Homework 09 - Regression and Classification

Alexis Bryant

2025-11-07

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 categorical variables into multiple binary indicator columns (e.g., “fuel_type_gas”, “fuel_type_diesel”), allowing categorical data to be used in regression or classification models.
Feature selection	Choosing the most informative predictors to improve model accuracy and reduce overfitting.
Classifier	An algorithm that predicts categorical outcomes (e.g., diabetic vs. non-diabetic).
Precision	TP / (TP + FP); proportion of predicted positives that are actually positive.
Recall	TP / (TP + FN); proportion of true positives correctly identified.
F1 Score	2 × (Precision × Recall) / (Precision + Recall); balances precision and recall.
Parsimonious model	A simple model that explains data well without unnecessary complexity.
Ridge regression	Linear regression with an L2 penalty that shrinks coefficients toward zero (reduces variance).
LASSO regression	Linear regression with an L1 penalty that can shrink coefficients to exactly zero (performs feature selection).
Cross-validation	Method of evaluating model performance by repeatedly training/testing on different subsets of the data.
Tree-based methods	Models that make predictions by splitting data hierarchically (e.g., decision trees, random forests).

Question 2

a. What shape does a perfect classifier look like on an ROC curve? What about a bad classifier?

A perfect classifier has an ROC curve that hugs the top-left corner which is when AUC equals 1 so it classifies all the positives and negatives correctly. A bad classifier goes along the diagonal line which is when AUC = 0.5.

b. Think about the formula for an F1 score. What does it mean when the F1 score is close to 1? Close to 0?

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

An F1 close to 1 means both precision and recall are high meaning the classifier is accurate and consistent.
An F1 near 0 means one or both are low meaning the model either misses positives or generates too many false alarms.

Question 3

	Linear Regression	Logistic Regression
Chart Shape	Straight line showing a continuous relationship between predictors and outcome.	S-shaped (sigmoid) curve showing probability between 0 and 1.
Dependent Variable Type	Continuous numeric variable (e.g., price, weight, mpg).	Categorical (binary) variable (e.g., yes/no, 0/1).
Purpose	Regression — predicts continuous outcomes.	Classification — predicts categorical outcomes.
Range of Output Variable	(-∞, ∞)	(0, 1), representing predicted probabilities.
Method	Ordinary Least Squares (OLS) — minimizes sum of squared errors.	Maximum Likelihood Estimation (MLE) — finds parameters that maximize likelihood of observed outcomes.
Example of Use	Predicting house price from square footage and number of rooms.	Predicting diabetes diagnosis from glucose, BMI, and age.

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).

Its important to train and then test the the model because when the model is fit on the data and then evaluated on the same data it can lead to overfiting due to the model already having “seen” the data.So you train on 75% and test the remaining 25% to check generalization and if the accuracy is good on the test set then the model is also good.

Question 5

a. First, load the housing.csv data set. Look at the data in some useful way. Why is linear regression appropriate here?

```
library(tidyverse)
library(caret)



housing <- read_csv("housing.csv")

head(housing)
```

longitude	latitude	housing_median_age	total_rooms	population	households	median_income
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl
-122.23	37.88	41	880	322	126	8.325
-122.22	37.86	21	7099	2401	1138	8.301
-122.24	37.85	52	1467	496	177	7.257
-122.25	37.85	52	1274	558	219	5.643
-122.25	37.85	52	1627	565	259	3.846
-122.25	37.85	52	919	413	193	4.036

6 rows | 1-7 of 8 columns

```
summary(housing)
```

```
##    longitude         latitude     housing_median_age  total_rooms
##  Min.   :-124.3   Min.   :32.54   Min.   : 1.00      Min.   :    2
##  1st Qu.:-121.8   1st Qu.:33.93   1st Qu.:18.00      1st Qu.: 1448
##  Median :-118.5   Median :34.26   Median :29.00      Median : 2127
##  Mean   :-119.6   Mean   :35.63   Mean   :28.64      Mean   : 2636
##  3rd Qu.:-118.0   3rd Qu.:37.71   3rd Qu.:37.00      3rd Qu.: 3148
##  Max.   :-114.3   Max.   :41.95   Max.   :52.00      Max.   :39320
##    population      households     median_income     median_house_value
##  Min.   :    3   Min.   :   1.0   Min.   : 0.4999   Min.   : 14999
##  1st Qu.:  787   1st Qu.: 280.0   1st Qu.: 2.5634   1st Qu.:119600
##  Median : 1166   Median : 409.0   Median : 3.5348   Median :179700
##  Mean   : 1425   Mean   : 499.5   Mean   : 3.8707   Mean   :206856
##  3rd Qu.: 1725   3rd Qu.: 605.0   3rd Qu.: 4.7432   3rd Qu.:264725
##  Max.   :35682   Max.   :6082.0   Max.   :15.0001   Max.   :500001
```

```
colSums(is.na(housing))
```

```
##          longitude           latitude housing_median_age        total_rooms
##                  0                  0                  0                  0
##         population         households      median_income median_house_value
##                  0                  0                  0                  0
```

Linear regression is appropriate because the response variable median_house_value is continuous. Predictors like median_income, housing_median_age, and total_rooms are numeric. We assume approximate linear relationships between predictors and the target variable.

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

```
set.seed(123)
housing <- housing %>% select_if(is.numeric)

train_index <- createDataPartition(housing$median_house_value, p = 0.75, list = FALSE)
train_data <- housing[train_index, ]
test_data <- housing[-train_index, ]

preproc <- preProcess(train_data, method = c("center", "scale"))
train_scaled <- predict(preproc, train_data)
test_scaled <- predict(preproc, test_data)
```

c. Fit the model.

```
model <- lm(median_house_value ~ ., data = train_scaled)

summary(model)
```

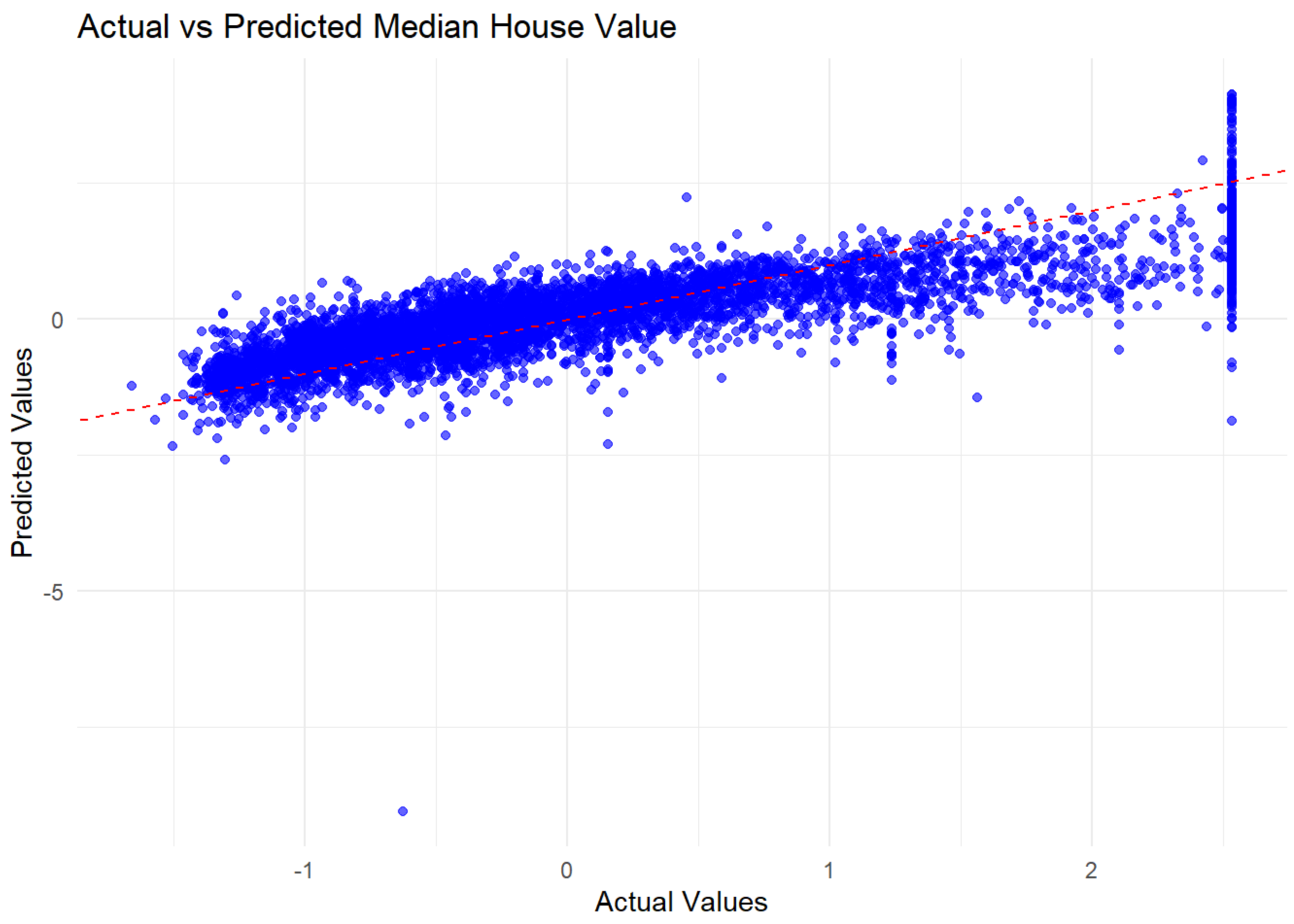
```
##
## Call:
## lm(formula = median_house_value ~ ., data = train_scaled)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6829 -0.3842 -0.1018  0.2679  4.2195
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         3.415e-15  4.888e-03   0.000   1.0000
## longitude          -7.183e-01  1.440e-02 -49.872   <2e-16 ***
## latitude           -7.717e-01  1.455e-02 -53.037   <2e-16 ***
## housing_median_age  1.291e-01  5.494e-03  23.506   <2e-16 ***
## total_rooms        -2.684e-02  1.530e-02  -1.754   0.0794 .
## population         -4.610e-01  1.241e-02 -37.135   <2e-16 ***
## households          5.245e-01  1.693e-02  30.980   <2e-16 ***
## median_income       6.285e-01  6.040e-03 104.057   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6082 on 15473 degrees of freedom
## Multiple R-squared:  0.6302, Adjusted R-squared:  0.6301
## F-statistic:  3768 on 7 and 15473 DF,  p-value: < 2.2e-16
```

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

```
predictions <- predict(model, newdata = test_scaled)

results <- tibble(
  Actual = test_scaled$median_house_value,
  Predicted = predictions
)

ggplot(results, aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.6, color = "blue") +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(
    title = "Actual vs Predicted Median House Value",
    x = "Actual Values",
    y = "Predicted Values"
  ) +
  theme_minimal()
```

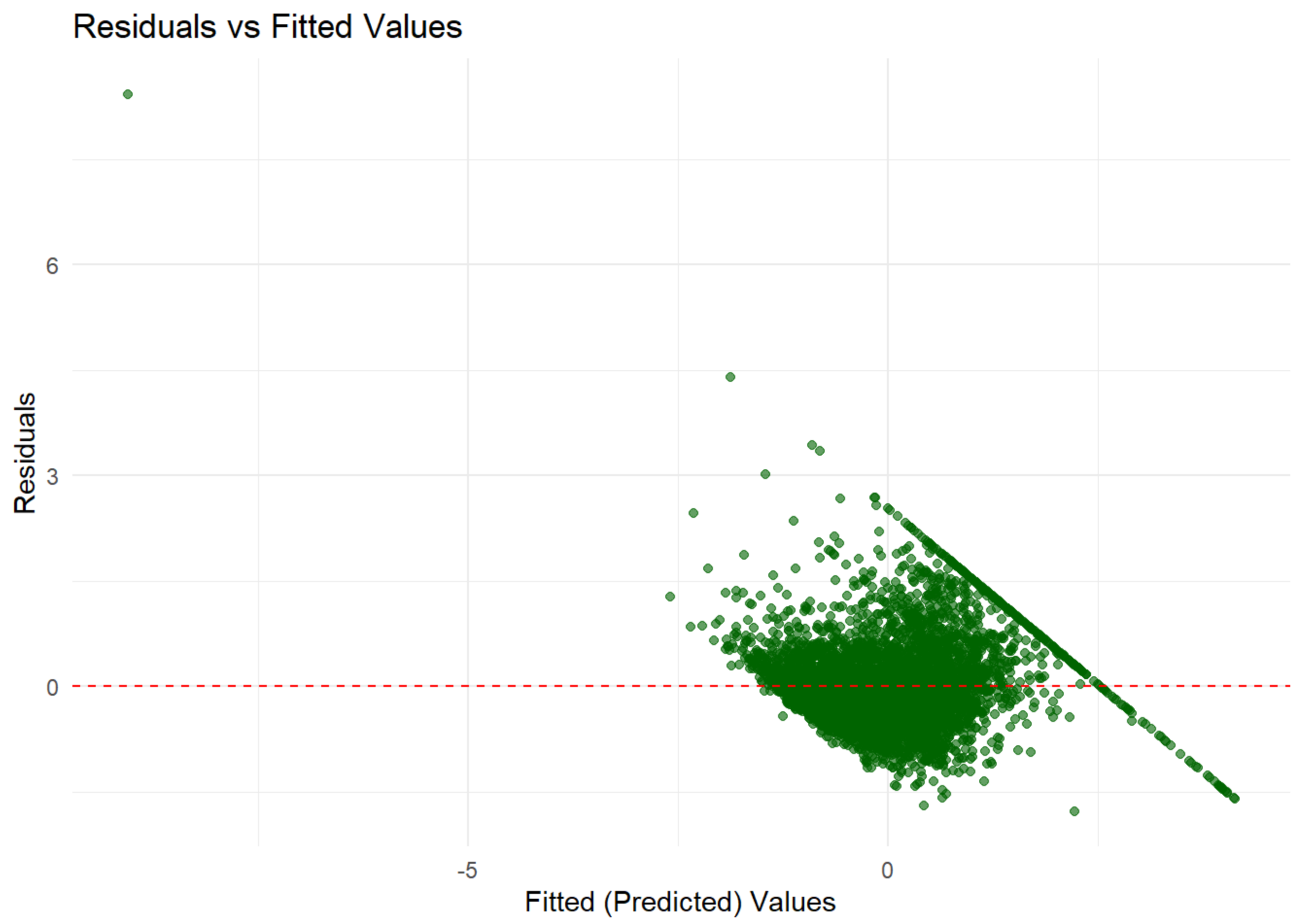


The plot shows pretty strong predictive accuracy as the points seem to follow along the perfect predictions dashed line pretty closely.

e. Make a residuals plot.

```
results <- results %>%
  mutate(Residuals = Actual - Predicted)

ggplot(results, aes(x = Predicted, y = Residuals)) +
  geom_point(alpha = 0.6, color = "darkgreen") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(
    title = "Residuals vs Fitted Values",
    x = "Fitted (Predicted) Values",
    y = "Residuals"
  ) +
  theme_minimal()
```



The residual plots shows that they are not randomly scattered around 0 which would be ideal. So I would not say that the model captures the relationship the best but its not completely far off.

Question 6

a. First, load the diabetes.csv data set. Look at the data in some useful way. Why is logistic regression appropriate here?

```
library(tidyverse)
library(caret)
library(glmnet)
diabetes <- read_csv("diabetes.csv")
glimpse(diabetes)
```

```
## Rows: 768
## Columns: 9
## $ Pregnancies              <dbl> 6, 1, 8, 1, 0, 5, 3, 10, 2, 8, 4, 10, 10, 1, …
## $ Glucose                  <dbl> 148, 85, 183, 89, 137, 116, 78, 115, 197, 125…
## $ BloodPressure            <dbl> 72, 66, 64, 66, 40, 74, 50, 0, 70, 96, 92, 74…
## $ SkinThickness            <dbl> 35, 29, 0, 23, 35, 0, 32, 0, 45, 0, 0, 0, 0, …
## $ Insulin                  <dbl> 0, 0, 0, 94, 168, 0, 88, 0, 543, 0, 0, 0, 0, …
## $ BMI                      <dbl> 33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31.0, 35.…
## $ DiabetesPedigreeFunction <dbl> 0.627, 0.351, 0.672, 0.167, 2.288, 0.201, 0.2…
## $ Age                      <dbl> 50, 31, 32, 21, 33, 30, 26, 29, 53, 54, 30, 3…
## $ Outcome                  <dbl> 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, …
```

The outcome variable is binary so logistic regression would be appropriate.

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

```
scale <- function(a){ (a - min(a)) / (max(a) - min(a)) }

set.seed(123)
train_idx <- runif(nrow(diabetes)) < 0.75
train <- diabetes[train_idx, ]
test  <- diabetes[!train_idx, ]

train <- train %>% mutate(across(where(is.numeric), scale))
test  <- test %>% mutate(across(where(is.numeric), scale))
```

c. Fit the model.

```
model <- glm(Outcome ~ ., data = train, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = Outcome ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4834  -0.6962  -0.3956   0.6975   3.0210
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -8.4200     0.8268 -10.184  < 2e-16 ***
## Pregnancies                1.6903     0.6650   2.542   0.0110 *
## Glucose                    7.4451     0.8871   8.393  < 2e-16 ***
## BloodPressure             -1.4281     0.7340  -1.946   0.0517 .
## SkinThickness             -0.3911     0.7976  -0.490   0.6239
## Insulin                   -1.2607     0.9116  -1.383   0.1667
## BMI                        6.3446     1.2121   5.234 1.66e-07 ***
## DiabetesPedigreeFunction   2.1300     0.8296   2.567   0.0102 *
## Age                        1.4178     0.6856   2.068   0.0386 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 744.25  on 574  degrees of freedom
## Residual deviance: 524.03  on 566  degrees of freedom
## AIC: 542.03
##
## Number of Fisher Scoring iterations: 5
```

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

```
pred_probs <- predict(model, newdata = test, type = "response")
pred_labels <- ifelse(pred_probs > 0.5, 1, 0)

conf <- table(Predicted = pred_labels, Actual = test$Outcome)
accuracy <- sum(diag(conf)) / sum(conf)

conf
```

```
##          Actual
## Predicted   0   1
##         0 107  32
##         1  19  35
```

```
accuracy
```

```
## [1] 0.7357513
```

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?

```
library(glmnet)
library(tidyverse)
library(caret)
diabetes$Outcome <- as.factor(diabetes$Outcome)


set.seed(123)
train_index <- createDataPartition(diabetes$Outcome, p = 0.75, list = FALSE)
train_data <- diabetes[train_index, ]
test_data <- diabetes[-train_index, ]


preproc <- preProcess(train_data[, -ncol(train_data)], method = c("center", "scale"))
train_scaled <- predict(preproc, train_data[, -ncol(train_data)])
test_scaled <- predict(preproc, test_data[, -ncol(test_data)])

x_train <- as.matrix(train_scaled)
y_train <- train_data$Outcome
x_test <- as.matrix(test_scaled)
y_test <- test_data$Outcome

lasso_model <- cv.glmnet(
  x_train,
  y_train,
  alpha = 1,
  family = "binomial",
  type.measure = "class"
)


lasso_model$lambda.min
```

```
## [1] 0.01501857
```

```
coef_lasso <- coef(lasso_model, s = "lambda.min")
coef_lasso
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                  s1
## (Intercept)              -0.8029985
## Pregnancies               0.2497610
## Glucose                   0.9512493
## BloodPressure            -0.1068222
## SkinThickness             .
## Insulin                   .
## BMI                       0.5325463
## DiabetesPedigreeFunction  0.1680215
## Age                       0.1607026
```

```
lasso_pred_prob <- predict(lasso_model, newx = x_test, s = "lambda.min", type = "response")



lasso_pred_class <- ifelse(lasso_pred_prob > 0.5, 1, 0)

conf_matrix <- table(Predicted = lasso_pred_class, Actual = y_test)
conf_matrix
```

```
##          Actual
## Predicted   0   1
##         0 108  28
##         1  17  39
```

```
lasso_accuracy <- mean(lasso_pred_class == y_test)
lasso_accuracy
```

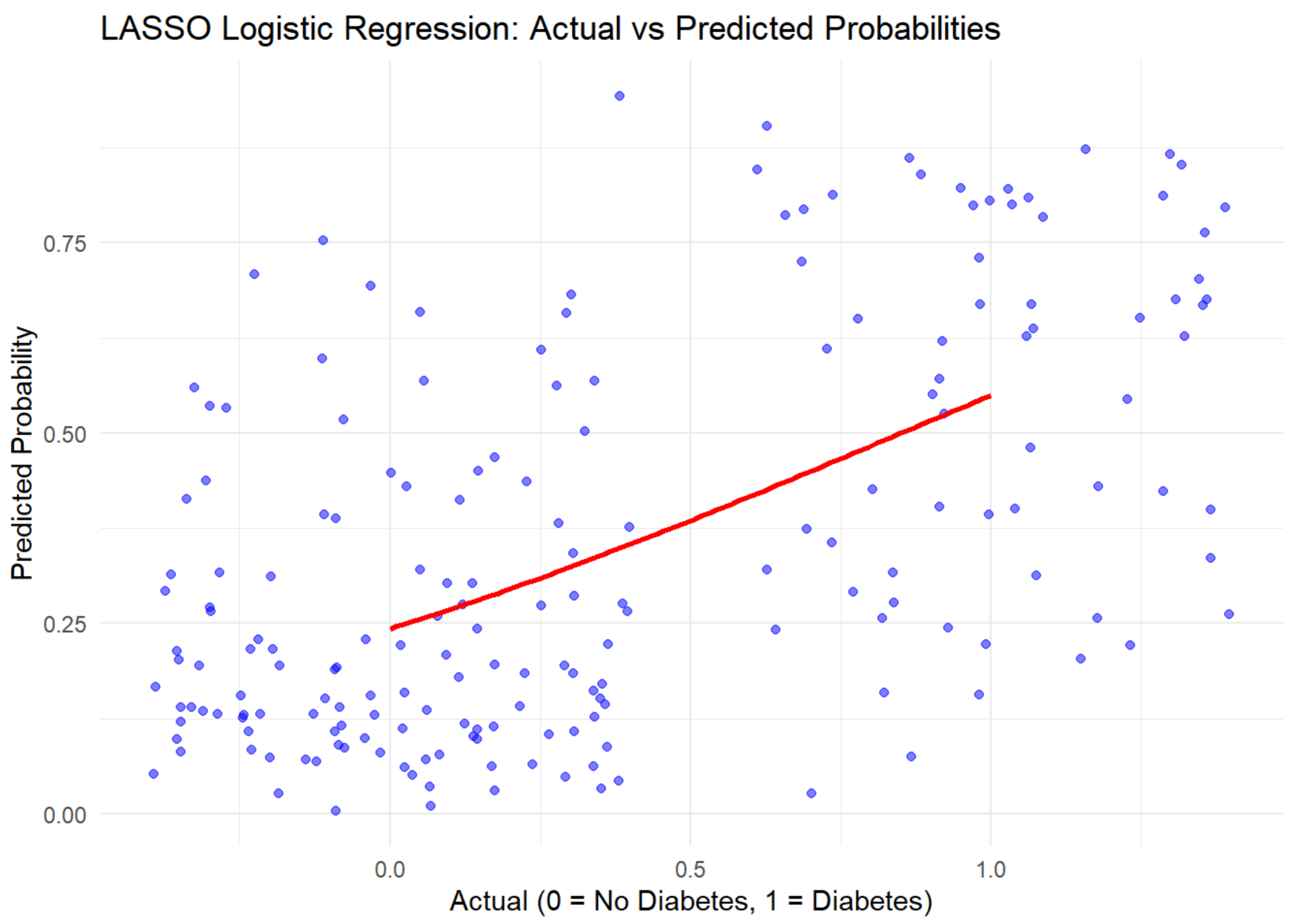
```
## [1] 0.765625
```

Pregnancies, Glucose, BMI,BloodPressure, DiabetesPedigreeFunction, and Age are the most important variables since they do not cross or go to zero.LASSO improves model generalization be penalizing complexity and it keeps variables that meaningfully contribute to the prediction, but it can lose a little bit of accuracy.

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

```
results_lasso <- tibble(
  Actual = as.numeric(as.character(y_test)),
  Predicted_Prob = as.numeric(lasso_pred_prob)
)

ggplot(results_lasso, aes(x = Actual, y = Predicted_Prob)) +
  geom_jitter(alpha = 0.5, color = "blue") +
  geom_smooth(method = "glm", method.args = list(family = "binomial"), se = FALSE, color = "re
d") +
  labs(
    title = "LASSO Logistic Regression: Actual vs Predicted Probabilities",
    x = "Actual (0 = No Diabetes, 1 = Diabetes)",
    y = "Predicted Probability"
  ) +
  theme_minimal()
```


LASSO Logistic Regression: Actual vs Predicted Probabilities
Predicted Probability
Actual (0 = No Diabetes, 1 = Diabetes)