Prediction on whether it will rain tomorrow in Australia INF552 Final Project

HE CHANG, 5670-5275-76
RUI HU, 2350-3082-89
ZIQIAO GAO, 2157-3718-27
FANLIN QIN, 5317-9738-58

In this project, we select a challenge[1] from Kaggle[2] on predicting whether or not it will rain in the next day in Australia. We would apply four machine learning algorithms, Decision Tree, Logistic Regression, Pocket Perceptron Learning & Support Vector Machine, and Neural Network, to examine the possibility of making an accurate prediction. By the end of this paper, we shall compare the prediction results of different machine learning methods and present a reflection on what we find inspiring and valuable.

Additional Key Words and Phrases: machine learning, binary classification, weather data, rainfall, Australia

1 INTRODUCTION

Making a binary classification on a set of instance data has been the major practice throughout this semester, so our team decide to focus on problems that need binary classification to produce meaningful insights for solutions. After several serious team discussions, we land on constructing the casual relationships between weather data in Australia and whether or not it will rain in the next day[1].

Although this topic may sound common, it is actually interesting as well as meaningful because weather is closely related to our lives. If we can increase the probability of correctly predicting rainfall condition, our daily productivity efficiency can be greatly improved. For example, we can prepare in advance for our next day plan, such as wearing waterproof clothes and avoiding going outside for a long time. Furthermore, it is a crucial contribution for country like Australia to better manage national industries including agriculture and animal husbandry. Australian government would benefit a lot form being able to predict rainfall in terms of adjusting schedules. A instance would be determining how long the farm irrigation system should be working.

In our project, we apply several machine learning algorithms taught in this semester to examine the possibility of making an accurate prediction. We use Decision Tree, Logistic Regression, Support Vector Machine and Neural Network. Firstly, we explain how we pre-process the raw weather data for best model training purposes. Then we discuss about how we actually implement and optimize these algorithms for this topic. Finally, We show all the results and evaluate their performance under different criterion.

2 MATERIALS

In the following, we briefly present our dataset, machine learning algorithms and evaluation methods used in this project.

2.1 Dataset

In our training dataset[1] that we download from Kaggle, there are 142193 instances of weather data collected from period of 2008 to 2017 in Australia. Each instance data contains 24 dimensions that capture a mixture of both categorical and numerical features.

To acquire test data, we first browse the official website of Australian Bureau of Meteorology and search for recent raw weather data. Unfortunately, it takes us a long time to collect only a limited amount of weather data because the website is not bug-free and a lot of weather stations are no longer valid. The amount of collected data is too little to be comprehensive and representational. Therefore, we randomly split our original dataset to create a training subset and a testing subset. The training subset takes 80% while testing subset takes 20%.

Although there are some missing values from the dataset, we believe it is acceptable because several unavoidable factors would cause the weather data unavailability and data loss. Here is a list of exemplary data features followed by their datatype and notion:

- Date (categorical): the date of observation in format "YYYY-MM-DD".
- Location (categorical): the location where weather data is collected.
- MaxTemp (float): the maximum temperature in degrees Celsius.
- WindGustDir (categorical): the direction of the strongest wind gust in the 24 hours to midnight.
- Pressure9am (float): the atmospheric pressure reduced to mean sea level at 9am.
- RainTomorrow (boolean): the label variable that indicates whether it rains in the next day.

2.2 Decision Tree

Decision Tree is a tree structure model that can be a either binary tree or a non-binary tree. It represents a mapping relationship between object attributes and object values. Generated from top to bottom, each tree node may have two or more child nodes, leading to different results. While each non-leaf node represents a test on a feature attribute with branches representing the output of this feature attribute in a certain value range, the leaf node stores one of the binary categories.

The decision-making process starts from the root node, testing the corresponding feature attributes in the item to be classified and selecting the output branch according to its value until it reaches the leaf node. Finally, it uses the category stored in the leaf node as the decision result.

There are several advantages of Decision Tree model. The first one is that Decision Tree is easy to understand and implement, because it can directly reflect the characteristics of data. The second is that it is able to produce feasible and good results for large data sources in a relatively short time. Finally, it is easy to evaluate the decision model through static testing, where a credibility of the model can be determined.

2.3 Logistic Regression

Logistic Regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. It analyzes the relationship between multiple independent variables and categorical dependent variables, and estimates the likelihood of an event by fitting the data to a logistic curve. From the perspective of machine learning, it is a supervised learning classification algorithm which is used to predict observations to a discrete set of classes. Therefore, Logistic Regression is a potential feasible classifier for our raining problem.

The Logistic Regression algorithm works as follows.

First of all, Linear Equation. Logistic Regression algorithm works by implementing a linear equation with independent or explanatory variables to predict a response value. For example, in this dataset, we consider variables like the Maximum Temperature, the Minimum Temperature, Rainfall as the explanatory variables and they are denoted by x_1, x_2 ... x_n. The probability of raining is the target variable and it is denoted by z. The linear equation would be given mathematically with the following equation:

$$z(x) = w_1x_1 + w_2x_2 + ... + w_nx_n \tag{1}$$

Secondly, Sigmoid Function. The predicted response value, denoted by z, is then converted into a probability value that lies between 0 and 1. We use the sigmoid function in order to map predicted values to probability values. This sigmoid function then maps any real value into a probability value between 0 and 1.

A sigmoid function is a special case of the Logistic Function. It is given by the following mathematical formula:

$$\theta(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

Thirdly, Decision Boundary. The sigmoid function returns a probability value between 0 and 1. This probability value is then mapped to a discrete class which is either "0" or "1". In order to map this probability value to a discrete class (rain or not), we select a threshold value. This threshold value is called decision boundary. Above this threshold value, we will map the probability values into class 1 and below which we will map values into class 0. In our case, the decision boundary is set to 0.5. So, if the probability value is greater than 0.5, we will say that tomorrow is going to rain, versa vice.

Finally, Making Predictions. Now, we know about sigmoid function and decision boundary in logistic regression. After fitting data into the Logistic Regression model, having the parameters w_n, we can use our knowledge of sigmoid function and decision boundary to write a prediction function. A prediction function in logistic regression returns the probability of the raining being positive, Yes or True.

2.4 Pocket Perceptron Learning & Support Vector Machine

The Perceptron Learning is a supervised algorithm that generates binary classifiers. Because Perceptron Learning's final classification accuracy would be affected by the initial states of assigned random weights, we will implement Perceptron Learning in a 'Pocket' way to setup different initial random weights and acquire the best classification Perceptron model with maximized prediction accuracy.

Support Vector Machines, also named Support Vector network is a supervised learning model associated with learning algorithms that analyze data classification and data regression. Different from Pocket Perceptron Learning and other binary classification algorithm, Support Vector Machines model not only separates data into 2 groups, but also tries to create the separation so that the distance to the 'gap' from each group is maximized. The testing data is then mapped and predicted to belong to a category based on the side of the gap on which they fall. In addition to performing linear classification, SVM can efficiently perform a non-linear classification using what is called the Kernel Trick that implicitly maps the linear non-separable data into a higher dimension feature space. Using the classification results in higher dimension space where data becomes separable, SVM is able to perform binary classification in the original space.

2.5 Neural Network

Neural Networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated. Neural networks have various architectures for practical applications.

For our task, we choose the multi-layer Feed-forward Neural Network. This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each "`neurons`" in one layer has directed connections to the "`neurons`" of the subsequent layer. In many applications the units of these networks apply a sigmoid function as an activation function. The universal approximation theorem for Neural Networks states that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer perceptrons with just one hidden layer. This result holds for a wide range of activation functions.

Multi-layer Neural Networks use a variety of learning techniques with the most popular being back-propagation. Here, the output values are compared with the correct answer to compute the value of some pre-defined error-function. By various techniques, the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small. In this case, one would say that the network has learned a certain target function.

The basic sklearn configuration setup of our Neural Network Model is as the following:

- Hidden Layer count: 1.
- Hidden layer size: 100.
- Learning rate: 0.1.
- Solving method: Stochastic Gradient Descendent.
- Activation function: Sigmoid function.

We also try a multi-layers FNN model using PyTorch, the model configuration is as the following:

- Input Layer count: 1
- Input Layer size: 23
- Hidden Layer count: 2.
- Hidden layer size: 200, 150.
- Learning rate: decaying the learning rate of each parameter group by 0.95 every 30 epochs.
- Solving method: Stochastic Gradient Descendent.
- Activation function: ReLU function for the first hidden layer, Tanh function for the second hidden layer, CELU function for the output layer.

2.6 Evaluation measures

In order to evaluate models performance in raining prediction, we not only measure the accuracy of each model, but also use the measures of precision, recall, accuracy and F-measure[4].

2.6.1 Precision. Precision[4], also called positive predictive value, is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). It identifies the proportion of correctly predicted positive outcome. It is more concerned with the positive class than the negative class.

2.6.2 Recall. Recall[4], is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been). It identifies the proportion of correctly predicted actual positives.

2.6.3 F-Measure. The F-measure[4] (also F-score or F1 score) is a measure of a test's accuracy. It is the harmonic mean of the precision and recall, considers both the precision and the recall of the test.

$$F - Measure = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{3}$$

3 METHODS

In this section we are mainly discussing how we handle data processing and how we implement machine learning algorithms.

3.1 Data Pre-processing

Because some features of instance data is not numerical, such as location and wind direction, we need to pre-process our dataset to make each instance available to fit into the machine learn models. We keep the values already in either integer or float datatype while we mainly target categorical and boolean variables. We basically assign a unique integer id starting from 1 to the categorical feature columns and we use 0 to mark 'No' and 1 to mark 'Yes' for the boolean data. Also, in case we encounter missing data, we basically use a special value of -1 to differentiate it from other valid data which would not possibly be negative. For temperature related data, we use -274 to indicate missing data because regular temperature data would not reach below -274 degree Celsius.

On the other hand, after doing some research, we find that normalizing the data may boost the accuracy rate a little bit. We guess that different weather data features do not have similar ranges of values and it may take a long time for machine learning models to oscillate back and forth before it can finally find their way to the global/local minimum. It is the data normalization procedure that gets rid of extreme abnormal sample values and thus increases the probability of acquiring a larger gradient rate. So we normalize the data columns whose attribute value is numerical.

3.2 Decision Tree

In this project, we use the open source python library sklearn.tree. DecisionTreeClassifier() to implement the decision tree algorithm. The entire procedure is relatively simple. After getting the training data, we directly use the sklearn Decision Tree Classifier to fit the data and build the decision tree. Then we calculate the model accuracy by testing it with our testing dataset. In addition, we try both 'gini' and 'entropy' as the criterion for the classifier.

3.3 Logistic Regression

In the Logistic Regression method, we use the open source Python library sklearn.linear_model.LogisticRegression package to implement the Logistic Regression algorithm. The training data and test data are all come from the preprocessed data mentioned in 3.1, we randomly choose 80% of records as our training data and the rest of them are the testing data. During the training process, we choose "soga" solver and set the maximum iteration as 4000. Then, we use the library to fit and classify the data.

3.4 Pocket PLA & SVM

We are using the implementation of Perceptron Learning Algorithm from sklearn.linear_model. For each pocket training, we fit the training data into the model and get the mean accuracy after 1000 iterations of learning. Overall, we generate 700 PLA models from 700 pockets and we output the model with the best prediction accuracy. Fig. 1 shows the number of classifications of each PLA in the total 700 pocket iteration.

We are using the SVM model from the sklearn package to implement the binary classification on weather data. For the specific SVM model configuration, we utilize the Kernel Trick by transforming our test instance data with a polynomial kernel function in degree of 2.

3.5 Neural Network

With the basic configuration, we use the sklearn Neural Network model to fit the training weather data and make predictions on test data. However, the initial Neural Network setup gives us a relatively low accuracy compared to other model results. We believe

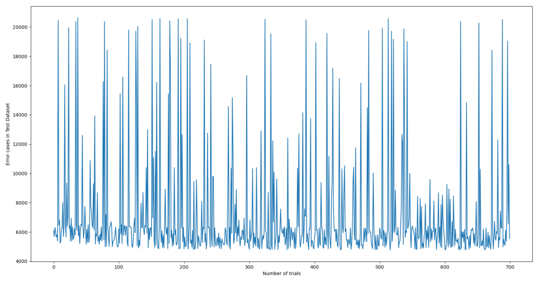

Fig. 1. The number of miss-classifications for each PLA model in 700 Pocket iterations

the prediction accuracy can be further improved because we are able to adjust a lot of factors of our Neural Network model.

First of all, we think that maybe the dataset itself can be further processed to help enhance the prediction accuracy. After analysis, we believe that the dataset might contain too many types of factorial features, and many of them are even mutual-affected. Therefore, we apply PCA algorithm to decrease the number of data dimensions to 80%, 75% and 50% of the original number of dimensions. Similarly, the month feature of the data could be grouped by season to decrease the probability that Neural Network model goes too over-fitting. Unfortunately, none of the modifications to those factors above brings satisfying changes to the result. Then we start to configure different learning rates, hidden layer size and batch sizes to see if we could find new outcomes because these standard factors are able to directly influence the performance of our Neural Network model. The result, however, still does not generate big changes to the prediction rate.

Secondly, we circle back to the model itself and deem that configuring different solving methods and activation functions might lead us to a new angle of observation. According to the official sklearn document, we replace Stochastic Gradient Descendent with 'lbfgs' optimizer and we use hyperbolic tan function as the activation function for the neurons in the hidden layer. Surprisingly, the prediction accuracy rises by 7%. Our analysis indicates that hyperbolic tan function solves the non-zero centered output issue of sigmoid function, where the output value of each neuron now can be both positive and negative centered by zero. Zero centered data would help to facilitate the model converging efficiency. On the other hand, although we do not know much about how the 'lbfgs' optimizer works, we realize one of the shortcoming of SGD is that the SGD algorithm can be negatively affected by noise data from certain training data samples.

Last but not least, we also try using a different third party library called PyTorch to architect a more versatile Neural Network model. We adjust the hidden layer to be more than one and we can even assign different activation functions for the neurons in different layers. The result from PyTorch Neural Network is very similar to our best prediction rate.

4 MACHINE LEARNING MODEL RESULTS

After fitting the weather history data into four machine learning models, we present the results in Table 1.

Table 1. Prediction Results

Algorithms	Accuracy	Precision	Recall	F-measure
Decision Tree	84.33 %	86.84 %	94.29 %	36.36 %
Logistic Regression	84.21 %	94.24 %	86.57 %	58.59 %
Pocket PLA & SVM	83.86 %	96.59 %	84.72 %	52.50 %
Neural Network	85.57 %	94.33 %	88.91 %	63.56 %

5 COMPARISON AND DISCUSSION

To begin with, we cannot guarantee that our models perform very well based on the accuracy values in Table 1. So we introduce null accuracy of the dataset to help evaluate our models. Null accuracy is calculated by always predicting the most frequent label type, which is 'no rain in the next day' in our datset. If a model generates an accuracy better than the null accuracy,it is at least better than us making a simple guess on the most frequent label. We can see that the mean model accuracy score is around 84% while the null accuracy score of the dataset is 77.49%. So, from this point, we can say that all of our models are doing a good job in predicting whether it will rain tomorrow.

Next, here is a brief comparison of the performance of our four machine learning models. In terms of prediction accuracy, although the four models share similar results, Neural Network has the best performance among the four machine learning algorithms. For precision[4], Pocket PLA & SVM has the best performance, followed by the Logistic Regression and Neural Network both with precision over 94%. And the Decision Tree can not have the same level of performance on Precision, meaning that it sometimes incorrectly predicts that it will not rain in the next day while it actually rains. For Recall[4], Decision Tree performs best with 94.29% Recall rate, which means that it can more correctly identify that tomorrow is going to rain than the others. For F-Measure[4], the Neural Network ranks top among the four, and followed by Logistic Regression, Pocket PLA & SVM and Decision Tree. In this dataset[1], its class's distribution are unbalanced, 77% of records are raining but both classes are important, then a classifier can get a high accuracy rate simply by choosing the majority class. In such a situation, we would choose the model that gets high F1 scores on both classes, as well as high accuracy rate. Therefore, among all four of machine learning models, the Neural Network has the best performance.

Our idea to explain the reason why Neural Network has the best accuracy and performance is that it involves multiple layers of neuron functions. For the relatively large number of data dimensions in the dataset[1], the more neuron functions participating in fitting the parameters, the better the trait of each data feature can be captured. With a better feature capture, Neural Network is able to fit closer to what each data essentially represents and give a more accurate prediction.

6 CONCLUSION

After constructing the machine learning models and evaluating the their performances, we are glad to summarize that making an feasibly accurate prediction on the rainfall conditions in Australia is quite promising. Based on the above analysis we can conclude that our classification models accuracy are good. If we can collect more instances of data with more dimensions of features, it is highly possible to acquire a even more accurate predicting ability. Also, another lesson we learn is that there are more than one criteria to evaluate the performance of a machine learning model. Besides accuracy, F-measure is also a very good indicator, and the judgment should be based on the specific dataset conditions too.

The most exciting thing is that we use the machine learning knowledge learned in this semester to solve problems in real life. For Australia, agriculture, animal husbandry and tourism take a significant portion of the entire national economy, and rainfall conditions would hugely affect those industry fields. Therefore, being able to make an accurate prediction on the rainfall conditions in Australia plays a critical role in adjusting country policies and thus avoiding loss caused by inappropriate activities in rain day.

In a word, we believe it is a good start for us to interpret this world in a machine learning view. Machine learning helps us to build a invisible relationship among various factors in our life. From this rainfall prediction project, all team members gain a lot of confidence in building machine learning models to solve real life problems.

7 CONTRIBUTION

7.1 He Chang

- Research and implement Neural Network model using sklearn package
- Collect raw weather for testing
- Write & edit Introduction section, Material section, Neural Network section, Conclusion section in final report document

7.2 Hu Rui

- Pre-process raw weather data;
- Implement, debug and optimize the Pocket PLA & SVM algorithm & Feed-forward Neural Network model using PyTorch
- Edit Pocket PLA & SVM section in final report document

7.3 Ziqiao Gao

- Research,implement, debug and optimize the Logistic Regression algorithm
- Introduce Evaluation Measure to the final project
- Write & edit Abstract section, Evaluation Measure section, Logistic Regression section, Result Section, Comparison & Discussion Section in final report document

7.4 Fanlin Qin

- Implement, debug and optimize the Decision Tree algorithm
- Edit Introduction section, Decision Tree section in final report document

REFERENCES

[1] Rain in Australia. https://www.kaggle.com/jsphyg/weather-dataset-rattle-package
[2] Kaggle. https://www.kaggle.com/
[3] PyTorch. https://pytorch.org/docs/stable/index.html
[4] Powers, David Martin. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation." (2011).