```
# Load data
train_data = read.csv("ProjectTrain.csv")
test_data = read.csv("ProjectTest.csv")

# Convert Sex and Embarked to factors
train_data$Sex = factor(train_data$Sex)
train_data$Embarked = factor(train_data$Embarked)
test_data$Sex = factor(test_data$Sex)
test_data$Embarked = factor(test_data$Embarked)

# Impute missing values of Age with mean
train_data$Age[is.na(train_data$Age)] = mean(train_data$Age, na.rm = TRUE)
test_data$Age[is.na(test_data$Age)] = mean(train_data$Age, na.rm = TRUE)

# Calculate the contingency table and chi-square statistic for Sex and
Embarked
table_SE = table(train_data$Sex, train_data$Embarked)
chisq_SE = chisq.test(table_SE)$statistic
# Calculate the correlation matrix for numeric variables
predictors <- train_data[, c("Pclass", "Age", "SibSp", "Parch")]
cor_matrix <- cor(predictors)
# Print the results
print("Contingency table for Sex and Embarked:")
print(table_SE)
print(paste0("Chi-square statistic for Sex and Embarked: ", chisq_SE))
print("Correlation matrix for numeric predictors:")
print(cor_matrix)


#p1
# Logistic regression model
log_model = glm(Survived ~ Pclass + Sex + Age + SibSp + Parch + Embarked,
                 data = train_data,
                 family = "binomial")
log_predictions = predict(log_model, newdata = test_data, type =
"response")
log_predictions = ifelse(log_predictions > 0.5, 1, 0)
log_cm = table(log_predictions, test_data$Survived)
log_accuracy = sum(diag(log_cm)) / sum(log_cm)
cat(paste("Logistic regression accuracy:", log_accuracy, "\n"))

# LDA model
library(MASS)
lda_model <- lda(Survived ~ Pclass + Sex + Age + SibSp + Parch + Embarked,
                 data = train_data)
lda_predictions <- predict(lda_model, newdata = test_data)
lda_cm <- table(lda_predictions$class, test_data$Survived)
lda_accuracy <- sum(diag(lda_cm)) / sum(lda_cm)
cat(paste("LDA accuracy:", lda_accuracy, "\n"))

# QDA model with regularization
qda_model <- qda(Survived ~ Pclass + Sex + Age + SibSp + Parch,
                     data = train_data)
qda_predictions_reg <- predict(qda_model, newdata = test_data)
qda_cm_reg <- table(qda_predictions_reg$class, test_data$Survived)
qda_accuracy_reg <- sum(diag(qda_cm_reg)) / sum(qda_cm_reg)
cat(paste("QDA with regularization accuracy:", qda_accuracy_reg, "\n"))
```

```
# Convert Sex and Embarked to factors
train_data$Sex <- factor(train_data$Sex)
train_data$Embarked <- factor(train_data$Embarked)
test_data$Sex <- factor(test_data$Sex)
test_data$Embarked <- factor(test_data$Embarked)
# Encode factors as numeric values
train_data$Sex <- as.numeric(train_data$Sex) - 1
train_data$Embarked <- as.numeric(train_data$Embarked)
test_data$Sex <- as.numeric(test_data$Sex) - 1
test_data$Embarked <- as.numeric(test_data$Embarked)
# Fit KNN model
library(class)
set.seed(123)
# KNN model with k=5
knn_model <- knn(train = train_data[, c("Pclass", "Sex", "Age", "SibSp",
"Parch", "Embarked")],
                 test = test_data[, c("Pclass", "Sex", "Age", "SibSp",
"Parch", "Embarked")],
                 cl = train_data$Survived,
                 k = 5)
knn_cm <- table(knn_model, test_data$Survived)
knn_accuracy <- sum(diag(knn_cm)) / sum(knn_cm)
cat(paste("KNN accuracy with k = 5:", knn_accuracy, "\n")) # Accuracy =
0.738

# KNN model with k=3
knn_model <- knn(train = train_data[, c("Pclass", "Sex", "Age", "SibSp",
"Parch", "Embarked")],
                 test = test_data[, c("Pclass", "Sex", "Age", "SibSp",
"Parch", "Embarked")],
                 cl = train_data$Survived,
                 k = 3)
knn_cm <- table(knn_model, test_data$Survived)
knn_accuracy <- sum(diag(knn_cm)) / sum(knn_cm)
cat(paste("KNN accuracy with k = 3:", knn_accuracy, "\n")) # Accuracy =
0.753

# KNN model with k=7
knn_model <- knn(train = train_data[, c("Pclass", "Sex", "Age", "SibSp",
"Parch", "Embarked")],
                 test = test_data[, c("Pclass", "Sex", "Age", "SibSp",
"Parch", "Embarked")],
                 cl = train_data$Survived,
                 k = 7)
knn_cm <- table(knn_model, test_data$Survived)
knn_accuracy <- sum(diag(knn_cm)) / sum(knn_cm)
cat(paste("KNN accuracy with k = 7:", knn_accuracy, "\n")) # Accuracy =
0.73

log_cm
lda_cm
qda_cm_reg
knn_cm

#p2
# Replace missing values in Cabin with "Not Available"
train_data$Cabin <- ifelse(train_data$Cabin == "", "Not Available",
"Available")
test_data$Cabin <- ifelse(test_data$Cabin == "", "Not Available",
```

```
"Available")

# Model 1: Logistic Regression with Cabin feature included
model1 <- glm(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare +
Embarked + Cabin, data=train_data, family="binomial")

# Model 2: Logistic Regression without Cabin feature
model2 <- glm(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare +
Embarked, data=train_data, family="binomial")

# Predict using model 1
pred1 = predict(model1, newdata=test_data, type="response")
pred1 = ifelse(pred1 > 0.5, 1, 0)

# Compute confusion matrix and accuracy for model 1
cm1 <- table(test_data$Survived, pred1)
accuracy1 <- sum(diag(cm1))/sum(cm1)

# Predict using model 2
pred2 <- predict(model2, newdata=test_data, type="response")
pred2 <- ifelse(pred2 > 0.5, 1, 0)

# Compute confusion matrix and accuracy for model 2
cm2 <- table(test_data$Survived, pred2)
accuracy2 <- sum(diag(cm2))/sum(cm2)

# Print the confusion matrices and accuracy scores
cat("Confusion matrix for Model 1:\n")
print(cm1)
cat("Accuracy for Model 1: ", accuracy1, "\n\n")
cat("Confusion matrix for Model 2:\n")
print(cm2)
cat("Accuracy for Model 2: ", accuracy2, "\n\n")




#p3
# Fit logistic regression model
model = glm(Survived ~ Sex + Pclass + Embarked + Cabin, data = train_data,
family = binomial)

# Obtain coefficient estimates
coef_estimates = summary(model)$coefficients[,1]

# Exponentiate to obtain odds ratios
odds_ratios = exp(coef_estimates)

# Print adjusted odds ratios
cat("Adjusted odds ratios:\n")
cat("Sex: ", odds_ratios[2], "\n")
cat("Pclass: ", odds_ratios[3], "\n")
cat("Embarked: ", odds_ratios[4], "\n")



#4
# define the threshold values
thresholds <- c(0.2, 0.5, 0.8)
```

```
# create empty vectors to store the metrics
accuracy <- precision <- recall <- f1 <- rep(0, length(thresholds))

# loop through the thresholds and calculate the metrics
for (i in seq_along(thresholds)) {
  # make predictions using the LR model and the current threshold
  predictions <- ifelse(predict(model, test_data, type="response") >
thresholds[i], 1, 0)

  # calculate the confusion matrix
  cm <- table(predictions, test_data$Survived)

  # calculate the evaluation metrics
  accuracy[i] <- sum(diag(cm)) / sum(cm)
  precision[i] <- cm[2,2] / sum(cm[,2])
  recall[i] <- cm[2,2] / sum(cm[2,])
  f1[i] <- 2 * precision[i] * recall[i] / (precision[i] + recall[i])
}

# print the results
metrics <- data.frame(Threshold=thresholds, Accuracy=accuracy,
Precision=precision, Recall=recall, F1=f1)
print(metrics)



#5
library(pROC)
lda_model = lda(Survived ~ Pclass + Sex + Age + SibSp + Parch + Embarked,
                 data = train_data)
lda_predictions = predict(lda_model, newdata = test_data)
roc_obj = roc(test_data$Survived, lda_predictions$posterior[,
2],percent=TRUE, plot=TRUE,
              ci=TRUE)
plot(roc_obj, main = "ROC Curve for LDA Model")

#6
#fit a KNN model
library(class)
set.seed(123)
#fit the model and evaluate its performance using 10-fold cross-
validation:
knn_model = knn(train = train_data[, c("Pclass", "Sex", "Age", "Parch")],
                 test = test_data[, c("Pclass", "Sex", "Age","Parch")],
                 cl = train_data$Survived,
                 k = 5)
knn_cm <- table(knn_model, test_data$Survived)
knn_accuracy <- sum(diag(knn_cm)) / sum(knn_cm)
cat(paste("KNN accuracy with k = 5:", knn_accuracy, "\n"))
```