Assignment 1

SURV 640

Sagnik Chakravarty

Setup

```
library(titanic)
library(caret)
library(pROC)
```

Data

In this notebook we use the Titanic data that is used on Kaggle (https://www.kaggle.com) as an introductory competition for getting familiar with machine learning. It includes information on a set of Titanic passengers, such as age, sex, ticket class and whether he or she survived the Titanic tragedy.

Source: https://www.kaggle.com/c/titanic/data

```
titanic <- titanic_train
str(titanic)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

We begin with some minor data preparations. The `lapply()` function is a handy tool if the task is to apply the same transformation (e.g. `as.factor()`) to multiple columns of a data frame.

```
titanic[, c(2:3,5,12)] <- lapply(titanic[, c(2:3,5,12)], as.factor)
```

The `age` variable has some NAs, as a quick and dirty solution we can create a categorized age variable with NAs as an additional factor level.

```
titanic$Age_c <- cut(titanic$Age, 5)
titanic$Age_c <- addNA(titanic$Age_c)
summary(titanic$Age_c)
```

```
## (0.34,16.3] (16.3,32.3] (32.3,48.2] (48.2,64.1] (64.1,80.1]        <NA>
##         100         346         188          69          11         177
```

Train and test set

Next we split the data into a training (80%) and a test (20%) part. This can be done by random sampling with `sample()`.

```r
set.seed(9395)
# Add code here

train_indices <- sample(1:nrow(titanic),
                        size = round(nrow(titanic)*0.8))
train <- titanic[train_indices, ]
test <- titanic[-train_indices, ]
cat('The dimension of training data:\t', dim(train), '\n',
    'The dimension of test data:\t', dim(test), '\n')
```

```
## The dimension of training data:   713 13
##  The dimension of test data:  178 13
```

Logistic regression

In this exercise we simply use logistic regression as our prediction method, since we want to focus on the evaluation part. Build a first logit model with `Survived` as the outcome and `Pclass`, `Sex`, `Age_c`, `Fare` and `Embarked` as features.

```r
log_model <- glm(Survived ~ Pclass+Sex+Age_c+Fare+Embarked,
                 data = train,
                 family = 'binomial')

summary(log_model)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age_c + Fare + Embarked,
##     family = "binomial", data = train)
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       1.799e+01  1.692e+03   0.011  0.99151
## Pclass2          -8.663e-01  3.287e-01  -2.635  0.00841 **
## Pclass3          -2.274e+00  3.241e-01  -7.016 2.28e-12 ***
## Sexmale          -2.545e+00  2.128e-01 -11.961  < 2e-16 ***
## Age_c(16.3,32.3] -8.525e-01  3.321e-01  -2.567  0.01026 *
## Age_c(32.3,48.2] -1.233e+00  3.759e-01  -3.281  0.00103 **
## Age_c(48.2,64.1] -1.580e+00  5.010e-01  -3.154  0.00161 **
## Age_c(64.1,80.1] -1.696e+01  6.976e+02  -0.024  0.98061
## Age_cNA          -1.179e+00  3.801e-01  -3.102  0.00192 **
## Fare             -1.112e-04  2.203e-03  -0.050  0.95976
## EmbarkedC        -1.417e+01  1.692e+03  -0.008  0.99332
## EmbarkedQ        -1.399e+01  1.692e+03  -0.008  0.99340
## EmbarkedS        -1.475e+01  1.692e+03  -0.009  0.99304
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 951.76  on 712  degrees of freedom
## Residual deviance: 624.39  on 700  degrees of freedom
```

```
## AIC: 650.39
##
## Number of Fisher Scoring iterations: 15
```

A quick look at the coefficients of the first logit model.

```
data.frame(
  Coefficient = log_model$coefficients
)
```

```
##                    Coefficient
## (Intercept)       1.799076e+01
## Pclass2          -8.662529e-01
## Pclass3          -2.274260e+00
## Sexmale          -2.545070e+00
## Age_c(16.3,32.3] -8.524552e-01
## Age_c(32.3,48.2] -1.233381e+00
## Age_c(48.2,64.1] -1.580223e+00
## Age_c(64.1,80.1] -1.695719e+01
## Age_cNA          -1.178943e+00
## Fare             -1.111658e-04
## EmbarkedC        -1.416625e+01
## EmbarkedQ        -1.399168e+01
## EmbarkedS        -1.474854e+01
```

Now, build an additional logit model that uses the same features, but includes at least one interaction or non-linear term.

```
mod_log_model <- glm(Survived ~ Pclass+Sex+Age_c*Fare+Embarked,
                     data = train,
                     family = 'binomial')
```

Again, summarize the resulting object.

```
summary(mod_log_model)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age_c * Fare + Embarked,
##     family = "binomial", data = train)
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)             1.845e+01  1.679e+03   0.011 0.991233
## Pclass2                -1.003e+00  3.436e-01  -2.919 0.003510 **
## Pclass3                -2.488e+00  3.487e-01  -7.134 9.75e-13 ***
## Sexmale                -2.585e+00  2.168e-01 -11.923  < 2e-16 ***
## Age_c(16.3,32.3]       -1.135e+00  4.421e-01  -2.568 0.010228 *
## Age_c(32.3,48.2]       -1.992e+00  5.143e-01  -3.873 0.000107 ***
## Age_c(48.2,64.1]       -1.935e+00  6.858e-01  -2.822 0.004773 **
## Age_c(64.1,80.1]       -1.621e+01  1.350e+03  -0.012 0.990417
## Age_cNA                -1.401e+00  4.984e-01  -2.810 0.004949 **
```

```
## Fare                  -1.290e-02  8.521e-03  -1.514 0.129960
## EmbarkedC             -1.405e+01  1.679e+03  -0.008 0.993325
## EmbarkedQ             -1.392e+01  1.679e+03  -0.008 0.993385
## EmbarkedS             -1.459e+01  1.679e+03  -0.009 0.993066
## Age_c(16.3,32.3]:Fare  7.973e-03  8.935e-03   0.892 0.372207
## Age_c(32.3,48.2]:Fare  1.906e-02  9.626e-03   1.980 0.047679 *
## Age_c(48.2,64.1]:Fare  9.718e-03  1.081e-02   0.899 0.368880
## Age_c(64.1,80.1]:Fare -2.479e-02  3.919e+01  -0.001 0.999495
## Age_cNA:Fare           5.701e-03  1.069e-02   0.533 0.593719
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 951.76  on 712  degrees of freedom
## Residual deviance: 616.56  on 695  degrees of freedom
## AIC: 652.56
##
## Number of Fisher Scoring iterations: 15
```

Prediction in test set

Given both logit objects, we can generate predicted risk scores/ predicted probabilities of `Survived` in the test set.

```
test$SurvPredictlogit <- predict(log_model, newdata = test, type = 'response')
test$SurvPredictlogitInteraction <- predict(mod_log_model, newdata = test, type = 'response')

head(test[, c('Survived', 'SurvPredictlogit', 'SurvPredictlogitInteraction')])
```

```
##   Survived SurvPredictlogit SurvPredictlogitInteraction
## 1        0       0.08088622                   0.08419932
## 2        1       0.92977319                   0.94525788
## 3        1       0.52862580                   0.54865547
## 4        1       0.88110542                   0.89955642
## 5        0       0.05671222                   0.04078152
## 7        0       0.29135061                   0.30396686
```

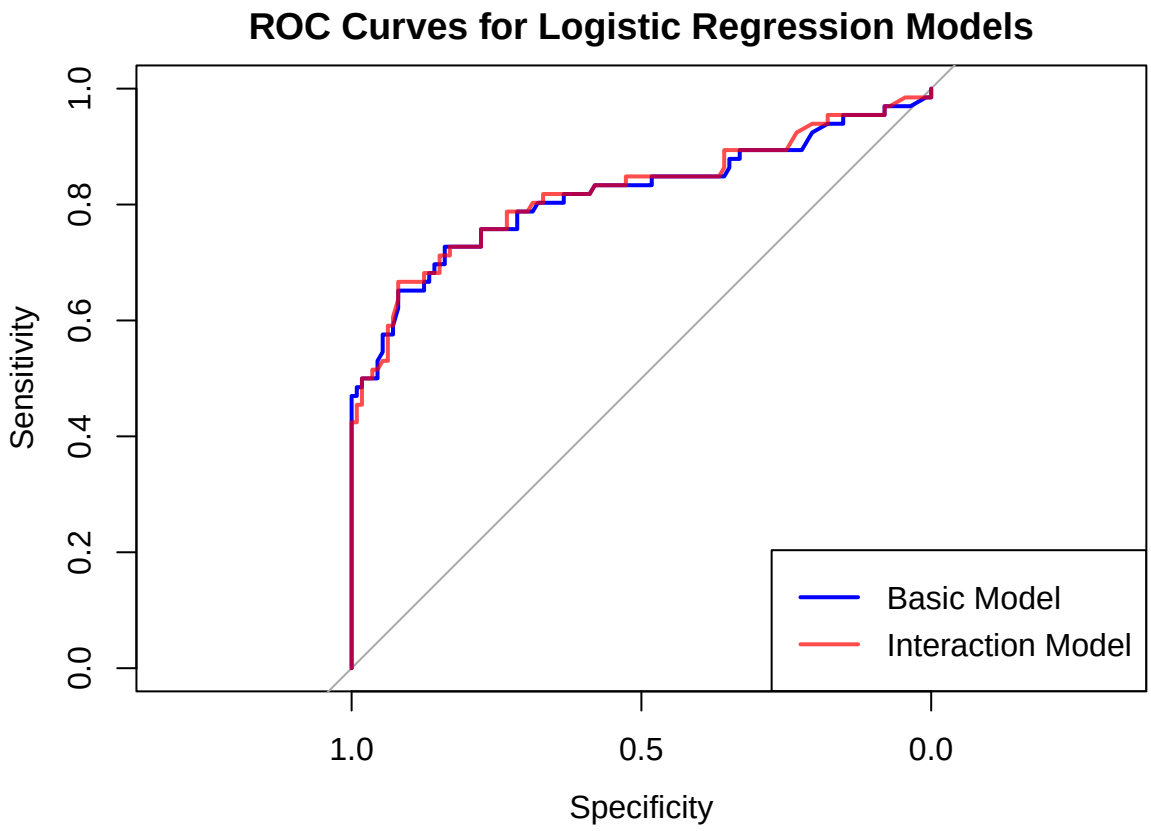
It is often useful to first get an idea of prediction performance independent of specific classification thresholds. Use the `pROC` (or `PRROC`) package to create roc objects for both risk score vectors.

```
library(pROC)
# ROC object for the basic logistic regression model
roc_logit <- roc(test$Survived, test$SurvPredictlogit, levels = c(0, 1))

# ROC object for the interaction logistic regression model
roc_logit_interaction <- roc(test$Survived, test$SurvPredictlogitInteraction, levels = c(0, 1))
# Plot the first ROC curve (basic model)
plot(roc_logit, col = "blue", main = "ROC Curves for Logistic Regression Models", lwd = 2)

# Plot the second ROC curve (interaction model) with transparency
plot(roc_logit_interaction, col = rgb(1, 0, 0, 0.7), add = TRUE, lwd = 2)  # Red color with 50% transpa
```

```r
# Add a legend
legend("bottomright", legend = c("Basic Model", "Interaction Model"),
       col = c("blue", rgb(1, 0, 0, 0.7)), lwd = 2)
```




```r
# AUC for the basic model
auc_logit <- auc(roc_logit)

# AUC for the interaction model
auc_logit_interaction <- auc(roc_logit_interaction)

# Print AUC values
print(paste("AUC for Basic Model:", round(auc_logit, 3)))
```

```
## [1] "AUC for Basic Model: 0.817"
```

```r
print(paste("AUC for Interaction Model:", round(auc_logit_interaction, 3)))
```

```
## [1] "AUC for Interaction Model: 0.821"
```

Now, you can print and plot the resulting `roc` objects.

```r
# Print ROC details for the basic model
print(roc_logit)
```

```
##
## Call:
## roc.default(response = test$Survived, predictor = test$SurvPredictlogit,     levels = c(0, 1))
##
## Data: test$SurvPredictlogit in 112 controls (test$Survived 0) < 66 cases (test$Survived 1).
## Area under the curve: 0.8166
```

```r
# Print ROC details for the interaction model
print(roc_logit_interaction)
```

```
##
## Call:
## roc.default(response = test$Survived, predictor = test$SurvPredictlogitInteraction,     levels = c(0
##
## Data: test$SurvPredictlogitInteraction in 112 controls (test$Survived 0) < 66 cases (test$Survived 1
## Area under the curve: 0.8212
```

In your own words, how would you interpret these ROC curves? What do you think about the ROC-AUCs we observe here?

- **Area Under the Curve (AUC)**:
  - The **basic logistic regression model** has an AUC of **0.8166**, indicating good discriminative ability.
  - The **interaction model** has a slightly higher AUC of **0.8212**, suggesting it performs marginally better at distinguishing between survivors and non-survivors.
- **What AUC Represents**:
  - AUC measures the ability of a model to distinguish between classes (e.g., survivors vs. non-survivors) regardless of the classification threshold.
  - An AUC of 0.5 indicates no discriminative power (equivalent to random guessing), while an AUC of 1.0 indicates perfect discrimination.
  - Both models here have AUCs well above 0.8, which is considered good. This suggests that the models have strong predictive performance.
- **Comparison of Models**:
  - The interaction model performs slightly better, with an AUC improvement of approximately **0.0046** (from 0.8166 to 0.8212).
  - The inclusion of interaction terms or non-linear transformations might capture relationships between variables that the basic model misses, leading to this small improvement.
- **Practical Implications**:
  - The small difference in AUC values suggests that while the interaction model has slightly better predictive performance, the improvement may not be substantial in practical terms.
  - Both models are likely adequate for most use cases, but the interaction model could be preferred if higher predictive accuracy is critical.

Start text...

end As a next step, we want to predict class membership given the risk scores of our two models. Here we use the default classification threshold, 0.5.

```
# Predicted class for the basic logistic regression model
test$Predicted_Class_Logit <- ifelse(test$SurvPredictlogit >= 0.5, 1, 0)

# Predicted class for the interaction logistic regression model
test$Predicted_Class_Logit_Interaction <- ifelse(test$SurvPredictlogitInteraction >= 0.5, 1, 0)

# View the predictions
head(test[, c("Survived", "SurvPredictlogit", "SurvPredictlogitInteraction",
                  "Predicted_Class_Logit", "Predicted_Class_Logit_Interaction")])
```

```
##   Survived SurvPredictlogit SurvPredictlogitInteraction Predicted_Class_Logit
## 1        0       0.08088622                  0.08419932                     0
## 2        1       0.92977319                  0.94525788                     1
## 3        1       0.52862580                  0.54865547                     1
## 4        1       0.88110542                  0.89955642                     1
## 5        0       0.05671222                  0.04078152                     0
## 7        0       0.29135061                  0.30396686                     0
##   Predicted_Class_Logit_Interaction
## 1                                 0
## 2                                 1
## 3                                 1
## 4                                 1
## 5                                 0
## 7                                 0
```

On this basis, we can use `confusionMatrix()` to get some performance measures for the predicted classes.

```
library(caret)

# Confusion matrix for the basic model
confusionMatrix(as.factor(test$Predicted_Class_Logit), as.factor(test$Survived))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 97 21
##          1 15 45
##
##                Accuracy : 0.7978
##                  95% CI : (0.7312, 0.8541)
##     No Information Rate : 0.6292
##     P-Value [Acc > NIR] : 8.829e-07
##
##                   Kappa : 0.5583
##
##  Mcnemar's Test P-Value : 0.4047
```

```
##
##             Sensitivity : 0.8661
##             Specificity : 0.6818
##          Pos Pred Value : 0.8220
##          Neg Pred Value : 0.7500
##              Prevalence : 0.6292
##          Detection Rate : 0.5449
##    Detection Prevalence : 0.6629
##       Balanced Accuracy : 0.7739
##
##        'Positive' Class : 0
##
```

```r
# Confusion matrix for the interaction model
confusionMatrix(as.factor(test$Predicted_Class_Logit_Interaction), as.factor(test$Survived))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 97 21
##          1 15 45
##
##                Accuracy : 0.7978
##                  95% CI : (0.7312, 0.8541)
##     No Information Rate : 0.6292
##     P-Value [Acc > NIR] : 8.829e-07
##
##                   Kappa : 0.5583
##
##  Mcnemar's Test P-Value : 0.4047
##
##             Sensitivity : 0.8661
##             Specificity : 0.6818
##          Pos Pred Value : 0.8220
##          Neg Pred Value : 0.7500
##              Prevalence : 0.6292
##          Detection Rate : 0.5449
##    Detection Prevalence : 0.6629
##       Balanced Accuracy : 0.7739
##
##        'Positive' Class : 0
##
```

Briefly explain potential limitations when measuring prediction performance as carried out in the last two code chunks.

1. Imbalanced Classes:

- **Prevalence**: The dataset shows a prevalence of 62.92% for class 0 (non-survivors), which means the classes are imbalanced. This can affect performance metrics, especially if the model is biased towards the majority class.

- **Impact**: The accuracy metric (79.78%) might seem high, but it could be misleading if the model simply predicts the majority class (0) most of the time. For instance, predicting `0` for most observations would still yield a high accuracy if most of the observations are `0`.

2. Accuracy vs. Other Metrics:

- **Accuracy**: While accuracy is a common metric, it is not always the best measure, especially in imbalanced datasets. A model could achieve high accuracy by predicting the majority class correctly most of the time, but it might not perform well in identifying the minority class (1, survivors in this case).
- **Alternative Metrics**: Sensitivity (recall) and specificity are more informative in this case. For example:
  - **Sensitivity**: The model correctly identifies 86.61% of survivors (class 1), which is good.
  - **Specificity**: The model correctly identifies 68.18% of non-survivors (class 0), which is lower, indicating room for improvement in detecting the majority class.

3. Positive Predictive Value (PPV) and Negative Predictive Value (NPV):

- **Positive Predictive Value**: The model predicts `Survived` (class 1) 82.20% of the time correctly, which is good, but this metric may not always be stable if the class distribution changes.
- **Negative Predictive Value**: The model correctly predicts `Not Survived` (class 0) 75.00% of the time. While it is decent, it's important to consider that PPV and NPV are sensitive to the prevalence of the classes.

Start text...

end