Homework 9

Homework 09

Student: Jama Brookes

This homework is based on the classification and regression lectures.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Question 1

In the table below, fill in the definition column with a short (no more than two sentence) definition for each vocab word. If it can be summarized by a formula, give the formula.

Vocab Word	Definition
One-hot coding	Converts categories in a column into separate binary/dichotomous columns.
Feature selection*	Selects only the most important variables for analysis.
Classifier	A model that assigns inputs to categories instead of numeric values.
Precision	= true positives / (true positives + false positives)
Recall	= true positives / (true positives + false negatives)
F1 Score	= 2x((precision*recall)/(precision+recall))
Parsimonious model	Simplest form of a model with the fewest parameters.
Ridge regression	Helps to not overfit a regression model by adding a penalty term to the loss function that is proportional to the square of the magnitude of the coefficients.

Vocab Word	Definition
LASSO regression	Helps to not overfit a regression model by adding a penalty term to the loss function based on absolute values of the coefficients.
Cross validation	Validates the model by separating data into a training set and validation set repeatedly to model performance and generalizability. This can also be done by using k-folds, which splits data into K folds used to test the set and uses data not in the folds to train.
Tree based methods	Splits features repeatedly through various decision trees (ie algorithms that make decisions about similarity in our data) to model data.

*Just give the general idea.

Question 2

a) What shape does a perfect classifier look like on an ROC curve? What about a bad classifier? A perfect classifier will look like a rectangle (ie a line with a corner hugging the top left corner) and a bad classifier looks like a straight line or y=x. #### b) Think about the formula for an F1 score. What does it mean when the F1 score is close to 1? Close to 0? If the F1 score is close to 1, it indicates a very good performing model because the model perfectly predicts true positives with no false positives nor false negatives. If the F1 score is close to 0, the model is poorly performing because there are either a lot of false positives or false negatives making precision or recall poor.

Question 3

Compare the following aspects of linear vs. logistic regression.

	Linear	Logistic
Chart Shape	linear	S-shaped
Dependent Variable Type	Continuous	Categorical
Purpose (regression or classification)	Regression	Classification
Range of output variable (y_i or p_i)	y_i	p_i
Method*	OLS	MLE
Example of use	Predict weight gain from calories consumed	Predict probability of heart disease

*Meaning ordinary least squares or maximum likelihood estimation

Question 4

Why is it important to train then test our model? How do we do that? (2-3 sentences. Not looking for code, just general explanation). If we do not train and then test our model, our model may only be valid for the data we have and not valid for other new data and we may overfit. We can first 1) split data for training and test, 2) fit model to the training set and 3) predict on the test set.

Question 5

This question runs through a linear regression example. We want to predict median house value based on the other variables. #### a) First, load the `housing.csv` data set. Look at the data in some useful way. Why is linear regression appropriate here?

```
housing_df <- read.csv("./hw9_data/housing.csv")

str(housing_df)
```

```
## 'data.frame':    20640 obs. of  8 variables:
##  $ longitude         : num  -122 -122 -122 -122 -122 ...
##  $ latitude          : num  37.9 37.9 37.9 37.9 37.9 ...
##  $ housing_median_age: int  41 21 52 52 52 52 52 52 42 52 ...
##  $ total_rooms       : int  880 7099 1467 1274 1627 919 2535 3104 2555 3549 ...
##  $ population        : int  322 2401 496 558 565 413 1094 1157 1206 1551 ...
##  $ households        : int  126 1138 177 219 259 193 514 647 595 714 ...
##  $ median_income     : num  8.33 8.3 7.26 5.64 3.85 ...
##  $ median_house_value: int  452600 358500 352100 341300 342200 269700 299200 241400 226700 261100 ..
```

```
head(housing_df)
```

```
##   longitude latitude housing_median_age total_rooms population households
## 1   -122.23    37.88                41         880        322        126
## 2   -122.22    37.86                21        7099       2401       1138
## 3   -122.24    37.85                52        1467        496        177
## 4   -122.25    37.85                52        1274        558        219
## 5   -122.25    37.85                52        1627        565        259
## 6   -122.25    37.85                52         919        413        193
##   median_income median_house_value
## 1        8.3252             452600
## 2        8.3014             358500
## 3        7.2574             352100
## 4        5.6431             341300
## 5        3.8462             342200
## 6        4.0368             269700
```

QUESTION 5a ANSWER: Linear regression is appropriate because our dependent variable (median_house_value) is continuous and the other predictor variables are also continuous or numerical.

```
# Scale data.
scale <- function(a){
  (a - min(a))/(max(a)-min(a))
}

d_num <- housing_df %>% select(where(is.numeric)) %>%
  mutate(across(where(is.numeric), scale))

head(d_num)
```

b) Scale data and split it 75/25 training/testing. Set seed = 123.

```
##   longitude  latitude housing_median_age total_rooms  population households
## 1 0.2111554 0.5674814          0.7843137  0.02233074 0.008940834 0.02055583
## 2 0.2121514 0.5653560          0.3921569  0.18050257 0.067210404 0.18697583
## 3 0.2101594 0.5642933          1.0000000  0.03726029 0.013817652 0.02894261
## 4 0.2091633 0.5642933          1.0000000  0.03235159 0.015555369 0.03584937
## 5 0.2091633 0.5642933          1.0000000  0.04132967 0.015751563 0.04242723
## 6 0.2091633 0.5642933          1.0000000  0.02332265 0.011491353 0.03157375
##   median_income median_house_value
## 1     0.5396684          0.9022664
## 2     0.5380271          0.7082466
## 3     0.4660281          0.6950507
## 4     0.3546986          0.6727828
## 5     0.2307761          0.6746385
## 6     0.2439208          0.5251545
```

```
# Set Seed
set.seed(123)

# Split data - 75% training and 25% testing.
train <- runif(nrow(d_num)) < 0.75
test <- !train
```

```
f = median_house_value ~  housing_median_age + total_rooms + population + households + median_income + l

m <- lm(f, data=d_num %>% filter(train))
summary(m)
```

c) Fit the model.

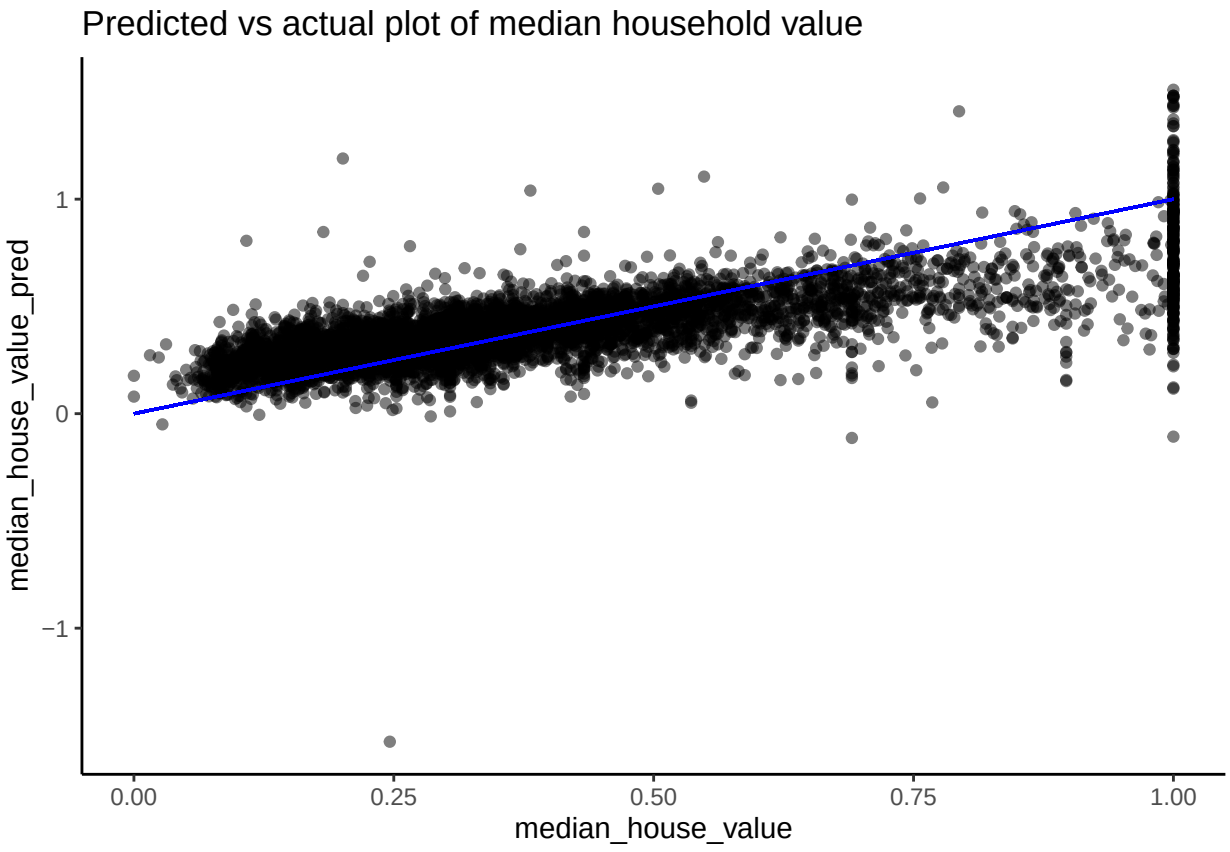
```
##
## Call:
## lm(formula = f, data = d_num %>% filter(train))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.27209 -0.10001 -0.02338  0.07256  0.95932
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -0.058345   0.004810  -12.13   <2e-16 ***
## housing_median_age  0.196973   0.005444   36.18   <2e-16 ***
## total_rooms        -1.078389   0.067938  -15.87   <2e-16 ***
## population         -3.209320   0.099893  -32.13   <2e-16 ***
## households          2.781972   0.066686   41.72   <2e-16 ***
## median_income       1.372480   0.010900  125.92   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1555 on 15558 degrees of freedom
```

```
## Multiple R-squared:  0.5711, Adjusted R-squared:  0.571
## F-statistic:  4144 on 5 and 15558 DF,  p-value: < 2.2e-16
```

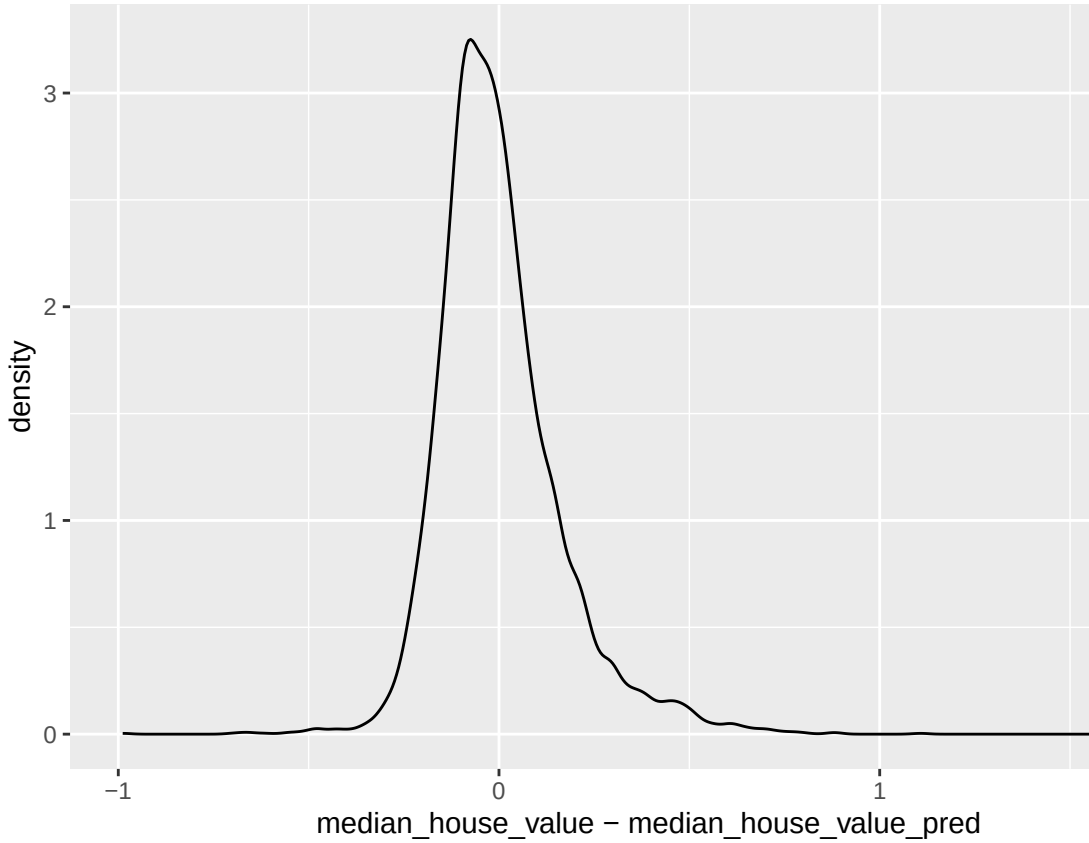
```r
# Prediction on test data.
dx <- d_num %>% filter(test)

dx <- dx %>%
  mutate(median_house_value_pred = predict(m, dx %>% as.data.frame()))

# Predicted vs actual plot.
ggplot(dx, aes(median_house_value, median_house_value_pred)) +
  geom_point(alpha = 0.5) +
  geom_segment(aes(x=0,y=0,xend=1,yend=1), color = "blue") +
  ggtitle("Predicted vs actual plot of median household value") +
  theme_classic()
```

d) Make predictions on test data and show them in an actual vs. predicted plot.

```
## Warning in geom_segment(aes(x = 0, y = 0, xend = 1, yend = 1), color = "blue"): All aesthetics have l
## i Please consider using `annotate()` or provide this layer with data containing
##   a single row.
```



```
# Residual density plot.
ggplot(dx, aes(median_house_value-median_house_value_pred)) +
  geom_density()
```



e) Make a residuals plot.

Question 6

This question runs through a logistic regression example. We want to predict diabetes diagnosis based on the other variables. #### a) First, load the `diabetes.csv` data set. Look at the data in some useful way. Why is logistic regression appropriate here?

```
diabetes_df <- read.csv("./hw9_data/diabetes.csv")

str(diabetes_df)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
##  $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
##  $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
##  $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
##  $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
##  $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
```

```
##  $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
##  $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

QUESTION 6a ANSWER: Logistic regression is appropiate here because the dependent variable (diabetes diagnosis; "Outcome") is a categorica/dichotomous variable (0/1).

```
# Scale
scale <- function(x){
  (x - min(x)) / (max(x) - min(x))
}

d <- diabetes_df %>% as_tibble() %>% filter(complete.cases(.)) %>%
  filter(complete.cases(.)) %>%
  transmute(
    Pregnancies              = Pregnancies %>% scale(),
    Glucose                  = Glucose %>% scale(),
    BloodPressure            = BloodPressure %>% scale(),
    SkinThickness            = SkinThickness %>% scale(),
    Insulin                  = Insulin %>% scale(),
    BMI                      = BMI %>% scale(),
    DiabetesPedigreeFunction= DiabetesPedigreeFunction %>% scale(),
    Age                      = Age %>% scale(),
    Outcome                  = Outcome
  )

d
```

b) Scale data and split it 75/25 training/testing. Set seed = 123.

```
## # A tibble: 768 x 9
##    Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI
##          <dbl>   <dbl>         <dbl>         <dbl>   <dbl> <dbl>
##  1      0.353   0.744         0.590         0.354   0     0.501
##  2      0.0588  0.427         0.541         0.293   0     0.396
##  3      0.471   0.920         0.525         0       0     0.347
##  4      0.0588  0.447         0.541         0.232   0.111 0.419
##  5      0       0.688         0.328         0.354   0.199 0.642
##  6      0.294   0.583         0.607         0       0     0.382
##  7      0.176   0.392         0.410         0.323   0.104 0.462
##  8      0.588   0.578         0             0       0     0.526
##  9      0.118   0.990         0.574         0.455   0.642 0.455
## 10      0.471   0.628         0.787         0       0     0
## # i 758 more rows
## # i 3 more variables: DiabetesPedigreeFunction <dbl>, Age <dbl>, Outcome <int>
```

```
# Set seed
set.seed(123)

# Split
n <- nrow(d)
```

```
train_idx <- sample.int(n, size = floor(0.1 * n))
d_train <- d %>% slice(train_idx)
d_test  <- d %>% slice(setdiff(seq_len(n), train_idx))

d_train %>% write_csv("derived_data/diabetes_train.csv")
d_test  %>% write_csv("derived_data/diabetes_test.csv")
```

```
f <- Outcome ~ .
m <- glm(f, data = d_train, family = binomial())

summary(m)
```

c) Fit the model.

```
##
## Call:
## glm(formula = f, family = binomial(), data = d_train)
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -8.2848     2.3250  -3.563 0.000366 ***
## Pregnancies                1.8716     1.5354   1.219 0.222848
## Glucose                    5.0362     2.1896   2.300 0.021446 *
## BloodPressure              0.2239     2.3883   0.094 0.925307
## SkinThickness              1.4403     2.3091   0.624 0.532776
## Insulin                   -1.2769     2.0595  -0.620 0.535259
## BMI                        7.2613     4.2227   1.720 0.085507 .
## DiabetesPedigreeFunction   0.7208     1.9228   0.375 0.707739
## Age                        1.0054     1.8917   0.531 0.595091
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 98.898  on 75  degrees of freedom
## Residual deviance: 76.074  on 67  degrees of freedom
## AIC: 94.074
##
## Number of Fisher Scoring iterations: 5
```

```
saveRDS(m, "derived_data/diabetes_logres.rds")
```

```
d_test <- read_csv("derived_data/diabetes_test.csv", show_col_types = FALSE)
m <- readRDS("derived_data/diabetes_logres.rds")

p <- predict(m, newdata = d_test, type = "response")
pred <- as.integer(p >= 0.5)
```

```r
truth <- d_test$Outcome

tp <- sum(pred == 1 & truth == 1)
fp <- sum(pred == 1 & truth == 0)
tn <- sum(pred == 0 & truth == 0)
fn <- sum(pred == 0 & truth == 1)
acc <- (tp + tn) / (tp + fp + tn + fn)

df <- tibble(
  measure = c("True Positive", "False Positive", "True Negative", "False Negative", "Accuracy"),
  value   = c(tp, fp, tn, fn, acc)
)

df
```

d) Make predictions on test data. Print a table with the number of true positives, false positives, true negatives, false negatives, and accuracy.

```
## # A tibble: 5 x 2
##   measure          value
##   <chr>            <dbl>
## 1 True Positive  147
## 2 False Positive  72
## 3 True Negative  379
## 4 False Negative  94
## 5 Accuracy         0.760
```

```r
# Packages
suppressPackageStartupMessages(library(glmnet))
suppressPackageStartupMessages(library(broom))
suppressPackageStartupMessages(library(gridExtra))

# Set Seed
set.seed(123)

# Load data
d_train <- read_csv("derived_data/diabetes_train.csv", show_col_types = FALSE)
d_test  <- read_csv("derived_data/diabetes_test.csv", show_col_types = FALSE)

# Response matrices
y_train <- d_train$Outcome
y_test  <- d_test$Outcome

# Predictor matrices
X_train <- model.matrix(Outcome ~ ., data = d_train)[, -1]
X_test  <- model.matrix(Outcome ~ ., data = d_test)[, -1]

set.seed(123)

# Fit LASSO with cross-validation
```

```r
cvfit <- cv.glmnet(X_train, y_train, alpha = 1, family = "binomial")

# Final LASSO model at best lambda
best_fit <- glmnet(X_train, y_train, alpha = 1, family = "binomial", lambda = cvfit$lambda.min)
best_fit$beta
```

e) Fit a LASSO-regularized logistic regression model. Again, set seed = 123. Which variables are the most important (which ones don't go to zero)? How does the LASSO model affect the accuracy?

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                                s0
## Pregnancies              1.073291
## Glucose                  3.455074
## BloodPressure            .
## SkinThickness            .
## Insulin                  .
## BMI                      3.507655
## DiabetesPedigreeFunction .
## Age                      .
```

```r
# Predict with test data
lasso_pred_prob <- predict(cvfit, newx = X_test, s = "lambda.min", type = "response")
lasso_pred <- as.integer (lasso_pred_prob >= 0.5)

# Calculate metrics
tp <- sum(lasso_pred == 1 & y_test == 1)
fp <- sum(lasso_pred == 1 & y_test == 0)
tn <- sum(lasso_pred == 0 & y_test == 0)
fn <- sum(lasso_pred == 0 & y_test == 1)
acc <- (tp + tn) / (tp + fp + tn + fn)

tibble(
  measure = c("True Positive", "False Positive", "True Negative", "False Negative", "Accuracy"),
  value = c(tp, fp, tn, fn, acc))
```

```
## # A tibble: 5 x 2
##   measure          value
##   <chr>            <dbl>
## 1 True Positive  120
## 2 False Positive  31
## 3 True Negative  420
## 4 False Negative 121
## 5 Accuracy         0.780
```

ANSWER 6e:

Most important variables are Pregnancies, Glucose, and BMI according to the most simplified LASSO model. Since the LASSO simplifies the model by dropping variables that are not needed, it helps to reduce overfitting and slightly improves the accuracy on the test data (0.78 vs 0.76), which we see in our data.

```r
library(ggplot2)

# Predicted probabilities from LASSO
lasso_prob <- predict(best_fit, newx = X_test, type = "response")

# Create a tibble for plotting
plot_df <- tibble(Actual = y_test, Predicted_Prob = as.numeric(lasso_prob))

# Plot: Actual vs Predicted Probability
ggplot(plot_df, aes(x = Predicted_Prob, y = Actual)) +
  geom_jitter(height = 0.05, width = 0.1, alpha = 0.6, color = "purple") +
  geom_smooth(method = "loess", se = FALSE, color = "blue") +
  labs(
    title = "Actual vs Predicted Probability (LASSO Regression)",
    x = "Actual Outcome",
    y = "Predicted Probability") +
  theme_minimal()
```

f) Make a plot of actual vs. predicted values for the LASSO model.

```
## `geom_smooth()` using formula = 'y ~ x'
```