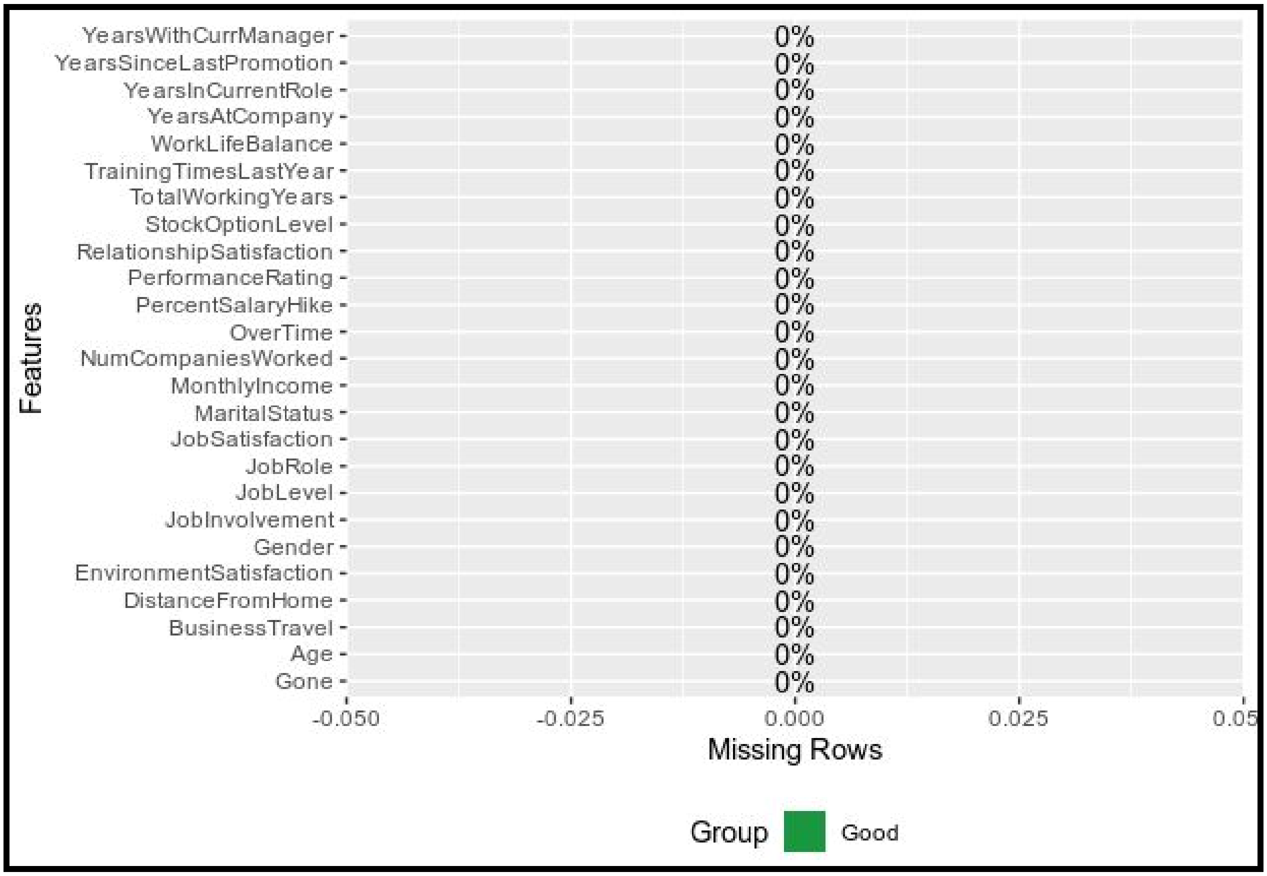
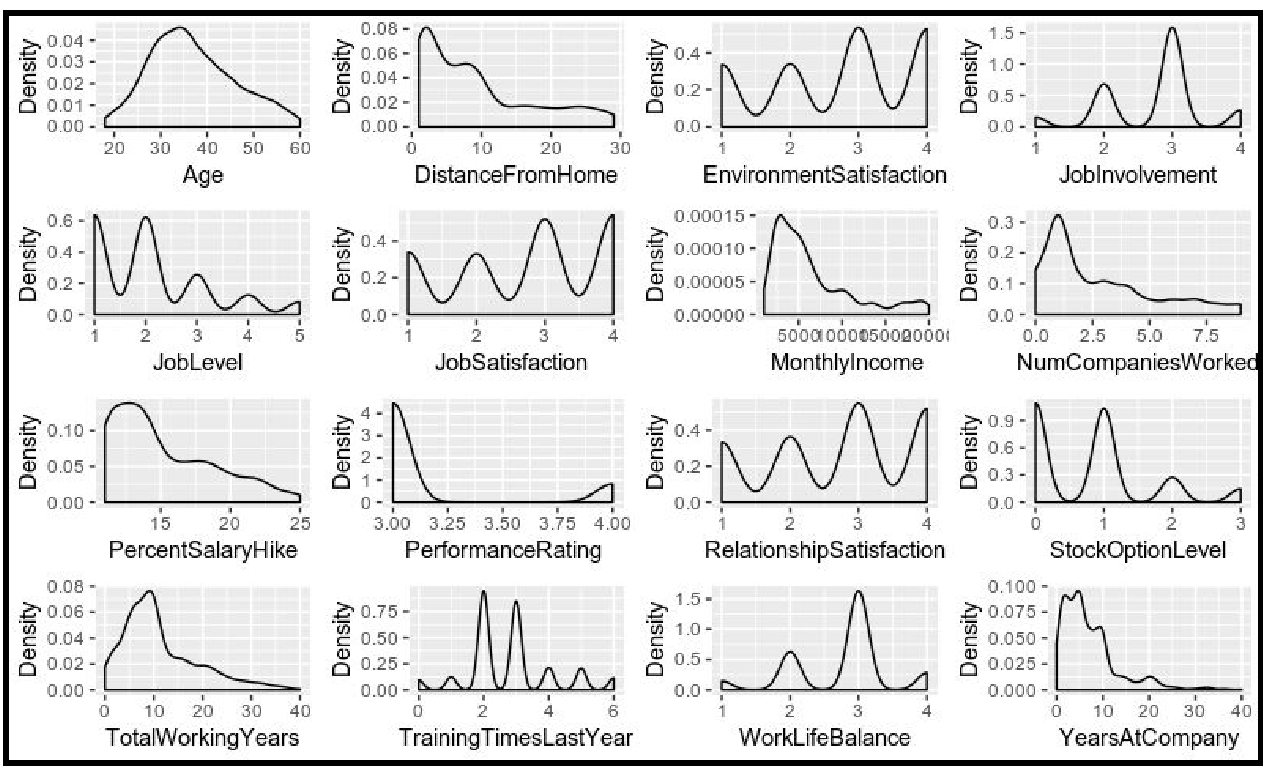
Using the HR_Churn dataset, so the following:

1) Perform all data inspections (it is a binary classification exercise with Gone as the response)

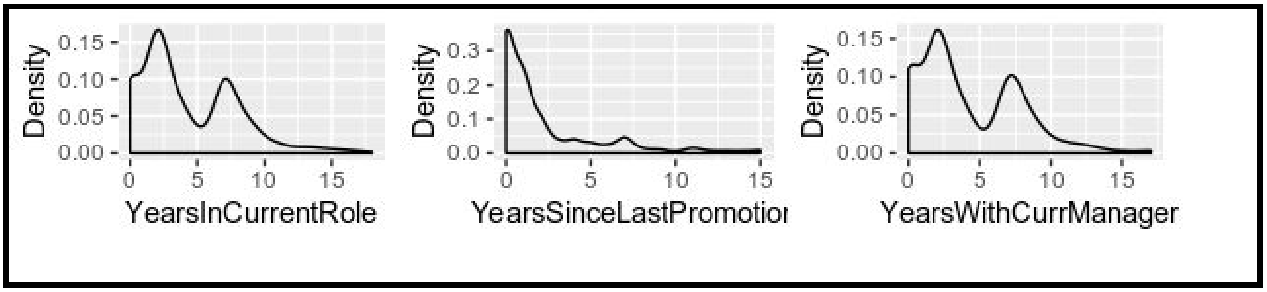
```
> glimpse(data_raw)
Observations: 1,470
Variables: 25
$ Gone                     <fct> Yes, No, Yes, No, No, No, No, No, N…
$ Age                      <int> 41, 49, 37, 33, 27, 32, 59, 30, 38,…
$ BusinessTravel           <fct> Travel_Rarely, Travel_Frequently, T…
$ DistanceFromHome         <int> 1, 8, 2, 3, 2, 2, 3, 24, 23, 27, 16…
$ EnvironmentSatisfaction  <int> 2, 3, 4, 4, 1, 4, 3, 4, 4, 3, 1, 4,…
$ Gender                   <fct> Female, Male, Male, Female, Male, M…
$ JobInvolvement           <int> 3, 2, 2, 3, 3, 3, 4, 3, 2, 3, 4, 2,…
$ JobLevel                 <int> 2, 2, 1, 1, 1, 1, 1, 1, 3, 2, 1, 2,…
$ JobRole                  <fct> Sales Executive, Research Scientist…
$ JobSatisfaction          <int> 4, 2, 3, 3, 2, 4, 1, 3, 3, 3, 2, 3,…
$ MaritalStatus            <fct> Single, Married, Single, Married, M…
$ MonthlyIncome            <int> 5993, 5130, 2090, 2909, 3468, 3068,…
$ NumCompaniesWorked       <int> 8, 1, 6, 1, 9, 0, 4, 1, 0, 6, 0, 0,…
$ OverTime                 <fct> Yes, No, Yes, Yes, No, No, Yes, No,…
$ PercentSalaryHike        <int> 11, 23, 15, 11, 12, 13, 20, 22, 21,…
$ PerformanceRating        <int> 3, 4, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3,…
$ RelationshipSatisfaction <int> 1, 4, 2, 3, 4, 3, 1, 2, 2, 2, 3, 4,…
$ StockOptionLevel         <int> 0, 1, 0, 0, 1, 0, 3, 1, 0, 2, 1, 0,…
$ TotalWorkingYears        <int> 8, 10, 7, 8, 6, 8, 12, 1, 10, 17, 6…
$ TrainingTimesLastYear    <int> 0, 3, 3, 3, 3, 2, 3, 2, 2, 3, 5, 3,…
$ WorkLifeBalance          <int> 1, 3, 3, 3, 3, 2, 2, 3, 3, 2, 3, 3,…
$ YearsAtCompany           <int> 6, 10, 0, 8, 2, 7, 1, 1, 9, 7, 5, 9…
$ YearsInCurrentRole       <int> 4, 7, 0, 7, 2, 7, 0, 0, 7, 7, 4, 5,…
$ YearsSinceLastPromotion  <int> 0, 1, 0, 3, 2, 3, 0, 0, 1, 7, 0, 0,…
$ YearsWithCurrManager     <int> 5, 7, 0, 0, 2, 6, 0, 0, 8, 7, 3, 8,…
```



YearsWithCurrManager
YearsSinceLastPromotion
YearsInCurrentRole
YearsAtCompany
WorkLifeBalance
TrainingTimesLastYear
TotalWorkingYears
StockOptionLevel
RelationshipSatisfaction
PerformanceRating
PercentSalaryHike
OverTime
NumCompaniesWorked
MonthlyIncome
MaritalStatus
JobSatisfaction
JobRole
JobLevel
JobInvolvement
Gender
EnvironmentSatisfaction
DistanceFromHome
BusinessTravel
Age
Gone
0%
Features
-0.050
-0.025
0.000
0.025
0.05
Missing Rows
Group
Good



Density
Age
DistanceFromHome
EnvironmentSatisfaction
JobInvolvement
JobLevel
JobSatisfaction
MonthlyIncome
NumCompaniesWorked
PercentSalaryHike
PerformanceRating
RelationshipSatisfaction
StockOptionLevel
TotalWorkingYears
TrainingTimesLastYear
WorkLifeBalance
YearsAtCompany



2) Alter the dataset, if required (i.e., remove unnecessary predictors, etc.)

3) Split into training and test with appropriate sizes for each

```
> set.seed(993)
> train_test_split = initial_split(data_raw, prop=.70)
> train_test_split
<1029/441/1470>
```

4) Create and implement a recipe as required

```
> cake
Data Recipe

Inputs:

      role #variables
   outcome          1
 predictor         24

Training data contained 1029 data points and no missing data.

Operations:

Dummy variables from BusinessTravel, Gender, JobRole, MaritalStatus, OverTime [trained]
Box-Cox transformation on Age, DistanceFromHome, ... [trained]
```

5) Run a cross-validated lasso model on the data and identify all relevant predictors

The relevant predictors are the predictors with coefficient values.

```
> coef(cv.lasso)
39 x 1 sparse Matrix of class "dgCMatrix"
                                              1
(Intercept)                        9.215526e+00
Age                               -3.886817e-01
DistanceFromHome                   7.004894e-02
EnvironmentSatisfaction           -2.723468e-01
JobInvolvement                    -2.441354e-01
JobLevel                          -2.644646e-01
JobSatisfaction                   -2.323246e-01
MonthlyIncome                     -1.272455e+00
NumCompaniesWorked                 6.699899e-02
PercentSalaryHike                 -1.198218e+00
PerformanceRating                  .
RelationshipSatisfaction           .
StockOptionLevel                  -5.476428e-02
TotalWorkingYears                 -4.364066e-03
TrainingTimesLastYear             -9.575895e-03
WorkLifeBalance                   -1.270784e-01
YearsAtCompany                     .
YearsInCurrentRole                -2.261339e-02
YearsSinceLastPromotion            .
YearsWithCurrManager              -2.240592e-02
BusinessTravel_Non.Travel         -6.948522e-02
BusinessTravel_Travel_Frequently   4.661320e-01
BusinessTravel_Travel_Rarely       .
Gender_Female                     -9.269181e-02
Gender_Male                        1.810319e-16
JobRole_Healthcare.Representative  .
JobRole_Human.Resources            .
JobRole_Laboratory.Technician      2.022270e-01
JobRole_Manager                    .
JobRole_Manufacturing.Director     .
JobRole_Research.Director          .
```

6) Rerun cross-validated logistic regression and LDA models on the data with only relevant predictors incorporated.

```
set.seed(3854)
train_x <-  model.matrix(Gone ~ . -1, data = train_clean2)
train_y <-  train_clean2$Gone
grid = 10^seq(10,-2,by=-.1)
cv.lasso2 <-  cv.glmnet(train_x, train_y, family="binomial", alpha=1, lambda=grid)
cv.lasso2
plot(cv.lasso2)
```

```
> lda.fit
Linear Discriminant Analysis

1029 samples
  34 predictor
   2 classes: 'No', 'Yes'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 925, 927, 927, 925, 926, 926, ...
Resampling results:

  ROC        Sens       Spec
  0.8276416  0.9660029  0.4231618
```

7) Generate confusion matrices on the test data and describe the results in terms of sensitivity, specificity, positive predictive value, negative predictive value, and kappa for both logistic and LDA.

Sensitivity: We are 96% sure that the positive results (those who are going to leave) in our sample are actually positive (left in reality) and identified which appears to be a great result.
Specificity: It is the value the shows all those who were not going to leave, be correctly identified, the value here is only 42% which does not appear to be a healthy number.
Pos pred value- In LDA the pos pred value is 0.89; there are 12 people we predict they are going leave but they didn't leave and we predict 27 people are going to leave and they left.
In Lasso the pos pred value is 0.87; there are 6 people we predict they are going leave but they didn't leave and we predict 19 people are going to leave and they left.
Neg pred value- In LDA the Neg pred value is 0.69; there are 358 people we predict they are not going to leave and they didn't leave and we predict 44 people are not going to leave but they left.
In Lasso the Neg pred value is 0.76; there are 364 people we predict they are not going to leave and they didn't left and we predict 52 people are not going to leave but they left.

- As discussed in class false negative predictors are more dangerous than false positive predictors. The false negative predictor of LDA predicted 44 people are not going to leave while real data they left and in Lasso 53 people are not going to leave but they left. So, in this case the LDA prediction is more accurate that lasso in terms of false negative predictors.

Kappa- LDA 0.42 , Lasso 0.34 this means that the probability of LDA is better than Lasso.

```
> lda.pred = predict(lda.fit, test_clean2)
> confusionMatrix(lda.pred, test_clean2$Gone)
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  358  44
       Yes  12  27

               Accuracy : 0.873
                 95% CI : (0.8383, 0.9026)
    No Information Rate : 0.839
    P-Value [Acc > NIR] : 0.02741

                  Kappa : 0.4253
 Mcnemar's Test P-Value : 3.435e-05

            Sensitivity : 0.9676
            Specificity : 0.3803
         Pos Pred Value : 0.8905
         Neg Pred Value : 0.6923
             Prevalence : 0.8390
         Detection Rate : 0.8118
   Detection Prevalence : 0.9116
      Balanced Accuracy : 0.6739

       'Positive' Class : No
```

```
> confusionMatrix(table(lasso.classes,test_y))
Confusion Matrix and Statistics

             test_y
lasso.classes  No Yes
          No  364  52
          Yes   6  19

               Accuracy : 0.8685
                 95% CI : (0.8333, 0.8986)
    No Information Rate : 0.839
    P-Value [Acc > NIR] : 0.04988

                  Kappa : 0.3405
 Mcnemar's Test P-Value : 3.446e-09

            Sensitivity : 0.9838
            Specificity : 0.2676
         Pos Pred Value : 0.8750
         Neg Pred Value : 0.7600
             Prevalence : 0.8390
         Detection Rate : 0.8254
   Detection Prevalence : 0.9433
      Balanced Accuracy : 0.6257

       'Positive' Class : No
```

library(MASS)
library(recipes)
library(rsample)
library(car)
library(DataExplorer)
library(polycor)
library(tidyverse)
library(ROCR)
library(caret)
library(glmnet)

data_raw = read.csv(file="/home/neo/Downloads/HR_Churn.csv", header=TRUE, sep=",")
glimpse(data_raw)

plot_missing(data_raw)
plot_density(data_raw)

set.seed(993)
train_test_split = initial_split(data_raw, prop=.70)
train_test_split

```
train_tbl = training(train_test_split)
test_tbl = testing(train_test_split)

cake = recipe(Gone ~., data=train_tbl) %>%
  step_dummy(all_nominal(), -all_outcomes(), one_hot=TRUE) %>%
  step_BoxCox(all_predictors(), -all_outcomes()) %>%
  prep(data=train_tbl)

cake

train_clean = bake(cake,new_data=train_tbl)
test_clean = bake(cake,new_data=test_tbl)

train_clean$Gone = as.factor(train_clean$Gone)
test_clean$Gone = as.factor(test_clean$Gone)
glimpse(train_clean)

set.seed(3432)
train_x <-  model.matrix(Gone ~ . -1, data = train_clean)
train_y <-  train_clean$Gone
grid = 10^seq(10,-2,by=-.1)
cv.lasso <-  cv.glmnet(train_x, train_y, family="binomial", alpha=1, lambda=grid)
plot(cv.lasso)

best_lambda = cv.lasso$lambda.min
coef(cv.lasso)

train_clean2 = train_clean %>%
select(-PerformanceRating,-RelationshipSatisfaction,-YearsAtCompany,-YearsSinceLast
Promotion)
glimpse(train_clean2)
test_clean2 = test_clean %>%
select(-PerformanceRating,-RelationshipSatisfaction,-YearsAtCompany,-YearsSinceLast
Promotion)

set.seed(3854)
train_x <-  model.matrix(Gone ~ . -1, data = train_clean2)
train_y <-  train_clean2$Gone
grid = 10^seq(10,-2,by=-.1)
cv.lasso2 <-  cv.glmnet(train_x, train_y, family="binomial", alpha=1, lambda=grid)
names(cv.lasso2)
plot(cv.lasso2)
```

```
best_lambda2 = cv.lasso2$lambda.min
best_lambda2
coef(cv.lasso2)

test_x <-  model.matrix(Gone ~ . -1, data = test_clean2)
test_y <-  test_clean2$Gone

lasso.pred = predict(cv.lasso2, s=best_lambda2,newx=data.matrix(test_x))
lasso.pred

lasso.classes = ifelse(lasso.pred>0,"Yes","No")
lasso.classes

confusionMatrix(table(lasso.classes,test_y))

control = trainControl(method="repeatedcv", number=10, repeats=3,
summaryFunction=twoClassSummary,classProbs=TRUE, savePredictions="final")
lda.fit = train(Gone ~., data=train_clean2, method="lda", metric="ROC",
trControl=control)
lda.fit

lda.pred = predict(lda.fit, test_clean2)
confusionMatrix(lda.pred, test_clean2$Gone)

#grid = expand.grid(alpha=1, lambda=10^seq(10,-2,length=1000))
#control = trainControl(method="repeatedcv", number=10, repeats=3)
#cv.lasso = train(Gone ~., data=train_clean,
                 #method="glmnet",
                 #trControl=control,
                 #tuneGrid=grid)

#cv.lasso$lambda.min
#coef(cv.lasso, s=0.1)
```

Selam

```
library(MASS)
library(recipes)
library(rsample)
library(car)
library(DataExplorer)
library(polycor)
library(tidyverse)
library(ROCR)
library(caret)
library(glmnet)

data_raw= HR_Churn
glimpse(data_raw)
hetcor(data_raw)


plot_missing(data_raw)
plot_density(data_raw)

set.seed(993)
train_test_split = initial_split(data_raw, prop=.70)
train_test_split

train_tbl = training(train_test_split)
test_tbl = testing(train_test_split)

cake = recipe(Gone ~., data=train_tbl) %>%
  step_dummy(all_nominal(), -all_outcomes(), one_hot=TRUE) %>%
  step_BoxCox(all_predictors(), -all_outcomes()) %>%
  prep(data=train_tbl)

cake

train_clean = bake(cake,new_data=train_tbl)
test_clean = bake(cake,new_data=test_tbl)

train_clean$Gone = as.factor(train_clean$Gone)
test_clean$Gone = as.factor(test_clean$Gone)
glimpse(train_clean)

set.seed(3432)
train_x <-  model.matrix(Gone ~ . -1, data = train_clean)
train_y <-  train_clean$Gone
```

```
grid = 10^seq(10,-2,by=-.1)
cv.lasso <-  cv.glmnet(train_x, train_y, family="binomial", alpha=1, lambda=grid)
plot(cv.lasso)

best_lambda = cv.lasso$lambda.min
coef(cv.lasso)

train_clean2 = train_clean %>%
select(-PerformanceRating,-RelationshipSatisfaction,-YearsAtCompany,-YearsSinceLast
Promotion)
glimpse(train_clean2)
test_clean2 = test_clean %>%
select(-PerformanceRating,-RelationshipSatisfaction,-YearsAtCompany,-YearsSinceLast
Promotion)

set.seed(3854)
train_x <-  model.matrix(Gone ~ . -1, data = train_clean2)
train_y <-  train_clean2$Gone
grid = 10^seq(10,-2,by=-.1)
cv.lasso2 <-  cv.glmnet(train_x, train_y, family="binomial", alpha=1, lambda=grid)
plot(cv.lasso2)

best_lambda = cv.lasso2$lambda.min
coef(cv.lasso2)

test_x <-  model.matrix(Gone ~ . -1, data = test_clean2)
test_y <-  test_clean2$Gone

lasso.pred = predict(cv.lasso2, s= best_lambda, newx= data.matrix(test_x), type =
"response" )
lasso.pred
lasso.classes = ifelse(lasso.pred>0.5, "Yes", "No")
lasso.classes
confusionMatrix(table(lasso.classes, test_clean2$Gone))


control = trainControl(method="repeatedcv", number=10, repeats=3,
summaryFunction=twoClassSummary,classProbs=TRUE, savePredictions="final")

lda.fit = train(Gone ~., data=train_clean2, method="lda", metric="ROC",
trControl=control)
lda.fit
```

```
lda.pred = predict(lda.fit, test_clean2)
confusionMatrix(lda.pred, test_clean2$Gone)

confusionMatrix()

#grid = expand.grid(alpha=1, lambda=10^seq(10,-2,length=1000))
#control = trainControl(method="repeatedcv", number=10, repeats=3)
#cv.lasso = train(Gone ~., data=train_clean,
#method="glmnet",
#trControl=control,
#tuneGrid=grid)

#cv.lasso$lambda.min
#coef(cv.lasso, s=0.1)
```