ISE 789-Midterm 2

Kimia Vahdat

#200262784

Question 1

a) *What are the differences between the model selection criteria: Mallow's CP, Leave-one-out Cross Validation, AIC and BIC?*

To completely explain the differences between aforementioned criteria, first I explain each of the separately and the compare them with each other.

- Mallow's Cp:

$$\frac{SSE_p}{\sigma^2} - (n - 2p)$$

Where the denominator is the estimation of variance in full model and p is the number of parameters in the model. We can observe that as we increase p, the first term decreases (because we are explaining more variation in the model) but the second term, which consist of 2p, increases. So, it has a trade off between number of parameters and variance and can be used to choose the best number of variables.

- LOOCV: This criterion only has closed form relation in linear regression, which is,

$$\sum_{i=1}^{n} \frac{e_i^2}{1 - h_i} \approx MSE + 2\sigma^2 tr(H)$$

This criterion in each iteration put aside one of the observations and form the model using the rest of the observations. Then, it uses that one observation as a test set and compute the MSE for it. It repeats this procedure for n times. As you can see, this method is very time consuming. Fortunately, there is a closed form solution for it in linear regression, but not in other models.

- AIC:

$$-2\log[L(\hat{\beta})] + 2p$$

Same as previous methods, it has 2 components, where one of them increases with number of parameters and the other decrease, so they can create a tradeoff between them.

- BIC:

$$-2\log[L(\hat{\beta})] + \log[n] * p$$

This method is very similar to AIC, the only difference is in the coefficient of p in the relation, which is now dependent on sample size.

Comparison:

Asymptotic view:

As $n \to \infty$ there is a theorem which claims that, Leave-one-out cross validation and Mallow's Cp are very close to each other. It worth mentioning that LOOCV is very time consuming when n is large so, although it gives a realistic view of our model, it might not be practical to use it. Also, asymptotically Mallow's Cp and AIC are equivalent. In fact, we can show that $AIC = Cp + K_n$ where, K is a constant which is a function of n. Since Cp and AIC involve less calculation than leave-one-out, they have advantages when n is large. In addition to these, if n is very large (larger than 100) this means, BIC penalizes p more than AIC; in this case, BIC has very little chance of choosing too big a model if *n* is sufficient (i.e. probably under estimate).

General view:

When n is not very large, log[n] would be smaller than 2, therefore, BIC has less penalization than AIC and their results would be kind of like each other. In general, because of fixed coefficient of p (2), AIC always has a chance of choosing too big a model, regardless of *n*. On the other hand, BIC has a larger chance than AIC, for any given *n*, of choosing too small a model. So, depending on what model and our goal we might choose either AIC and BIC for comparison.

Another important observation is that, there is no guarantee that Cp, AIC or BIC choose best *predictive* model, but LOOCV does that. In other words, Cp, AIC or BIC probably over fit the data and might not be the best choice if our goal is prediction. At last, if the best model is among those we are testing and depending on our goal, either of these methods might be useful.

b) *Describe what the difference is between ridge regression and lasso in terms of*

To simplify the rest of the solution, first take a look at objective function of LASSO and Ridge.

LASSO:

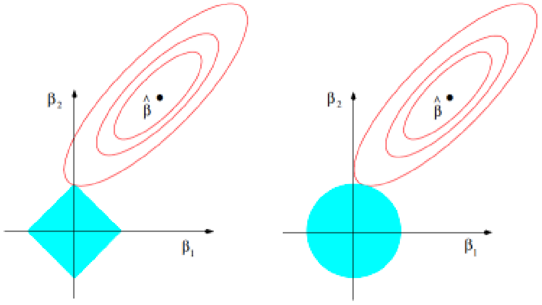
$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

Ridge Regression:

$$min\left\{\frac{1}{N} ||y - X\beta||_2^2 + \lambda||\beta||_2^2\right\}$$

i. *Regularization:* Regularization is a technique to avoid high variance in our model. To do this we will introduce a penalty term to the cost function. By adding this penalty term, we will prevent the coefficients of the linear function from getting too large. Introducing this bias should decrease the variance and thus prevent overfitting. As shown above, the regularization term in LASSO is the absolute values of β, while in Ridge it's second degree norm of coefficients.

As you can see in the following picture, the graph of constraints is different in Ridge vs. LASSO, which is cause by the regularization term in these models. This will affect their model selection too.



ii. *Model Selection:* Based on what was said for the previous part, difference in their regularization term affects their model selection too. To be more specific, LASSO shrinks some β's to 0 so it does model selection; on the other hand, Ridge does not necessarily make β's 0, but it shrinks them to be small. Since Ridge does not shrink to 0, it's not the best model for model selection; to explain more, we need to decide based on the values of β's whether to include them in model or not, but LASSO provides us with exact 0 and nonzero values for β's.

iii. *Sparsity:* As said before, LASSO set some of β's to 0 so, it produces a sparse solution, where Ridge only shrinks β's to be small but not 0 so it will not be considered as a sparse solution.

c) *The following two plots (attached at the end of the homework) show the progression of the coefficients with the penalty constant λ, where the red line shows the optimal λ. The plots are for fitting and selecting variables for a regression problem where there are eight different predictors.*

i. *What model would you select based on each of these two plots and why?*

Based on what explained about LASSO and Ridge Regression models, we can decide which variables to choose. The right-hand side plot is representing the LASSO model on the data. We can observe that, at the optimal λ only coefficients for "lcavol", "svi" and "lweight" has values and the rest of them are 0. So, the model that LASSO selects has only these variables in it.

The left-hand side plot depicts Ridge Regression. At the optimal λ, only one of the variables, "lcp" is very close to 0 and the rest of the variables, although not very large, have positive values, so the model that Ridge chooses has all variables except for "lcp".

Another thing that we might notice in these plots are the trend of coefficients' values. We can see that in both plots, all variables have increasing trend as we increase the λ except for three variables "gleason"," age" and "lcp". Of course, in LASSO plot these variables are set to 0 at the optimal.

ii. *How are the two plots different and why?*

The x-axis in these plots are different from each other. The horizontal axis for the LASSO plot is the shrinkage factor, s:

$$s = \frac{\lambda}{\sum_{j=1}^{p} |\hat{\beta}_j|}$$

Whereas, for the Ridge plot is degrees of freedom of λ.

The y-axis is the same in both plots which is coefficients estimations. We can see that the range for coefficient estimations in both plots are the same (from -0.2 to 0.6).

Also, if you look more closely, you can see that the LASSO plot is a piecewise linear function while, Ridge plot is a smooth one. The reason is again in their difference in their regularization term. Since, LASSO's regularization term is absolute values of coefficients, their plot would be linear. Also, because the penalty term for Ridge is the square of coefficients, their plot would be a curve and not a line.

LASSO plot is much sparser, because it does not include a lot of coefficients (set a value of 0 for them). And at last, LASSO plot can be used in model selection but Ridge plot is not that helpful in selecting the variables.

Question 2:

Let (Yi , Xi) for i = 1, ... , n, where Yi has a distribution from the exponential family of distributions; its general density formula is

$$f(y;\theta) = h(y)e^{\eta(\theta)T(y)-B(\theta)}$$

We want to estimate the parameter β defined by $g\{\mathbb{E}(Yi\,|Xi\,)\} = \beta Xi$ where g is a link function.

a) *If the observations Yi are binary, i.e. taking only values of 0 or 1, what is the link function g and how you would derive it?*

If Y only takes values of 0 and 1, its distribution can be Bernoulli. We know that the Bernoulli density function is

$$f(y;\theta) = \theta^{y}(1-\theta)^{1-y}$$

Where, θ is the probability of taking value 1 and the expected value for y. Now we need to reformulate its density function so it would be in the same format as general density formula:

$$E[y] = \theta$$

$$\begin{aligned} f(y;\theta) &= \exp(ylog[\theta] + (1-y)\log[1-\theta]) \\ &= \exp(y(\log[\theta] - \log[1-\theta]) + \log[1-\theta]) \\ &= \exp\left(y\left(\log\left[\frac{\theta}{1-\theta}\right]\right) + \log[1-\theta]\right) \end{aligned}$$

Based on above representation we can say,

$$h(y) = 1$$

$$\eta(\theta) = \log\left[\frac{\theta}{1-\theta}\right]$$

$$T(y) = y \text{ } and \text{ } B(\theta) = -\log[1-\theta]$$

$$g(E[y|x]) = g(\theta) = \eta(\theta) = \log\left[\frac{\theta}{1-\theta}\right]$$

We can set g(E[y|x]) to be the same as η(θ), because the support range for this function is -∞ to ∞, and they are both function of expectation of y.

b) *If the observations Yi are count values, i.e. taking only values of 0, 1, 2,3, . . . , what is a common distribution that is assumed for Yi? For this distribution, what is the link function g and how you would derive it?*

The common distribution is Poisson. Same as before we first try to write its density function in the format of general density function, in which θ is the expectation of y (here it is the same as λ, or rate parameter in Poisson distribution)

$$f(y;\lambda) = \frac{e^{-\lambda}\lambda^{y}}{y!} \rightarrow f(y;\theta) = \frac{e^{-\theta}\theta^{y}}{y!}$$

$$f(y;\theta) = \frac{1}{y!}\exp(ylog(\theta) - \theta)$$

Based on above representation we can say,

$$h(y) = \frac{1}{y!}$$

$$\eta(\theta) = \log[\theta]$$

$$T(y) = y \; and \; B(\theta) = \theta$$

$$g(E[y|x] = \lambda) = g(\theta) = \eta(\theta) = \log[\theta]$$

Again, since the support range for this function is also infinity and is a function of λ or the expectation of y (θ), we can accept it as our link function for Poisson.

c) *Assume that the distribution of Yi is Gamma, which has a density function*

$$f(y;\alpha,\beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)}y^{\alpha-1}\exp(-\beta y)$$

Note: $\mathbb{E}(Y) = \alpha/\beta$. Assuming α fixed and known, what is the link function g and how you would derive it?

Same as before, first we need to reformulate the density function, so it would be a function of expectation of y (i.e. θ= α/β) and in the format of general density function.

$$\begin{aligned} f(y;\alpha,\beta) &= \frac{y^{\alpha-1}}{\Gamma(\alpha)}\exp(\alpha\log[\beta] - \beta y) \\ &= \frac{y^{\alpha-1}}{\Gamma(\alpha)}\exp(\alpha\log[\beta] - \alpha\log[\alpha] - \beta y + \alpha\log[\alpha]) \\ &= \frac{y^{\alpha-1}}{\Gamma(\alpha)}\exp\left(\alpha\left(log\left[\frac{\beta}{\alpha}\right]\right) - \alpha\left(\frac{\beta}{\alpha}\right)y + \alpha\, log[\alpha]\right) \end{aligned}$$

Now, since, α is a fixed number, we can consider it as a constant and replace α/β with θ and achieve our goal.

$$f(y;\theta) = \frac{y^{\alpha-1}}{\Gamma(\alpha)}\exp\left(\alpha\left(log\left[\frac{1}{\theta}\right]\right) - \alpha\left(\frac{1}{\theta}\right)y + \alpha\, log[\alpha]\right)$$

Based on above representation we can say,

$$h(y) = \frac{y^{\alpha-1}}{\Gamma(\alpha)}$$

$$\eta(\theta) = -\frac{1}{\theta}$$

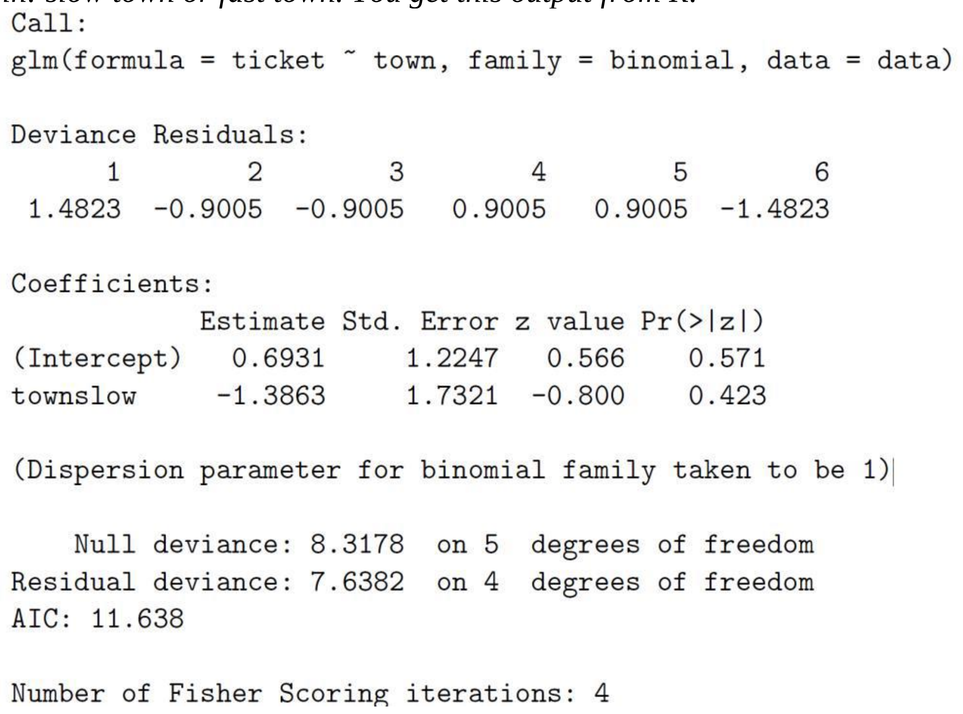
$$T(y) = \alpha y \; and \; B(\theta) = \alpha\left(log\left[\frac{1}{\theta}\right]\right) + \alpha\, log[\alpha] = \alpha\log\left[\frac{\alpha}{\theta}\right]$$

$$g\left(E[y|x] = \frac{\alpha}{\beta}\right) = g(\theta) = \eta(\theta) = -\frac{1}{\theta}$$

Support for this function is infinity and is a function of expectation of y (i.e. $\frac{\alpha}{\beta} = \theta$).

Question 3

Let Y = 1 if a driver gets a ticket. Consider a categorical predictor indicating which town a driver is in: slow town or fast town. You get this output from R:



```
Call:
glm(formula = ticket ~ town, family = binomial, data = data)

Deviance Residuals:
      1        2        3        4        5        6  
 1.4823  -0.9005  -0.9005   0.9005   0.9005  -1.4823  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.6931     1.2247   0.566    0.571
townslow     -1.3863     1.7321  -0.800    0.423

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8.3178  on 5  degrees of freedom
Residual deviance: 7.6382  on 4  degrees of freedom
AIC: 11.638

Number of Fisher Scoring iterations: 4
```

a) *Specify the model and link function.*

As shown in the call section, model is GLM with binomial family and default link function for this family is logit.

$$logit = \log\left(\frac{p}{1-p}\right)$$

Which is what we derived in the previous question. Also, p here is the probability of getting a ticket. So our model would be,

$$logit = \beta_0 + \beta_1 "townslow"$$

Where "townslow" is a dummy variable which takes a value of 1 if the driver is in slow town and 0 otherwise. The estimation of both of these parameters is shown in the output. At last we would have the following as our model.

$$logit = 0.69 + (-1.39)"townslow"$$

$$p = \frac{e^{0.69+(-1.39)"townslow"}}{1 + e^{0.69+(-1.39)"townslow"}}$$

b) *What is the odds ratio for getting a ticket while in town fast?*

The odds ratio is $\frac{p}{1-p}$, and to get its value when in a town fast is equivalent to get its value for when "townslow" variable is 0.

$$\widehat{adds} = \exp(0.69 + (-1.39)"\text{townslow}") = \exp(0.69) = 1.99$$

c) *What is the expected probability of getting a ticket while in fast town?*

Expected probability of getting a ticket is p so based on previous part's answer,

$$\frac{p}{1-p} = 1.99 \rightarrow \hat{p} = \frac{1.99}{1+1.99} = 0.66$$

This means the probability of getting a ticket while in a fast town is 0.66.

d) *Compute the confidence interval for the effect of being in slow town.*

To compute confidence interval for the effect of being in a slow town, we need to compute the confidence interval for β_1 , which can be done using its estimation and standard error.

$$CI = \hat{\beta} \pm std.Error * Z_{\alpha/2}$$

If we consider a significance level of 0.05, we would get,

$$CI = -1.38 \pm 1.73 * 1.96 = [-4.77, 2.01]$$

Question 4:

Greene and Shaffer (1992), cited in Fox (1997), analyzed decisions by the Canadian Federal Court of Appeals on cases filed by refugee applicants who had been turned down by the Immigration and Refugee Board. We are interested in the fact that there are large differences among judges.

It is possible, of course, that the judges get to hear very different cases. In this analysis you will control for an expert assessment of whether the case had merit, the city where the original application was filed (Toronto, Montreal or other) and the language in which it was filed (English, French). An additional predictor is the logit of the success rate for all cases from the applicant's country. The country itself is also available. Fox's dataset is restricted to the 10 (of 12) judges who were present in the court during the entire period, and to countries of origin that produced at least 20 appeals during this period. There are384 observations on 8 variables.

a) ***Modeling and Inference by judge.*** *Fit a logistic regression model where only the judge categorical variable is included in the model. Use Iacobucci as the reference judge to facilitate the comparisons with other judges. Interpret the coefficient for judge Desjardins. Verify whether the predicted probability for this judge is in agreement with that one provided in the R output. Test the hypothesis that the probability of granting leave to appeal is the same for all judges.*

First, we read the data in R, and set the base level for judges to be judge "Iacobucci". Then we create a model just using Judge names, the result of the model is shown in the summary of the model. We can see that only the intercept is significant, and the rest of the variables have p-values greater than 0.01. This means we fail to reject the null hypothesis for these variables (i.e. their coefficients can be set to 0 based on p-value).

```
data<-read.csv("judges_and_immigration.txt",sep = " ",header = T)

data <- within(data, JudgeName <- relevel(JudgeName, ref = "Iacobucci")
)

attach(data)

model1<-glm(GrantedAppeal~JudgeName,family= binomial(link = logit))

summary(model1)

##
```

```
## Call:
## glm(formula = GrantedAppeal ~ JudgeName, family = binomial(link = lo
git))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1213  -0.9005  -0.8687   1.4083   1.8465
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -0.9651     0.4155  -2.323   0.0202 *
## JudgeNameDesjardins   0.6137     0.5121   1.198   0.2308
## JudgeNameHeald        0.2719     0.5455   0.498   0.6182
## JudgeNameHugessen     0.4370     0.4917   0.889   0.3741
## JudgeNameMacGuigan    0.1849     0.4888   0.378   0.7052
## JudgeNameMahoney      0.8315     0.5537   1.502   0.1331
## JudgeNameMarceau     -0.4212     0.6501  -0.648   0.5170
## JudgeNamePratte       0.2719     0.5289   0.514   0.6072
## JudgeNameStone        0.2719     0.5559   0.489   0.6247
## JudgeNameUrie        -0.5390     0.8853  -0.609   0.5426
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 491.57  on 383  degrees of freedom
## Residual deviance: 483.67  on 374  degrees of freedom
## AIC: 503.67
##
## Number of Fisher Scoring iterations: 4
```

The coefficient estimate for judge “Desjardins” is 0.61. The interpretation is that his measure of effect relative to judge “Iacobucci” is 0.61. which means, if we change from judge “Iacobucci” to judge “Desjardins” the log of odds ratio will increase by 0.61.

The predicted probability for this judge can be computed using command predict, with type “link” which will give us the prediction of link function (that is logit ratio). When we get the predicted link function, we can easily compute the predicted probability. The calculations are shown in the output of the code and the result is 0.41.

To verify this result with the one that R gives us, we need to run the same command predict, with one difference, which is instead of type “logit” we put type “response”. We can observe that we get the same result as computed before.

```
link<-predict(model1,newdata = list(JudgeName="Desjardins"),type = "lin
k")

Prob<-exp(link)/(exp(link)+1)

Prob

##         1

## 0.4130435

response<-predict(model1,newdata = list(JudgeName="Desjardins"),type =
"response")

response

##         1

## 0.4130435

# They are the same!
```

To test whether the probability of granting leave to appeal is the same for all judges, we need to do a hypothesis test, in which null hypothesis and alternative hypothesis is as follows,

$$H_0: \beta_i = \beta_j = 0 \;\; for\ all\ i \neq j\ and\ i, j = 1,2, \dots, p$$

$$H_a: otherwise$$

For this test we can use likelihood ratio test, which follows a F distribution and can be computed in R. To do so, we need to use deviance for null model, which is the model corresponding our null hypothesis, and residual deviance for full model. The F test would be

$$\frac{(D_{sub} - D_{full})/(df_{sub} - df_{full})}{D_{full}/df_{full}} \sim F_{(df_{sub}-df_{full}),df_{full}}$$

```
null<-model1$null.deviance

full<-model1$deviance

test<-((null-full)/(model1$df.null-model1$df.residual))/(full/model1$df.resid
ual)

# P-value is:
```

```
pf(test,model1$df.null-model1$df.residual,model1$df.residual,lower.tail = F)
## [1] 0.7282361
```

Since the p-value is larger than 0.05, we do not have enough evidence to reject null hypothesis. This means, we do not have enough evidence that judges are different from each other.

b) ***Modeling and Inference - full model****. Fit a logistic regression model with all available independent variables included in the model. This model would allow to assess the the probability of granting leave to appeal for each judge while controlling for confounding variables. Interpret the coefficient for judge Desjardins in this model, explaining carefully what it means to adjust for the other predictors. Has the adjustment reduced the contrast with Iacobucci? Test the hypothesis that the probability of granting leave is the same for all judges after adjusting for the control variables.*

   In this part, we are including all variables except for country, for which we include the success rate instead. We can see the result in the summary of the model. In this model only “JudgeNameMarceau” and “Merityes” are significant.

   The coefficient for judge “Desjardins” is -0.1, which means if we consider all other variables to be constant and if we change the judge from “Iacobucci” (the base) to this judge, the logit of the response (granted appeal) will decrease by 0.1.

   Previously, when we only had judges in the model, the coefficient of this judge was 0.61, but now it is -0.1. In other words, previously if we changed judge from the base to Desjardins, we would get an increase in the log odds of 0.61, but now we get a decrease in log odds by 0.1. This means the contrast between this judge and the base judge has changed (reduced) due to adding different other variables into model. The reason for this could be collinearity between different variables. There are some variables in the model that have high correlation with each other, and they affect other coefficients as well. For example, variable language and city, obviously have high correlation with each other so this could be one of the reasons that contrast has decreased.

```
model2<-glm(GrantedAppeal~JudgeName+Merit+Language+SuccessRate+City,family= binomial(link = logit))

summary(model2)
##
## Call:
## glm(formula = GrantedAppeal ~ JudgeName + Merit + Language +
##     SuccessRate + City, family = binomial(link = logit))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5784  -0.8111  -0.7316   1.0674   2.1844
```

```
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -0.42814    0.72885  -0.587   0.5569
## JudgeNameDesjardins -0.10074    0.55975  -0.180   0.8572
## JudgeNameHeald      -0.03349    0.57142  -0.059   0.9533
## JudgeNameHugessen    0.07583    0.56601   0.134   0.8934
## JudgeNameMacGuigan  -0.15262    0.52349  -0.292   0.7706
## JudgeNameMahoney     0.39014    0.58359   0.669   0.5038
## JudgeNameMarceau    -1.42980    0.72603  -1.969   0.0489 *
## JudgeNamePratte      0.06138    0.56816   0.108   0.9140
## JudgeNameStone       0.06508    0.57674   0.113   0.9102
## JudgeNameUrie       -1.14864    0.92637  -1.240   0.2150
## Merityes             1.36593    0.27142   5.032 4.84e-07 ***
## LanguageFrench      -0.24374    0.55495  -0.439   0.6605
## SuccessRate          0.17073    0.22180   0.770   0.4414
## Cityother           -0.63606    0.62232  -1.022   0.3067
## CityToronto         -0.50186    0.55931  -0.897   0.3696
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 491.57  on 383  degrees of freedom
## Residual deviance: 450.66  on 369  degrees of freedom
## AIC: 480.66
##
## Number of Fisher Scoring iterations: 4
```

To test if the probability of granting leave is the same for all judges in this model, we form another model called null.model, in which all variables of Model2 are included except for judges. The null hypothesis in this test is the same as null hypothesis in previous part:

$$H_0: \beta_i = \beta_j = 0 \;\; for\ all\ i \neq j\ and\ i, j \in \{Judges\}$$

$$H_a: otherwise$$

To test this hypothesis, we again form the F statistic and calculate the p-value for this test. We observe a value of 0.40 for the p-value, which is larger than 0.05; therefore, we do not have enough evidence to reject the null hypothesis, so we cannot say there is any difference between judges even after adjusting control variables.

```
null.model<-glm(GrantedAppeal~Merit+Language+SuccessRate+City,family= binomia
l(link = logit))

null<-null.model$deviance

full<-model2$deviance

## Hypothesis testing

test<-((null-full)/(null.model$df.residual-model2$df.residual))/(full/model2$
df.residual)

pf(test,null.model$df.residual-model2$df.residual,model2$df.residual,lower.ta
il = F)
```

```
## [1] 0.4014993
```

c) ***Goodness-of-fit****. How do the two models compare in terms of goodness-of-fit?*
To compare these models in terms of goodness of fit, we have different options. One option is to compare their deviances with each other. if the deviance for one model is smaller than the other model, we can conclude that the first model is performing better. At last, I compare their AIC as well.

```
# comparing deviance of two models

model2$dev

## [1] 450.6574

model1$deviance

## [1] 483.6683

## model 2 has smaller deviance

# Comparing aic

model2$aic

## [1] 480.6574

model1$aic

## [1] 503.6683

# model 2 has less aic
```

We can see that the second model (the one that contained all the variables) has smaller deviance, so it's acting a little better than the first model.
At last, if we compare their AIC's we see that AIC of the second model is smaller and therefore better than the first model. We can also, do a rigorous likelihood ratio test and calculate the p-value for the hypothesis that two tests are the same. We can see that p-value is much smaller than 0.05, so we reject the null hypothesis and two models are not the same.

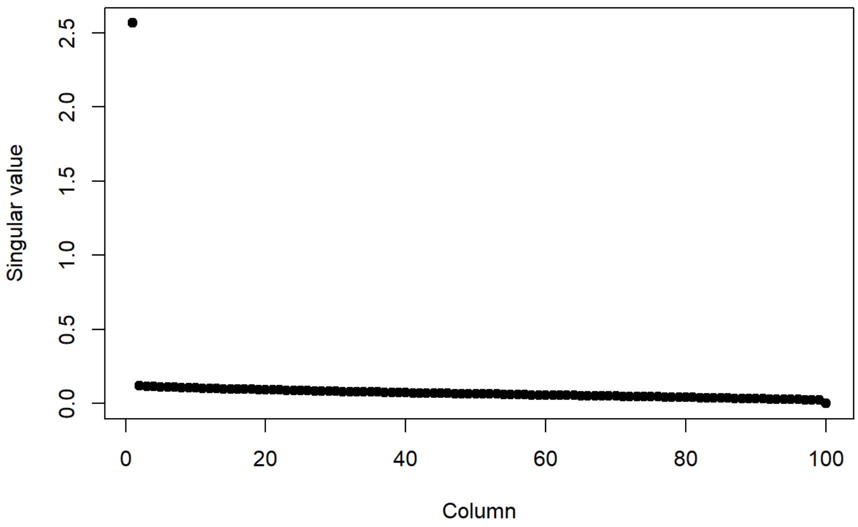
```
## Checking if two models are the same?
library(lmtest)
lrtest(model1,model2)
## Likelihood ratio test
##
## Model 1: GrantedAppeal ~ JudgeName
## Model 2: GrantedAppeal ~ JudgeName + Merit + Language + SuccessRate +
##     City
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1  10 -241.83
## 2  15 -225.33  5 33.011  3.745e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question 5:

This question was done with R, which I attached the output of the code here.

Question 5 Part a

```
q5<-read.csv("Question5.csv",sep=",",header = F)
#install.packages("devtools")
library(devtools)
## Warning: package 'devtools' was built under R version 3.4.4
## Warning: package 'usethis' was built under R version 3.4.4
####################### Part a
# svd approach
zvars <- scale(q5, center = T, scale = F)
z.svd <- svd(zvars)
plot(z.svd$d, xlab = "Column", ylab = "Singular value", pch = 19)
```



```
f<-0
i<-0
while(f<0.95){
    i<-i+1
  f<-sum(z.svd$d[1:i]^2)/sum(z.svd$d^2)

}
# number of variables is i-1=12
k=i-1
k
```

```
## [1] 12 # Number of columns chosen
```

```
## PC-Scores
sigma<-matrix(0,k,k)
diag(sigma)<-z.svd$d[1:k]
Scores <- z.svd$u[,1:k] %*% sigma
head(Scores)
```

```
##             [,1]         [,2]          [,3]         [,4]          [,5]
## [1,] -0.23615588  0.0078452481 -3.881341e-03  0.0061489362 -0.0046434380
## [2,]  0.17523640  0.0021618642 -5.494114e-05  0.0127631225  0.0009483052
## [3,] -0.01063137 -0.0040113386  2.210083e-02 -0.0094399765  0.0035500226
## [4,]  0.11456789  0.0004908275  2.131046e-02  0.0066886145 -0.0008319519
## [5,] -0.09727160 -0.0169102373 -1.278879e-02 -0.0047200610 -0.0066711536
## [6,]  0.15921029 -0.0128455703 -9.154337e-03 -0.0001018231 -0.0280053930
##               [,6]         [,7]         [,8]          [,9]         [,10]
## [1,]  0.0003055315 -0.029753873  0.008180912 -0.0043287318  0.0219333608
## [2,]  0.0048879313  0.009598281  0.007137941 -0.0024919740 -0.0057708099
## [3,]  0.0094771125 -0.014996891  0.007930775 -0.0003147998  0.0001266277
## [4,] -0.0007561800  0.018055760  0.004198599 -0.0199914641  0.0112062638
## [5,] -0.0195902420 -0.001250679 -0.011440247 -0.0079811101  0.0050493908
## [6,] -0.0105301117  0.005889005  0.016433470  0.0055718267 -0.0026176193
##             [,11]        [,12]
## [1,] -0.003739071  0.011231530
## [2,]  0.001614618  0.018356497
## [3,]  0.008546983  0.004835034
## [4,] -0.001872298  0.006194889
## [5,] -0.024011274 -0.015427012
## [6,] -0.010613667 -0.005542646
```

Question 5 Part b

```
# eigen decomposition
w <- t(zvars)%*%zvars
a <- eigen(w)$values
avec<-eigen(w)$vectors
f<-0
i<-0
while(f<0.95){
  i<-i+1
  f<-sum(a[1:i])/sum(a)


}
# number of variables is i-1=12
k=i-1
k
```

```
## [1] 12 # number of columns chosen
```

```
# PC-Scores
```

```
pca.scores<- zvars %*% avec[,1:k]
head(pca.scores)
```

```
##             [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  0.23615588 -0.0078452481  3.881341e-03 -0.0061489362 -0.0046434380
## [2,] -0.17523640 -0.0021618642  5.494114e-05 -0.0127631225  0.0009483052
## [3,]  0.01063137  0.0040113386 -2.210083e-02  0.0094399765  0.0035500226
## [4,] -0.11456789 -0.0004908275 -2.131046e-02 -0.0066886145 -0.0008319519
## [5,]  0.09727160  0.0169102373  1.278879e-02  0.0047200610 -0.0066711536
## [6,] -0.15921029  0.0128455703  9.154337e-03  0.0001018231 -0.0280053930
##               [,6]         [,7]         [,8]          [,9]         [,10]
## [1,]  0.0003055315 -0.029753873 -0.008180912 -0.0043287318  0.0219333608
## [2,]  0.0048879313  0.009598281 -0.007137941 -0.0024919740 -0.0057708099
## [3,]  0.0094771125 -0.014996891 -0.007930775 -0.0003147998  0.0001266277
## [4,] -0.0007561800  0.018055760 -0.004198599 -0.0199914641  0.0112062638
## [5,] -0.0195902420 -0.001250679  0.011440247 -0.0079811101  0.0050493908
## [6,] -0.0105301117  0.005889005 -0.016433470  0.0055718267 -0.0026176193
##             [,11]        [,12]
## [1,]  0.003739071  0.011231530
## [2,] -0.001614618  0.018356497
## [3,] -0.008546983  0.004835034
## [4,]  0.001872298  0.006194889
## [5,]  0.024011274 -0.015427012
## [6,]  0.010613667 -0.005542646
```