Jaden Obi

Please use R for all the following problems except when exclusively it is stated to use other computational tools. Whenever it is necessary, set the seed at 1. Also, to perform any statistical tests use $\alpha = .05$.

1. Predicting Philadelphia Housing Prices. The file "PhilaHousing.csv" contains information collected by the US Bureau of the Census concerning housing in the area of Phila, PA. The dataset includes information on 506 census housing tracts in the Philadelphia area. The goal is to predict the median house price in new tracts based on information such as crime rate, pollution, and number of rooms. The dataset contains 11 numerical variables, and the response is the median house price (MEDV). Description of each variable is provided below:

Description of Variables for Philadelphia Housing.

CRIM	Per capita crime rate by town
ZN	Proportion of residential land zoned for lots over 25,000 ft2
INDUS	Proportion of nonretail business acres per town
NOX	Nitric oxide concentration (parts per 10 million)
RM	Average number of rooms per dwelling
AGE	Proportion of owner-occupied units built prior to 1940
DIS	Weighted distances to five Phila employment centers
RAD	Ind3ex of accessibility to radial highways
TAX	Full-value property-tax rate per $10,000
PTRATIO	Pupil/teacher ratio by town
LSTAT	Percentage lower status of the population
MEDV	Median value of owner-occupied homes in $1000s

```
PhilaHousing.df <- read.csv("PhilaHousing.csv")
```

(a) Partition your data set and allocate 25% for the training and the rest for the validation.

```
train.rows <- sample(rownames(PhilaHousing.df), dim(PhilaHousing.df)[1]*0.25)
train.data <- PhilaHousing.df[train.rows, ]
valid.rows <- sample(setdiff(rownames(PhilaHousing.df), train.rows), dim(PhilaHousing.df)[1]*0.75)
valid.data <- PhilaHousing.df[valid.rows, ]
```

(b) Why should the data be partitioned into training and validation sets? What will the training set be used for? What will the validation set be used for?

The main purpose of splitting the dataset into a training set and a validation set is to prevent our model from overfitting, which means that the model becomes really good at classifying the samples in the training set but cannot generalize and make accurate classifications on the data it has not seen before.

The training set will be used to find the best model that fits as accurately as possible. The validation set helps to analyze the ability of the model to fit with new and unseen observations, and it also tests the performance of the model used with the training set.

(c) Create a correlation matrix. Based on your correlation matrix, create a simple regression model by using the measurement (variable) that has the second highest correlation with our response variable (MEDV = Median Value of Owner-Occupied Home in $1000).

```
View(train.data)
names(train.data)
head(train.data[c("crim", "zn", "indus", "nox", "rm", "age", "dis", "rad", "tax", "ptratio", "lstat", "medv")], 4)
X <- train.data[c("crim", "zn", "indus", "nox", "rm", "age", "dis", "rad", "tax", "ptratio", "lstat", "medv")]
View(X)
CM1 <- cor(X)
View(CM1)
```

	crim	zn	indus	nox	rm	age	dis	rad	tax	ptratio	lstat	medv
crim	1.0000000	-0.1912655	0.3312756	0.4319242	-0.1961756	0.3412968	-0.3356458	0.5688486	0.5035718	0.2892102	0.4743604	-0.4409797
zn	-0.1912655	1.0000000	-0.5571770	-0.5593149	0.3559119	-0.6365205	0.6802573	-0.3242151	-0.3477077	-0.4994062	-0.5135818	0.4908354
indus	0.3312756	-0.5571770	1.0000000	0.7722066	-0.3874410	0.7112127	-0.7267770	0.5334209	0.6571394	0.5317752	0.6956437	-0.5744410
nox	0.4319242	-0.5593149	0.7722066	1.0000000	-0.2959781	0.7655409	-0.7748638	0.6872043	0.7166242	0.4244889	0.7108501	-0.5305143
rm	-0.1961756	0.3559119	-0.3874410	-0.2959781	1.0000000	-0.1999978	0.1719268	-0.1444068	-0.2344279	-0.3436907	-0.5891185	0.7757949
age	0.3412968	-0.6365205	0.7112127	0.7655409	-0.1999978	1.0000000	-0.7523892	0.4937198	0.5193221	0.4087738	0.6820879	-0.4461553
dis	-0.3356458	0.6802573	-0.7267770	-0.7748638	0.1719268	-0.7523892	1.0000000	-0.4796743	-0.5129631	-0.3408612	-0.5813597	0.3287877
rad	0.5688486	-0.3242151	0.5334209	0.6872043	-0.1444068	0.4937198	-0.4796743	1.0000000	0.8788352	0.5003532	0.5214232	-0.4572965
tax	0.5035718	-0.3477077	0.6571394	0.7166242	-0.2344279	0.5193221	-0.5129631	0.8788352	1.0000000	0.5313902	0.5386393	-0.5481144
ptratio	0.2892102	-0.4994062	0.5317752	0.4244889	-0.3436907	0.4087738	-0.3408612	0.5003532	0.5313902	1.0000000	0.4580451	-0.5803848
lstat	0.4743604	-0.5135818	0.6956437	0.7108501	-0.5891185	0.6820879	-0.5813597	0.5214232	0.5386393	0.4580451	1.0000000	-0.7398777
medv	-0.4409797	0.4908354	-0.5744410	-0.5305143	0.7757949	-0.4461553	0.3287877	-0.4572965	-0.5481144	-0.5803848	-0.7398777	1.0000000

```
Simple_Reg <- lm(medv~rm, data=train.data)
summary(Simple_Reg)
```

(d) Analyze your model and evaluate your model fully based on the three criteria we discussed in class regarding the simple linear regression model.

```
Pred_Y_SR <-fitted(Simple_Reg)
Residual_SR <-rstandard(Simple_Reg)
plot(Pred_Y_SR, Residual_SR)
qqnorm(Residual_SR,  ylab="Standardized Residuals",  xlab="Normal Scores", main="Normal Probability Plot")
qqline(Residual_SR)
```

Multiple Regression Analysis:

2. Use the previous data set ("PhilaHousing.csv") to perform/answer the following parts:

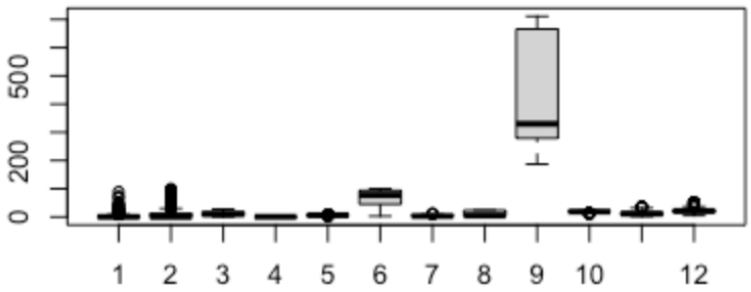
(a) Partition your data set and allocate 10% for the training and the rest for the validation.

```
train.rows <- sample(rownames(PhilaHousing.df), dim(PhilaHousing.df)[1]*0.10)
train.data <- PhilaHousing.df[train.rows, ]
valid.rows <- sample(setdiff(rownames(PhilaHousing.df), train.rows), dim(PhilaHousing.df)[1]*0.90)
valid.data <- PhilaHousing.df[valid.rows, ]
```

(b) Create Box and Plot Displays for all of your numerical variables.

```
names(PhilaHousing.df)
```

```
View(PhilaHousing.df)
attach(PhilaHousing.df)
par(mfrow=c(2, 2))
for (var in names(PhilaHousing.df)) {
  if (is.numeric(PhilaHousing.df[[var]])) {
    boxplot(PhilaHousing.df[[var]], main=paste("Boxplot of", var))
  }
}
boxplot(crim, zn, indus, nox, rm, age, dis, rad, tax, ptratio, lstat, medv)
```



(c) Based on the correlation matrix you computed in Problem 1, Part (c) fit a multiple linear regression model to the median house price (MEDV) as a function of three highest correlated variables to predict the median house price from the predictors in the model.

```
Multiple_Reg <- lm(medv~lstat+rm+ptratio, data=train.data)
summary(Multiple_Reg)
```

(d) Analyze your model and evaluate your model fully based on the four criteria we discussed in class regarding the multiple linear regression model.

```
Pred_Y_MR <- fitted(Multiple_Reg)
SR_MR <- rstandard(Multiple_Reg)
plot(Pred_Y_MR, SR_MR)
qqnorm(SR_MR, ylab="Standardized Residuals", xlab="Normal Scores", main="Normal Probability Plot Multiple Regression")
qqline(SR_MR)
vif(Multiple_Reg)
```

3. Data were collected on the nutritional information and consumer rating of 77 breakfast cereals (data is saved in "Cereals.csv"). The consumer rating is a rating of cereal "healthiness" for consumer information (not a rating by consumers). For each cereal, the data include 13 numerical variables, and we are interested in reducing this dimension. For each cereal, the information is based on a bowl of cereal rather than a serving size, because most people simply fill a cereal bowl (resulting in constant volume, but not weight). The description of the different variables is given below:

Variable	Description
mfr	Manufacturer of cereal (American Home Food Products, General Mills, Kellogg, etc.)

type	Cold or hot
calories	Calories per serving
protein	Grams of protein
fat	Grams of fat
sodium	Milligrams of sodium
fiber	Grams of dietary fiber
carbo	Grams of complex carbohydrates
sugars	Grams of sugars
potass	Milligrams of potassium
vitamins	Vitamins and minerals: 0, 25, or 100, indicating the typical percentage of FDA recommended
shelf	Display shelf (1, 2, or 3, counting from the floor)
weight	Weight in ounces of one serving
cups	Number of cups in one serving
rating	Rating of the cereal calculated by consumer reports

```
Cereals.df <- read.csv("Cereals.csv")
```

(a) Partition your data set and allocate 70% for the training and the rest for the validation.

```
train.rows <- sample(rownames(Cereals.df), dim(Cereals.df)[1]*0.70)

train.data <- Cereals.df[train.rows, ]

valid.rows <- sample(setdiff(rownames(Cereals.df), train.rows), dim(Cereals.df)[1]*0.30)

valid.data <- Cereals.df[valid.rows, ]
```

(b) Use the training data set to construct a correlation table (using only the numerical variables). Identify the 4 highest correlated variables. Which pair has the lowest correlation?

```
View(train.data)

names(train.data)

head(train.data[c("calories", "protein", "fat", "sodium", "fiber", "carbo", "sugars", "potass", "vitamins", "shelf", "weight",
"cups", "rating")], 4)

X <- train.data[c("calories", "protein", "fat", "sodium", "fiber", "carbo", "sugars", "potass", "vitamins", "shelf", "weight", "cups",
"rating")]

View(X)

CM1 <- cor(X)

View(CM1)
```

	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rating
calories	1.00000000	-0.04732713	0.501394590	0.263193609	-0.38739585	0.33303326	0.59400023	-0.146037485	0.35375420	0.18367517	0.6673077	0.18985921	-0.71679873
protein	-0.04732713	1.00000000	0.119205904	-0.107327279	0.46234799	-0.03881449	-0.29920223	0.518414542	-0.04401105	0.05146676	0.1447294	-0.21889473	0.50582583
fat	0.50139459	0.11920590	1.000000000	-0.007862598	-0.08790621	-0.23776024	0.34697020	0.121518013	-0.03088089	0.25659688	0.1886282	-0.02349005	-0.46879761
sodium	0.26319361	-0.10732728	-0.007862598	1.000000000	-0.05310298	0.25676914	0.07919977	-0.008523566	0.34941216	0.06154715	0.3130665	0.06216572	-0.37105982
fiber	-0.38739585	0.46234799	-0.087906213	-0.053102982	1.00000000	-0.43366394	-0.13983975	0.907606592	-0.04951604	0.28449082	0.1935083	-0.56969371	0.61692924
carbo	0.33303326	-0.03881449	-0.237760239	0.256769140	-0.43366394	1.00000000	-0.41193901	-0.409187507	0.21531288	-0.27647236	0.1594985	0.37177335	0.01882983
sugars	0.59400023	-0.29920223	0.346970200	0.079199771	-0.13983975	-0.41193901	1.00000000	0.020326057	0.22101036	0.26770306	0.4861724	-0.02479749	-0.76467382
potass	-0.14603749	0.51841454	0.121518013	-0.008523566	0.90760659	-0.40918751	0.02032606	1.000000000	0.02852707	0.37813390	0.3834378	-0.54851526	0.41516255
vitamins	0.35375420	-0.04401105	-0.030880892	0.349412164	-0.04951604	0.21531288	0.22101036	0.028527070	1.00000000	0.30589313	0.4307647	0.08068536	-0.28802721
shelf	0.18367517	0.05146676	0.256596881	0.061547153	0.28449082	-0.27647236	0.26770306	0.378133901	0.30589313	1.00000000	0.2870258	-0.34358734	-0.12328536
weight	0.66730765	0.14472943	0.188628209	0.313066498	0.19350834	0.15949847	0.48617237	0.383437805	0.43076469	0.28702578	1.0000000	-0.20513509	-0.31655888
cups	0.18985921	-0.21889473	-0.023490051	0.062165724	-0.56969371	0.37177335	-0.02479749	-0.548515260	0.08068536	-0.34358734	-0.2051351	1.00000000	-0.26236724
rating	-0.71679873	0.50582583	-0.468797613	-0.371059823	0.61692924	0.01882983	-0.76467382	0.415162553	-0.28802721	-0.12328536	-0.3165589	-0.26236724	1.00000000

(c) Conduct a principal components analysis on the training data (using the numerical variables).

```
numerical_vars <- train.data[sapply(train.data, is.numeric)]
```

```
numerical_vars <- na.omit(numerical_vars)

pca_result <- prcomp(numerical_vars, scale. = TRUE)

summary(pca_result)
```

```
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10    PC11    PC12    PC13    PC14
Standard deviation     1.9589 1.7989 1.3814 1.11856 0.97990 0.87836 0.79654 0.70722 0.62085 0.57854 0.32001 0.23122 0.14856 0.05287
Proportion of Variance 0.2741 0.2311 0.1363 0.08937 0.06859 0.05511 0.04532 0.03573 0.02753 0.02391 0.00731 0.00382 0.00158 0.00020
Cumulative Proportion  0.2741 0.5052 0.6415 0.73091 0.79950 0.85460 0.89992 0.93565 0.96318 0.98709 0.99441 0.99822 0.99980 1.00000
```

(d) Based on your output, how many principal component(s) should be used? List and discuss your suggested principal component(s). What proportion of the original information can be represented by your suggested principal component(s)?

(e) Write your principal component scores as a CSV file. Submit it as part of your MS Word file.

```
principal_components <- pca_result$x

write.csv(principal_components, "principal_components.csv", row.names = FALSE)
```

4. The file "Banks.csv" includes data on a sample of 20 banks. The "Financial Condition" column records the judgment of an expert on the financial condition of each bank. This outcome variable takes one of two possible values—weak or strong—according to the financial condition of the bank. The predictors are two ratios used in the financial analysis of banks: TotLns&Lses/Assets is the ratio of total loans and leases to total assets and TotExp/Assets is the ratio of total expenses to total assets. The target is to use the two ratios for classifying the financial condition of a new bank.

   Run a logistic regression model (on the entire dataset) that models the status of a bank as a function of the two financial measures provided. Specify the success class as weak (this is similar to creating a dummy that is 1 for financially weak banks and 0 otherwise) and use the default cutoff value of 0.5.

(a) Write the estimated equation that associates the financial condition of a bank with its two predictors in three formats:

The logit as a function of the predictors

The odds as a function of the predictors

The probability as a function of the predictors

```
Banks.df <- read.csv("Banks.csv")

names(Banks.df)

summary(Banks.df)

View(Banks.df)

table(Banks.df$Financial.Condition)

Banks.df$Financial.Condition <- ifelse(Banks.df$Financial.Condition == "weak", 1, 0)

logistic_model <- glm(Financial.Condition ~ TotLns.Lses.Assets + TotExp.Assets, data = Banks.df, family = binomial)

summary(logistic_model)

predicted_probabilities <- predict(logistic_model, type = "response")
```

```
predicted_class <- ifelse(predicted_probabilities >= 0.5, 1, 0)
```

(b) Interpret the estimated coefficients (coefficients of TotLns&Lses/Assets and TotExp/Assets) in the context of this problem.

```
Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)       -2.557e+01  2.530e+05       0        1
TotLns.Lses.Assets  1.256e-14  3.209e+05       0        1
TotExp.Assets      -8.988e-14  1.968e+06       0        1
```

(c) Evaluate the performance of your Logistic Regression by using the **Confusion Matrix**.

```
logistic_model <- glm(Financial.Condition ~ TotLns.Lses.Assets + TotExp.Assets, data = Banks.df, family = binomial)

predicted_probabilities <- predict(logistic_model, newdata = Banks.df, type = "response")

Banks.df$pred_class <- ifelse(predicted_probabilities >= 0.5, 1, 0)

Banks.df$pred_class <- factor(Banks.df$pred_class, levels = c("0", "1"))

Banks.df$Financial.Condition <- factor(Banks.df$Financial.Condition, levels = c("0", "1"))

confusion_matrix <- confusionMatrix(Banks.df$Financial.Condition, Banks.df$pred_class)

print(confusion_matrix)
```

5. A management consultant is studying the roles played by experience and training in a system administrator's ability to complete a set of tasks in a specified amount of time. In particular, she is interested in discriminating between administrators who are able to complete given tasks within a specified time and those who are not. Data are collected on the performance of 75 randomly selected administrators. They are stored in the file "SystemAdministrators.csv".

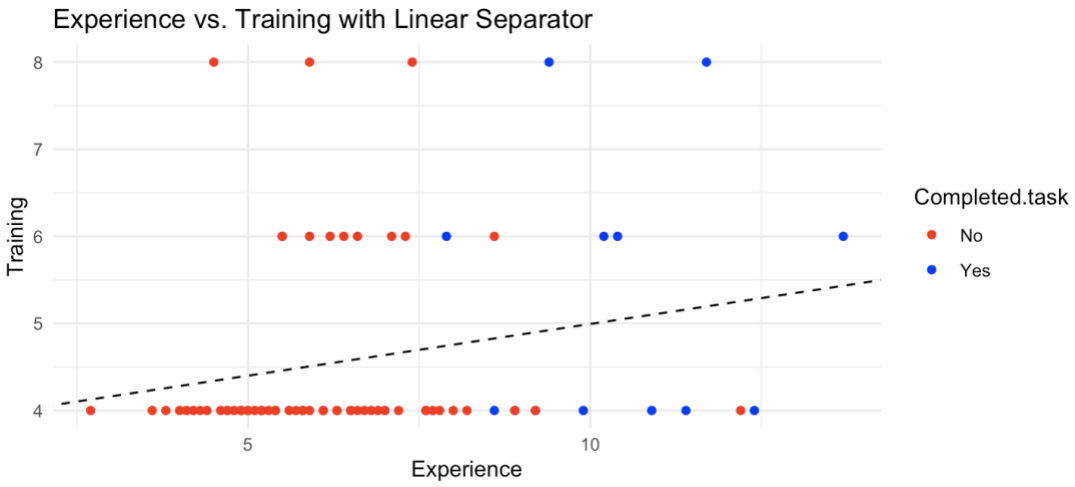
   Using these data, the consultant performs a discriminant analysis. The variable Experience measures months of full-time system administrator experience, while Training measures number of relevant training credits. The dependent variable Completed is either Yes or No, according to whether or not the administrator completed the tasks.

```
SystemAdminstrators.df <- read.csv("SystemAdministrators.csv")

names(SystemAdminstrators.df)

summary(SystemAdminstrators.df)

View(SystemAdminstrators.df)
```

(a) Create a scatter plot of Experience vs. Training using color or symbol to differentiate administrators who completed the tasks from those who did not complete them. See if you can identify a line that separates the two classes with minimum misclassification.

```
install.packages("tidyverse")

library(tidyverse)

library(ggplot2)

library(tidyverse)

library(caret)
```

```
ggplot(SystemAdminstrators.df, aes(x = Experience, y = Training, color = Completed.task)) +
  geom_point() +
  labs(title = "Experience vs. Training", x = "Experience", y = "Training") +
  scale_color_manual(values = c("Yes" = "blue", "No" = "red")) +
  theme_minimal()
fit <- lm(Training ~ Experience, data = SystemAdminstrators.df)
summary(fit)
ggplot(SystemAdminstrators.df, aes(x = Experience, y = Training, color = Completed.task)) +
  geom_point() +
  geom_abline(slope = coef(fit)[2], intercept = coef(fit)[1], color = "black", linetype = "dashed") +
  labs(title = "Experience vs. Training with Linear Separator", x = "Experience", y = "Training") +
  scale_color_manual(values = c("Yes" = "blue", "No" = "red")) +
  theme_minimal()
```



(b) Use R to run a discriminant analysis with both predictors using the entire dataset as training data. Among those who completed the tasks, what is the percentage of administrators who are classified incorrectly as failing to complete the tasks?

```
SystemAdminstrators.df$Completed.task <- ifelse(SystemAdminstrators.df$Completed.task == 'Yes', 1, 0)
glm.fit <- glm(Completed.task ~ Experience + Training, data = SystemAdminstrators.df, family = binomial())
da.reg <- lda(SystemAdminstrators.df[,1:2], SystemAdminstrators.df$Completed.task)
da.reg
predicted <- predict(da.reg, SystemAdminstrators.df[,1:2])
predicted$class
predicted$posterior
SystemAdminstrators.df$Completed.task <-as.factor(SystemAdminstrators.df$Completed.task)
predicted$class <-as.factor(predicted$class)
confusionMatrix(predicted$class, SystemAdminstrators.df$Completed.task)
```

(6) The file EastWestAirlinesCluster.csv contains information on 3999 passengers who belong to an airline's frequent flier program. For each passenger, the data include information on their mileage history and on different ways they accrued or spent miles in the last year. The goal is to try to identify clusters of passengers that have similar characteristics for the purpose of targeting different segments for different types of mileage offers. Description of each variable is given below:

Field Name	Data Type	Max Data Length	Raw Data or Telcom Created Field?	Description
ID#	NUMBER		Telcom	Unique ID
Balance	NUMBER	8	Raw	Number of miles eligible for award travel
Qual_miles	NUMBER	8	Raw	Number of miles counted as qualifying for Topflight status
cc1_miles	CHAR	1	Raw	Number of miles earned with freq. flyer credit card in the past 12 months:
cc2_miles	CHAR	1	Raw	Number of miles earned with Rewards credit card in the past 12 months:
cc3_miles	CHAR	1	Raw	Number of miles earned with Small Business credit card in the past 12 months:
note: miles bins:				1 = under 5,000
				2 = 5,000 - 10,000
				3 = 10,001 - 25,000
				4 = 25,001 - 50,000
				5 = over 50,000
Bonus_miles	NUMBER		Raw	Number of miles earned from non-flight bonus transactions in the past 12 months
Bonus_trans	NUMBER		Raw	Number of non-flight bonus transactions in the past 12 months
Flight_miles_12mo	NUMBER		Raw	Number of flight miles in the past 12 months
Flight_trans_12	NUMBER		Raw	Number of flight transactions in the past 12 months
Days_since_enroll	NUMBER		Telcom	Number of days since Enroll_date
Award?	NUMBER		Telcom	Dummy variable for Last_award (1=not null, 0=null)

(a) Apply hierarchical clustering with Euclidean distance and Single Linkage's method. Make sure to normalize the data first. How many clusters appear?

```
EastWestAirlinesCluster.df <- read.csv("EastWestAirlinesCluster.csv")

EastWestAirlinesCluster.df <- EastWestAirlinesCluster.df[,-1]

EastWestAirlinesCluster.norm <- sapply(EastWestAirlinesCluster.df, scale)

d.norm <- dist(EastWestAirlinesCluster.norm[, c(1-12)], method = "euclidean")

hc1 <- hclust(d.norm, method = "single")

plot(hc1, hang = -1, label = EastWestAirlinesCluster.df$ID., cex = 0.5, main="Cluster Dendrogram", col = "blue")

d.norm_total <- dist(EastWestAirlinesCluster.norm[,c(1-12)], method = "euclidean")

d.norm_total
```

(b) What would happen if the data were not normalized?

Normalizing data is an essential step. If it isn't, there could be misleading principal components. In addition, features with large variances might dominate the principal components.

7. Predicting Housing Median Prices. The file PhilaHousing1.csv contains information on 506 census tracts in Philadelphia, where for each tract multiple variables are recorded. The last column (CAT.MEDV) was derived from MEDV, such that it obtains the value 1 if MEDV > 30 and 0 otherwise. Consider the goal of predicting the median value (MEDV) of a tract, given the information in the first 11 columns.

```
summary(PhilaHousing1.df)
```

(a) Partition the data into training (40%) and validation (60%) sets.

```
PhilaHousing1.df <- read.csv("PhilaHousing1.csv")
```

```
train.rows <- sample(rownames(PhilaHousing1.df), dim(PhilaHousing1.df)[1]*0.40)
train.data <- PhilaHousing1.df[train.rows, ]
valid.rows <- sample(setdiff(rownames(PhilaHousing1.df), train.rows), dim(PhilaHousing1.df)[1]*0.60)
valid.data <- PhilaHousing1.df[valid.rows, ]
```