626 hw3

1. Utilize the training datasets to train a model that predicts the whether car price is less than $12,000.

Step 1. Import data, imputation, set categorical variables

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
df.small <- read.csv("C:/Users/linzh/Desktop/ECON 626/hw3/final_predcomp_training_data_small.csv")
df.large <- read.csv("C:/Users/linzh/Desktop/ECON 626/hw3/final_predcomp_training_data_large.csv")

# as df.small is a subset of df.large, identify the diff rows between those 2 datasets to get train dat
df.train <- anti_join(df.large, df.small)
```

```
## Joining, by = c("id", "price", "year", "odometer", "cylinders", "transmission", "drive", "type", "pa
```

```
# fill in na in year
df.train$year[is.na(df.train$year)] <- median(df.train$year, na.rm=TRUE)

#categorical variable: cylinders
unique(df.train$cylinders)
```

```
## [1]  4  6 NA  8 10 12  5  3
```

```
df.train$cylinders[is.na(df.train$cylinders)] <- 9

#categorical variable: drive
unique(df.train$drive)
```

```
## [1] "fwd" ""    "rwd" "4wd"
```

```r
mapping <- c("fwd" = 0, "rwd" = 1, "4wd" = 2)
df.train$drive <- mapping[df.train$drive]
df.train$drive[is.na(df.train$drive)] <- 3

#categorical variable: fuel
unique(df.train$fuel)
```

```
## [1] "gas"      "other"    "diesel"   ""         "hybrid"   "electric"
```

```r
mapping <- c("gas" = 0, "diesel" = 1, "hybrid" = 2, "electric" = 3, "other" = 4)
df.train$fuel <- mapping[df.train$fuel]
df.train$fuel[is.na(df.train$fuel)] <- 4

#categorical variable: transmission
unique(df.train$transmission)
```

```
## [1] "automatic" "other"     "manual"    ""
```

```r
mapping <- c("automatic" = 0, "manual" = 1, "other" = 2)
df.train$transmission <- mapping[df.train$transmission]
df.train$transmission[is.na(df.train$transmission)] <- 2

#categorical variable: condition
unique(df.train$condition)
```

```
## [1] "excellent" "good"      ""          "like new"  "fair"      "new"
## [7] "salvage"
```

```r
mapping <- c("new" = 0, "excellent" = 1, "good" = 2, "like new" = 3, "fair" = 4, "salvage" = 5)
df.train$condition <- mapping[df.train$condition]
df.train$condition[is.na(df.train$condition)] <- 6

#categorical variable: odometer
df.train$odometer <- cut(df.train$odometer, breaks=c(0, 50000, 100000, 150000, 200000, 250000,
                         300000, 350000, 400000, 500000), labels=c(0, 1, 2, 3, 4, 5, 6, 7, 8))
mapping <- c("0" = 0, "1" = 1, "2" = 2, "3" = 3, "4" = 4, "5" = 5, "6" = 6, "7" = 7, "8" = 8)
df.train$odometer <- mapping[df.train$odometer]
df.train$odometer[is.na(df.train$odometer)] <- 9

#categorical variable: manufacturer
df.train$manufacturer <- as.factor(df.train$manufacturer)
n <- length(unique(df.train$manufacturer))
mapping <- c(0:n)
df.train$manufacturer <- mapping[df.train$manufacturer]

# set cars with price below 12,000 to be 1, and above to be 0.
df.train$price.category <- cut(df.train$price, breaks=c(-Inf, 12000, Inf), labels=c(1, 0))
```

Step 2. Build a logistic model for categorical data

```
fit <- glm(price.category ~ year + odometer + transmission + condition + cylinders + drive
         + manufacturer + fuel + description_badcredit, data = df.train, family = 'binomial')
summary(fit)
```

```
##
## Call:
## glm(formula = price.category ~ year + odometer + transmission +
##     condition + cylinders + drive + manufacturer + fuel + description_badcredit,
##     family = "binomial", data = df.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0430  -0.8798   0.3545   0.8527   5.3616
##
## Coefficients:
##                          Estimate Std. Error  z value Pr(>|z|)
## (Intercept)           -2.726e+02  1.866e+00 -146.060   <2e-16 ***
## year                   1.349e-01  9.270e-04  145.498   <2e-16 ***
## odometer              -2.157e-01  2.007e-03 -107.488   <2e-16 ***
## transmission           6.197e-01  1.027e-02   60.347   <2e-16 ***
## condition              3.109e-03  2.170e-03    1.433    0.152
## cylinders              2.139e-01  2.585e-03   82.747   <2e-16 ***
## drive                  2.400e-01  4.365e-03   54.966   <2e-16 ***
## manufacturer          -7.530e-03  3.815e-04  -19.741   <2e-16 ***
## fuel                   2.228e-01  6.495e-03   34.301   <2e-16 ***
## description_badcredit  2.229e-01  1.231e-02   18.099   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 373697  on 270000  degrees of freedom
## Residual deviance: 289679  on 269991  degrees of freedom
## AIC: 289699
##
## Number of Fisher Scoring iterations: 5
```

```
# in sample validation
train_pred <- predict(fit, newdata = df.train, type = "response")
train_pred <- cut(train_pred, breaks=c(-Inf, 0.5, Inf), labels = c(1, 0))

# confusion matrix and accuracy
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
comparison <- table(train_pred, df.train$price.category)
confusionMatrix(comparison)
```

```
## Confusion Matrix and Statistics
```

```
##
##
## train_pred      1      0
##          1  97634  25967
##          0  30982 115418
##
##                Accuracy : 0.7891
##                  95% CI : (0.7875, 0.7906)
##     No Information Rate : 0.5236
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5765
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7591
##             Specificity : 0.8163
##          Pos Pred Value : 0.7899
##          Neg Pred Value : 0.7884
##              Prevalence : 0.4764
##          Detection Rate : 0.3616
##    Detection Prevalence : 0.4578
##       Balanced Accuracy : 0.7877
##
##        'Positive' Class : 1
##
```

Accuracy: 0.7891
Sensitivity: 0.7591
Specificity: 0.8163

Step 3. Fit test data

```
# fill in na in year
df.small$year[is.na(df.small$year)] <- median(df.small$year, na.rm=TRUE)

#categorical variable: fuel
mapping <- c("gas" = 0, "diesel" = 1, "hybrid" = 2, "electric" = 3, "other" = 4)
df.small$fuel <- mapping[df.small$fuel]
df.small$fuel[is.na(df.small$fuel)] <- 4

#categorical variable: cylinders
df.small$cylinders[is.na(df.small$cylinders)] <- 9

#categorical variable: drive
mapping <- c("fwd" = 0, "rwd" = 1, "4wd" = 2)
df.small$drive <- mapping[df.small$drive]
df.small$drive[is.na(df.small$drive)] <- 3

#categorical variable: transmission
mapping <- c("automatic" = 0, "manual" = 1, "other" = 2)
df.small$transmission <- mapping[df.small$transmission]
df.small$transmission[is.na(df.small$transmission)] <- 2
```

```r
#categorical variable: condition
mapping <- c("new" = 0, "excellent" = 1, "good" = 2, "like new" = 3, "fair" = 4, "salvage" = 5)
df.small$condition <- mapping[df.small$condition]
df.small$condition[is.na(df.small$condition)] <- 6

#categorical variable: odometer
df.small$odometer <- cut(df.small$odometer, breaks=c(0, 50000, 100000, 150000, 200000, 250000,
                         300000, 350000, 400000, 500000), labels=c(0, 1, 2, 3, 4, 5, 6, 7, 8))
mapping <- c("0" = 0, "1" = 1, "2" = 2, "3" = 3, "4" = 4, "5" = 5, "6" = 6, "7" = 7, "8" = 8)
df.small$odometer <- mapping[df.small$odometer]
df.small$odometer[is.na(df.small$odometer)] <- 9

#categorical variable: manufacturer
df.small$manufacturer <- as.factor(df.small$manufacturer)
n <- length(unique(df.small$manufacturer))
mapping <- c(0:n)
df.small$manufacturer <- mapping[df.small$manufacturer]

# set cars with price above 12,000 to be 1, and below to be 0.
df.small$price.category <- cut(df.small$price, breaks=c(-Inf, 12000, Inf), labels=c(1, 0))

# prediction
test_pred <- predict(fit, newdata = df.small, type = "response")
test_pred <- cut(test_pred, breaks=c(-Inf, 0.5, Inf), labels = c(1,0))

library(caret)
comparison <- table(test_pred, df.small$price.category)
confusionMatrix(comparison)
```

```
## Confusion Matrix and Statistics
##
##
## test_pred     1     0
##         1 10898  2812
##         0  3467 12822
##
##                Accuracy : 0.7907
##                  95% CI : (0.786, 0.7953)
##     No Information Rate : 0.5212
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5799
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7586
##             Specificity : 0.8201
##          Pos Pred Value : 0.7949
##          Neg Pred Value : 0.7872
##              Prevalence : 0.4788
##          Detection Rate : 0.3633
##    Detection Prevalence : 0.4570
##       Balanced Accuracy : 0.7894
```

```
##
##        'Positive' Class : 1
##
```

Accuracy: 0.7907
Sensitivity : 0.7586
Specificity : 0.8201

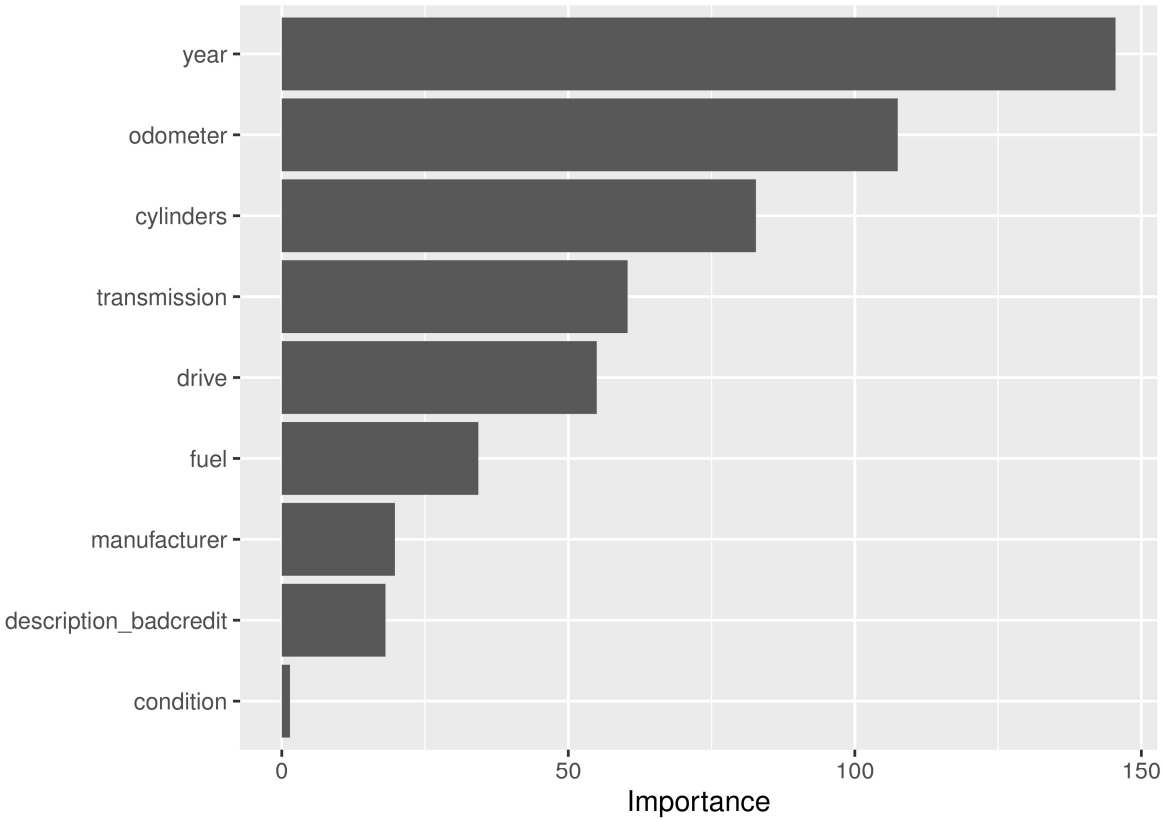
Q2. Draw a figure that demonstrates which features are most important in constructing more accurate predictions.

```
library(vip)
```

```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

```
vip(fit)
```



Q3. Construct your own simulated data set that enables you to replicate the general pattern shown in ISLR Figure 2.17 for the KNN algorithm.

```r
library(FNN)
library(class)
```

```
##
## Attaching package: 'class'
```

```
## The following objects are masked from 'package:FNN':
##
##     knn, knn.cv
```

```r
# randomly select 29999 entries from train data to keep it to the same size as test data
train_data <- data.frame(df.train$price.category, df.train$year, df.train$odometer,
          df.train$cylinders, df.train$transmission, df.train$drive, df.train$manufacturer,
          df.train$condition, df.train$fuel, df.train$description_badcredit)
set.seed(1)
train_data <- sample_n(train_data, 29999)
train.y <- train_data[,1]
train.x <- train_data[,-1]

# create a data frame for test data
test_data <- data.frame(df.small$price.category, df.small$year, df.small$odometer,
          df.small$cylinders, df.small$transmission, df.small$drive,  df.small$manufacturer,
          df.small$condition, df.small$fuel,  df.small$description_badcredit)
test.y <- test_data[,1]
test.x <- test_data[,-1]

# get train data error
set.seed(1)
k <- 1:100
train.error <- c()
for (i in 1:100){
    knn.mod <-  knn(train = train.x, test = train.x, cl = train.y, k = i)
    train.error[i] <- mean(train.y != knn.mod)
}

# get test data error
set.seed(1)
test.error <- c()
for (i in 1:100){
    knn.mod <-  knn(train = train.x, test = test.x, cl = train.y, k = i)
    test.error[i] <- mean(test.y != knn.mod)
}
```

```r
# knn plot
(plot(1/k, train.error, col = 'blue', xlab = '1/k', ylab = 'error rate', ylim =c(0, 0.2))
 + lines(1/k, train.error, col = 'blue') + points(1/k, test.error, col = 'orange')
 + lines(1/k, test.error, col = 'orange') + abline(h = min(test.error), lty = 2))
```

```
## integer(0)
```

```
#add legends
legend(0.3,0.2, legend ='test data error', cex = 0.8, box.lty=0)
legend(0.3,0.05, legend ='train data error', cex = 0.8, box.lty=0)
```