable (any actual value).

3.3. Neural Network

Neural network [6] is a model that contains multiple layers of neurons, calculates the gradient of neuron parameters by the gap between outputs and correct labels, and updates these parameters by back propagation to achieve the training goal. The neuron model is a model with input, output and computation functions. Input can be likened to a neuron's dendrite, output to a neuron's axon, and computation to a cell's nucleus.

3.4. Trivial System (Nearest Mean)

In this experiment, we use the nearest mean method [13] implemented in the previous job as the trivial system. This method calculates the average distance from the data in the test set to the data in the training set to determine the data in the test set belongs to the category. In order to make the classification non-linear, some kernel functions can be used instead of the basic distance computing functions.

3.5. Baseline Method

For the baseline method of this experiment, we choose the probability of each class in the training set as our prior knowledge. For the test data, we choose one class randomly for each sample according to the prior probability, then we obtain the classification performance on the test set.

4. Datasets and Experiments

4.1. Dataset Usage

During the training, we selected the training set contained in the data set, which contains 185 pieces of data, each of which contains 10 dimensions. One of these dimensions is time, a trait we discard at the time of final training. The test set contained in the dataset also has 10 dimensions, with 61 entries. The test set will not be used until the model has been trained.

In the course of our training, we need to cross-validate the training process in order to compare the performance of each model fairly and accurately, and to find the corresponding parameter combination of the best performance for each model. We chose the scikit-learn Library's GridSearchCV function, which trains different parameter combinations for the model and cross-checks to find the best one.

Before formal training, we also need to do some processing on the original data set, such as checking and filling the data for missing values, and standardizing the data to stabilize the training, and feature enhancement of existing data to improve the performance of the model.

4.2. Preprocessing

When the variance and mean range of each feature of the data are very different, the model can not be trained stably, so the performance of the model is reduced. So before we do formal training, we tend to standardize the data, subtract the mean from the data and divide it by the standard deviation, so that the characteristics of the data are constrained to a certain extent, to enhance the numerical stability of the model.

In order to have a better understanding of the characteristics of the data and to understand the degree of correlation between the characteristics of the data and the final prediction label, we plot the correlation between the characteristics of the data, and visualized in Figure 1. We can see that there is a strong correlation between features “RH”, “Rain” and the label.

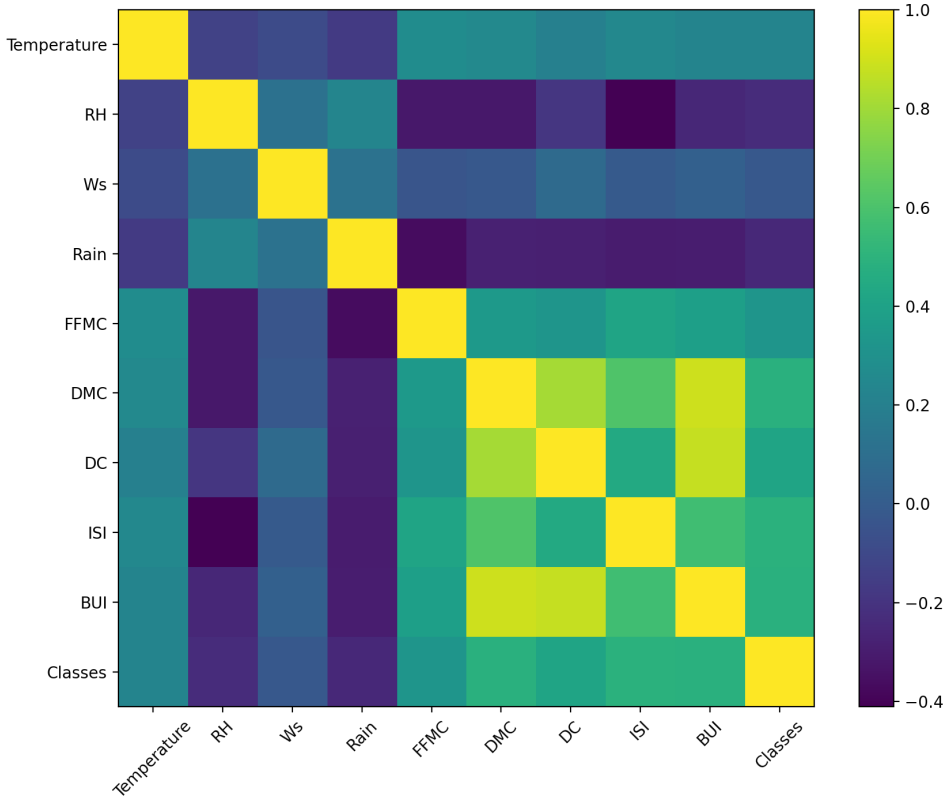
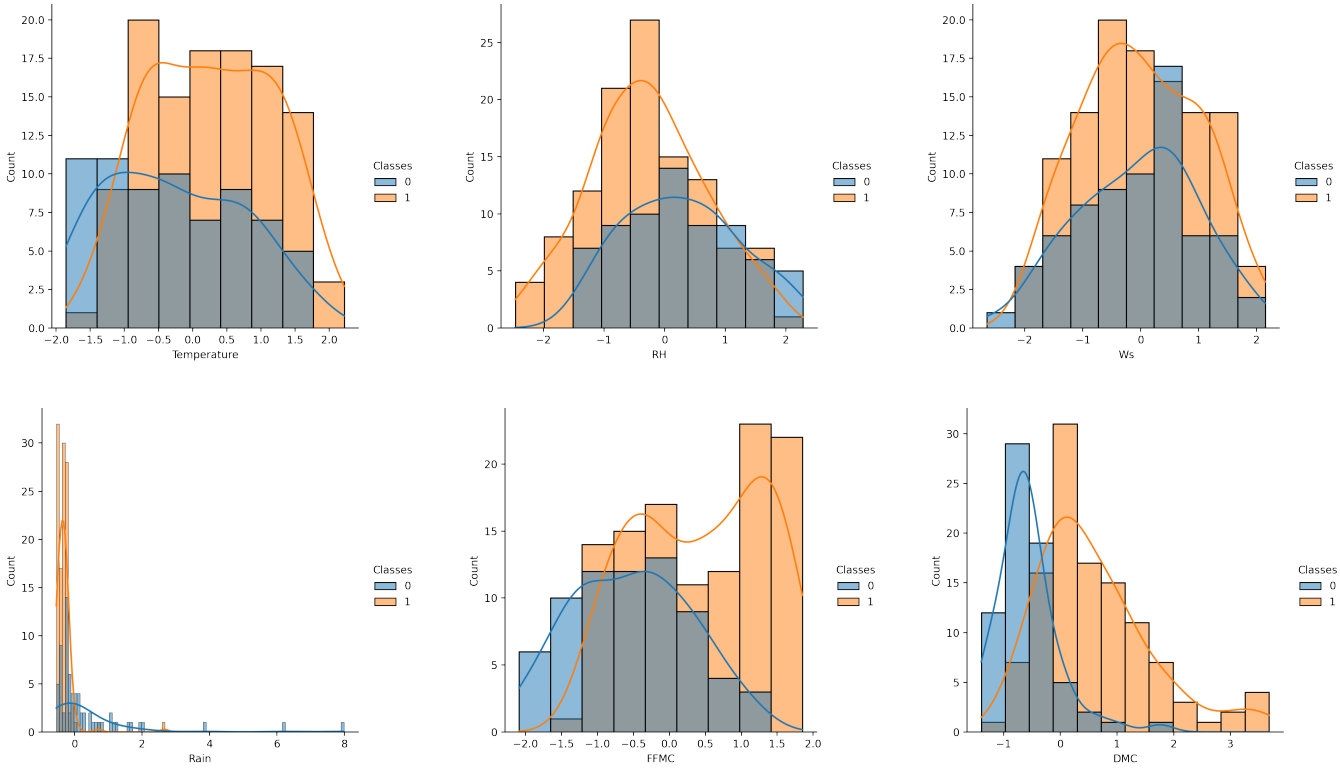


Figure 1 The heat map of correlation between the features and the target

4.3. Feature engineering

First, we visualize the feature distribution of the original data in Figure 2. We choose a histogram to look at the distribution of features under different categories. We can see that some of these features are very different in different categories, such as feature temperature and ISI. These may play a greater role in the correct classification of the model.

On the basis of the original features, we have further enhanced the features. Because our task is related to the prediction of time series, we deal with each feature by a sliding window. For the current time, we chose the average, maximum, and minimum of the feature for the first five days as additional features for the day. Since such a feature can not be acquired in the initial time, we pad them with zero in the first 5 rows. With this data enhancement, the final training set will contain 36 dimensions.



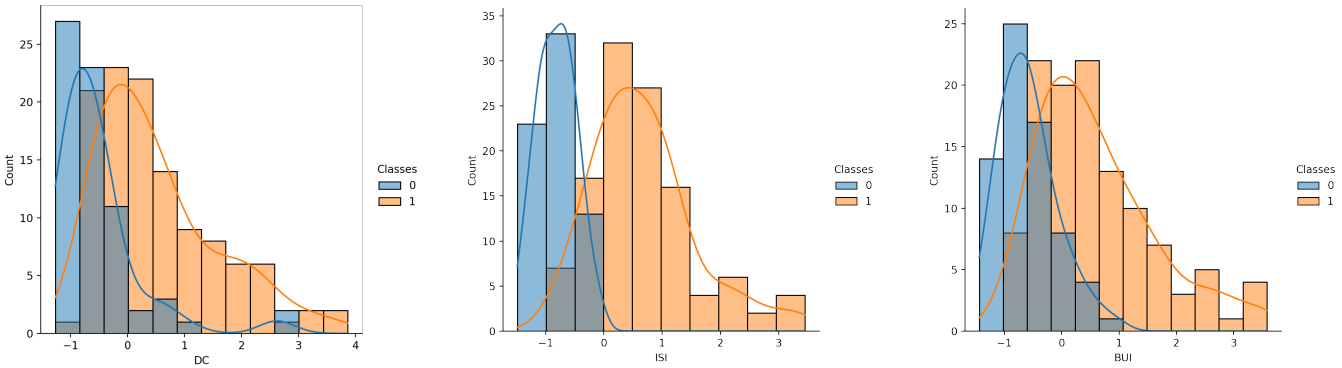


Figure 2 The distribution of different features of data via different classes

4.4. Feature dimensionality adjustment

In this experiment, we do not consider the reduction of the feature dimension of the data because we only enhance the time feature of the sliding window of the data and did not introduce redundant features. We want the final model to show which features are important, so we do not delete them in advance.

4.5. Training, classification, and model selection

In this section we will describe our specific training process. For each model, we first use the original data set for cross-validation, and select the model that performs best on the validation set from the alternative parameter combinations, the model is then trained with a full set of training data. After training, we deploy the model to the test set and get the final test performance score. Similarly, in order to compare whether our feature-enhanced dataset adds to the model's classification performance, we ran the same process over the enhanced dataset and compared the results. Finally, the average prediction accuracy and the annotation error of the verification set under the different parameter combination of each model are presented quantitatively in Appendix A.

For different models, we need to measure three metrics. The first indicator is the accuracy rate, that is, the difference between the predicted value and the true value, if it equals to , the prediction is correct, otherwise it is wrong, and finally we take the average value as the prediction accuracy.

$$\mathrm{acc} = \frac{\sum_{i=1}^{n} 1(f(x_i) = y_i)}{n}$$

where, n is the number of samples and 1(·) is the indicator function.

The second metric we need to measure the performance is F1-score. It can be compute by the following equation.

$$F1 = \frac{2PR}{P + R}$$
$$P = \frac{TP}{TP + FP}$$
$$R = \frac{TP}{TP + FN}$$

where, TP is the true positive number of samples, FP is the false positive number of samples and FN is the false negative number of samples.

The third metric is the confusion matrix which can represent the classification result intuitively. The matrix has two rows and two columns, where the first row and the second row denotes the true label and false label respectively, and the first column and the second column denotes the positive prediction and the negative prediction respectively.

5. Results and Analysis: Comparison and Interpretation

5.1. Performance analysis

In this section, we compare performance across different models and on raw and enhanced data sets. The advantages and disadvantages of different models and the combination of parameters are compared comprehensively in this part. Finally, we visualize the data features considered important by the model and draw some interesting conclusions.

According to the final accuracy measure, the method of support vector machine and neural network achieves the best performance, which is related to the strong classification ability of the two methods. The method of logistic regression also performed better. But the other two baseline approaches, nearest mean and trivial system, do not work as well, and the trivial system approach assembles random performance.

Although the classification accuracy of support vector machine and neural networks is similar, they have different classification preferences for different categories. The difference can be seen in the confusion matrix of Figure 3, where the support vector machine predicts more of the negative class as the positive class, while the neural network does the opposite.

From the point of view of data enhancement, the performance of the enhanced data is no better than that of the original data, this may be because the enhanced features in the time dimension do not contain much

useful information for fire prediction. These feature important content we will discuss in the later part.

By comparing f1-score with accuracy, we find that f1-score is lower than accuracy because f1-score takes precision and recall into account and calculates their harmonic mean. This approach is more rigorous and fair than the direct approach, so many data mining tasks tend to use f1-score as an indicator. This evaluation index f1-score can also effectively test the ability of classification balance of the model.

5.2. Feature Importance

We visualize the importance of the features given by the logistic model in Figure 4. From the figure, we can see that the most important feature for the model is “ISI”. It is also a very intuitive feature of how the fire is spreading. So this feature is the most important for predicting fires. At the same time, there are several features for the classification of the contribution to a greater degree. For example, “DC” and “DMC” represent drought and moisture respectively. Judging from the occurrence of fire, soil and air humidity is also a more important indicator of the emergence of a fire. Combining the heat map of correlation coefficient and the histogram of feature distribution, we can see that the values of the correlation coefficient are higher for these more important features, moreover, the feature distribution of each category in the histogram is quite different.

There is also an enhanced feature that is very important which is the maximum rainfall in these five days. It's perfectly understandable that fire is difficult to occur in heavy rainfall, so this feature can be used as an effective indicator to determine whether there is a fire.

The analysis of these characteristics tells us that in the prevention of forest fires, we need to pay special attention to the forest air and soil moisture, if the weather is very dry, we should prepare well for fire prevention. When it rains heavily, we do not have to worry about the happen of fire. And after the fire, we should also have a clear understanding of the spread of the fire, the index will clearly indicate the trend of the fire.

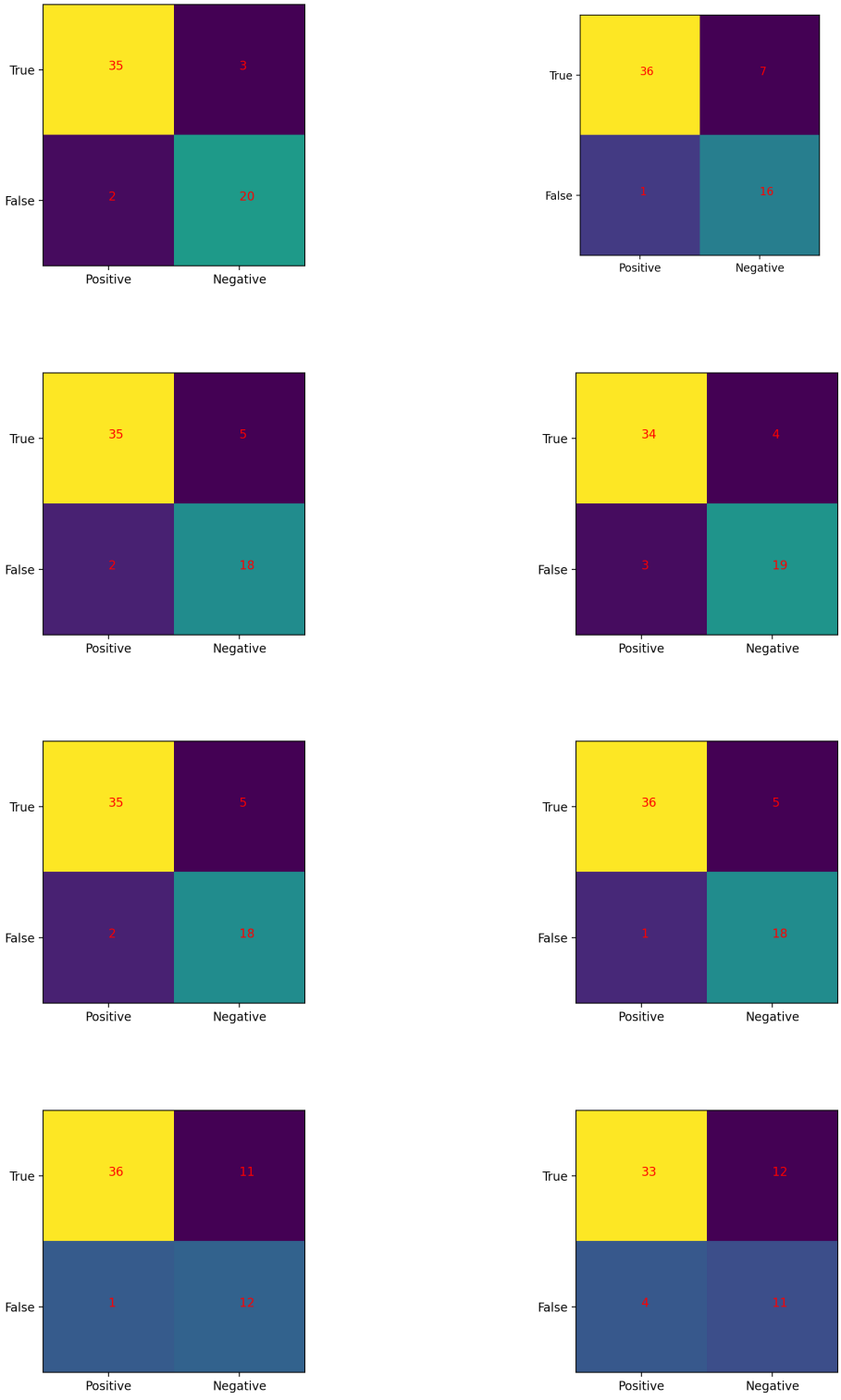


Figure 3 The confusion matrix of different algorithms. (The first column: the original dataset, the second column: the augmented dataset. The first row: logistic regression, the second row: support vector machine, the third row: neural networks, the fourth row: nearest mean.)

Table 1 The performance of different algorithms on the original and augmented dataset

Algorithm	Dataset	Validation Accuracy Mean ± Std	Test Accuracy	F1-score
Logistic Regression	Original	0.924±0.031	0.883	0.873
	Augmented	0.907±0.036	0.883	0.876
Support Vector Machine	Original	0.924±0.031	0.917	0.911
	Augmented	0.863±0.031	0.867	0.850
Neural Network	Original	0.929±0.027	0.883	0.873
	Augmented	0.875±0.046	0.900	0.890
Kernel Nearest Mean	Original	0.864±0.084	0.800	0.762
	Augmented	0.771±0.093	0.733	0.692
Trivial System	Original	0.462±0.045	0.400	0.482
	Augmented	0.462±0.045	0.400	0.482

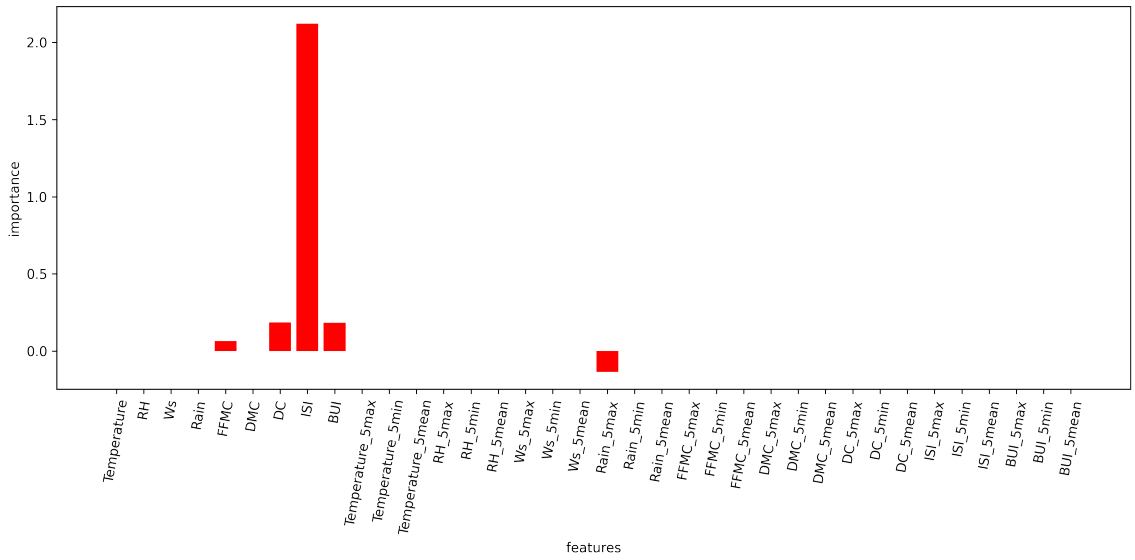


Figure 4 The feature importance given by the logistic regression model

6. Libraries used and what you coded yourself

In our code implementation, we compare different algorithms, such as logistic regression, support vector machine, neural network, kernel nearest mean and trivial system method. The first three method we apply the Scikit-learn library, and we implement the last two methods from scratch. Also, to make it easier to present our results, and to reuse some functional modules, we also implemented some basic code that was included in our submission.

7. Summary and conclusions

In this experiment, we use the method of machine learning to analyze the situation of forest fire in Algeria, and find out some useful features for fire prediction, it also explores the internal function model and mechanism, and provides guidance for accurate prediction and prevention of forest fire. In future work, we can choose more powerful algorithms, such as Random Forest [14][15] and XGBoost [16][17], to enhance the performance of the model and provide better interpretability, which we will address in the future.

References

[1] Davis, Kenneth Pickett. "Forest fire, control and use." Forest fire, control and use. (1959).

[2] Stephens, Scott L., and Lawrence W. Ruth. "Federal forest‐fire policy in the United States." Ecological applications 15.2 (2005): 532-542.

[3] Pham, Binh Thai, et al. "Performance evaluation of machine learning methods for forest fire modeling and prediction." Symmetry 12.6 (2020): 1022.

[4] Noble, William S. "What is a support vector machine?." Nature biotechnology 24.12 (2006): 1565-1567.

[5] Hosmer Jr, David W., Stanley Lemeshow, and Rodney X. Sturdivant. Applied logistic regression. Vol. 398. John Wiley & Sons, 2013.

[6] Hecht-Nielsen, Robert. "Theory of the backpropagation neural network." Neural networks for perception. Academic Press, 1992. 65-93.

[7] Faroudja ABID et al. , â€œPredicting Forest Fire in Algeria using Data Mining Techniques: Case Study of the Decision Tree Algorithmâ€ , International Conference on Advanced Intelligent Systems for Sustainable Development (AI2SD 2019) , 08 - 11 July , 2019, Marrakech, Morocco.

[8] Slavia, Athaya Putri, Edi Sutoyo, and Deden Witarsyah. "Hotspots forecasting using autoregressive integrated moving average (ARIMA) for detecting forest fires." 2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS). IEEE, 2019.

[9] Kerdprasop, Nittaya, et al. "Forest Fire Area Estimation using Support Vector Machine as an Approximator." IJCCI. 2018.

[10] Sakellariou, Stavros, et al. "Remotely sensed data fusion for spatiotemporal geostatistical analysis of forest fire hazard." Sensors 20.17 (2020): 5014.

[11] Zhang, Guoli, Ming Wang, and Kai Liu. "Forest fire susceptibility modeling using a convolutional neural network for Yunnan province of China." International Journal of Disaster Risk Science 10.3 (2019): 386-403.

[12] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." the Journal of machine Learning research 12 (2011): 2825-2830.

[13] Veenman, Cor J., and Marcel JT Reinders. "The nearest subclass classifier: A compromise between the nearest mean and nearest neighbor classifier." IEEE Transactions on Pattern Analysis and Machine Intelligence 27.9 (2005): 1417-1429.

[14] Belgiu, Mariana, and Lucian Drăguţ. "Random forest in remote sensing: A review of applications and future directions." ISPRS journal of photogrammetry and remote sensing 114 (2016): 24-31.

[15] Biau, Gérard, and Erwan Scornet. "A random forest guided tour." Test 25.2 (2016): 197-227.

[16] Chen, Tianqi, et al. "Xgboost: extreme gradient boosting." R package version 0.4-2 1.4 (2015): 1-4.

[17] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016.

Appendix

Appendix A: Parameters and Results

Table 2 The logistic regression model's performance on the original dataset with different parameters

params	mean_val_score	std_val_score
{'C': 1, 'penalty': 'l2', 'solver': 'liblinear'}	0.9240	0.0313
{'C': 1, 'penalty': 'l1', 'solver': 'saga'}	0.9186	0.0380
{'C': 1, 'penalty': 'l2', 'solver': 'newton-cg'}	0.9186	0.0380
{'C': 1, 'penalty': 'l2', 'solver': 'lbfgs'}	0.9186	0.0380
{'C': 1, 'penalty': 'l2', 'solver': 'sag'}	0.9186	0.0380
{'C': 1, 'penalty': 'l2', 'solver': 'saga'}	0.9186	0.0380
{'C': 10, 'penalty': 'l1', 'solver': 'saga'}	0.9186	0.0380
{'C': 10, 'penalty': 'l2', 'solver': 'newton-cg'}	0.9186	0.0380
{'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}	0.9186	0.0380
{'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}	0.9186	0.0380

Table 3 The logistic regression model's performance on the augmented dataset with different parameters

params	mean_val_score	std_val_score
{'C': 0.1, 'penalty': 'l1', 'solver': 'saga'}	0.9075	0.0369
{'C': 1, 'penalty': 'l1', 'solver': 'saga'}	0.9024	0.0651
{'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}	0.8970	0.0492
{'C': 0.1, 'penalty': 'l2', 'solver': 'newton-cg'}	0.8802	0.0448
{'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}	0.8802	0.0448
{'C': 1, 'penalty': 'l2', 'solver': 'newton-cg'}	0.8751	0.0629
{'C': 1, 'penalty': 'l2', 'solver': 'lbfgs'}	0.8751	0.0629
{'C': 1, 'penalty': 'l2', 'solver': 'sag'}	0.8749	0.0698

{'C': 0.1, 'penalty': 'l2', 'solver': 'sag'}	0.8748	0.0508
{'C': 1, 'penalty': 'l2', 'solver': 'liblinear'}	0.8748	0.0742

Table 4 The support vector machine's performance on the original dataset with different parameters

params	mean_val_score	std_val_score
{'C': 1, 'gamma': 'scale', 'kernel': 'linear'}	0.9240	0.0313
{'C': 1, 'gamma': 'auto', 'kernel': 'linear'}	0.9240	0.0313
{'C': 10, 'gamma': 'scale', 'kernel': 'linear'}	0.9132	0.0395
{'C': 10, 'gamma': 'auto', 'kernel': 'linear'}	0.9132	0.0395
{'C': 10, 'gamma': 'auto', 'kernel': 'sigmoid'}	0.9131	0.0201
{'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}	0.8913	0.0342
{'C': 0.1, 'gamma': 'auto', 'kernel': 'linear'}	0.8913	0.0342
{'C': 1, 'gamma': 'auto', 'kernel': 'sigmoid'}	0.8913	0.0452
{'C': 10, 'gamma': 'scale', 'kernel': 'sigmoid'}	0.8908	0.0530
{'C': 1, 'gamma': 'scale', 'kernel': 'sigmoid'}	0.8859	0.0358

Table 5 The support vector machine's performance on the augmented dataset with different parameters

params	mean_val_score	std_val_score
{'C': 1, 'gamma': 'auto', 'kernel': 'sigmoid'}	0.8638	0.0315
{'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}	0.8584	0.0445
{'C': 0.1, 'gamma': 'auto', 'kernel': 'linear'}	0.8584	0.0445
{'C': 10, 'gamma': 'scale', 'kernel': 'sigmoid'}	0.8529	0.0459
{'C': 1, 'gamma': 'scale', 'kernel': 'sigmoid'}	0.8473	0.0526
{'C': 1, 'gamma': 'scale', 'kernel': 'linear'}	0.8423	0.0578
{'C': 1, 'gamma': 'auto', 'kernel': 'linear'}	0.8423	0.0578
{'C': 10, 'gamma': 'auto', 'kernel': 'sigmoid'}	0.8420	0.0449

{'C': 0.01, 'gamma': 'scale', 'kernel': 'linear'}	0.8309	0.0651
{'C': 0.01, 'gamma': 'auto', 'kernel': 'linear'}	0.8309	0.0651

Table 6 The neural network's performance on the original dataset with different parameters

params	mean_val_score	std_val_score
{'activation': 'tanh', 'alpha': 0.5, 'learning_rate': 'adaptive', 'solver': 'adam'}	0.9294	0.0274
{'activation': 'tanh', 'alpha': 0.001, 'learning_rate': 'constant', 'solver': 'adam'}	0.9239	0.0266
{'activation': 'tanh', 'alpha': 0.001, 'learning_rate': 'adaptive', 'solver': 'adam'}	0.9239	0.0266
{'activation': 'tanh', 'alpha': 0.01, 'learning_rate': 'adaptive', 'solver': 'adam'}	0.9239	0.0266
{'activation': 'tanh', 'alpha': 0.001, 'learning_rate': 'invscaling', 'solver': 'adam'}	0.9185	0.0171
{'activation': 'tanh', 'alpha': 0.01, 'learning_rate': 'constant', 'solver': 'adam'}	0.9185	0.0296
{'activation': 'tanh', 'alpha': 0.1, 'learning_rate': 'adaptive', 'solver': 'adam'}	0.9185	0.0171
{'activation': 'tanh', 'alpha': 0.5, 'learning_rate': 'constant', 'solver': 'adam'}	0.9185	0.0296
{'activation': 'tanh', 'alpha': 0.5, 'learning_rate': 'invscaling', 'solver': 'adam'}	0.9185	0.0296
{'activation': 'tanh', 'alpha': 0.1, 'learning_rate': 'invscaling', 'solver': 'adam'}	0.9183	0.0302

Table 7 The neural network's performance on the augmented dataset with different parameters

params	mean_val_score	std_val_score
{'activation': 'tanh', 'alpha': 0.001, 'learning_rate': 'constant', 'solver': 'adam'}	0.8752	0.0467

{'activation': 'relu', 'alpha': 0.5, 'learning_rate': 'adaptive', 'solver': 'adam'}	0.8749	0.0531
{'activation': 'tanh', 'alpha': 0.1, 'learning_rate': 'constant', 'solver': 'adam'}	0.8748	0.0588
{'activation': 'relu', 'alpha': 0.1, 'learning_rate': 'invscaling', 'solver': 'adam'}	0.8695	0.0360
{'activation': 'relu', 'alpha': 0.01, 'learning_rate': 'constant', 'solver': 'adam'}	0.8643	0.0564
{'activation': 'tanh', 'alpha': 0.5, 'learning_rate': 'invscaling', 'solver': 'adam'}	0.8641	0.0567
{'activation': 'tanh', 'alpha': 0.5, 'learning_rate': 'adaptive', 'solver': 'adam'}	0.8641	0.0452
{'activation': 'relu', 'alpha': 0.1, 'learning_rate': 'constant', 'solver': 'adam'}	0.8641	0.0296
{'activation': 'relu', 'alpha': 0.1, 'learning_rate': 'adaptive', 'solver': 'adam'}	0.8641	0.0452
{'activation': 'tanh', 'alpha': 0.001, 'learning_rate': 'adaptive', 'solver': 'lbfgs'}	0.8640	0.0749