Neural Networks

Yingying Ji, 15220162202134

2019/05/19

Introduction

In this report, I will implement the methods used in the "Supermarket Entry" case by another simulated data I have used before, the weather data. The methods include logistic regression, classification tree, cross-validation approach, random forest, boosting, SVM and neural network methods. Since we can control the complexity of a model by regularization or by limiting the number of hidden units in neural network,we chould campare the results between it and other methods to see whcih is the best.

Data Description

This dataset contains daily weather observations from numerous Australian weather stations. [1] And before I use it, I have already processed the original data with Excel by cleaning some unuseful data.

Name of Variables	Interpretion
MinTemp	The minimum temperature in degrees celsius
MaxTemp	The maximum temperature in degrees celsius
Rainfall	The amount of rainfall recorded for the day in mm
Evaporation	The so-called Class A pan evaporation (mm) in the 24 hours to 9am
Sunshine	The number of hours of bright sunshine in the day
WindGustSpeed	The speed (km/h) of the strongest wind gust in the 24 hours to midnight
WindSpeed9am	Wind speed (km/hr) averaged over 10 minutes prior to 9am
WindSpeed3pm	Wind speed (km/hr) averaged over 10 minutes prior to 3pm
Humidity9am	Humidity (percent) at 9am
Humidity3pm	Humidity (percent) at 3pm
Pressure9am	Atmospheric pressure (hpa) reduced to mean sea level at 9am
Pressure3pm	Atmospheric pressure (hpa) reduced to mean sea level at 3pm
Cloud9am	Fraction of sky obscured by cloud at 9am. This is measured in "oktas"
Cloud3pm	Fraction of sky obscured by cloud at 3pm
Temp9am	Temperature (degrees C) at 9am
Temp3pm	Temperature (degrees C) at 3pm
RainTomorrow	1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0

Codes and the Interpretation of Results[2]

Prepare the data

```
install.packages("neuralnet")
install.packages("NeuralNetTools")
```

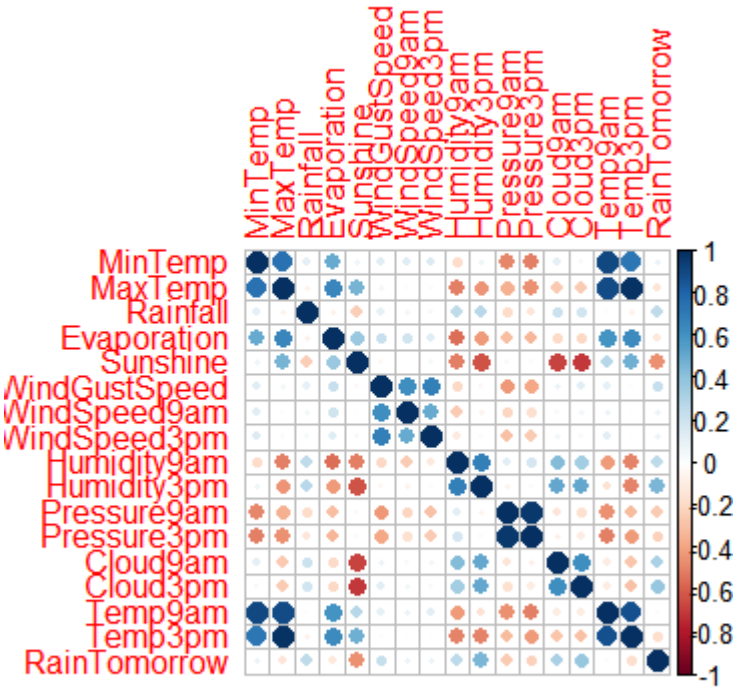
[1] The data are from ***Kaggle Dataset***.

[2] Since the code would have some problems running here, I put the whole codes and results in another file, where you can run to check the results.

```
library(caret)
library(glmnet)
library(corrplot)
library(rpart)
library(rpart.plot)
library(randomForest)
library(gbm)
library(e1071)
library(nnet)
library(neuralnet)
library(NeuralNetTools)
library(gridExtra)
rm(list=ls())
data <- read.csv("C:/Users/15068/Desktop/Microeconometrics/HW6/weather.csv",header = T)
```

Read the data

```
summary(data)
corrplot(cor(data))
data[,1:16] = scale(data[,1:16]) # scale the data
data$RainTomorrow = as.factor(data$RainTomorrow)
nvar = ncol(data) - 1
```



-

Create training and test sets

```
set.seed(123)
train =  createDataPartition(data$RainTomorrow,p=0.5,list=F)
data_train = data[train,]
data_test = data[-train,]
ytrue = data_test$RainTomorrow
```

Logistic Regression

```
fit <- glm(RainTomorrow ~.,data_train,family='binomial')
summary(fit)
result=coeftest(fit)
sum(result[,4]>0.05)
```

- The result is 4, which means (16-4=)12 varibales are statistically significant.

Test error in logistic regression

```
phat = predict(fit,data_test,type="response")
yhat = as.numeric(phat > 0.5)
table(ytrue,yhat)
1-mean(yhat==ytrue)
```

- The result is 0.147156, which means the misclassification error rate equals to 14.716%.

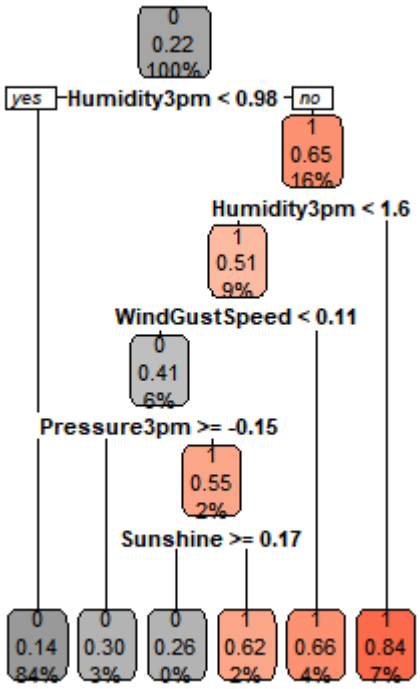
Claasification Tree

```
set.seed(100)
fit = rpart(RainTomorrow ~.,data_train)
rpart.plot(fit,box.palette=list("Grays", "Reds"))
```

Test error in classification tree

```
yhat = predict(fit,data_test,type="class")
table(ytrue,yhat)
1-mean(yhat==ytrue)
```

- The result is 0.1610315, which means the misclassification error rate equals to 16.103%.
- We can see that the error rate increased by 1.387%. It seems that the prediction results by logistic regression can be preciser than by classification tree.



-

Random Forest

```
set.seed(100)
fit=train(RainTomorrow~.,data=data_train,method="rf",
          trControl=trainControl(method = "cv"),
          tuneLength=10)
fit$bestTune
fit = randomForest(RainTomorrow ~.,data=data_train,mtry=8)
```

- Here I use cross validation to select the best mtry. And the result is 8.

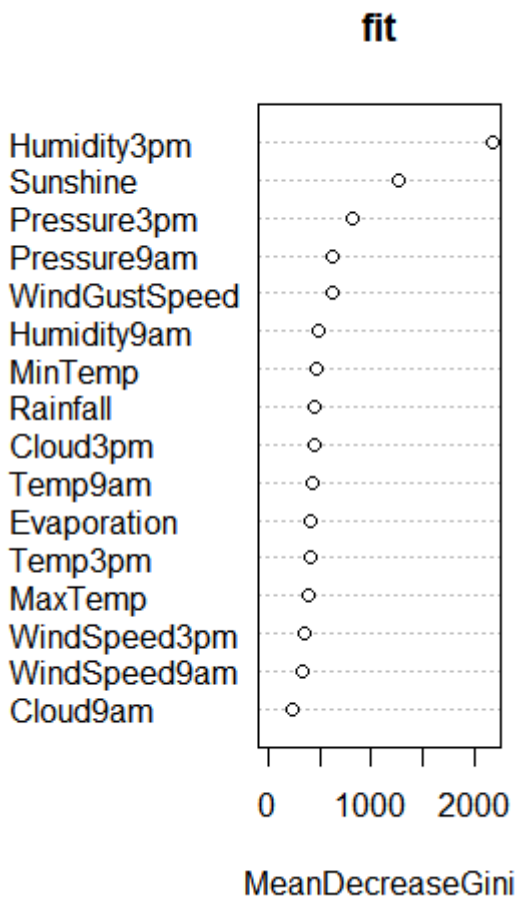
Test error in random forest

```
yhat = predict(fit,data_test)
table(ytrue,yhat)
1-mean(yhat==ytrue)
```

- The result is 1401322, which means the misclassification error rate equals to 14.013%.
- We can see that the error rate decreased by 2.09%.

The importance of variables

```
varImpPlot(fit)
```

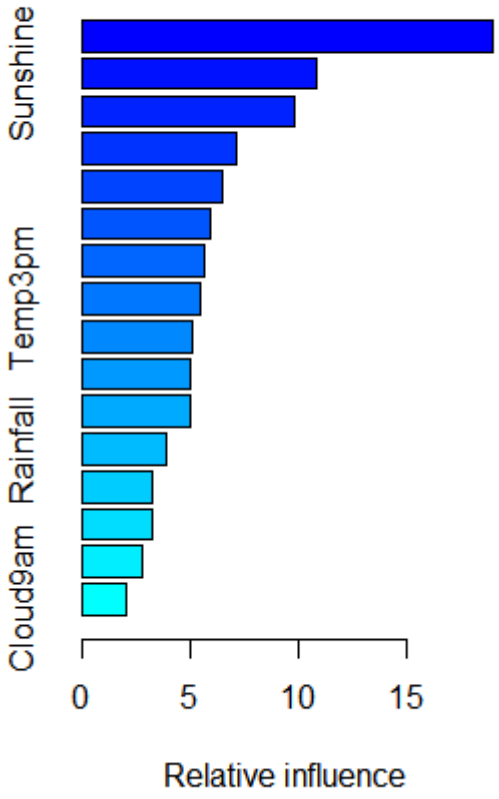


- From the graph, since the greater the "MeanDecreaseGini", the greater the importance of the variable, we can see that the humidty at 3pm is the most important factor to tomorrow's weather. In the meanwhile, the fraction of sky obscured by cloud at 9am has no effects basically.

Boosting

```
ntree = 5000
data_boost = transform(data_train,RainTomorrow=as.numeric(RainTomorrow)-1)
fit = gbm(RainTomorrow~.,data_boost,distribution="adaboost",
          n.trees=ntree,
          interaction.depth = 10,
          shrinkage = 0.1)
summary(fit)
```

- The boosting method shows almost the same resluts as the ransom forest: the humidty at 3pm is the most important factor to tomorrow's weather. In the meanwhile, the fraction of sky obscured by cloud at 9am has no effects basically.



-

Test error in boosting

```
phat = predict(fit,data_test,n.trees=ntree,type="response")
yhat = as.numeric(phat>0.5)
table(ytrue,yhat)
1-mean(yhat==ytrue)
```

- The result is 0.143472, which means the misclassification error rate equals to 14.347%.

Test error in using SVM

```
set.seed (100)
fit= svm(RainTomorrow~.,data_train,kernel="radial",gamma=0.01,cost=30,scale=F)
yhat = predict(fit,data_test,type="class")
table(ytrue,yhat)
1-mean(yhat==ytrue)
```

- The result is 0.1435753, which means the misclassification error rate equals to 14.358%. The error rate increased by 0.011% comparing with using boosting method.

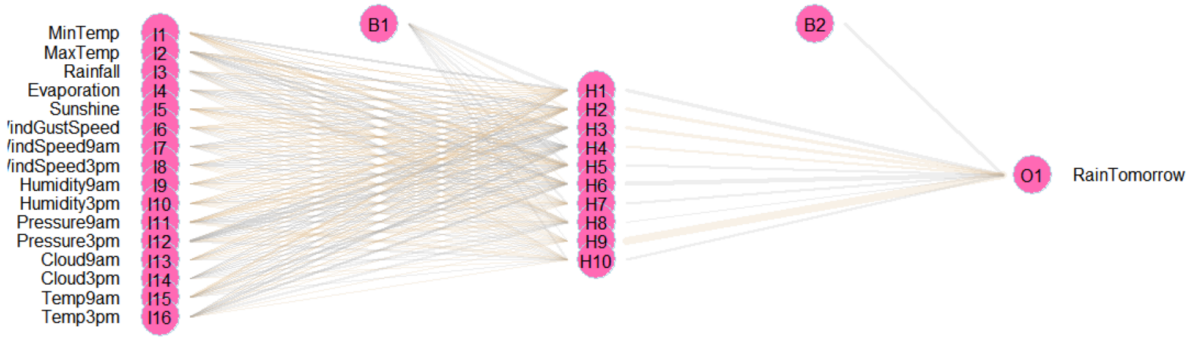
Test error in using Neural Net

```
set.seed(100)
fit = nnet(RainTomorrow ~.,data=data_train,
           size=10,maxit=10000,MaxNWts=10000,decay=0.1)
yhat = predict(fit,data_test,type="class")
table(ytrue,yhat)
1-mean(yhat==ytrue)
```

- The result is 0.1422669, which means the misclassification error rate equals to 14.227%. The error rate increased by 0.133% comparing with using SVM method.

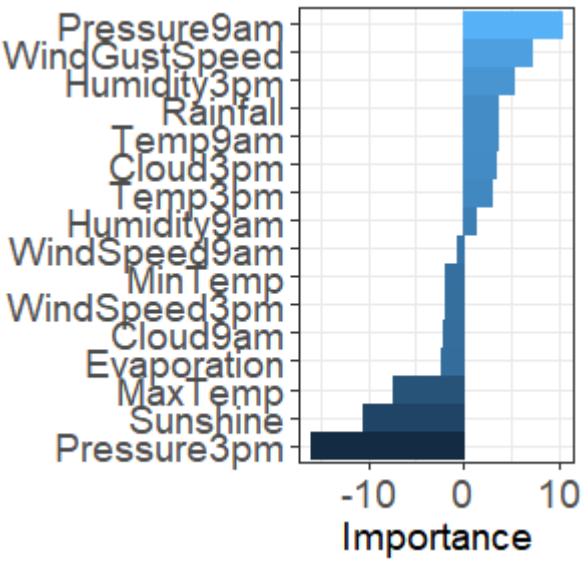
Vidualize Network

```
plotnet(fit,alpha_val=.2,
        circle_col="hotpink",
        pos_col="burlywood",
        neg_col="darkgray")
```

- 

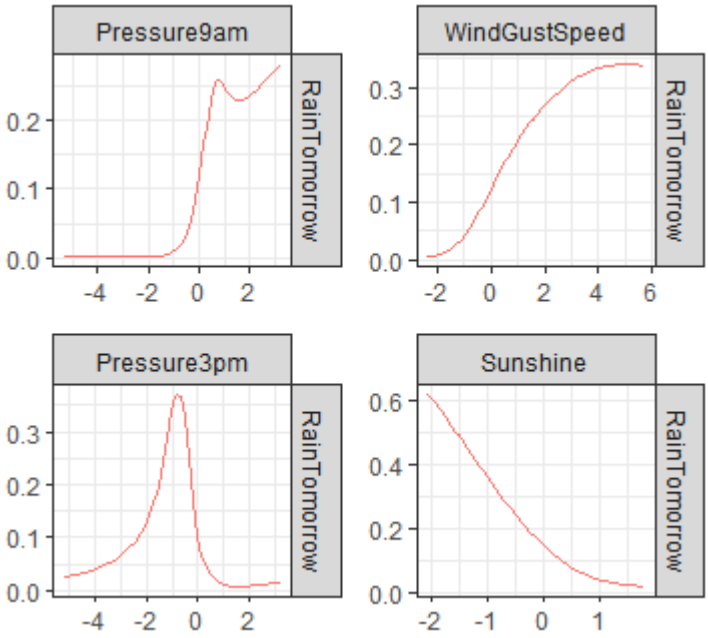
Importance plots based on weights

```
h = olden(fit)
h + coord_flip() + theme(axis.text=element_text(size=14),axis.title=element_text(size=14))
```

- 

Partial dependence plots

```
h1 = lekprofile(fit,xsel=c("Pressure9am"),group_vals=0.5) +
  theme(legend.position="none",axis.title=element_blank())
h2 = lekprofile(fit,xsel=c("WindGustSpeed"),group_vals=0.5) +
  theme(legend.position="none",axis.title=element_blank())
h3 = lekprofile(fit,xsel=c("Pressure3pm"),group_vals=0.5) +
  theme(legend.position="none",axis.title=element_blank())
h4 = lekprofile(fit,xsel=c("Sunshine"),group_vals=0.5) +
  theme(legend.position="none",axis.title=element_blank())
grid.arrange(h1,h2,h3,h4,ncol=2)
```



-
- Here are the most four important variables we selected compared with others in this model.