R Notebook

Code

2/12/23

This is an R Markdown (http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

Linear models for classification are a type of machine learning algorithm used to predict the categorical outcome of a given set of input variables. They work by drawing a straight line, or hyperplane, in the feature space that best separates the different classes of data. The goal of a linear model is to find the coefficients that define the hyperplane in a way that minimizes the distance between the hyperplane and the training examples. Linear models are relatively simple and fast to train, making them suitable for large datasets and real-time applications. They also have a clear and interpretable decision boundary that can help identify important features in the data. However, their performance can suffer when the relationship between the input variables and the output classes is more complex, or when there is significant overlap between the classes. In such cases, more sophisticated models like non-linear or ensemble methods may be more appropriate.

Hide

```
data(ParisH)
```

```
Warning: data set ‘ParisH’ not found
```

Hide

```
str(ParisH)
```

```
'data.frame':   10000 obs. of  18 variables:
 $ squareMeters     : int  75523 80771 55712 32316 70429 39223 58682 86929 51522 39686 ...
 $ numberOfRooms    : int  3 39 58 47 19 36 10 100 3 42 ...
 $ hasYard          : int  0 1 0 0 1 0 1 1 0 0 ...
 $ hasPool          : int  1 1 1 0 1 1 1 0 0 0 ...
 $ floors           : int  63 98 19 6 90 17 99 11 61 15 ...
 $ cityCode         : int  9373 39381 34457 27939 38045 39489 6450 98155 9047 71019 ...
 $ cityPartRange    : int  3 8 6 10 3 8 10 3 8 5 ...
 $ numPrevOwners    : int  8 6 8 4 7 6 9 4 3 8 ...
 $ made             : int  2005 2015 2021 2012 1990 2012 1995 2003 2012 2021 ...
 $ isNewBuilt       : int  0 1 0 0 1 0 1 1 1 1 ...
 $ hasStormProtector: int  1 0 0 1 0 1 1 0 1 1 ...
 $ basement         : int  4313 3653 2937 659 8435 2009 5930 6326 632 5198 ...
 $ attic            : int  9005 2436 8852 7141 2429 4552 9453 4748 5792 5342 ...
 $ garage           : int  956 128 135 359 292 757 848 654 807 591 ...
 $ hasStorageRoom   : int  0 1 1 0 1 0 0 0 1 1 ...
 $ hasGuestRoom     : int  7 2 9 3 4 1 5 10 5 3 ...
 $ price            : num  7559082 8085990 5574642 3232561 7055052 ...
 $ category         : chr  "Basic" "Luxury" "Basic" "Basic" ...
```

Hide

```
str(ParisH)
```


```
'data.frame':   10000 obs. of  18 variables:
 $ squareMeters     : int  75523 80771 55712 32316 70429 39223 58682 86929 51522 39686 ...
 $ numberOfRooms    : int  3 39 58 47 19 36 10 100 3 42 ...
 $ hasYard          : int  0 1 0 0 1 0 1 1 0 0 ...
 $ hasPool          : int  1 1 1 0 1 1 1 0 0 0 ...
 $ floors           : int  63 98 19 6 90 17 99 11 61 15 ...
 $ cityCode         : int  9373 39381 34457 27939 38045 39489 6450 98155 9047 71019 ...
 $ cityPartRange    : int  3 8 6 10 3 8 10 3 8 5 ...
 $ numPrevOwners    : int  8 6 8 4 7 6 9 4 3 8 ...
 $ made             : int  2005 2015 2021 2012 1990 2012 1995 2003 2012 2021 ...
 $ isNewBuilt       : int  0 1 0 0 1 0 1 1 1 1 ...
 $ hasStormProtector: int  1 0 0 1 0 1 1 0 1 1 ...
 $ basement         : int  4313 3653 2937 659 8435 2009 5930 6326 632 5198 ...
 $ attic            : int  9005 2436 8852 7141 2429 4552 9453 4748 5792 5342 ...
 $ garage           : int  956 128 135 359 292 757 848 654 807 591 ...
 $ hasStorageRoom   : int  0 1 1 0 1 0 0 0 1 1 ...
 $ hasGuestRoom     : int  7 2 9 3 4 1 5 10 5 3 ...
 $ price            : num  7559082 8085990 5574642 3232561 7055052 ...
 $ category         : chr  "Basic" "Luxury" "Basic" "Basic" ...
```


Hide

```
dim(ParisH)
```

```
[1] 10000    18
```

Hide

```
head(ParisH)
```

	squareMeters <int>	numberOfRo… <int>	hasY… <int>	hasP… <int>	floors <int>	cityCode <int>	cityPartRange <int>	numPrevOwn <ir
1	75523	3	0	1	63	9373	3	
2	80771	39	1	1	98	39381	8	
3	55712	58	0	1	19	34457	6	
4	32316	47	0	0	6	27939	10	
5	70429	19	1	1	90	38045	3	
6	39223	36	0	1	17	39489	8	

6 rows | 1-10 of 18 columns

Hide

```
ParisH$category<-ifelse(ParisH$category=='Luxury',1,0)

df <- ParisH[,c(1,2,5,6,7,8,9,12,13,14,17,18)]
set.seed(1234)
i <- sample(1:nrow(df), nrow(df)*0.80, replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Data exploration. ParisH is the Paris Housing dataset that is imported. Luxury was changed to 1 and Basic to 0 in the Category column for logisitc regression.

Hide

```


glm1 <- glm(category~., data=train, family="binomial")
summary(glm1)
```

```
Call:
glm(formula = category ~ ., family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2527  -0.4854  -0.2847  -0.1635   3.0881

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)   -7.973e+00  8.164e+00  -0.977  0.32878
squareMeters  -5.199e-02  1.645e-03 -31.611  < 2e-16 ***
numberOfRooms -1.162e-03  1.317e-03  -0.882  0.37773
floors        -3.323e-02  1.719e-03 -19.329  < 2e-16 ***
cityCode       1.492e-06  1.303e-06   1.145  0.25206
cityPartRange -3.684e-02  1.330e-02  -2.771  0.00559 **
numPrevOwners -1.306e-02  1.340e-02  -0.974  0.32982
made           2.013e-03  4.072e-03   0.494  0.62105
basement      -9.068e-06  1.322e-05  -0.686  0.49263
attic          5.424e-06  1.312e-05   0.413  0.67937
garage        -1.115e-04  1.465e-04  -0.761  0.44658
price          5.199e-04  1.645e-05  31.611  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6071.0  on 7999  degrees of freedom
Residual deviance: 4624.8  on 7988  degrees of freedom
AIC: 4648.8

Number of Fisher Scoring iterations: 6
```

This is the logistic regression model. The deviance residuals shows the min, 1q, median, 3q, and max for the residuals of the model. The coefficients shows the estimated coefiicients for each predicted variable, along with their standared errors and p values. For this model, price and squremeters have a strong negative correlation with the house being basic, which makes sense since luxury houses would be more expensive. The std.error of the model is low indicating there is not high variance in the model.The resudial deviance is lower than the null deviance which states that predictor variables are important in predicitng the model.

Hide

```
library(e1071)
nb1 <- naiveBayes(category~., data=train)
nb1
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
       0        1
0.873625 0.126375

Conditional probabilities:
   squareMeters
Y       [,1]     [,2]
  0 49867.96 28863.61
  1 49394.73 28511.23

   numberOfRooms
Y       [,1]     [,2]
  0 50.47675 28.89362
  1 49.99505 28.14209

   floors
Y       [,1]     [,2]
  0 50.33953 28.89959
  1 50.09397 28.66488

   cityCode
Y       [,1]     [,2]
  0 50149.19 29102.29
  1 50993.77 29418.69

   cityPartRange
Y       [,1]     [,2]
  0 5.518386 2.871980
  1 5.424332 2.924624

   numPrevOwners
Y       [,1]     [,2]
  0 5.532980 2.857412
  1 5.464886 2.835682

   made
Y       [,1]     [,2]
  0 2005.384 9.287300
  1 2005.469 9.519223

   basement
Y       [,1]     [,2]
  0 5041.638 2880.570
  1 4965.571 2854.985
```

```
   attic
Y       [,1]     [,2]
  0 5020.756 2890.110
  1 4984.869 2922.181

   garage
Y       [,1]     [,2]
  0 552.3979 261.1439
  1 554.2938 259.2161

   price
Y      [,1]    [,2]
  0 4992784 2886369
  1 4948969 2851185
```

This is the Naive Bayes classifier. It is showing that the data that it received had an 87% chance of being basic. Price and square meters and rooms and floors being greater has a big impact on whether the house is luxury. THe values for the second column are much lower than the first column.

Hide

```
p1 <- predict(nb1, newdata=test, type="class")
table(p1, test$category)
```

```
p1     0    1
  0 1746  254
  1    0    0
```

Hide

```
mean(p1==test$category)
```

```
[1] 0.873
```

Hide

```
probs <- predict(glm1, newdata=test, type="response")
pred <- ifelse(probs>0.5, 1, 0)
acc <- mean(pred==test$category)
print(paste("accuracy = ", acc))
table(pred, test$category)
```

The Naiive Bayes received a very slightly higher accuracy by 0.3%. I think this happened because the features do have some independece but also are related to each other a little bit, like squareMeters and price. It only required an accuracy of 87.3%. However, this was greater than the accuracy of the Logistic moel which received an accuracy of 87%. This is probably because multicollinearity between suareMeters and price.

Naive Bayes is a simple and fast algorithm that is easy to implement and can handle high dimensional data with many variables. It performs well in the presence of irrelevant variables, as the algorithm assumes that variables are conditionally independent given the class label, so irrelevant variables are automatically filtered out. It is also

robust to noise and missing values in the data. However, Naive Bayes assumes that the variables are independent, which is not always the case in practice, leading to reduced accuracy in some datasets. The algorithm tends to be overly confident in its predictions, as it always chooses the class with the highest probability, even when the difference between the probabilities is small. Naive Bayes may perform poorly when the class distributions are imbalanced or when there is overlap between the classes.

Logistic regression, on the other hand, can model the probability of the outcome variable, which is useful for estimating the effects of predictors on the outcome. It is a powerful tool for modeling the relationship between the dependent and independent variables, and can handle both continuous and categorical variables. It is less prone to overfitting than Naive Bayes, and the coefficients can be interpreted easily. However, logistic regression can be sensitive to outliers and multicollinearity in the data. It may also require a larger sample size than Naive Bayes to achieve stable estimates. Logistic regression assumes a linear relationship between the independent and dependent variables, which may not hold in some cases, leading to reduced accuracy.

I used accuracy in logisic model and mean in naiive bayes model Accuracy is a widely used metric for classification and is easy to interpret. It provides a straightforward measure of how well a classification algorithm performs, as it represents the proportion of correct predictions over the total number of predictions. This metric is useful for binary and multiclass classification problems, as it takes into account both true positive and true negative predictions. However, accuracy may not be the best metric to use when the classes in the dataset are imbalanced, as a model can achieve a high accuracy by simply predicting the majority class all the time. It also does not take into account the false positive and false negative predictions, which are important in many applications. Therefore, accuracy should be used in combination with other classification metrics to provide a more comprehensive evaluation of the model’s performance.

Mean is a measure of central tendency of a distribution and is useful in many statistical analyses. It can provide information on the average value of a set of data points and can help identify outliers. However, mean is not always the best metric to use, as it can be sensitive to extreme values in the dataset, especially in the presence of outliers. In such cases, other measures of central tendency such as median or mode may be more appropriate. Mean is also not applicable to categorical or ordinal data, which are common in classification problems.