FINAL EXAM

Luis J. Cuervo

Question 1

Data analysis can help to further understand your data, as well as to predict some features of new data, based on the other features.

Here are three examples:

Can I group my customers?

It could be the case where we would want to identify groups within the dataset in order to understand how the people of this dataset are related with one another. Using the clustering methodology, the algorithm would group similar data into as many groups (clusters) as we want.

This could help us to identify different profiles in the dataset and maybe develop specific products or services targeting each of these profiles.

Can I predict credit risk?

It is possible. By developing a classification algorithm, we would be able to predict whether new customers have good or bad credit risks. As it is stated in the example question, we would develop a decision tree algorithm where class is the dependent variable and the rest of variables are independent.

This methodology would also help to understand which are the important. By plotting the structure of the tree we would be able to identify the features that are most important in the tree. Those who spread positive and negative data the most will have high relevance in the categorization.

Can I identify tendencies in specific groups of people?:

We may be interested in looking at specific groups of the dataset that we are interested in. For example, we may be interested in looking at the customers with ages between 18 and 30, and study their saving status or some other parameters. This way we would understand their situation and needs, so maybe we could develop a marketing strategy with this info in mind.

This can be achieved very easily by building a dataframe with our data and then manually fixing the conditions that we want to study, while getting rid of those that we consider irrelevant.

CASE 1: German credit dataset

Classification

We will now try to predict credit risk:

```
# 1 Read dataset
ds = read.csv("credit-g.csv", sep=",", header = TRUE)
head(ds)
```

```
##   checking_status duration                  credit_history            purpose
## 1            '<0'        6 'critical/other existing credit'           radio/tv
## 2      '0<=X<200'       48                  'existing paid'           radio/tv
## 3   'no checking'       12 'critical/other existing credit'          education
## 4            '<0'       42                  'existing paid' furniture/equipment
## 5            '<0'       24             'delayed previously'          'new car'
## 6   'no checking'       36                  'existing paid'          education
##   credit_amount     savings_status employment installment_commitment
## 1          1169 'no known savings'      '>=7'                      4
## 2          5951             '<100'   '1<=X<4'                      2
## 3          2096             '<100'   '4<=X<7'                      2
## 4          7882             '<100'   '4<=X<7'                      2
## 5          4870             '<100'   '1<=X<4'                      3
## 6          9055 'no known savings'   '1<=X<4'                      2
##        personal_status other_parties residence_since  property_magnitude age
## 1        'male single'          none               4       'real estate'  67
## 2 'female div/dep/mar'          none               2       'real estate'  22
## 3        'male single'          none               3       'real estate'  49
## 4        'male single'     guarantor               4    'life insurance'  45
## 5        'male single'          none               4 'no known property'  53
## 6        'male single'          none               4 'no known property'  35
##   other_payment_plans    housing existing_credits                  job
## 1                none        own                2              skilled
## 2                none        own                1              skilled
## 3                none        own                1 'unskilled resident'
## 4                none 'for free'                1              skilled
## 5                none 'for free'                2              skilled
## 6                none 'for free'                1 'unskilled resident'
##   num_dependents own_telephone foreign_worker class
## 1              1           yes            yes  good
## 2              1          none            yes   bad
## 3              2          none            yes  good
## 4              2          none            yes  good
## 5              2          none            yes   bad
## 6              2           yes            yes  good
```

```
ds$class <- as.factor(ds$class)

l_b = length(ds$class[ds$class == "bad"])
l_g = length(ds$class[ds$class == "good"])

# Number of people with class "bad"
l_b
```

```
## [1] 300
```

```
# Number of people with class "good"
l_g
```

```
## [1] 700
```

We can see that our dataset is not balanced, we have more good classes than bad classes. We want to work with a balanced dataset so that the algorithm does not tend to predict data as good to achieve higher accuracy. Also, balancing the data will make are accuracy metrics more reliable.

```
ds.good = which(ds$class=="good")[1:(l_g - l_b)]

ds <- ds[-c(ds.good),]
length(ds$class[ds$class == "bad"])
```

```
## [1] 300
```

```
length(ds$class[ds$class == "good"])
```

```
## [1] 300
```

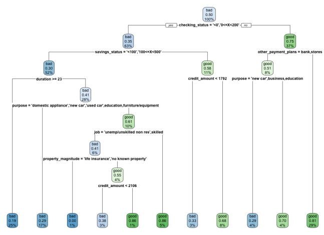
```
# Now it is balanced.

# Split dataset:
train <- createDataPartition (y=ds$class, p=0.8,list=FALSE)
ds.train <- ds[train,]
ds.test <- ds[-train,]

# Train decision tree:
tree <- rpart(class ~ ., ds.train, method="class")
tree
```

```
## n= 480
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 480 240 bad (0.5000000 0.5000000)
##     2) checking_status='<0','0<=X<200' 303 107 bad (0.6468647 0.3531353)
##       4) savings_status='<100','100<=X<500' 250  76 bad (0.6960000 0.3040000)
##         8) duration>=22.5 118  22 bad (0.8135593 0.1864407) *
##         9) duration< 22.5 132  54 bad (0.5909091 0.4090909)
##          18) purpose='domestic appliance','new car','used car',education,furniture/equipment 83  24 bad (0.710
8434 0.2891566) *
##          19) purpose=business,radio/tv,repairs,retraining 49  19 good (0.3877551 0.6122449)
##            38) job='unemp/unskilled non res',skilled 27  11 bad (0.5925926 0.4074074)
##              76) property_magnitude='life insurance','no known property' 7   0 bad (1.0000000 0.0000000) *
##              77) property_magnitude='real estate',car 20   9 good (0.4500000 0.5500000)
##               154) credit_amount< 2105.5 13   5 bad (0.6153846 0.3846154) *
##               155) credit_amount>=2105.5 7   1 good (0.1428571 0.8571429) *
##            39) job='high qualif/self emp/mgmt','unskilled resident' 22   3 good (0.1363636 0.8636364) *
##       5) savings_status='>=1000','500<=X<1000','no known savings' 53  22 good (0.4150943 0.5849057)
##        10) credit_amount< 1791.5 15   5 bad (0.6666667 0.3333333) *
##        11) credit_amount>=1791.5 38  12 good (0.3157895 0.6842105) *
##     3) checking_status='>=200','no checking' 177  44 good (0.2485876 0.7514124)
##       6) other_payment_plans=bank,stores 37  18 good (0.4864865 0.5135135)
##        12) purpose='new car',business,education 17   5 bad (0.7058824 0.2941176) *
##        13) purpose='used car',furniture/equipment,radio/tv 20   6 good (0.3000000 0.7000000) *
##       7) other_payment_plans=none 140  26 good (0.1857143 0.8142857) *
```

```
rpart.plot(tree)
```



The decision tree algorithm is represented here, where the input is at the top of the tree and it goes down through the branches until it is classified on one of the leaves at the bottom. The numbers at center of each leaf or node tell us the amount of files that reach them having class good. It can also be interpreted as the probability of the output being good once that node is reached. Therefore, we will want a high number on the leaves that classify as good, and a low number of the leaves that classify as bad so that the algorithm is certain about the labeling that it makes. We can see that is the case in most part of our tree.

On the other hand, the percentage at the bottom represents the amount of data that has reached that specific leaf or node (Percentage over the total amount of data samples).

Looking at the top of the decision tree plot we can see that initially all data have a 50% chance of being class bad, which makes sense since our data is balanced. Then the first node checks the checking status, and if it satisfies the stated condition, it will go to the left where we it will have a 36% chance of having good credit risk. So the checking status has high importance regarding credit risk.

To further analyze the rest of the tree we must only look at each branch of the tree like we just did, knowing that lines going to the left of each node satisfy the stated condition while those that go to the right don't.

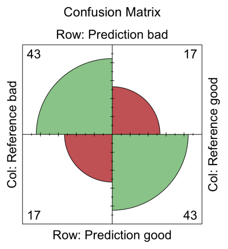
Let's now proceed to evaluate the algorithm.

First we will look at the confusion matrix:

```
predictions <- predict(tree, ds.test, type = "class")
cm = confusionMatrix(predictions, ds.test$class)

ctable <- as.table(matrix(cm[[2]], nrow = 2, byrow = TRUE, dimnames=list(list("Prediction bad", "Prediction goo
d"), list("Reference bad", "Reference good"))))

fourfoldplot(ctable, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1, main = "Confusion Matrix")
```



This figure shows the correct predictions in green and the wrong predictions in red, where columns represent the real value of the data labels and rows the predicted value. We can see that the green values clearly overrule the red values, so our model starts with a good foot. Also, the two green circles have similar value, so our algorithms is not more inclined to label the data as positive or negative, which is also good.

Let's study now accuracy metrics:

```
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   43   17
##       good  17   43
##
##                Accuracy : 0.7167
##                  95% CI : (0.6272, 0.7951)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 1.138e-06
##
##                   Kappa : 0.4333
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7167
##             Specificity : 0.7167
##          Pos Pred Value : 0.7167
##          Neg Pred Value : 0.7167
##              Prevalence : 0.5000
##          Detection Rate : 0.3583
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.7167
##
##        'Positive' Class : bad
##
```

Our algorithm stays around 70% of accuracy (we performed more than one execution on the model), which means that 70% of the data is labeled correctly. That is quite high, given that it is a complex tree.

Also, we consider our result satisfactory because we have high sensitivity and specificity. Sensitivity measures the amount of correctly predicted positive data (true positives) over the total amount of positive labeled data (total positives). In this case, the positive class is "bad". Specificity is the same but with negative data. So, just like we mentioned earlier, the fact that both green circles have similar size is represented in the sensitivity and specificity having similar values. Again, this will mean that our algorithm is not inclined to label data as positive or negative.

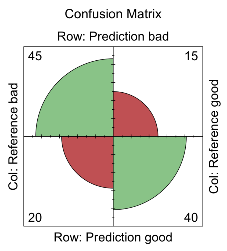
KNN

```
knn.fit <- train(class~ ., data=ds.train, method="knn", preProcess=c("center", "scale"))
knn.fit
```

```
## k-Nearest Neighbors
##
## 480 samples
##  20 predictor
##   2 classes: 'bad', 'good'
##
## Pre-processing: centered (48), scaled (48)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 480, 480, 480, 480, 480, 480, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.6027539  0.2071095
##   7  0.6141543  0.2301436
##   9  0.6204261  0.2429962
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```
knn.predict <- predict(knn.fit, newdata=ds.test)
CM = confusionMatrix (knn.predict, ds.test$class)

ctable <- as.table(matrix(CM[[2]], nrow = 2, byrow = TRUE, dimnames=list(list("Prediction bad", "Prediction goo
d"), list("Reference bad", "Reference good"))))
fourfoldplot(ctable, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1, main = "Confusion Matrix")
```



```
CM
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction bad good
##       bad   45   20
##       good  15   40
##                                           
##                Accuracy : 0.7083          
##                  95% CI : (0.6184, 0.7877)
##     No Information Rate : 0.5             
##     P-Value [Acc > NIR] : 2.866e-06       
##                                           
##                   Kappa : 0.4167          
##                                           
##  Mcnemar's Test P-Value : 0.499           
##                                           
##             Sensitivity : 0.7500          
##             Specificity : 0.6667          
##          Pos Pred Value : 0.6923          
##          Neg Pred Value : 0.7273          
##              Prevalence : 0.5000          
##          Detection Rate : 0.3750          
##    Detection Prevalence : 0.5417          
##       Balanced Accuracy : 0.7083          
##                                           
##        'Positive' Class : bad             
## 
```

We obtained accuracy values that are not so different from the decission tree algorithm (around 65% of accuracy). However, most of the times the obtained sensitivity is considerably different from specificity, menaing that there is a tendency towards possitive results if sensitivity is higher, or towards negative results if specificity is higher.

Which is better?

We come to the conclusion that the decision tree algorithm is better in all aspects. The average accuracy of all the trainings that we have done is higher than that of knn. Furthermore, it is less computationally expensive and it allows interpretation of the model. KNN on the other hand gives more unbalanced results, requires more computational power and acts as a black box, meaning that we don’t really know what is going on when the algorithm is implemented.

Analysis explanation

We have developed a rather satisfactory methodology of predicting credit risk. By feeding it other information about the person (such as his/her existing credit, age, personal status…) the program will categorize the person as good or bad credit risk.

This algorithm was trained on real data, meaning that it can be trained further if we obtain more data in the future. With more training it is very likely to provide better results. That is one more reason for the bank to take good care of its data.

However, this algorithm is not 100% reliable (we saw that it achieves around 70% of accuracy). That is why it would be convenient for a human being to overlook the results of the algorithms, specially when considering giving big amounts of credit. Nevertheless it can be a useful tool to use and it can be used to make instant decisions when considering small amounts of credit.

CASE 2: Insurance Company

Why is regression appropiate?

A regression analysis is useful to predict a numeric value based on other parameters, under the assumption that the dependent value can be predicted based on the other independent variables. In this case we want to predict the final amount of compensation based on other parameters.

Linear regression is more than able to do so and it will give us perspective on what parameters are really important for the prediction.

How is it different from the previous case?

The main difference between this case and the previous one is that the output data is not categorical like in the previous case. Instead it is numerical. So the prediction made by the algorithm will not be a class label like “good” or “bad”, it will be a specific number with infinite possible values.

Also, the very structure of the algorithm is different. Where as previously we either had the tree structure of the decision tree or the black box of knn, the model here will be quite simpler. We will just have a series of parameters that when multiplied by their corresponding independent variable value and added up together, they will result in the numerical value that we are looking for

Procedure

```
df<-read.csv("insurance.csv", sep=",", stringsAsFactors=TRUE)
df$ClaimNumber = NULL
df$DateReported = NULL
df$DateTimeOfAccident = NULL
df$ClaimDescription = NULL

# Regression algorithm
model = lm(UltimateIncurredClaimCost ~ ., data = df)
model
```

```
## 
## Call:
## lm(formula = UltimateIncurredClaimCost ~ ., data = df)
## 
## Coefficients:
##             (Intercept)                      Age  
##               2.2516545                0.0071958  
##                 GenderM           MaritalStatusS  
##              -0.1886459               -0.0218987  
##          MaritalStatusU        DependentChildren  
##               0.1928446                0.1234945  
##         DependentsOther              WeeklyWages  
##               0.2666938                0.0009025  
##       PartTimeFullTimeP         HoursWorkedPerWeek  
##               0.0751282               -0.0060219  
##       DaysWorkedPerWeek  InitialIncurredCalimsCost  
##              -0.0640052                0.7473189
```

By looking at the coefficients drawn by the regression model, we could come to the conclusion that the higher the coefficients, the more relevant their corresponding predictor would be. However that is not necessarily true because these predictors could have different order of magnitudes. For example, the values of the variable HoursWorkedPerWeek will almost always be higher than that of DaysWorkedPerWeek and therefore the resulting contribution to the linear regression will be higher even if they were to have similar coefficients.

Instead, we will look at the p-value of each predictor variable. Variables with a high p-value will likely have no effect on the independent variable (It tests the null hypothesis)

```
summary(model)
```

```
## 
## Call:
## lm(formula = UltimateIncurredClaimCost ~ ., data = df)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -3.7357 -0.4496 -0.0571  0.3574  6.5032 
## 
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                2.252e+00  2.559e-01   8.800  < 2e-16 ***
## Age                        7.196e-03  1.892e-03   3.804 0.000147 ***
## GenderM                   -1.886e-01  4.800e-02  -3.930 8.79e-05 ***
## MaritalStatusS            -2.190e-02  4.855e-02  -0.451 0.652012    
## MaritalStatusU             1.928e-01  6.920e-02   2.787 0.005377 ** 
## DependentChildren          1.235e-01  4.075e-02   3.031 0.002472 ** 
## DependentsOther            2.667e-01  1.676e-01   1.591 0.111805    
## WeeklyWages                9.025e-04  9.207e-05   9.803  < 2e-16 ***
## PartTimeFullTimeP          7.513e-02  9.221e-02   0.815 0.415337    
## HoursWorkedPerWeek        -6.022e-03  4.116e-03  -1.463 0.143612    
## DaysWorkedPerWeek         -6.401e-02  5.097e-02  -1.256 0.209388    
## InitialIncurredCalimsCost  7.473e-01  1.309e-02  57.109  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.8578 on 1988 degrees of freedom
## Multiple R-squared:  0.6986, Adjusted R-squared:  0.6969 
## F-statistic: 418.8 on 11 and 1988 DF,  p-value: < 2.2e-16
```

R already marks the most relevant predictors (those with the lowest P-value) with asterisks. We will consider Age, Gender, WeeklyWages and InitialIncurredCalimsCost as the most relevant variables, followed by MaritalStatusU and DependentChildren.

```
model2 = lm(UltimateIncurredClaimCost ~ Age + Gender + WeeklyWages + InitialIncurredCalimsCost + MaritalStatus +
DependentChildren, data = df)

coefficients = summary(model2)$coefficients
coefficients
```

```
##                              Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept)                1.7526483734 1.282120e-01 13.6699242 9.947429e-41
## Age                        0.0071681175 1.899102e-03  3.7744770 1.650347e-04
## GenderM                   -0.2265705610 4.706707e-02 -4.8137812 1.592469e-06
## WeeklyWages                0.0007910944 8.690996e-05  9.1024598 2.084067e-19
## InitialIncurredCalimsCost  0.7529299360 1.305467e-02 57.6751578 0.000000e+00
## MaritalStatusS            -0.0196812653 4.867766e-02 -0.4043182 6.860221e-01
## MaritalStatusU             0.2033056904 6.936642e-02  2.9308950 3.418361e-03
## DependentChildren          0.1346506719 4.052749e-02  3.3224531 9.084389e-04
```

```
#We will predict the Dependent variable using a linear regression formula like the following:

sprintf('y = %.3f + %.3f*Age - %.3f*Gender + %.3f*WeeklyWages + %.3f*CalimsCost - %.3f*M.S.S + %.3f*M.S.U + %.3f*
DependentChildren + error', coefficients[1], coefficients[2], coefficients[3], coefficients[4], coefficients[5],
coefficients[6], coefficients[7], coefficients[8])
```

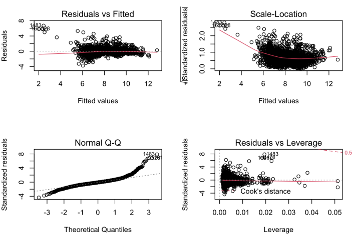
```
## [1] "y = 1.753 + 0.007*Age - -0.227*Gender + 0.001*WeeklyWages + 0.753*CalimsCost - -0.020*M.S.S + 0.203*M.S.U
+ 0.135*DependentChildren + error"
```

Evaluate model:

We will now study the quality of the model. First, by looking at the median value of the residuals, we can see that it is quite close to zero, which means that the “line” drawn by the model is quite centered in respect to the data (because the mean distance of the points to the line is close to zero).

Then, of course, we will look at the R-squared value. It is 0.695, an acceptable value although it could be higher (so we must not worry about over-fitting). It means that most of the points will fall close to the line. Let’s look at the residual plots:

```
layout(matrix(1:4,2,2))
plot(model2)
```



Residuals versus fitted seams quite linear, meaning we don’t have non linear relationships. Normal Q-Q is for the most part linear, which is good Scale-location does not seem good though. All the fitted values are positives and the line is not completely horizontal. This mean that the variables do not have the same variance. Residuals vs Leverage shows that there are not many outliers (the cook’s distance line can barely be seen). This makes sense given that we previously saw that the max value of the residuals was 6.7

All in all we could say that is a fairly good model.

Invented case

```
invented.case = data.frame(Age=21, Gender="F", WeeklyWages= 320, InitialIncurredCalimsCost=7, MaritalStatus="M",
DependentChildren=1)

# Prediction
predict(model2, invented.case)
```

```
##        1 
## 7.561489
```

Case 3. Customer Segmentation at RetailMart

Is clustering a good option?

Clustering is the perfect procedure for this objective. It will define a similarity metric between the data and group all the data samples into as many groups as we ask based on similarity.

However, some interpretation must be done on the clusters. The algorithm will group the data, yes, but it is up to us to deduce what are the characteristics of these groups This should be rather easy given that we will only work with three variables.

How is clustering different?

Clustering belongs to a branch of machine learning known as unsupervised ML. It is differentiated from the previous methods in the fact that its data is not labeled. Our previous data already had the independent variable stated, we trained the algorithm by showing it various examples with different labels. However, now our data is unlabeled. We are asking the algorithm to find the clusters on its own, without it knowing the correct answer to the question.

Explain the results

Clustering works by trying to assign to each cluster a centroid (a value of the variables that whose neighbors will be included in the cluster) for each variable. It starts by randomly selecting the centroids and finding the nearest values to each centroid. Then, it assigns the new centroid at the “mass center” of each resulting cluster and repeat the procedure until the clusters don’t change. These centroids are shown in the first table of the results shown. They are useful to deduce the separation of each cluster, if wee look at how apart the centroids are from one another.

Looking at the pie chart, we can see the amount of data samples that fall into each cluster: 42% to cluster0, 18.2% to cluster 1 and 39.1% to cluster2. (NOTE: We considered that the pie charts corresponds to the bar plots based on the indexes since the colors and the indexes do not match with those of the bar plots)

Let’s now look at the boxplots. Boxplots are very useful to deduce the distance between data samples within the clusters. If the distance is small we will say that the elements of the cluster are homogeneous.

Let’s focus first on cluster 0: it is homogeneous on amount and frequency, but it is not homogeneous in recency. This means that 42% of the data have very similar values of amount and frequency. In fact, all the elements of this cluster seem to have the same value for frequency (something around 1).

Cluster1 on the other hand only is homogeneous on recency. For amount, only the first three quarters of the data stay between 0 and 59.7k, the 4th quarter gets as high as 19.91k. Frequency is also very heterogeneous. We could say that in a way this cluster is the opposite of cluster0 in terms of homogeneity, and if we look at the position of the centroids we can see that they are very far away from those of cluster0.

Finally, Cluster2 is the most balanced in terms of homogeneity. It is quite similar to cluster0 (the centroids are quite close too) with the difference that it is less homogeneous in amount and frequency, but more in recency. But the main difference relies on the fact that frequency is different than that fixed value of the elements of cluster0.

So basically there are three kinds of customers:

Type 0 (cluster 0): Esporadic customers

Customers that have come one or two times to Walmart, have spent a small amount of money and came at very different points of time. They represent 42.6% of the customers

Type 1 (cluster 1): Loyal customers

They come frequently to the store, have spent the largest amount of money and came for the last time very recently. 18.2% of the customers

Type 2 (cluster 2): Frequent customers

Repeated customers. Customers that are in between the other two types. They spend medium/small amounts of money, come every now and then to the store and most of them came to the store not so long ago.