Additional Example

This example replicates what you need to do for assignment 2. Similar tasks are performed in this example, although using a different dataset.

Lets get startted by loading some data from the internet. This is the “adult” dataset that contains a number of predictors and the target variable if the individuals earn above $50k.

adult <- read.table('https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data',   
 sep = ',', fill = F, strip.white = T)  
  
colnames(adult) <- c('age', 'workclass', 'fnlwgt', 'educatoin',   
 'educatoin\_num', 'marital\_status', 'occupation', 'relationship', 'race', 'sex', 'capital\_gain', 'capital\_loss', 'hours\_per\_week', 'native\_country', 'income')  
  
summary(adult)

## age workclass fnlwgt educatoin   
## Min. :17.00 Length:32561 Min. : 12285 Length:32561   
## 1st Qu.:28.00 Class :character 1st Qu.: 117827 Class :character   
## Median :37.00 Mode :character Median : 178356 Mode :character   
## Mean :38.58 Mean : 189778   
## 3rd Qu.:48.00 3rd Qu.: 237051   
## Max. :90.00 Max. :1484705   
## educatoin\_num marital\_status occupation relationship   
## Min. : 1.00 Length:32561 Length:32561 Length:32561   
## 1st Qu.: 9.00 Class :character Class :character Class :character   
## Median :10.00 Mode :character Mode :character Mode :character   
## Mean :10.08   
## 3rd Qu.:12.00   
## Max. :16.00   
## race sex capital\_gain capital\_loss   
## Length:32561 Length:32561 Min. : 0 Min. : 0.0   
## Class :character Class :character 1st Qu.: 0 1st Qu.: 0.0   
## Mode :character Mode :character Median : 0 Median : 0.0   
## Mean : 1078 Mean : 87.3   
## 3rd Qu.: 0 3rd Qu.: 0.0   
## Max. :99999 Max. :4356.0   
## hours\_per\_week native\_country income   
## Min. : 1.00 Length:32561 Length:32561   
## 1st Qu.:40.00 Class :character Class :character   
## Median :40.00 Mode :character Mode :character   
## Mean :40.44   
## 3rd Qu.:45.00   
## Max. :99.00

Lets get rid of some of the variables that we do not want to include in our model. We can easily set these attributes to “NULL”. We also need to convert the charachter attributes to factor.

adult$fnlwgt<-NULL  
adult$educatoin <-NULL  
adult$occupation<-NULL  
adult$native\_country<-NULL  
adult$marital\_status<-NULL  
adult$race<-NULL  
adult$relationship<-NULL  
adult$workclass <-NULL  
  
adult$income=as.factor(adult$income)  
adult$sex=as.factor(adult$sex)  
  
summary(adult)

## age educatoin\_num sex capital\_gain   
## Min. :17.00 Min. : 1.00 Female:10771 Min. : 0   
## 1st Qu.:28.00 1st Qu.: 9.00 Male :21790 1st Qu.: 0   
## Median :37.00 Median :10.00 Median : 0   
## Mean :38.58 Mean :10.08 Mean : 1078   
## 3rd Qu.:48.00 3rd Qu.:12.00 3rd Qu.: 0   
## Max. :90.00 Max. :16.00 Max. :99999   
## capital\_loss hours\_per\_week income   
## Min. : 0.0 Min. : 1.00 <=50K:24720   
## 1st Qu.: 0.0 1st Qu.:40.00 >50K : 7841   
## Median : 0.0 Median :40.00   
## Mean : 87.3 Mean :40.44   
## 3rd Qu.: 0.0 3rd Qu.:45.00   
## Max. :4356.0 Max. :99.00

Now we call the caret library and the class library that we will later. Since we have an input categorical attribute (sex), we need to have that attribute converted into a dummy variable. After transformation, instead of one column gender that takes two values of male or female, we will have two attributes gender.male and gender.female that each take values of either 0 or 1. This makes it possible to apply the knn algorithm.

We will use the dummyVars function of the caret packge to create a model for this. We call the result model ‘dummies’ and then we apply it to our dataset using the predict() function. We then convert the result into a data frame (defualt is a matrix). Note that this has removed the income attribute from the results (we don’t want to convert this attribute as this is our target/output attribute).

Look at the first few rows of the results. We now have two separate columns sex.male and sex.female.

***Note*** You don’t have categorical predictors in your dataset for assignment 2, so this step could be skipped (you still need to call the libraries for the following tasks)

library(caret)

## Warning: package 'caret' was built under R version 4.1.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.1.2

## Loading required package: lattice

library(class)  
  
dummies <- dummyVars(income ~ ., data = adult)  
adult\_dummy=as.data.frame(predict(dummies, newdata = adult))

## Warning in model.frame.default(Terms, newdata, na.action = na.action, xlev =  
## object$lvls): variable 'income' is not a factor

head(adult\_dummy)

## age educatoin\_num sex.Female sex.Male capital\_gain capital\_loss  
## 1 39 13 0 1 2174 0  
## 2 50 13 0 1 0 0  
## 3 38 9 0 1 0 0  
## 4 53 7 0 1 0 0  
## 5 28 13 1 0 0 0  
## 6 37 14 1 0 0 0  
## hours\_per\_week  
## 1 40  
## 2 13  
## 3 40  
## 4 40  
## 5 40  
## 6 40

Next we need to normalize the data, or otherwise the calculation of distance will be affected by the unit of the variables. We can normalized in range 0-1 (min-max normalization or range normalization) or even better, we can normalize by removing the mean an dividing data by their standard deviation (z-transformation). We can use the preProcess function of caret for this.

***Important Note:*** Note that the preprocess function of caret creates a model. We will then need to apply this model to our data to apply the normalization effect. We will also keep this model and apply it to the future data that we want to pass to the model for predictions.

Now, look at the summary of the transformed dataset (i.e. adult\_norm), the mean of all columns are 0, as expected.

***Note***: If the method is set to “range” you would get the min-max normalization.

Norm\_model <- preProcess(adult\_dummy,   
 method = c("center", "scale"))  
adult\_norm=predict(Norm\_model,adult\_dummy)  
summary(adult\_norm)

## age educatoin\_num sex.Female sex.Male   
## Min. :-1.5822 Min. :-3.52960 Min. :-0.7031 Min. :-1.4223   
## 1st Qu.:-0.7758 1st Qu.:-0.42005 1st Qu.:-0.7031 1st Qu.:-1.4223   
## Median :-0.1160 Median :-0.03136 Median :-0.7031 Median : 0.7031   
## Mean : 0.0000 Mean : 0.00000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.6905 3rd Qu.: 0.74603 3rd Qu.: 1.4223 3rd Qu.: 0.7031   
## Max. : 3.7696 Max. : 2.30080 Max. : 1.4223 Max. : 0.7031   
## capital\_gain capital\_loss hours\_per\_week   
## Min. :-0.1459 Min. :-0.2167 Min. :-3.19398   
## 1st Qu.:-0.1459 1st Qu.:-0.2167 1st Qu.:-0.03543   
## Median :-0.1459 Median :-0.2167 Median :-0.03543   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.00000   
## 3rd Qu.:-0.1459 3rd Qu.:-0.2167 3rd Qu.: 0.36951   
## Max. :13.3944 Max. :10.5933 Max. : 4.74289

Now its time to add back the target attribute (i.e. income)

adult\_norm$income=adult$income

Now, lets start dividing the data into train and validation. Note that we also could have first split the data and then apply the dummy variable encoding and normalization steps individually to each of the train and validation sets. However, doing it this way, saves us time as we don’t need to repeat the same steps for each set separately.

We will use createDataPartition() function of the caret to divide the data. We can also pass a target variable (income) so that the split is not too imbalanced (i.e., the percentage of <50 and >50 are similar in both train and validation sets).

Train\_Index = createDataPartition(adult$income,p=0.6, list=FALSE) # 60% reserved for Train  
Train.df=adult\_norm[Train\_Index,]  
Validation.df=adult\_norm[-Train\_Index,]

***Task 1*** Use the train set and knn method with k=1 to predict if someone with the following attributes will earn more than $50k or not.

age=35, educatoin\_num=12, sex=Female, capital\_gain=1000, capital\_loss=0, hours\_per\_week=35

We first need to create a dataframe that contains the records of this individual (note how sex is dummy encoded)

To\_Predict=data.frame(age=35, educatoin\_num=12,  
 sex.Female=0,sex.Male=0,  
 capital\_gain=1000,capital\_loss=0,  
 hours\_per\_week=35)  
  
print(To\_Predict)

## age educatoin\_num sex.Female sex.Male capital\_gain capital\_loss  
## 1 35 12 0 0 1000 0  
## hours\_per\_week  
## 1 35

Now, we need to apply the normalization to this record. We MUST use the same model that we applied to the original dataset for norlamization:

To\_Predict\_norm=predict(Norm\_model,To\_Predict)  
print(To\_Predict\_norm)

## age educatoin\_num sex.Female sex.Male capital\_gain capital\_loss  
## 1 -0.2625757 0.7460277 -0.7030605 -1.422309 -0.01051398 -0.2166562  
## hours\_per\_week  
## 1 -0.4403715

Now, we will use the knn function to make the prediction. Note that the first 7 columns are our predictors (input attributes) and income is our target or label (Train.df$income is the same as Train.df[,8]). We can see that the prediction is that this individiual will earn less than $50k.

Prediction <-knn(train=Train.df[,1:7],   
 test=To\_Predict\_norm[,1:7],  
 cl=Train.df$income,  
 k=1)  
print(Prediction)

## [1] <=50K  
## Levels: <=50K >50K

***Task 2*** What is the best choice of k that balances between overfitting and underfiting?

There are different approaches you can use here. One is to manually write a for-loop and try different values of k and see which work best of the validation set (example file on the course github).

However, the caret package has made life easier here. The train function of the caret package can be used to try different values of hyper-parameters for different algorithms (200+ algorithms, including knn!). It does a great job of cross-fold validation (and repeated cross fold validation) behind the scene and will tell us what is the best hyper-parameter and will automatically use that value and train a model for us (You are welcome!).

All what we need to do, is to tell the train function what form of crodss validation we want and the range of the k values (our hyper-parameter). In this examples we use 2 repeats of 3 fold cross validation (more fold and repeats are better but will take longer). We also try k from 1 to 10. You have to be patient since this part may take a while to calculate (we are building 6 models (2\*3) for each k).

As you can see, The final value used for the model was k = 10, which has the best accuracy. We save the best model into the Knn.model object (can be any other name too)

set.seed(123) # setting the seed of the random number generator will make sure that the results are reproducible  
  
fitControl <- trainControl(method = "repeatedcv",  
 number = 3,  
 repeats = 2)  
  
searchGrid=expand.grid(k = 1:10)  
  
Knn.model=train(income~.,   
 data=Train.df,  
 method='knn',  
 tuneGrid=searchGrid,  
 trControl = fitControl,)  
  
Knn.model

## k-Nearest Neighbors   
##   
## 19537 samples  
## 7 predictor  
## 2 classes: '<=50K', '>50K'   
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold, repeated 2 times)   
## Summary of sample sizes: 13024, 13025, 13025, 13024, 13025, 13025, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 1 0.7995343 0.4222494  
## 2 0.8017351 0.4249543  
## 3 0.8159900 0.4538900  
## 4 0.8168602 0.4549895  
## 5 0.8221067 0.4664572  
## 6 0.8222859 0.4657553  
## 7 0.8242565 0.4692863  
## 8 0.8246914 0.4693069  
## 9 0.8257152 0.4717993  
## 10 0.8255104 0.4709171  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 9.

***Task 3*** Show the confusion matrix for the validation data that results from using the best k.

Now that we have our care model for the best k (Knn.model), we can use the predict function of the caret package to make predictions on validation set. Yes, caret makes life easy!

predictions<-predict(Knn.model,Validation.df)

Now, we need to compare the predictions against the actual income labels in the validation set to compute the Confusion matrix! Again, the caret package has a function for this, called confusionMatrix().

confusionMatrix(predictions,Validation.df$income)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction <=50K >50K  
## <=50K 9207 1548  
## >50K 681 1588  
##   
## Accuracy : 0.8289   
## 95% CI : (0.8223, 0.8353)  
## No Information Rate : 0.7592   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4831   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9311   
## Specificity : 0.5064   
## Pos Pred Value : 0.8561   
## Neg Pred Value : 0.6999   
## Prevalence : 0.7592   
## Detection Rate : 0.7069   
## Detection Prevalence : 0.8258   
## Balanced Accuracy : 0.7188   
##   
## 'Positive' Class : <=50K   
##

***Task 4*** Consider another individual with the following attributes:

age=55, educatoin\_num=16, sex=Female=1, capital\_gain=10000,capital\_loss=0, hours\_per\_week=55

Use the m odel with the best k to predict if this individual will earn above or below $50k.

Same as before, all we need to do is to convert the information into a dataframe and then normalize it and then to us the predict function of caret to apply the best model on it! We can see that this individual is predicted to earn above 50k.

To\_Predict=data.frame(age=55, educatoin\_num=16,  
 sex.Female=1,sex.Male=0,  
 capital\_gain=10000,capital\_loss=0,  
 hours\_per\_week=55)  
  
To\_Predict\_norm=predict(Norm\_model,To\_Predict)  
predict(Knn.model,To\_Predict\_norm)

## [1] >50K  
## Levels: <=50K >50K