Report Format

Thursday, June 6, 2019 7:25 PM

Disclaimer:
These notes were created prior to the June 2019 Sitting. The syllabus has likely changed since this version of the exam. See the latest syllabus as well as the June 2019 solution once it is released for the latest information.

Executive Summary, Findings and Recommendations

1) State the business problem and how this adds ($$$) value
2) Err on the side of redundancy with the rest of the report (graders take off points for what is **not** included)
3) Talk at a high level about the modeling process. Use simple words. Write in Crayon.

   i.e., "Two models were used; a decision tree to predict X and a GLM to predict Y. The decision tree highlights interaction effects. For instance, for modeling healthcare costs, a person with heart disease will be impacted differently by high blood pressure than a person without heart disease. Decision trees pick this up."

   5) Highlight key takeaways from the model

   From the GLM, these five factors most impact the business outcome

   i. Variable X1 – Y increases as X1 decreases
   ii. Variable X2 – People with a high X2 have a high Y
   iii. Variable X3- …

4) Depending on the model, include an interpretation
   a. For a GLM with a continuous response, include a table with a set of "rules"
   b. For a glm with a categorical response (logistic regression), include a table that shows how the predicted value changes
      i. Log-odds needs to be transformed to probability
      ii. With log link, percent change in Xi changes the probability of the outcome by Bi percent. Example: https://stats.idre.ucla.edu/r/dae/logit-regression/
      iii. With non-log link, there is no easy interpretation
   c. For a decision tree, include a set of yes/no questions

5) Narrate the data process. Make it sound like it was difficult. Explain how you fixed things. **Tell the grader that you found their planted errors (See "Common Traps by the SOA" section).**
6) Mention all potential limitations of the analysis such as
   a. Model results limited to the range of the training data
   b. Change in data over time (covariate shift)
   c. Selection bias (how the data was collected)
   d. Legal issues (was personal information used? Would this be attainable in a real

application?)

Data Exploration and Feature Selection

Tips to save time with EDA:

7) To find counts by categorical variables, use the dplyr `count()` function instead of the *terrible* `table()` function that they tell you to use. https://dplyr.tidyverse.org/reference/tally.html
8) Do not waste time formatting in Excel. Just paste copy/paste results directly into Word. Aesthetics are not graded except in the Executive Summary

What the grader wants to see:

9) A histogram of the response (Y) distribution
10) That you removed the planted errors that the SOA put in the data
11) Plots of any variables that were transformed (i.e., log transform)
12) Words. Narrate the process.
    a. A long, long, time ago, I was given data by the SOA
    b. The SOA wants to fail candidates, and so they put mistakes in the data and template code
    c. …(Shows histogram of errors)…
    d. Then I found that X23 was right-skewed and so a log transform was applied
    e. …(Shows before and after histogram)..

Model Selection and Validation: Decision Tree

Explain the pros and cons of the modeling approaches used. Imagine that you the SOA did not tell you what model to use but that you *chose* to use the given model approach. Explain that there is no "silver bullet" of modeling.

Decision Trees

Pros

1. Easy to interpret
2. Captures interaction effects
3. Captures non-linearities
4. Handles continuous and categorical data
5. Handles missing values

Cons

6. Is a "weak learner" because of low predictive power
7. Does not work on small data sets
8. Is often a simplification of the underlying process because all observations at terminal nodes have equal predicted values

9. Is biased towards selecting high-cardinality features because more possible split points for these features tend to lead to overfitting
10. High variance (which can be alleviated with stricter parameters) leads the "easy to interpret results" to change upon retraining
11. Unable to predict beyond the range of the training data for regression (because each predicted value is an average of training samples)

Random Forests

Pros

12. Resilient to overfitting due to bagging
13. Only one parameter to tune (mtry, the number of features considered at each split)
14. Very good a multi-class prediction
15. Nonlinearities
16. Interaction effects
17. Deal with unbalanced and missing data*Usually requires over/undersamplin

Cons

18. Does not work on small data sets
19. Weaker performance than other methods (GBM, NN)
20. Unable to predict beyond training data for regression

GLMs

Pros

21. Easy to interpret
22. Handles skewed data through different response distributions
23. Models the average response which leads to stable predictions on new data
24. Handles continuous and categorical data
25. Works well on small data sets

Cons

26. Strict assumptions (see "what to include" below)
27. Unable to detect non-linearity directly (although this can manually be addressed through feature engineering)
28. Sensitive to outliers
29. Low predictive power

Elastic Net/Lasso/Ridge

Pros

30. All benefits from GLMS
31. Automatic variable selection
32. Better predictive power than GLM (usually)

Cons

33. All cons of GLMs

GBMs

Pros

34. High prediction accuracy
35. Closest model to a "silver bullet" that exists
36. Nonlinearities, interaction effects, resilient to outliers, corrects for missing values
37. Deals with class imbalance directly through by weighting observations

Cons

38. Requires large sample size
39. Longer training time
40. Does not detect linear combinations of features. These must be engineered
41. Can overfit if not tuned correctly

Common Traps by the SOA

Friday, June 7, 2019 2:32 PM

Trap 1: Having a GLM with a rank-deficient fit

Solution: One of the variables must be a linear combination of the others. For example, in the December exam, there were percentage columns that all added up to one.

P1+P2+P3 = 1

Removing one of these columns fixes the issue.

Trap 2: Not scaling variables prior to running kmeans or PCA.

Solution: Scale with the scale() function

Trap 3: Including obvious errors in data. For example, in the December exam, there were several ways of spelling the same names of mines. Sometimes they were called "closed" and others "abandoned".

Solution: Rename them

Trap 4: Including blatant outliers in data. For instance, a person with a negative age

Solution: Remove them **and tell the grader that you did**

Trap 5: Including data that is illegal to use in modeling such as racial status

Solution: Remove this from the model **and tell the grader that you did**

Trap 6: Including snippets of code which contain errors

Solution: Read the documentation using ?function_name for any code snippets provided in the .Rmd file

Model Building

Friday, June 7, 2019 2:33 PM

- Insure that reference levels are set to those with the most exposure. This should be done automatically.
    - Using dplyr: mutate(factor_with_ordered_levels = fct_infreq(factor))
    - For character factors, the default is by alphabetical order
    - For numeric "factors", the default is the lowest ordinal value
- For glmnet, is it not possible to fit non-normal (i.e., gamma, poisson, inverse gaussian, etc) distributions. If this is needed, use glm()
- If using a GLM for a strictly positive quantity (i.e., clalims), then use a distribution family which is strickly positive (i.e., gamma, inverse gausian, lognormal)
- Use offsets with Poisson GLMs when exposure is involved.
- Recall the assumptions of linear models
    - Errors centered at 0
    - Errors have no autocorrelation (time component)
    - The variance is constant (homoskedastic)
    - No key variables omitted
    - No correlation between predictors
    - No correlation between predictors with error
    - Features are not strongly correlated
    - No time trend (autocorrelation)
- Be able to interpret all GLM types
    - Continuous outcome with log-link -> exponent of the coefficient is the multiplicative change in the response. For log-transformed variables, this becomes a power transform x^b where b is the original model output.
    - Binary outcome with logit or probit link -> No easy interpretation. Look at the examples of observations and test what factors increase or decrease the probability of the outcome.
- Be able to fit kmeans and hclustering
    - Ensure variables are centered and scaled first
- Be able to perform PCA

Rstudio Tips

Sunday, June 9, 2019 11:25 AM

1) My preference is to force Rstudio to put output in the Console and View Pane. This makes it easier to copy results to clipboard and doesn't auto-scroll when running chunks.

https://bookdown.org/yihui/rmarkdown/notebook.html

By default, RStudio enables inline output (Notebook mode) on all R Markdown documents, so you can interact with any R Markdown document as though it were a notebook. If you have a document with which you prefer to use the traditional console method of interaction, you can disable notebook mode by clicking the gear button in the editor toolbar, and choosing `Chunk Output in Console` (Figure 3.4).

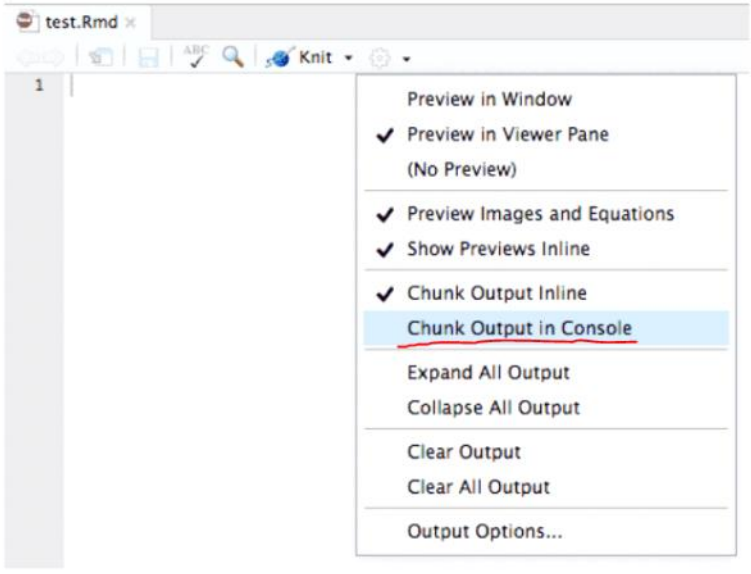

FIGURE 3.4: Send the R code chunk output to the console.

2) To save time, copy and paste directly from the R console into Word. Then change the font or reformat in word directly

3) On the actual exam, Prometric did give me a printed out version of the Problem Statement. This was easier than in practice where I was switching back and forth between two windows on a single monitor. To make practice more realistic, print out the problem statement first during practice runs.

Base R

Sunday, June 30, 2019 7:40 PM

A note on the code sections:
These are very incomplete. The best way to learn code is to read the documentation and test out simple examples. To learn about a specific function, type ?function_name into R.

Those serious about learning R should read https://r4ds.had.co.nz/

Apply a function to a vector

```
sapply(1:3, function(x) x^2)
data.frame(mean = sapply(data.AQ, mean),
        var = sapply(data.AQ, var))
```

Arithmetic

%% Modulo
^ Power

Vector comparisons

```
c(4, 5, 6) > 5

seq(2, 3, by=0.5)    #2.0 2.5 3.0
x[-4]                #all but the fourth element
x[x < 0]  or x[x %in% c(2,3,4)]        #filter elements of x
m[2,]                #row
m[,2]                #column
```

Create a new data frame

```
planets_df <- data.frame(name,type,diameter,rotation,rings)
summary(planets_df)
glimpse(planets_df)
```

Select rows

```
subset(planets_df,diameter < 1)
termLife[termLife$FACE>0, ]            #REMEMBER THE COMMA
```

Select columns

```
planets_df[c(1,2,3)]
planets_df[1:3]
planets_df[c("Name1","Name2")]
pairs(df[,c(4,5,7)])
```

GLMs

Sunday, June 30, 2019 7:42 PM

Read the CAS monograph on GLMs from beginning to end.

https://www.casact.org/pubs/monographs/papers/05-Goldburd-Khare-Tevet.pdf

Summary of families and link functions

"Family" - distribution of residuals of response variable
"Link" - function relating how changes in the data X relate to changes in the linear predictor mu.

Family	Response Variable Distribution Interpretation	Range	Canonical Link	Other Link choices
binomial	Probabilities, yes/no occurrances	[0,1]	(link = "logit")	Probit, cloglog
gaussian	Normal distribution, variables scaling linearly/additively	(-Inf,Inf)	(link = "identity")	
Gamma	Variables responding exponentially/multiplicatively	(0,Inf)	(link = "inverse")	
inverse.gaussian		(0,Inf)	(link = "1/mu^2")	
poisson	Count	Non-negints	(link = "log")	
quasi			(link = "identity", variance = "constant")	
quasibinomial			(link = "logit")	
quasipoisson			(link = "log")	

R Code

Gaussian with weights

```
mod5 <- glm(AvgClaims ~ District + Group + Age, data = Insurance, weights = Holders, family = gaussian)
mod5$coefficients
pred5 <- predict(mod5,type="response")
```

Poisson with Offset

```
mod2 <- glm(Claims ~ District + Group + Age + offset(log(Holders)), data = Insurance, family = poisson)
mod2$coefficients
pred2 <- predict(mod2,type="response")
```

LRT = Likelihood Ratio Test (Compare two GLMs)

```
# anova(object, ..., dispersion = NULL, test = NULL)
  # test - "Chisq", "LRT", "Rao", "F" or "Cp"
anova(glm.freq, glm.freq2, test = NULL)

#Single term deletions
drop1(glm.freq, test = "LRT") #AIC is the default
AIC is based on deviance, but penalizes more complicated models
plot(glm.sev) is a good way to analyse residuals
```

Offsets

```
mod1 <- glm(Claims ~ District + Group + Age, data = Insurance,family = poisson)
mod1$coefficients
pred1 <- predict(mod1,type="response")

#Note how an offset is treated as another predictor. Also, because it is treated as a variable, the
logarithm should be used as the offset.
mod2 <- glm(Claims ~ District + Group + Age + offset(log(Holders)), data = Insurance, family = poisson)
mod2$coefficients
pred2 <- predict(mod2,type="response")

sse1 <- sum((Insurance$Claims - pred1)^2)
sse2 <- sum((Insurance$Claims - pred2)^2)

sse1
sse2
```

GLM Regularization (ElasticNet)

Sunday, June 30, 2019 7:43 PM

Read chapter 6.2 of ISLR. Ideally, read ILSR from cover to cover.

http://www-bcf.usc.edu/~gareth/ISL/

TL;DR

1) Lasso performs variable selection whereas Ridge only makes the coefficients smaller
2) Ridge can sometimes have a better fit

```
#CHUNK 14
library(glmnet)

#Place the X values in a matrix (required for the glmnet function)
X <-as.matrix(df[,3:8])

# Set up the formula (model form)
formula.lm <- as.formula("y~X1+X2+X3+X4+X5+X6")

# Fit the model, lambda = 0 forces ordinary least squares and makes alpha irrelevant
# Higher lambda means higher levels of regularization
model.lm <- glmnet(X, y = df$y,family = "gaussian", alpha = 0, lambda = 0)

# Predict results (so we can plot the line)
df$pred <- predict(model.lm, newx = X)

# Plot the results
p1 <- ggplot(data = df, aes(x = x, y = y)) + geom_point(color = "blue", size = 3) + geom_line(aes(y=df
$pred))
p1

#CHUNK 16
library(glmnet)

#Here is a clever way to create the data matrix. This one doesn't require keeping track of which columns
contain the features.
X <- model.matrix(formula.lm, data = df)

#Lambda has arbitrarily been set to 0.1. Alpha = 0 implies ridge regression (1 implies lasso and anything
between is elasticnet). Also, note that the default is to standardize the features, but the estimated
coefficients are on the scale and location of the original values.
model.lm.ridge <- glmnet(X, y = df$y,
                         family = "gaussian",
                         alpha = 0,
                         lambda = 0.1)

# Predict results (so we can plot the line)
```

```
df$pred_ridge01 <- predict(model.lm.ridge, newx = X)

# Plot the results
p1 <- ggplot(data = df, aes(x = x, y = y)) + geom_point(color = "blue", size = 3) + geom_line(aes(y=df
$pred_ridge01))
p1
```

Hyperparameter tuning
Alpha and Lambda are not able to be optimized, but still need to be selected
Trial and Error or Cross Validation
Cross Validation: Repeating the validation step with different training and test samples

Trees

Sunday, June 30, 2019 7:51 PM

Classification

```
tree <- rpart(Credit ~ CreditAmount + Age + CreditHistory + Employment, data = credit)
dt1  <- rpart(dt1.f, data = data.training, method = "class",
        control = rpart.control(minbucket = 5, cp = 0.01, maxdepth = 5),
        parms = list(split = "gini"))


rpart.plot(tree, extra = 4)
rpart.plot(pdt1)

#Run CHUNK 6 to extract the optimal complexity of the tree by taking the one with the minimum xerror:
#xerror means cross validation error
dt1$cptable[which.min(dt1$cptable[, "xerror"]), "CP"]

# prune the tree
pdt1<- prune(dt1, cp = dt1$cptable[which.min(dt1$cptable[, "xerror"]), "CP"])
```

Regression

```
dt2  <- rpart(dt2.f,
        data = AutoClaim.training,
        method = "anova",
        control = rpart.control(minbucket = 10,
                        cp = 0,
                        maxdepth = 10),
        parms = list(split = "information"))
```

RandomForests

```
model.rf <- randomForest(formula = formula.rf,
                data = data.training,
                ntree = 50,
                mtry = 3, # The number of features to use at each split.
                sampsize = floor(0.6 * nrow(data.training)), # The number of observations to use in each tree.
                nodesize = 100, # The minimum number of observations in each leaf node of a tree - this controls complexity.
                importance = TRUE
                )

#Confusion Matrix is a useful function
confusionMatrix(predictions,data.testing$target)
```

Parameter Tuning with Caret

Read the official documentation: https://topepo.github.io/caret/model-training-and-tuning.html

TL;DR version:

Cross Validation

```
library(caret)
set.seed(1000)
ctrl <- trainControl(method = "cv", number = 10)

model.Hitters.glm <- train(Salary ~.,
                data = data.Hitters.train,
                method = "glmnet",
                trControl = ctrl
                )
```

Tuning a random forest's mtry parameter

```
rfGrid <- expand.grid(mtry = c(1,3,5,7)) # in randomForest, the parameter mtry = The number of features to select at each split.
ctrl <- trainControl(method = "repeatedcv",
                     number = 5,
                     repeats = 3,# We want to do 5-fold cross validation (repeated 3 times for robustness)
                     sampling = "down") # This is undersampling - other methods include "up" (oversampling), "SMOTE" and "ROSE" (hybrid methods).

model.rf.tuned <- train(target ~.,
                        data = data.training,
                        method = "rf", # This is so we use the randomForest algorithm.
                        trControl = ctrl,
                        tuneGrid = rfGrid,
                        # We can specify the other parameters for the randomForest model here if we wish to. If we don't they will take on their default values.
                        ntree = 50, # The default is 500, setting to 50 will save us a lot of computation time but may not produce the best results.
                        importance = TRUE
                        )
```

Variable Importance

```
imp <- varImp(model.rf.tuned)
plot(imp, top = 20) # top = 20 makes the results more readable.
```

Partial Dependence

```
library(pdp)

partial(model.rf.tuned, train = data.training,  pred.var = "issage", plot= TRUE, rug = TRUE, smooth = TRUE)
```

PCA

```
d.pca.2d <- prcomp(diamonds.2d.pca, center = TRUE, scale. = TRUE)
summary(d.pca.2d)

#prcomp stores the calculated principal components in "x". Here the first one is attached to the data set as an 11th variable.

tlPCA.new <- tlPCA
tlPCA.new$PC1 <- tlPCA.2$x[,1]

#We now remove the six variables are being replaced with the single principal component

tlPCA.new <- tlPCA.new[ , c(2,4,8,9,11)]
```

K-means Clustering

```
#always need to scale the inputs
df <- df.raw
df$a <- scale(df$a)
df$b <- scale(df$b)
km2 <- kmeans(df, 2)
df$group2 <- as.factor(km2$cluster)

#Another example
for (i in 1:6)
{ set.seed(1000)
  cluster <- kmeans(dataframe,centers=i,nstart=10)
  r <- cluster$betweenss/cluster$totss
  print(paste("clusters:",i,"ratio:",r))
}
```