CMSC/LING/STAT 208: Machine Learning

Abhishek Chakraborty [Much of the content in these slides have been adapted from *ISLR2* by James et al. and *HOMLR* by Boehmke & Greenwell]

Your Turn!!!

You will work with a dataset containing information on employee attrition. Please load the dataset using the code below.

```
attrition <- readRDS("attrition.rds")
```

Objective: The task is to predict `Attrition` (Yes/No) using the rest of the variables in the data (predictors/features).

- **Step 1**: Investigate the dataset
  - What are the types of features? - categorical or numeric
  - If categorical, are they ordinal or nominal? If ordinal, are their levels in appropriate order? You can use the `levels` function to check the ordering.
  - Are there any features with missing entries?
  - Are there any zv/nzv features?
- **Step 2**: Split the data into training and test sets (70-30 split)
- **Step 3**: Perform required data preprocessing and create the blueprint. If using `step_dummy()`, set `one_hot = FALSE`.
- **Step 4**: Implement 5-fold CV (1 repeat) to compare the performance of the following models. Use `metric = "Accuracy"`.
  - a logistic regression model (`method = "glm"` and `family = "binomial"`)
  - a KNN classifier with the optimal `K` chosen by CV (`method = "knn"`). Use a grid of `K` values $1, 2, \ldots, 10$.

What is the optimal `K` chosen? How do the models compare in terms of the CV accuracies?

- **Step 5**: Build your final optimal model. Obtain probability and class label predictions for the test set (use threshold of 0.5). Create the corresponding confusion matrix and report the test set accuracy. Also, create the ROC curve for the optimal model and report the AUC.

Your Turn!!! Step 1

```
glimpse(attrition)    # types of variables
```

```
## Rows: 1,470
## Columns: 31
## $ Age                      <int> 41, 49, 37, 33, 27, 32, 59, 30, 38, 36, 35, 2…
## $ Attrition                <fct> Yes, No, Yes, No, No, No, No, No, No, No, No,…
## $ BusinessTravel           <fct> Travel_Rarely, Travel_Frequently, Travel_Rare…
## $ DailyRate                <int> 1102, 279, 1373, 1392, 591, 1005, 1324, 1358,…
## $ Department               <fct> Sales, Research_Development, Research_Develop…
## $ DistanceFromHome         <int> 1, 8, 2, 3, 2, 2, 3, 24, 23, 27, 16, 15, 26, …
## $ Education                <fct> College, Below_College, College, Master, Belo…
## $ EducationField           <fct> Life_Sciences, Life_Sciences, Other, Life_Sci…
## $ EnvironmentSatisfaction  <fct> Medium, High, Very_High, Very_High, Low, Very…
## $ Gender                   <fct> Female, Male, Male, Female, Male, Male, Femal…
## $ HourlyRate               <int> 94, 61, 92, 56, 40, 79, 81, 67, 44, 94, 84, 4…
## $ JobInvolvement           <fct> High, Medium, Medium, High, High, High, Very_…
## $ JobLevel                 <int> 2, 2, 1, 1, 1, 1, 1, 1, 3, 2, 1, 2, 1, 1, 1, …
## $ JobRole                  <fct> Sales_Executive, Research_Scientist, Laborato…
## $ JobSatisfaction          <fct> Very_High, Medium, High, High, Medium, Very_H…
## $ MaritalStatus            <fct> Single, Married, Single, Married, Married, Si…
## $ MonthlyIncome            <int> 5993, 5130, 2090, 2909, 3468, 3068, 2670, 269…
## $ MonthlyRate              <int> 19479, 24907, 2396, 23159, 16632, 11864, 9964…
## $ NumCompaniesWorked       <int> 8, 1, 6, 1, 9, 0, 4, 1, 0, 6, 0, 0, 1, 0, 5, …
## $ OverTime                 <fct> Yes, No, Yes, Yes, No, No, Yes, No, No, No, N…
## $ PercentSalaryHike        <int> 11, 23, 15, 11, 12, 13, 20, 22, 21, 13, 13, 1…
## $ PerformanceRating        <fct> Excellent, Outstanding, Excellent, Excellent,…
## $ RelationshipSatisfaction <fct> Low, Very_High, Medium, High, Very_High, High…
## $ StockOptionLevel         <int> 0, 1, 0, 0, 1, 0, 3, 1, 0, 2, 1, 0, 1, 1, 0, …
## $ TotalWorkingYears        <int> 8, 10, 7, 8, 6, 8, 12, 1, 10, 17, 6, 10, 5, 3…
## $ TrainingTimesLastYear    <int> 0, 3, 3, 3, 3, 2, 3, 2, 2, 3, 5, 3, 1, 2, 4, …
## $ WorkLifeBalance          <fct> Bad, Better, Better, Better, Better, Good, Go…
## $ YearsAtCompany           <int> 6, 10, 0, 8, 2, 7, 1, 1, 9, 7, 5, 9, 5, 2, 4,…
## $ YearsInCurrentRole       <int> 4, 7, 0, 7, 2, 7, 0, 0, 7, 7, 4, 5, 2, 2, 2, …
## $ YearsSinceLastPromotion  <int> 0, 1, 0, 3, 2, 3, 0, 0, 1, 7, 0, 0, 4, 1, 0, …
## $ YearsWithCurrManager     <int> 5, 7, 0, 0, 2, 6, 0, 0, 8, 7, 3, 8, 3, 2, 3, …
```

Numerical variables are represented as `<int>`, categorical variables are represented as `<fct>`.

Your Turn!!! Step 1

```
# checking the levels of ordinal variables

levels(attrition$BusinessTravel)
```

```
## [1] "Non-Travel"        "Travel_Frequently" "Travel_Rarely"
```

```
levels(attrition$Education)
```

```
## [1] "Below_College" "College"       "Bachelor"      "Master"
## [5] "Doctor"
```

```
levels(attrition$EnvironmentSatisfaction)
```

```
## [1] "Low"       "Medium"    "High"      "Very_High"
```

```
levels(attrition$JobInvolvement)
```

```
## [1] "Low"       "Medium"    "High"      "Very_High"
```

Your Turn!!! Step 1

```
# checking the levels of ordinal variables

levels(attrition$JobSatisfaction)
```

```
## [1] "Low"       "Medium"    "High"      "Very_High"
```

```
levels(attrition$PerformanceRating)
```

```
## [1] "Excellent"   "Outstanding"
```

```
levels(attrition$RelationshipSatisfaction)
```

```
## [1] "Low"       "Medium"    "High"      "Very_High"
```

```
levels(attrition$WorkLifeBalance)
```

```
## [1] "Bad"    "Good"   "Better" "Best"
```

Your Turn!!! Step 1

```r
# reorder levels of 'BusinessTravel'

attrition$BusinessTravel <- factor(attrition$BusinessTravel, levels = c("Non-Travel", "Travel_Rarely", "Travel_Frequently"))

levels(attrition$BusinessTravel)
```

```
## [1] "Non-Travel"        "Travel_Rarely"     "Travel_Frequently"
```

Your Turn!!! Step 1

```r
sum(is.na(attrition))  # no missing entries
```

```
## [1] 0
```

Your Turn!!! Step 1

```r
nearZeroVar(attrition, saveMetrics = TRUE)  # no zv/nzv features
```

```
##                          freqRatio percentUnique zeroVar   nzv
## Age                       1.012987     2.9251701   FALSE FALSE
## Attrition                 5.202532     0.1360544   FALSE FALSE
## BusinessTravel            3.765343     0.2040816   FALSE FALSE
## DailyRate                 1.200000    60.2721088   FALSE FALSE
## Department                2.154709     0.2040816   FALSE FALSE
## DistanceFromHome          1.014423     1.9727891   FALSE FALSE
## Education                 1.437186     0.3401361   FALSE FALSE
## EducationField            1.306034     0.4081633   FALSE FALSE
## EnvironmentSatisfaction   1.015695     0.2721088   FALSE FALSE
## Gender                    1.500000     0.1360544   FALSE FALSE
## HourlyRate                1.035714     4.8299320   FALSE FALSE
## JobInvolvement            2.314667     0.2721088   FALSE FALSE
## JobLevel                  1.016854     0.3401361   FALSE FALSE
## JobRole                   1.116438     0.6122449   FALSE FALSE
## JobSatisfaction           1.038462     0.2721088   FALSE FALSE
## MaritalStatus             1.431915     0.2040816   FALSE FALSE
## MonthlyIncome             1.333333    91.7687075   FALSE FALSE
## MonthlyRate               1.000000    97.0748299   FALSE FALSE
## NumCompaniesWorked        2.644670     0.6802721   FALSE FALSE
## OverTime                  2.533654     0.1360544   FALSE FALSE
## PercentSalaryHike         1.004785     1.0204082   FALSE FALSE
## PerformanceRating         5.504425     0.1360544   FALSE FALSE
## RelationshipSatisfaction  1.062500     0.2721088   FALSE FALSE
## StockOptionLevel          1.058725     0.2721088   FALSE FALSE
## TotalWorkingYears         1.616000     2.7210884   FALSE FALSE
## TrainingTimesLastYear     1.114053     0.4761905   FALSE FALSE
## WorkLifeBalance           2.595930     0.2721088   FALSE FALSE
## YearsAtCompany            1.146199     2.5170068   FALSE FALSE
## YearsInCurrentRole        1.524590     1.2925170   FALSE FALSE
## YearsSinceLastPromotion   1.627451     1.0884354   FALSE FALSE
## YearsWithCurrManager      1.307985     1.2244898   FALSE FALSE
```

Your Turn!!! Step 2

```
# split data

set.seed(042324)

train_index <- createDataPartition(attrition$Attrition, p = 0.7, list = FALSE)

attrition_train <- attrition[train_index, ]

attrition_test <- attrition[-train_index, ]
```

Your Turn!!! Step 3

```
# create recipe, blueprint, prepare, and bake

attrition_recipe <- recipe(formula = Attrition ~ ., data = attrition_train)   # sets up the type and role of variables


blueprint <- attrition_recipe %>%

  # convert ordinal categorical features to integers
  step_integer(BusinessTravel, Education, EnvironmentSatisfaction, JobInvolvement,
               JobSatisfaction, PerformanceRating, RelationshipSatisfaction,
               WorkLifeBalance) %>%

  # center and scale features
  step_center(all_numeric()) %>%
  step_scale(all_numeric()) %>%

  # create dummy variables for nominal categorical features
  step_dummy(all_nominal(), -all_outcomes(), one_hot = FALSE)


prepare <- prep(blueprint, data = attrition_train)    # estimate feature engineering parameters based on training data


baked_train <- bake(prepare, new_data = attrition_train)   # apply the blueprint to training data for building final/optimal model

baked_test <- bake(prepare, new_data = attrition_test)    # apply the blueprint to test data for future use
```

Your Turn!!! Step 4

```
# perform CV

set.seed(042324)

cv_specs <- trainControl(method = "repeatedcv", number = 5, repeats = 1)   # 5-fold CV (1 repeat)


# CV with logistic regression

logistic_fit <- train(blueprint,
                  data = attrition_train,
                  method = "glm",
                  family = "binomial",
                  trControl = cv_specs,
                  metric = "Accuracy")

logistic_fit
```

```
## Generalized Linear Model
##
## 1030 samples
##   30 predictor
##    2 classes: 'No', 'Yes'
##
## Recipe steps: integer, center, scale, dummy
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 824, 824, 824, 824, 824
## Resampling results:
##
##   Accuracy   Kappa
##   0.8669903  0.4451545
```

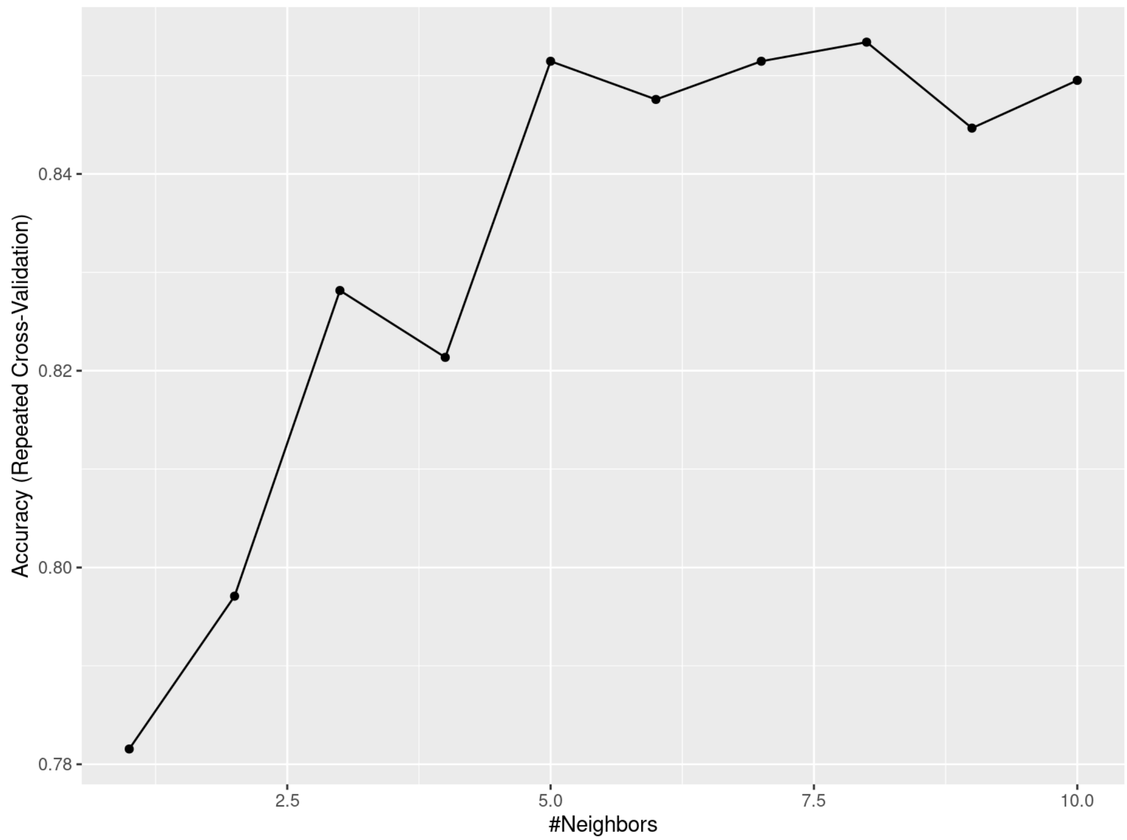
Your Turn!!! Step 4

```r
# CV with KNN

set.seed(042324)

k_grid <- expand.grid(k = seq(1, 10, by = 1))

knn_fit <- train(blueprint,
                 data = attrition_train,
                 method = "knn",
                 trControl = cv_specs,
                 tuneGrid = k_grid,
                 metric = "Accuracy")

knn_fit
```

```
## k-Nearest Neighbors
##
## 1030 samples
##   30 predictor
##    2 classes: 'No', 'Yes'
##
## Recipe steps: integer, center, scale, dummy
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 824, 824, 824, 824, 824
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.7815534  0.08241854
##    2  0.7970874  0.15783093
##    3  0.8281553  0.14031222
##    4  0.8213592  0.11676215
##    5  0.8514563  0.20214411
##    6  0.8475728  0.16595651
##    7  0.8514563  0.15650159
##    8  0.8533981  0.16823007
##    9  0.8446602  0.09498814
##   10  0.8495146  0.13065544
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 8.
```

Your Turn!!! Step 4

```
ggplot(knn_fit)
```



Your Turn!!! Step 5

```r
# build final optimal model and obtain predictions on test set

final_model <- glm(Attrition ~ ., data = baked_train, family = binomial)    # build final model

final_model_prob_preds <- predict(object = final_model, newdata = baked_test, type = "response")   # probability predictions on test data

threshold <- 0.5

final_model_class_preds <- factor(ifelse(final_model_prob_preds > threshold, "Yes", "No"))   # class label predictions on test data
```

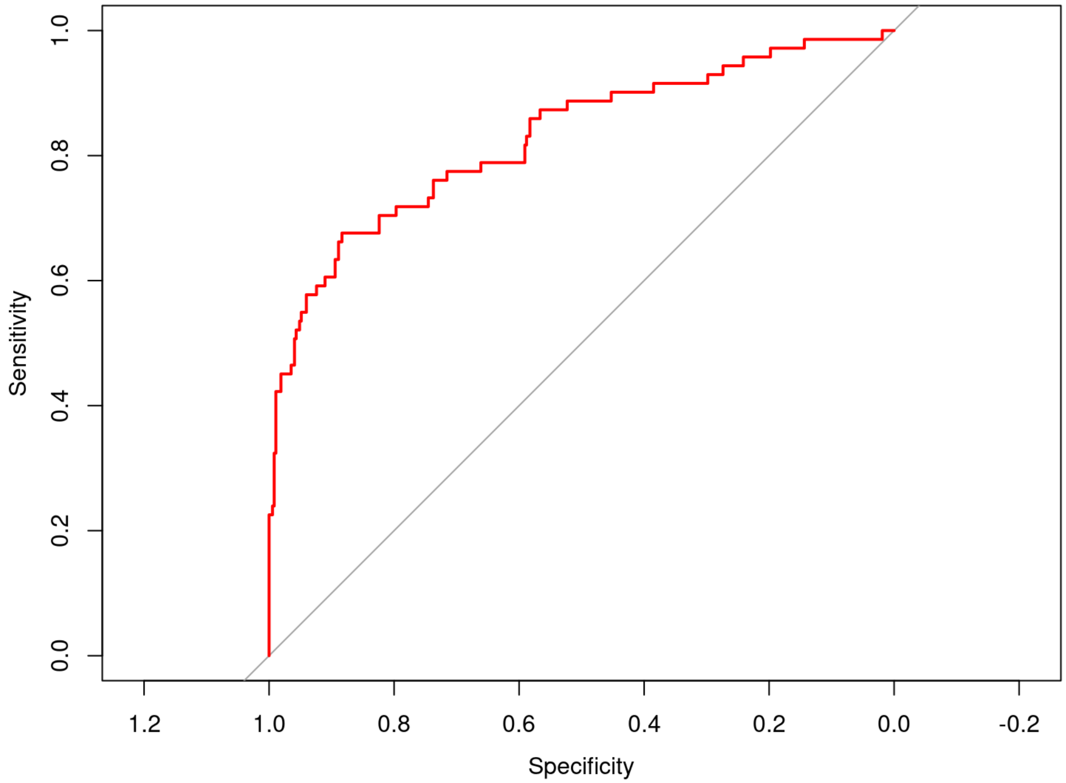
Your Turn!!! Step 5

```
# create confusion matrix

confusionMatrix(data = final_model_class_preds, reference = baked_test$Attrition, positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  360  39
##        Yes   9  32
##
##                Accuracy : 0.8909
##                  95% CI : (0.858, 0.9185)
##     No Information Rate : 0.8386
##     P-Value [Acc > NIR] : 0.001159
##
##                   Kappa : 0.514
##
##  Mcnemar's Test P-Value : 2.842e-05
##
##             Sensitivity : 0.45070
##             Specificity : 0.97561
##          Pos Pred Value : 0.78049
##          Neg Pred Value : 0.90226
##              Prevalence : 0.16136
##          Detection Rate : 0.07273
##    Detection Prevalence : 0.09318
##       Balanced Accuracy : 0.71316
##
##        'Positive' Class : Yes
##
```

Your Turn!!! Step 5

```
# create ROC cuvre and compute AUC

library(pROC)

roc_object <- roc(response = baked_test$Attrition, predictor = final_model_prob_preds)

plot(roc_object, col = "red")
```




```
auc(roc_object)
```

```
## Area under the curve: 0.8278
```