ADVANCED MATHEMATICAL STATISTICS – 522

Homework – 3

Banoth Jeevan Kumar

1) A description of the datasets (10%)

2) Description of data preprocessing, how the data is split as training and testing set (20%)

3) A conclusion or summary of results (40%). The reported accuracy needs to be averaged over 100 runs.

4) Include your R code as appendix to the report (20%)

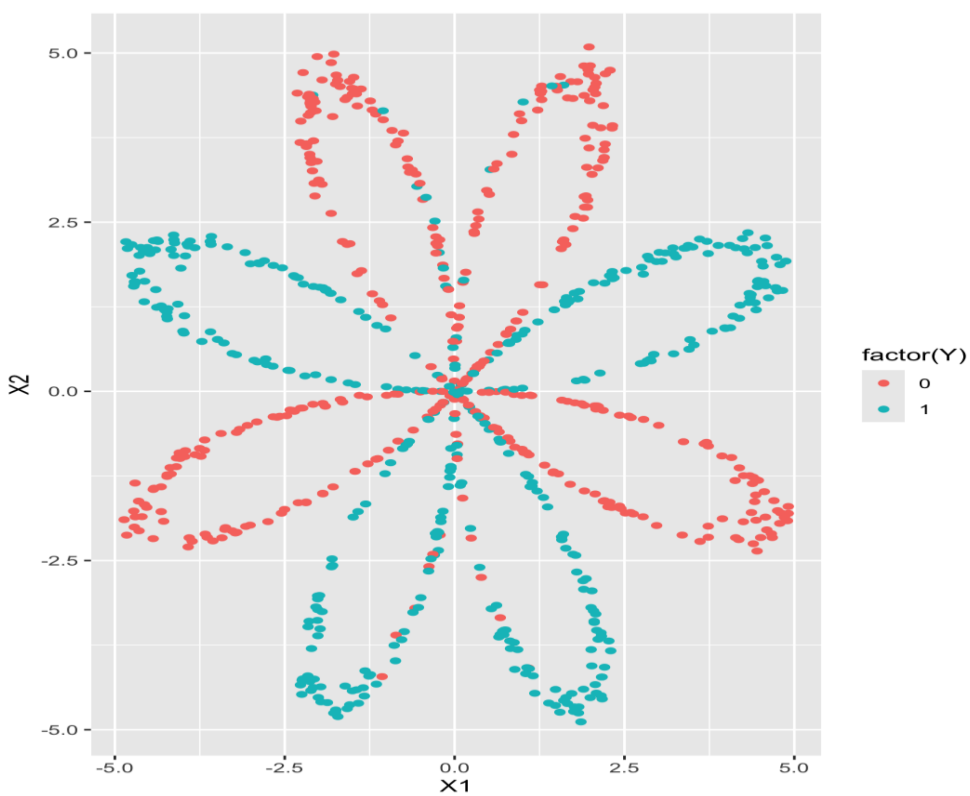
5) Clarity of the report carries 10%.

Ans:

I believe that creating a non-linear dataset would be beneficial in showcasing the superiority of Neural Networks over Logistic Regression. The dataset I plan to generate will consist of randomly generated planar data that is non-linear in nature. This dataset will require multiple decision boundaries to accurately classify or predict outcomes.

```
planar_dataset <- function(){ # A function makes it easier to modify the parameters, if required
set.seed(123) # To ensure reproducibility
m <- 800 # Total length of the data frame
N <- m/2 # Sample size
D <- 2 # Dimensions
X <- matrix(0, nrow = m, ncol = D) # matrix where each row is a single example
Y <- matrix(0, nrow = m, ncol = 1) # label vector (0 for red, 1 for blue)
a <- 5 # Max value of the flower spread
for(j in 0:1){ # For loop to generate a random sample
ix <- seq((N*j)+1, N*(j+1)) # range for sequence generation
t <- seq(j*3.12,(j+1)*3.12,length.out = N) + rnorm(N, sd = 0.2) # petal inclination
r <- a*sin(4*t) + rnorm(N, sd = 0.2) # radius
X[ix,1] <- r*sin(t) # fills the first column of the X matrix
X[ix,2] <- r*cos(t) # fills the second column of the X matrix
Y[ix,] <- j # will have value between 0 & 1
}
d <- as.data.frame(cbind(X, Y)) # Create a data-frame
```

```
names(d) <- c('X1','X2','Y') # Rename the columns accordingly
d
}

df <- planar_dataset()
df <- df[sample(nrow(df)), ] # Shuffle dataset
head(df)

## X1 X2 Y
## 740 2.053348 -3.3966571 1
## 730 -1.897771 1.4929127 1
## 648 -3.592936 0.8797975 1
## 342 -1.210739 4.1583541 0
## 123 -1.155820 -1.0033961 0
## 700 -1.307497 1.0938884 1

ggplot(df, aes(x = X1, y = X2, color = factor(Y))) +
geom_point()
```




```
set.seed(123) # For reproducibility
```

```
index <- sample(1:nrow(df), 0.8 * nrow(df), replace = FALSE) # Do a 80-20 train-test split
train <- df[index,]
test <- df[-index,]
glimpse(train)

## Rows: 640
## Columns: 3
## $ X1 <dbl> -1.158604778, 0.812910525, -3.371047466, -0.155050665, -1.816881952…
## $ X2 <dbl> 4.09778115, -0.73494525, -2.01363648, 1.67141934, 2.62977897, 1.559…
## $ Y <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0…

glimpse(test)

## Rows: 160
## Columns: 3
## $ X1 <dbl> 2.05334818, -3.59293564, 4.72474337, 3.73321458, 1.55710728, 4.7002…
## $ X2 <dbl> -3.396657065, 0.879797525, 1.532745099, -1.992538553, 4.651756167, …
## $ Y <dbl> 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0…


logit_train <- glm(Y ~ ., data = train, family = "binomial")
summary(logit_train) # Logistic regression using training set

##
## Call:
## glm(formula = Y ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
## Min 1Q Median 3Q Max
## -1.6198 -1.2205 0.7633 1.1721 1.5631
##
## Coefficients:
## Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.095684 0.082008 1.167 0.243
## X1 0.002088 0.032355 0.065 0.949
## X2 -0.214626 0.034320 -6.254 4.01e-10 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
## Null deviance: 886.60 on 639 degrees of freedom
## Residual deviance: 844.39 on 637 degrees of freedom
## AIC: 850.39
##
## Number of Fisher Scoring iterations: 4
```

```
logLik(logit_train) # Log likelihood of training set
```

```
## 'log Lik.' -422.1966 (df=3)
```

```
with(logit_train, pchisq(null.deviance - deviance, df.null - df.residual, lower.tail = FALSE))
```

```
## [1] 6.826302e-10
```

```
pR2(logit_train)["McFadden"]
```

```
## fitting null model for pseudo-r2
## McFadden
## 0.04760882
```

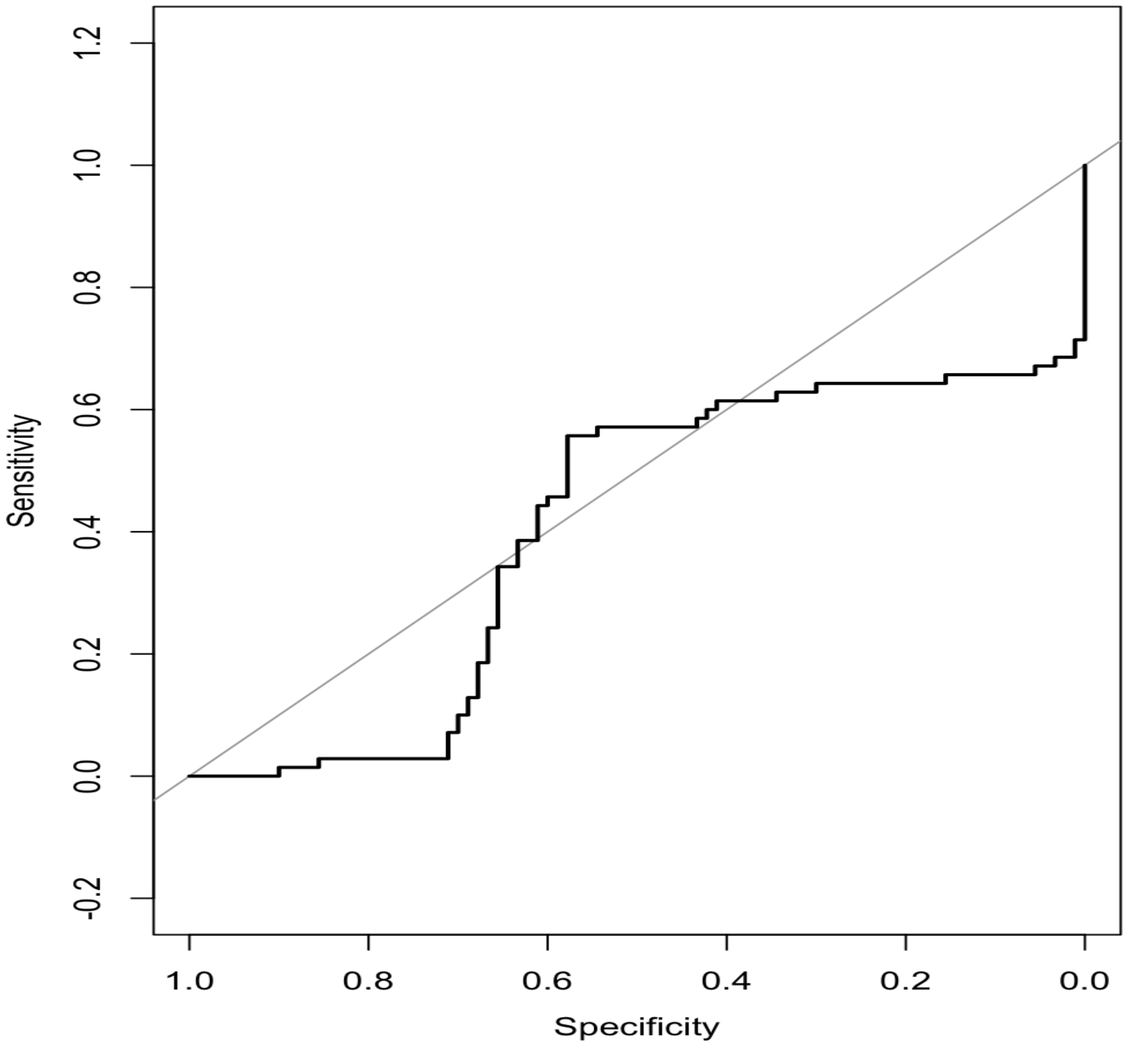
```
vif(logit_train) # Check for multicollinearity
```

```
## X1 X2
## 1.000042 1.000042
```

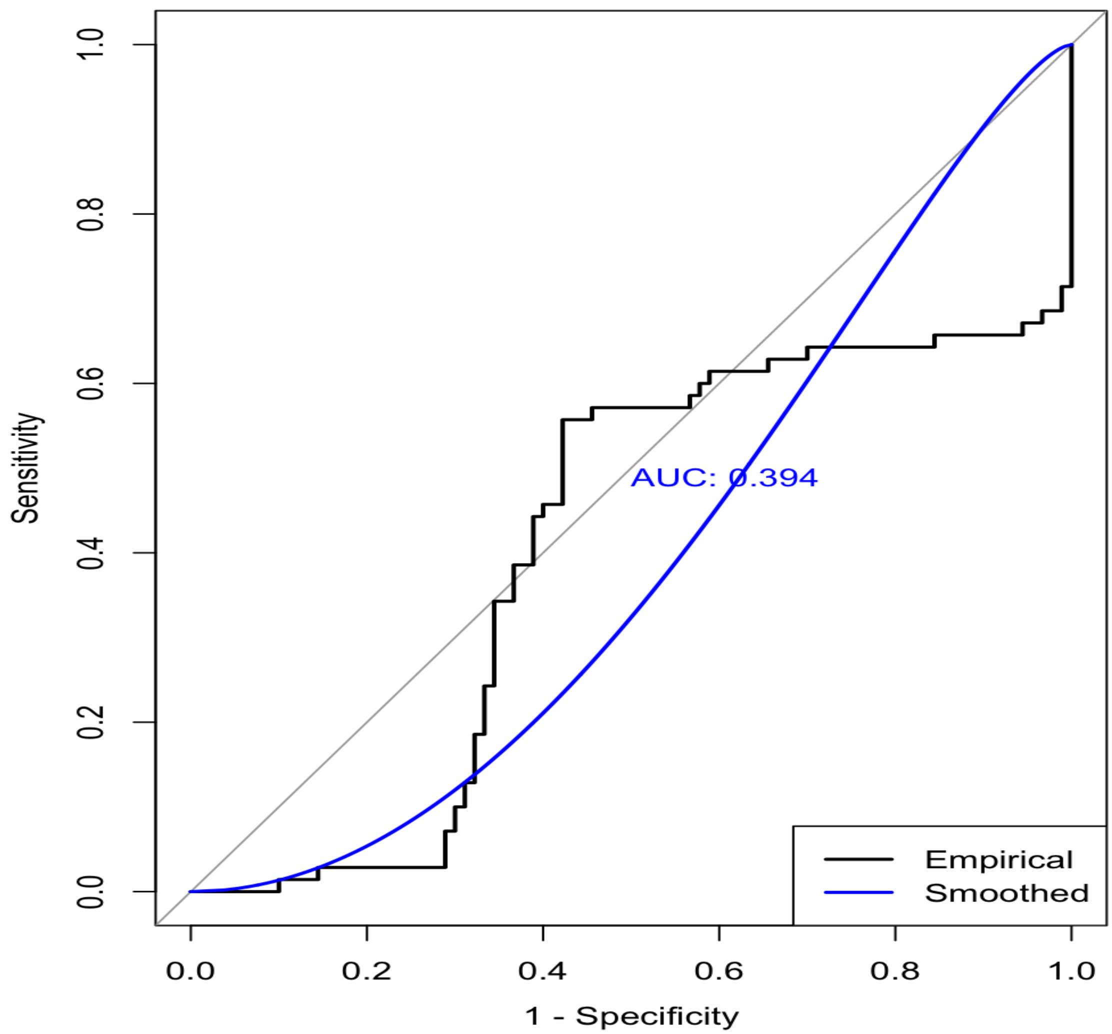
```
predictions_test <- predict(logit_train, test, type = "response")
predict_binary_test <- ifelse(predictions_test > 0.5, 1, 0)
cftable_test <- table(predict_binary_test, test$Y)
cftable_test
```

```
##
## predict_binary_test 0 1
## 0 35 30
## 1 55 40
accuracy_test <- sum(diag(cftable_test))/sum(cftable_test)
accuracy_test
## [1] 0.46875
sensitivity_test <- cftable_test[1]/(cftable_test[1] + cftable_test[2])
sensitivity_test
```

```
## [1] 0.3888889
specificity_test <- cftable_test[4]/(cftable_test[3] + cftable_test[4])
specificity_test
## [1] 0.5714286
ppv_test <- cftable_test[1]/(cftable_test[1] + cftable_test[3])
ppv_test
## [1] 0.5384615
npv_test <- cftable_test[4]/(cftable_test[2] + cftable_test[4])
npv_test

## [1] 0.4210526
error <- mean(predict_binary_test!=test$Y)
error
## [1] 0.53125
accuracy <- 1-error
accuracy
## [1] 0.46875
roc_test <- roc(test$Y,predictions_test)
## Setting levels: control = 0, case = 1
## Setting direction: controls > cases
plot.roc(roc_test, col=par("fg"),plot=TRUE,print.auc = FALSE, legacy.axes = TRUE, asp =NA)
roc_test
##
## Call:
## roc.default(response = test$Y, predictor = predictions_test)
##
## Data: predictions_test in 90 controls (test$Y 0) > 70 cases (test$Y 1).
## Area under the curve: 0.4049

plot_roc (roc_test)
```



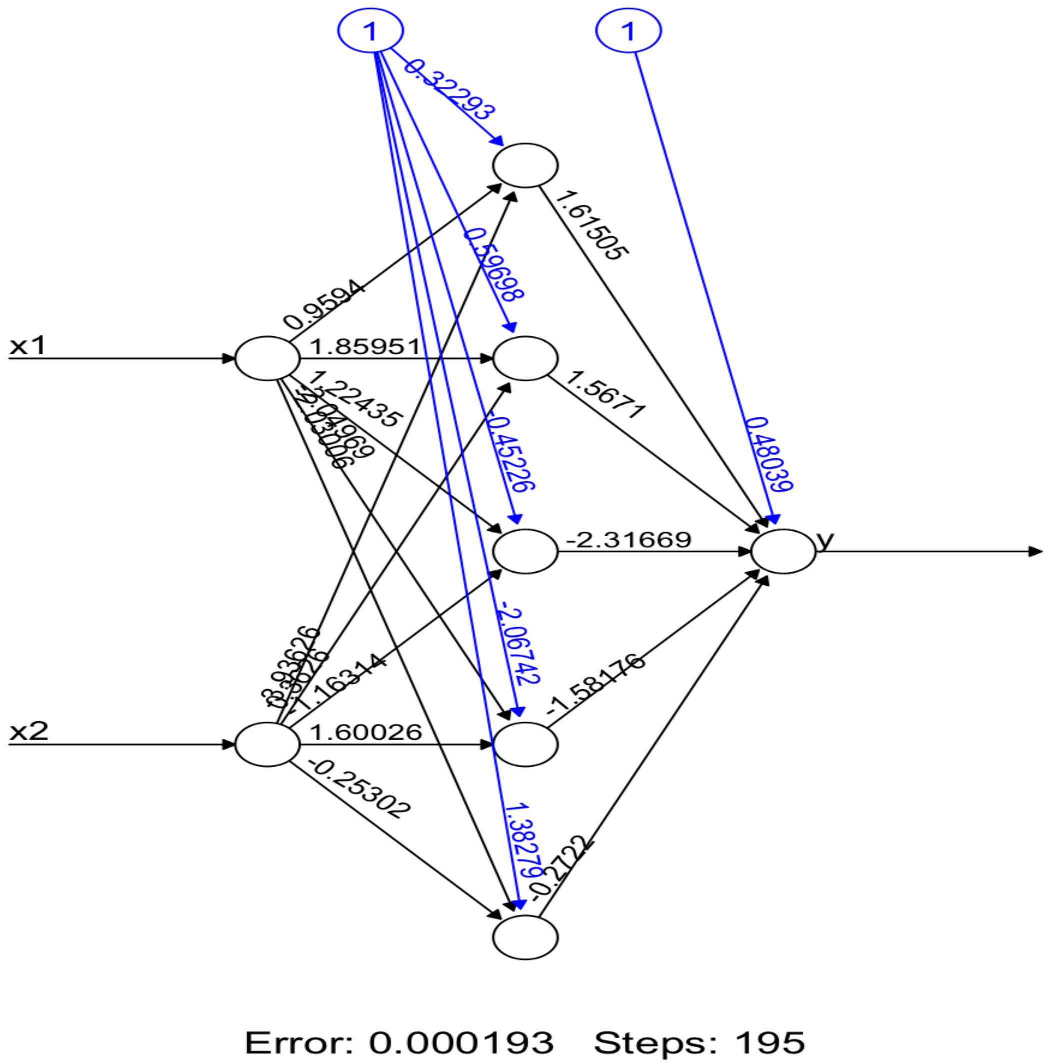
plot.roc(smooth(roc_test), col="blue", add = TRUE,plot=TRUE,print.auc = TRUE, legacy.axes = TRUE
, asp =NA)
legend("bottomright", legend=c("Empirical", "Smoothed"),
col=c(par("fg"), "blue"), lwd=2)



Based on the data analysis, it seems that Logistic Regression (LR) isn't a good fit for this dataset. This is because LR typically only creates one decision boundary, which isn't suitable for the complex, non-linear nature of the Planar dataset. With the use of Sine and Cosine functions, the dataset intentionally exists in a non-linear space, causing LR to perform poorly, as demonstrated by the AUC plot shown earlier.

```
seed <- 490
set.seed(seed)
planar_nn <- neuralnet(Y~., data = train, hidden = 5, rep = 100, stepmax = 10000) # Run a neural network
## Warning: Algorithm did not converge in 25 of 100 repetition(s) within the
```

```
## stepmax.
plot(planar_nn, rep = 1)
```




```
predict <- compute(planar_nn, test[,c(1:2)]) # Predict the trained model on the test set
#head(predict$net.result) # View the converged results - OPTIONAL!
p <- predict$net.result #Store the predicted values
prediction <- ifelse(p>0.5,1,0) # Create the decision boundaries
pred_table <- table(prediction,test$Y) # Generate a confusion matrix
pred_table
##
## prediction 0 1
## 0 85 6
```

```
## 1 5 64
error_nn <- 1-sum(diag(pred_table))/sum(pred_table) # Error of the NN
error_nn
## [1] 0.06875
accuracy_nn <- 1-error_nn # Accuracy of the NN
accuracy_nn
## [1] 0.93125
```

Based on the observations, it is evident that Neural Network (NN) significantly outperforms LR with an accuracy of around 94% compared to 47%. This superiority is mainly attributed to the nonlinear nature of the data. Moreover, adjusting the number of hidden layers has minimal impact on accuracy, but it does extend the processing time. Therefore, incorporating additional layers can be optimized with complementary techniques for enhanced performance. Now, let's examine a dataset where Logistic Regression (LR) surpasses Neural Network (NN) in terms of performance. Dataset Description: The tragic sinking of the Titanic remains one of the most well-known maritime disasters in history.

April 15, 1912 marked the fateful day when the supposedly unsinkable RMS Titanic tragically sank after hitting an iceberg. Tragically, there were not enough lifeboats for all the passengers on board, leading to the loss of 1502 out of 2224 individuals. Certain groups of people seemed to have had better chances of survival than others, despite some element of luck playing a role in the outcome. For this competition, participants will have access to two datasets - "train.csv" and "test.csv" - containing information about the passengers, such as their names, ages, genders, socio-economic classes, and more.

The train.csv file will include information on a group of passengers onboard (specifically 891 passengers) and will indicate whether they survived or not, which is also known as the “ground truth”. The test.csv dataset also has similar information but does not reveal the “ground truth” for each passenger. Your task is to make predictions about these outcomes. By analyzing the patterns in the train.csv data, you must predict whether the remaining 418 passengers onboard (as seen in test.csv) survived.

```
titanic_train <- read.csv (/Users/jeevankumarbanoth/Downloads/train.csv)
titanic_test <- read.csv (/Users/jeevankumarbanoth/Downloads/test.csv)

train <- titanic_train # The Training Set
```

```
test <- titanic_test # The Testing Set
glimpse(train)
```

```
## Rows: 891
## Columns: 12
## $ PassengerId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,…
## $ Survived <int> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1…
## $ Pclass <int> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3, 2, 3, 3…
## $ Name <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bradley (Fl…
## $ Sex <chr> "male", "female", "female", "female", "male", "male", "mal…
## $ Age <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, 39, 14, …
## $ SibSp <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, 0, 1, 0…
## $ Parch <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, 0, 0, 0…
## $ Ticket <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "113803", "37…
## $ Fare <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, 51.8625,…
## $ Cabin <chr> "", "C85", "", "C123", "", "", "E46", "", "", "", "G6", "C…
## $ Embarked <chr> "S", "C", "S", "S", "S", "Q", "S", "S", "S", "C", "S", "S"…
```

```
glimpse(test)
```

```
## Rows: 418
## Columns: 11
## $ PassengerId <int> 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903…
## $ Pclass <int> 3, 3, 2, 3, 3, 3, 3, 2, 3, 3, 3, 1, 1, 2, 1, 2, 2, 3, 3, 3…
## $ Name <chr> "Kelly, Mr. James", "Wilkes, Mrs. James (Ellen Needs)", "M…
## $ Sex <chr> "male", "female", "male", "male", "female", "male", "femal…
## $ Age <dbl> 34.5, 47.0, 62.0, 27.0, 22.0, 14.0, 30.0, 26.0, 18.0, 21.0…
## $ SibSp <int> 0, 1, 0, 0, 1, 0, 0, 1, 0, 2, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0…
## $ Parch <int> 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ Ticket <chr> "330911", "363272", "240276", "315154", "3101298", "7538",…
## $ Fare <dbl> 7.8292, 7.0000, 9.6875, 8.6625, 12.2875, 9.2250, 7.6292, 2…
## $ Cabin <chr> "", "", "", "", "", "", "", "", "", "", "", "", "B45", "",…
## $ Embarked <chr> "Q", "S", "Q", "S", "S", "S", "Q", "S", "C", "S", "S", "S"…
```

```
colSums(is.na(train) | train == " ") # Check for NAs and blank values
```

```
## PassengerId Survived Pclass Name Sex Age
## 0 0 0 0 0 177
## SibSp Parch Ticket Fare Cabin Embarked
```

```
## 0 0 0 0 0 0
```

```
colSums(is.na(test) | test == " ")
```

```
## PassengerId Pclass Name Sex Age SibSp
## 0 0 0 0 86 0
## Parch Ticket Fare Cabin Embarked
## 0 0 1 0 0
```

```
set.seed(444)
titanic_glm <- glm(Survived ~ Sex, data = train_copy, family = 'binomial')
summary(titanic_glm)
```

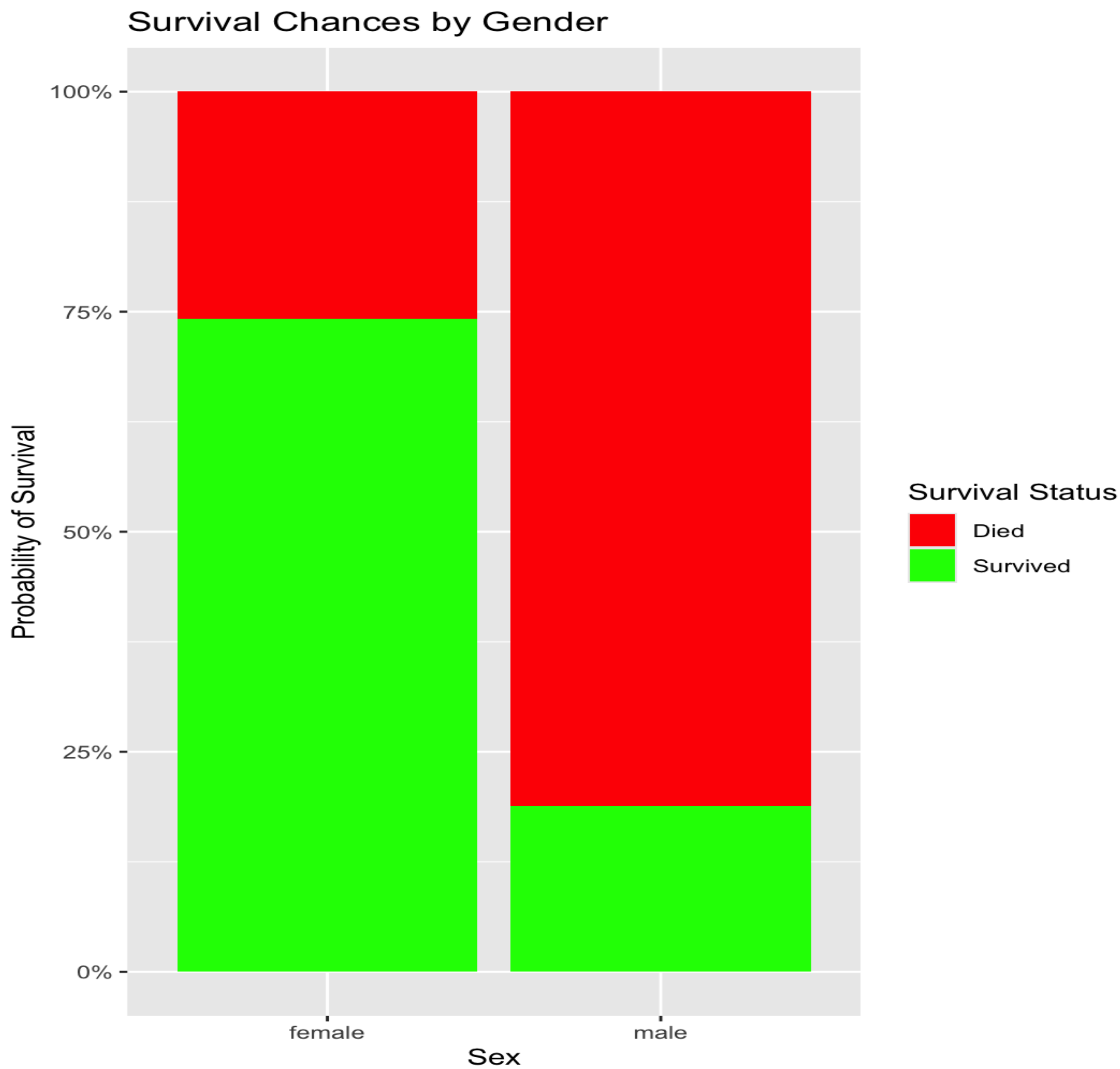
```
##
## Call:
## glm(formula = Survived ~ Sex, family = "binomial", data = train_copy)
##
## Deviance Residuals:
## Min 1Q Median 3Q Max
## -1.6462 -0.6471 -0.6471 0.7725 1.8256
##
## Coefficients:
## Estimate Std. Error z value Pr(>|z|)
## (Intercept) 1.0566 0.1290 8.191 2.58e-16 ***
## Sexmale -2.5137 0.1672 -15.036 < 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: 1186.7 on 890 degrees of freedom
## Residual deviance: 917.8 on 889 degrees of freedom
## AIC: 921.8
##
## Number of Fisher Scoring iterations: 4
```

```
# Test for accuracy

predict_sex_survived <- predict(titanic_glm,newdata = test_copy,type = 'response')

# Since Survived can only be either 1 or 0, write if statement to round up of down the response
```

```
predict_sex_survived <- ifelse(predict_sex_survived>0.5,1,0)
error_1 <- mean(predict_sex_survived!=test_copy$Survived)
accuracy_1 <- 1-error_1
accuracy_1
```

```
## [1] 0.7559809
```

```
prob_survival <- data.frame(prob_survival = titanic_glm$fitted.values,Survived =
train_copy$Surv
ived, Sex = train_copy$Sex)
ggplot(prob_survival, aes(fill = Survived, y = prob_survival, x = Sex)) +
geom_bar(position = "fill", stat = "identity") +
labs(x= "Survival Status", y = "Probability of Survival", title = "Survival chances by Gender"
) +
scale_y_continuous(labels = scales::percent_format()) +
scale_fill_discrete(labels = c("Died","Survived"))
```




```
logistic_complete <- glm(Survived~., data = train_copy, family = "binomial")
summary(logistic_complete)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = "binomial", data = train_copy)
##
## Deviance Residuals:
## Min 1Q Median 3Q Max
## -2.6834 -0.6053 -0.4060 0.6202 2.4785
```

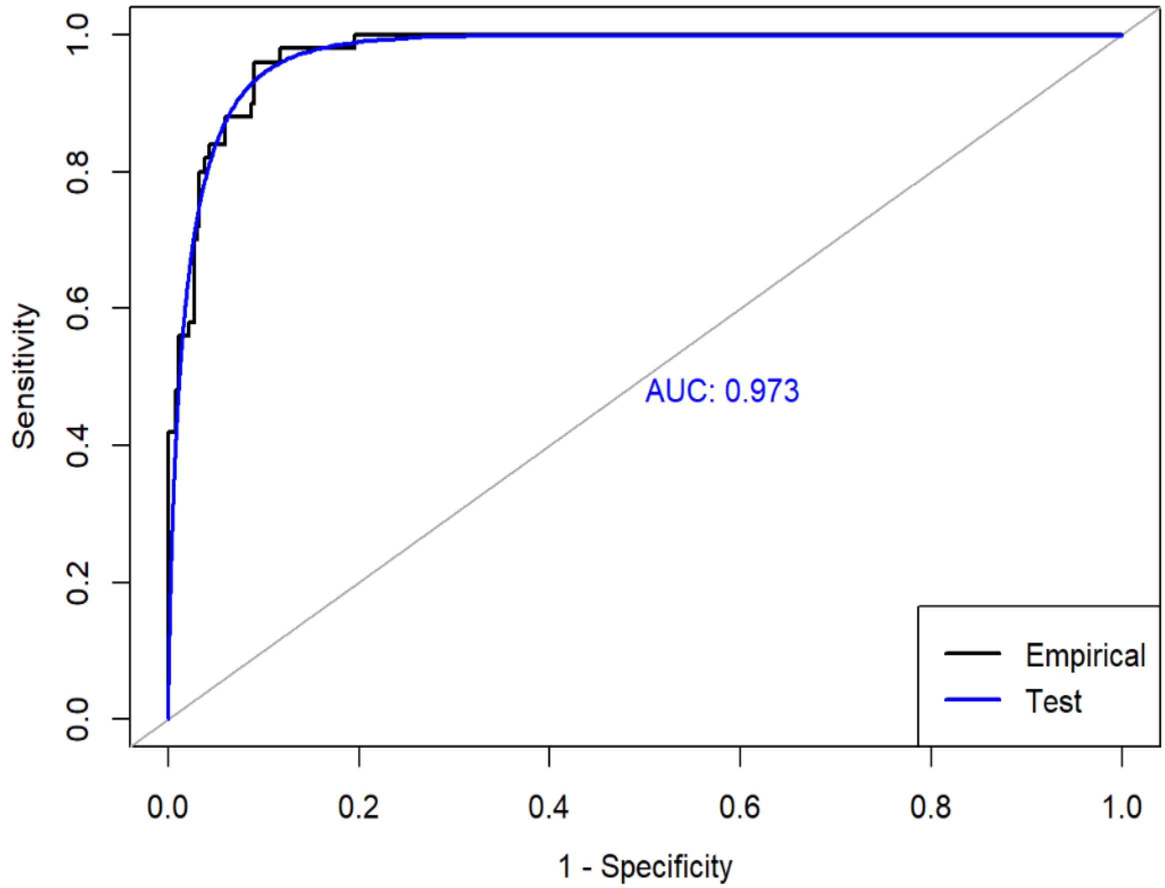
```
##
## Coefficients:
## Estimate Std. Error z value Pr(>|z|)
## (Intercept) 16.633165 608.445775 0.027 0.978191
## Pclass2 -1.056168 0.304843 -3.465 0.000531 ***
## Pclass3 -2.337544 0.311508 -7.504 6.19e-14 ***
## Sexmale -2.681168 0.201848 -13.283 < 2e-16 ***
## Age -0.043020 0.008119 -5.298 1.17e-07 ***
## SibSp -0.360844 0.111530 -3.235 0.001215 **
## Parch -0.099547 0.120556 -0.826 0.408955
## Fare 0.002006 0.002463 0.814 0.415414
## EmbarkedC -12.300751 608.445644 -0.020 0.983871
## EmbarkedQ -12.417556 608.445701 -0.020 0.983717
## EmbarkedS -12.718626 608.445631 -0.021 0.983323
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 1186.66 on 890 degrees of freedom
## Residual deviance: 779.97 on 880 degrees of freedom
## AIC: 801.97
##
## Number of Fisher Scoring iterations: 13
```

```
logistic_stepwise <- logistic_complete %>% stepAIC(direction = "both", trace = FALSE)
summary(logistic_stepwise)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age + SibSp, family = "binomial",
## data = train_copy)
##
## Deviance Residuals:
## Min 1Q Median 3Q Max
## -2.7628 -0.5958 -0.4020 0.6177 2.4872
##
## Coefficients:
## Estimate Std. Error z value Pr(>|z|)
## (Intercept) 4.282319 0.421806 10.152 < 2e-16 ***
```

```
## Pclass2 -1.311027 0.267854 -4.895 9.85e-07 ***
## Pclass3 -2.547895 0.258793 -9.845 < 2e-16 ***
## Sexmale -2.700613 0.194536 -13.882 < 2e-16 ***
## Age -0.044424 0.008044 -5.522 3.34e-08 ***
## SibSp -0.399100 0.106216 -3.757 0.000172 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 1186.66 on 890 degrees of freedom
## Residual deviance: 785.74 on 885 degrees of freedom
## AIC: 797.74
##
## Number of Fisher Scoring iterations: 5
AIC(logistic_complete,logistic_stepwise)
## df AIC
## logistic_complete 11 801.9660
## logistic_stepwise 6 797.7355
predict_logistic_test <- predict(logistic_stepwise,test_copy,type = "response")
predict_logistic_1 <- ifelse(predict_logistic_test>0.5,1,0)
cf_table_test <- table(predict_logistic_1,test_copy$Survived)
cf_table_test
##
## predict_logistic_1 0 1
## 0 262 0
## 1 106 50
acc_test <- sum(diag(cf_table_test))/sum(cf_table_test)
acc_test
## [1] 0.7464115
sens_test <- cf_table_test[1]/(cf_table_test[1] + cf_table_test[2])
sens_test
## [1] 0.7119565
spec_test <- cf_table_test[4]/(cf_table_test[3] + cf_table_test[4])
spec_test
## [1] 1
ppvtest <- cf_table_test[1]/(cf_table_test[1] + cf_table_test[3])
ppvtest

## [1] 1
```

```
npvtest <- cf_table_test[4]/(cf_table_test[2] + cf_table_test[4])
npvtest
## [1] 0.3205128

error_test <- mean(predict_logistic_1!=test_copy$Survived)
acc_test <- 1 - error_test
acc_test

## [1] 0.7464115
roc_test <- roc(test_copy$Survived,predict_logistic_test)
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

plot.roc(roc_test, col=par("fg"),plot=TRUE,print.auc = FALSE, legacy.axes = TRUE, asp =NA)
roc_test

##
## Call:
## roc.default(response = test_copy$Survived, predictor = predict_logistic_test)
##
## Data: predict_logistic_test in 368 controls (test_copy$Survived 0) < 50 cases (test_copy$Surv
ived 1).
## Area under the curve: 0.9743

plot.roc(smooth(roc_test), col="blue", add = TRUE,plot=TRUE,print.auc = TRUE, legacy.axes =
TRUE
, asp =NA)
legend("bottomright", legend=c("Empirical", "Test"),
col=c(par("fg"), "blue"), lwd=2)
```



Based on what we have seen, it is evident that Logistic Regression excels in making necessary classifications. Next, we will assess the effectiveness of Neural Networks using the same dataset.

```
set.seed(777)
control <- trainControl(method = "repeatedcv", number = 10, repeats = 10)
logistic_bayesian <- train(Survived ~
FamilySize+CabinPos+Deck+Pclass+Sex+Age+SibSp+Parch+Fare+E
mbarked+Title+Mother+Child, data = train, method = "bayesglm", trControl = control, family =
bin
omial(link = "logit"))
acc_bayesian <- paste0(round(max(logistic_bayesian$results$Accuracy),3)*100,'%')
cat('prediction accucary with bayesian logistic regression is ',acc_bayesian,'.\n')

## prediction accucary with bayesian logistic regression is 82.7% .

grid <- expand.grid(size = 1, decay = 0.01)
neuralnet <- train(Survived ~
FamilySize+CabinPos+Deck+Pclass+Sex+Age+SibSp+Parch+Fare+Embarked+
```

```
Title+Mother+Child, data = train, method = "nnet", trControl = control,maxit = 1000, trace = FAL
SE, tuneGrid = grid )
acc_nnet <- paste0(round(max(neuralnet$results$Accuracy),3)*100,'%')
cat('prediction accucary of neural net is ',acc_nnet,'.\n')

## prediction accucary of neural net is 82.9% .
```

"Using a basic Logistic Regression model, we were able to achieve about 77% accuracy. While Neural Networks achieved around 83% accuracy, Bayesian Logistic Regression, another form of Logistic Regression, was also successful in reaching the same accuracy level. Therefore, based on these results, it is evident that Logistic Regression is more suitable for this dataset compared to Neural Networks."