output: html_document: default pdf_document: default — Title: “Classification” Authors: Gaurang Goel (GXG190015), Anurag Diwate (AXD190004) Date: 02/18/2023

#Classification in general terms Classification is a fundamental problem in machine learning in which the class of a new observation is predicted based on its attributes. Linear models for classification are a type of method that uses a linear function to represent the relationship between features and classes. In other words, these models learn a feature space linear boundary that divides the distinct classes.

#Strengths and Weaknesses of Classification One of the strengths of linear models for categorization is their ease of interpretation. They are simple to understand and depict, making them an excellent tool for exploratory data analysis. Linear models are also capable of dealing with high-dimensional data and scaling effectively to big datasets. They can also provide probabilistic forecasts, which may help in decision making.

Linear models, on the other hand, have some restrictions. They may not perform well when the feature-class connection is not linear, and they may fail to capture complicated relationships between features. Moreover, linear models are sensitive to outliers and may struggle with unbalanced classes. Nevertheless, these flaws are frequently overcome by employing more complex approaches such as feature engineering, regularization, or ensemble methods.

#Data Source https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud (https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)

```
#importing libraries
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.4.1     ✓ purrr   1.0.1
## ✓ tibble  3.1.8     ✓ dplyr   1.1.0
## ✓ tidyr   1.3.0     ✓ stringr 1.5.0
## ✓ readr   2.1.4     ✓ forcats 1.0.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
# Importing dataset
data <- read.csv("e:/creditcard.csv", header = TRUE)

#attaching data
attach(data)

#Part A: Dividing the data into 80/20 train/test
set.seed(123)
trainIndex <- createDataPartition(data$Class, p = 0.8, list = FALSE)
train <- data[trainIndex, ]
test <- data[-trainIndex, ]

#Part B: Data Exploration
#names() method returns the names of the headers in the dataset
names(train)
```

```
##  [1] "Time"   "V1"     "V2"     "V3"     "V4"     "V5"     "V6"     "V7"
##  [9] "V8"     "V9"     "V10"    "V11"    "V12"    "V13"    "V14"    "V15"
## [17] "V16"    "V17"    "V18"    "V19"    "V20"    "V21"    "V22"    "V23"
## [25] "V24"    "V25"    "V26"    "V27"    "V28"    "Amount" "Class"
```

```
#nrow() method returns the number of rows in the dataset
nrow(train)
```

```
## [1] 227846
```

```
#nrow() method returns the number of columns in the dataset
ncol(train)
```

```
## [1] 31
```

```
#The summary() method provides an overview of each variable's distribution in the dataset.
summary(train)
```

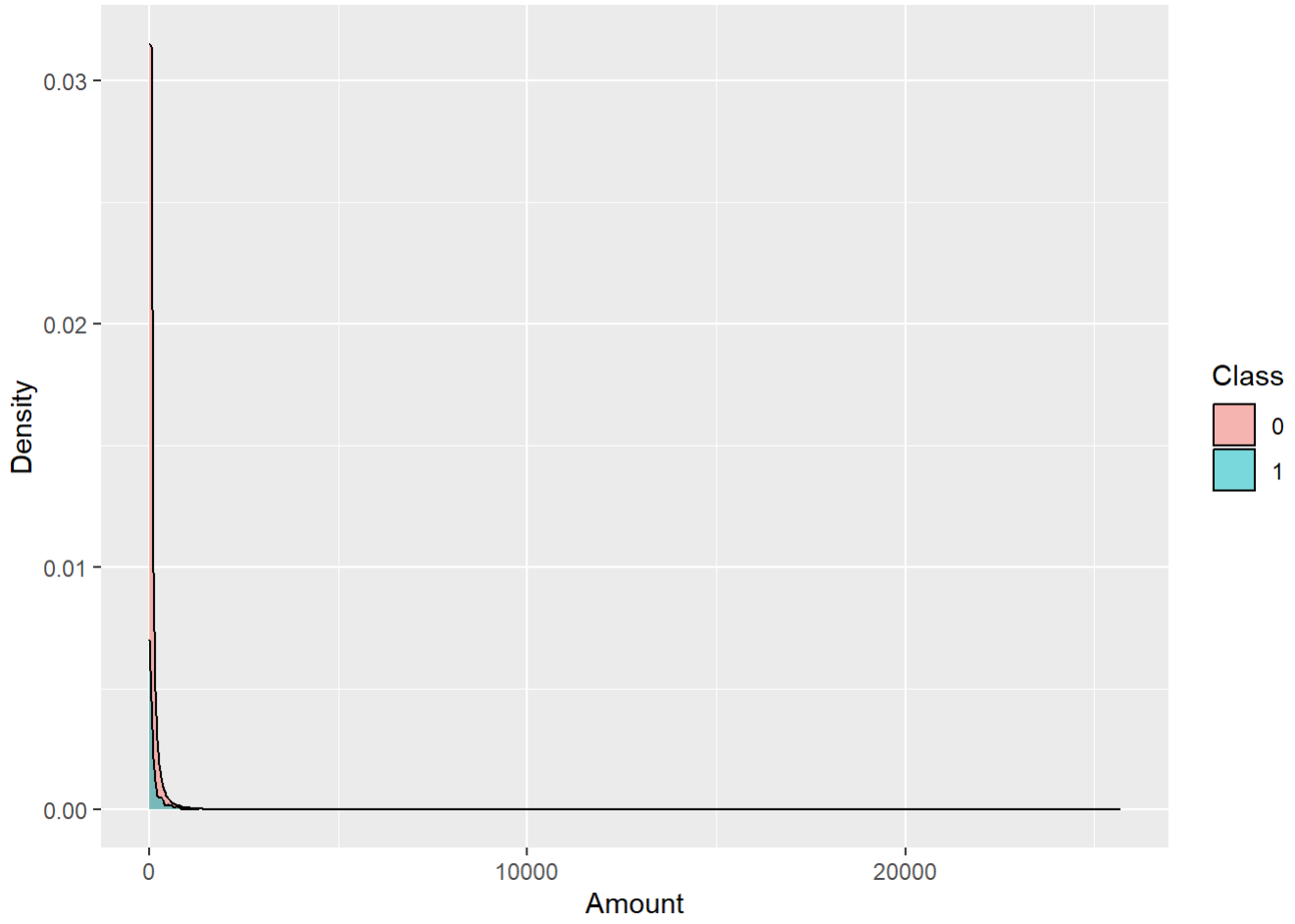
```
##       Time              V1                   V2                   V3
##  Min.   :     0   Min.   :-56.40751   Min.   :-72.71573   Min.   :-48.32559
##  1st Qu.: 54245   1st Qu.: -0.92091   1st Qu.: -0.59684   1st Qu.: -0.89049
##  Median : 84788   Median :  0.01692   Median :  0.06659   Median :  0.17970
##  Mean   : 94865   Mean   : -0.00086   Mean   :  0.00174   Mean   :  0.00027
##  3rd Qu.:139370   3rd Qu.:  1.31581   3rd Qu.:  0.80442   3rd Qu.:  1.02775
##  Max.   :172792   Max.   :  2.45189   Max.   : 22.05773   Max.   :  9.38256
##        V4                  V5                   V6
##  Min.   :-5.683171   Min.   :-113.74331   Min.   :-26.16051
##  1st Qu.:-0.850256   1st Qu.:  -0.69242   1st Qu.: -0.76871
##  Median :-0.022409   Median :  -0.05594   Median : -0.27562
##  Mean   :-0.002627   Mean   :  -0.00115   Mean   : -0.00063
##  3rd Qu.: 0.739540   3rd Qu.:   0.60994   3rd Qu.:  0.39724
##  Max.   :16.875344   Max.   :  34.80167   Max.   : 73.30163
##        V7                  V8                   V9
##  Min.   :-43.55724   Min.   :-73.21672   Min.   :-13.434066
##  1st Qu.: -0.55464   1st Qu.: -0.20861   1st Qu.: -0.642508
##  Median :  0.04025   Median :  0.02212   Median : -0.051192
##  Mean   : -0.00100   Mean   : -0.00195   Mean   :  0.000182
##  3rd Qu.:  0.57005   3rd Qu.:  0.32705   3rd Qu.:  0.597453
##  Max.   :120.58949   Max.   : 20.00721   Max.   : 15.594995
##       V10                 V11                  V12
##  Min.   :-24.588262   Min.   :-4.797473   Min.   :-18.683715
##  1st Qu.: -0.536870   1st Qu.:-0.765546   1st Qu.: -0.404669
##  Median : -0.092855   Median :-0.035046   Median :  0.140933
##  Mean   : -0.000605   Mean   :-0.001782   Mean   : -0.000091
##  3rd Qu.:  0.454017   3rd Qu.: 0.739648   3rd Qu.:  0.617885
##  Max.   : 23.745136   Max.   :12.018913   Max.   :  7.848392
##       V13                 V14                  V15
##  Min.   :-5.791881   Min.   :-19.214325   Min.   :-4.498945
##  1st Qu.:-0.647447   1st Qu.: -0.426030   1st Qu.:-0.583692
##  Median :-0.012773   Median :  0.051201   Median : 0.047257
##  Mean   : 0.001049   Mean   : -0.000157   Mean   :-0.000627
##  3rd Qu.: 0.663943   3rd Qu.:  0.492691   3rd Qu.: 0.649871
##  Max.   : 7.126883   Max.   : 10.526766   Max.   : 8.877742
##       V16                 V17                  V18
##  Min.   :-13.563273   Min.   :-25.16280   Min.   :-9.498746
##  1st Qu.: -0.468480   1st Qu.: -0.48332   1st Qu.:-0.499053
##  Median :  0.065990   Median : -0.06591   Median :-0.004552
##  Mean   : -0.000146   Mean   : -0.00046   Mean   :-0.000341
##  3rd Qu.:  0.522553   3rd Qu.:  0.39914   3rd Qu.: 0.499546
##  Max.   : 17.315112   Max.   :  9.25353   Max.   : 5.041069
##       V19                 V20                 V21
##  Min.   :-7.213527   Min.   :-54.49772   Min.   :-34.83038
##  1st Qu.:-0.456252   1st Qu.: -0.21134   1st Qu.: -0.22855
##  Median : 0.002498   Median : -0.06267   Median : -0.02982
##  Mean   :-0.000934   Mean   : -0.00056   Mean   : -0.00103
##  3rd Qu.: 0.456781   3rd Qu.:  0.13244   3rd Qu.:  0.18611
##  Max.   : 5.591971   Max.   : 39.42090   Max.   : 27.20284
##       V22                 V23                  V24
##  Min.   :-10.933144   Min.   :-44.80774   Min.   :-2.83663
##  1st Qu.: -0.542962   1st Qu.: -0.16202   1st Qu.:-0.35459
```

```
##  Median :  0.005536   Median : -0.01118   Median : 0.04074
##  Mean   : -0.000237   Mean   :  0.00018   Mean   :-0.00009
##  3rd Qu.:  0.529175   3rd Qu.:  0.14761   3rd Qu.: 0.44074
##  Max.   : 10.503090   Max.   : 22.08354   Max.   : 4.58455
##       V25                 V26                 V27
##  Min.   :-10.295397   Min.   :-2.604551   Min.   :-22.565679
##  1st Qu.: -0.316603   1st Qu.:-0.326462   1st Qu.: -0.070855
##  Median :  0.017095   Median :-0.051426   Median :  0.001261
##  Mean   :  0.000646   Mean   : 0.000532   Mean   : -0.000043
##  3rd Qu.:  0.351245   3rd Qu.: 0.241546   3rd Qu.:  0.091068
##  Max.   :  7.519589   Max.   : 3.517346   Max.   : 31.612198
##       V28                Amount             Class
##  Min.   :-15.43008   Min.   :    0.00   Min.   :0.00000
##  1st Qu.: -0.05293   1st Qu.:    5.60   1st Qu.:0.00000
##  Median :  0.01128   Median :   22.00   Median :0.00000
##  Mean   :  0.00053   Mean   :   88.16   Mean   :0.00169
##  3rd Qu.:  0.07846   3rd Qu.:   77.05   3rd Qu.:0.00000
##  Max.   : 33.84781   Max.   :25691.16   Max.   :1.00000
```

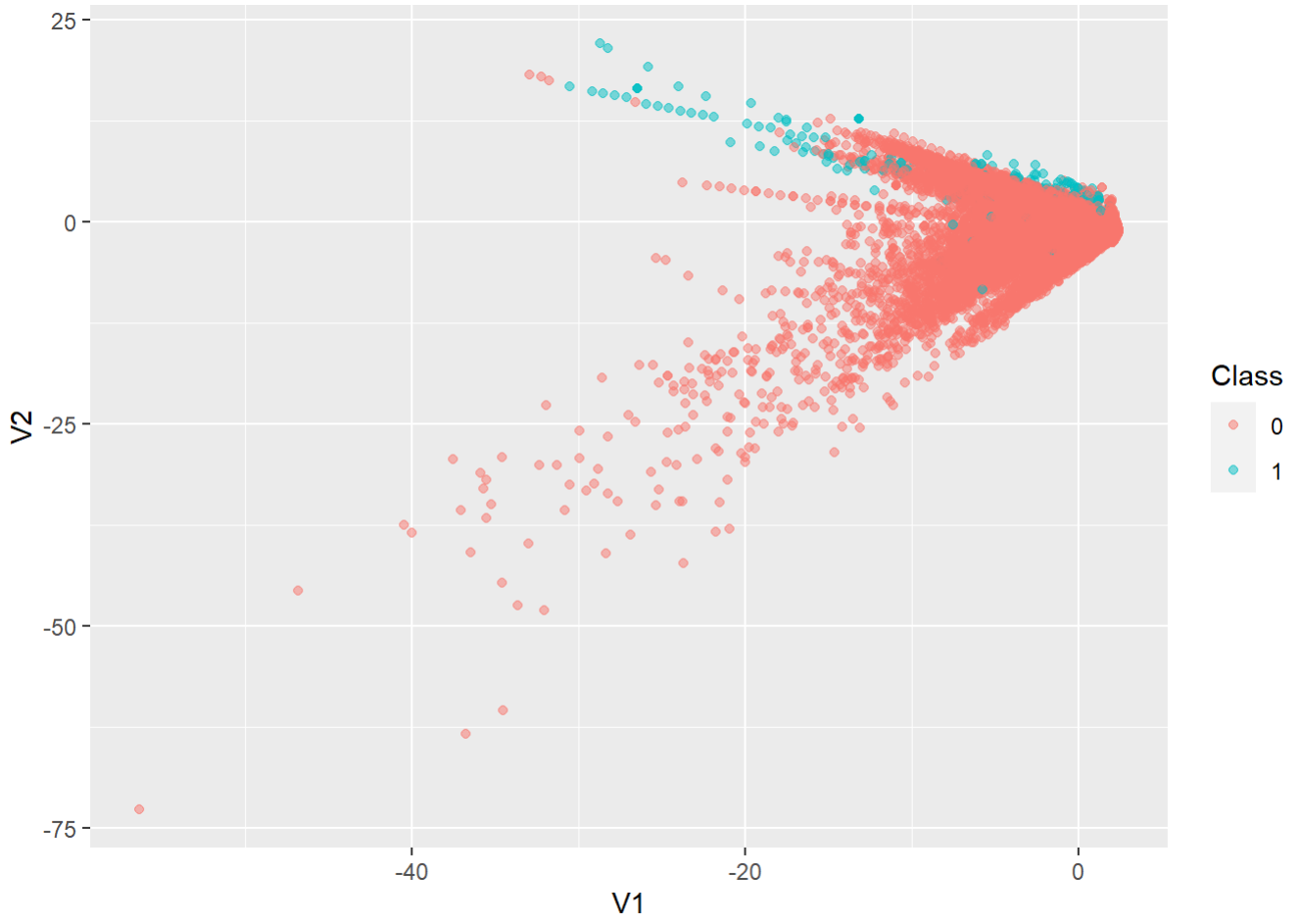
```
#The str() method displays the dataset's structure, including variable data types.
str(train)
```

```
## 'data.frame':    227846 obs. of  31 variables:
##  $ Time  : num  0 0 1 1 4 7 7 9 10 10 ...
##  $ V1    : num  -1.36 1.192 -1.358 -0.966 1.23 ...
##  $ V2    : num  -0.0728 0.2662 -1.3402 -0.1852 0.141 ...
##  $ V3    : num  2.5363 0.1665 1.7732 1.793 0.0454 ...
##  $ V4    : num  1.378 0.448 0.38 -0.863 1.203 ...
##  $ V5    : num  -0.3383 0.06 -0.5032 -0.0103 0.1919 ...
##  $ V6    : num  0.4624 -0.0824 1.8005 1.2472 0.2727 ...
##  $ V7    : num  0.2396 -0.0788 0.79146 0.23761 -0.00516 ...
##  $ V8    : num  0.0987 0.0851 0.2477 0.3774 0.0812 ...
##  $ V9    : num  0.364 -0.255 -1.515 -1.387 0.465 ...
##  $ V10   : num  0.0908 -0.167 0.2076 -0.055 -0.0993 ...
##  $ V11   : num  -0.552 1.613 0.625 -0.226 -1.417 ...
##  $ V12   : num  -0.6178 1.0652 0.0661 0.1782 -0.1538 ...
##  $ V13   : num  -0.991 0.489 0.717 0.508 -0.751 ...
##  $ V14   : num  -0.311 -0.144 -0.166 -0.288 0.167 ...
##  $ V15   : num  1.4682 0.6356 2.3459 -0.6314 0.0501 ...
##  $ V16   : num  -0.47 0.464 -2.89 -1.06 -0.444 ...
##  $ V17   : num  0.20797 -0.1148 1.10997 -0.68409 0.00282 ...
##  $ V18   : num  0.0258 -0.1834 -0.1214 1.9658 -0.612 ...
##  $ V19   : num  0.404 -0.1458 -2.2619 -1.2326 -0.0456 ...
##  $ V20   : num  0.2514 -0.0691 0.525 -0.208 -0.2196 ...
##  $ V21   : num  -0.0183 -0.2258 0.248 -0.1083 -0.1677 ...
##  $ V22   : num  0.27784 -0.63867 0.77168 0.00527 -0.27071 ...
##  $ V23   : num  -0.11 0.101 0.909 -0.19 -0.154 ...
##  $ V24   : num  0.0669 -0.3398 -0.6893 -1.1756 -0.7801 ...
##  $ V25   : num  0.129 0.167 -0.328 0.647 0.75 ...
##  $ V26   : num  -0.189 0.126 -0.139 -0.222 -0.257 ...
##  $ V27   : num  0.13356 -0.00898 -0.05535 0.06272 0.03451 ...
##  $ V28   : num  -0.02105 0.01472 -0.05975 0.06146 0.00517 ...
##  $ Amount: num  149.62 2.69 378.66 123.5 4.99 ...
##  $ Class : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
#Part C: Graphs
# Density plot
ggplot(train, aes(Amount, fill = factor(Class))) +
  geom_density(alpha = 0.5) +
  labs(x = "Amount", y = "Density", fill = "Class")
```




```
# Scatter plot of V1 and V2
ggplot(train, aes(V1, V2, color = factor(Class))) +
  geom_point(alpha = 0.5) +
  labs(x = "V1", y = "V2", color = "Class")
```




```
#Part D: Logistical Regression Model
log_model <- glm(Class ~ ., data = train, family = "binomial")
summary(log_model)
```

```
##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6403  -0.0285  -0.0189  -0.0122   4.3118
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.470e+00  2.807e-01 -30.177  < 2e-16 ***
## Time        -3.211e-06  2.552e-06  -1.258  0.20839
## V1           1.233e-01  4.839e-02   2.547  0.01086 *
## V2          -1.550e-02  6.446e-02  -0.241  0.80994
## V3           2.654e-03  6.100e-02   0.044  0.96530
## V4           6.894e-01  7.712e-02   8.939  < 2e-16 ***
## V5           5.457e-02  7.465e-02   0.731  0.46475
## V6          -1.392e-01  8.717e-02  -1.597  0.11029
## V7          -1.457e-01  7.483e-02  -1.947  0.05153 .
## V8          -1.725e-01  3.521e-02  -4.901 9.55e-07 ***
## V9          -3.556e-01  1.168e-01  -3.045  0.00233 **
## V10         -7.345e-01  1.019e-01  -7.211 5.56e-13 ***
## V11         -1.823e-02  9.165e-02  -0.199  0.84231
## V12          1.320e-01  9.646e-02   1.368  0.17117
## V13         -2.744e-01  9.289e-02  -2.954  0.00314 **
## V14         -5.992e-01  6.869e-02  -8.724  < 2e-16 ***
## V15         -1.288e-01  9.719e-02  -1.325  0.18508
## V16         -2.016e-01  1.305e-01  -1.545  0.12242
## V17          6.154e-02  7.624e-02   0.807  0.41957
## V18         -4.317e-02  1.358e-01  -0.318  0.75062
## V19          8.628e-02  1.058e-01   0.816  0.41478
## V20         -4.668e-01  9.268e-02  -5.037 4.74e-07 ***
## V21          3.860e-01  6.701e-02   5.760 8.40e-09 ***
## V22          6.102e-01  1.506e-01   4.051 5.11e-05 ***
## V23         -1.279e-01  6.830e-02  -1.872  0.06118 .
## V24          1.507e-01  1.675e-01   0.900  0.36821
## V25         -2.805e-02  1.491e-01  -0.188  0.85073
## V26          6.545e-02  2.126e-01   0.308  0.75818
## V27         -8.302e-01  1.360e-01  -6.105 1.03e-09 ***
## V28         -2.883e-01  1.022e-01  -2.822  0.00477 **
## Amount       9.044e-04  4.506e-04   2.007  0.04472 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5684.4  on 227845  degrees of freedom
## Residual deviance: 1754.3  on 227815  degrees of freedom
## AIC: 1816.3
##
## Number of Fisher Scoring iterations: 11
```

```
#The summary contains metrics for evaluating the model, such as deviation, AIC, and likelihood r
atio test. The AIC assesses the trade-off between model complexity and goodness of fit, whereas
 the deviation reflects the gap between anticipated and actual values. The likelihood ratio test
compares the current model to a null model with no predictors and returns a p-value showing whet
her the current model is significantly better than the null model. Overall, the model summary is
an excellent tool for assessing the logistic regression model and suggesting areas for improveme
nt.

#Part E: Naïve Bayes Model
library(e1071)
nb_model <- naiveBayes(Class ~ ., data = train)
nb_model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##           0           1
## 0.998310262 0.001689738
##
## Conditional probabilities:
##    Time
## Y       [,1]     [,2]
##   0 94890.48 47507.65
##   1 79860.10 48184.32
##
##    V1
## Y          [,1]     [,2]
##   0  0.00740885 1.932901
##   1 -4.88639316 7.128115
##
##    V2
## Y           [,1]     [,2]
##   0 -0.004798741 1.641011
##   1  3.867319951 4.367791
##
##    V3
## Y          [,1]     [,2]
##   0  0.01256929 1.460488
##   1 -7.26811935 7.325462
##
##    V4
## Y          [,1]     [,2]
##   0 -0.01046061 1.399779
##   1  4.62538083 2.868327
##
##    V5
## Y           [,1]     [,2]
##   0  0.004503653 1.363200
##   1 -3.338808334 5.580357
##
##    V6
## Y           [,1]     [,2]
##   0  0.001747588 1.332378
##   1 -1.407842835 1.870314
##
##    V7
## Y           [,1]     [,2]
##   0  0.008840788 1.187781
##   1 -5.813152477 7.584867
##
```

```
##    V8
## Y           [,1]     [,2]
##   0 -0.002771184 1.176258
##   1  0.481534185 7.190409
##
##    V9
## Y           [,1]     [,2]
##   0  0.004688844 1.091160
##   1 -2.662470701 2.596888
##
##    V10
## Y           [,1]     [,2]
##   0  0.009246005 1.044319
##   1 -5.820703369 5.102794
##
##    V11
## Y           [,1]     [,2]
##   0 -0.008330652 1.003939
##   1  3.867088122 2.643666
##
##    V12
## Y         [,1]      [,2]
##   0  0.0106774 0.9477549
##   1 -6.3618687 4.6697379
##
##    V13
## Y           [,1]      [,2]
##   0  0.001223421 0.9940628
##   1 -0.102097938 1.0828370
##
##    V14
## Y          [,1]      [,2]
##   0  0.01191161 0.8977033
##   1 -7.13024005 4.3136176
##
##    V15
## Y            [,1]      [,2]
##   0 -0.0004786208 0.9154533
##   1 -0.0884254463 1.0552750
##
##    V16
## Y           [,1]      [,2]
##   0  0.006911794 0.8452431
##   1 -4.170237462 3.8995826
##
##    V17
## Y          [,1]      [,2]
##   0  0.01095445 0.7509774
##   1 -6.74433901 7.1226709
##
##    V18
## Y           [,1]      [,2]
```

```
##   0  0.003518636 0.8251347
##   1 -2.280731156 2.9367788
##
##    V19
## Y           [,1]      [,2]
##   0 -0.002026151 0.8107966
##   1  0.644302710 1.5437516
##
##    V20
## Y           [,1]      [,2]
##   0 -0.001244412 0.7792088
##   1  0.402485116 1.3860709
##
##    V21
## Y           [,1]      [,2]
##   0 -0.002178711 0.7210648
##   1  0.680381563 4.1045657
##
##    V22
## Y            [,1]      [,2]
##   0 -0.0002308207 0.7238815
##   1 -0.0038244612 1.5596719
##
##    V23
## Y            [,1]      [,2]
##   0  0.0002707705 0.6185419
##   1 -0.0560149139 1.7222627
##
##    V24
## Y            [,1]      [,2]
##   0  7.543347e-05 0.6061957
##   1 -9.774902e-02 0.5212982
##
##    V25
## Y           [,1]      [,2]
##   0 0.0005272097 0.5214561
##   1 0.0708755337 0.8202603
##
##    V26
## Y           [,1]      [,2]
##   0 0.0004257687 0.4825454
##   1 0.0631377084 0.4887205
##
##    V27
## Y            [,1]     [,2]
##   0 -0.0002412351 0.402310
##   1  0.1169885093 1.486415
##
##    V28
## Y           [,1]      [,2]
##   0 0.0003755148 0.3368746
##   1 0.0916224446 0.5324919
```

```
##
##    Amount
## Y        [,1]     [,2]
##   0  88.11217 252.6090
##   1 117.78574 244.7329
```

```
#Based on the training data, the Naive Bayes model learns the conditional probability of each ch
aracteristic for each class (fraudulent or non-fraudulent transaction). Specifically, the model
 evaluated the likelihood of a particular feature value appearing in a fraudulent or non-fraudul
ent transaction and utilized these probabilities to categorize fresh transactions in the test da
ta. Naive Bayes is a probabilistic model that implies independence between characteristics given
a class, resulting in a simpler model and faster training times than other models. This assumpti
on, however, may not always hold true in practice, and the model may suffer from underfitting if
crucial relationships are not recorded.

#Part F: Predict and Evaluate
# Logistic regression predictions
log_pred <- predict(log_model, newdata = test, type = "response")
log_pred_class <- ifelse(log_pred > 0.5, 1, 0)

# Naïve Bayes predictions
nb_pred <- predict(nb_model, newdata = test)
nb_pred_class <- as.numeric(as.character(nb_pred)) - 1

# Logistic Regression Metrics
log_confusion <- confusionMatrix(as.factor(log_pred_class), as.factor(test$Class))
log_accuracy <- log_confusion$overall[1]
log_sensitivity <- log_confusion$byClass[1]
log_specificity <- log_confusion$byClass[2]
log_precision <- log_confusion$byClass[3]
log_f1_score <- log_confusion$byClass[4]

# Naive Bayes Metrics
nb_pred_class <- factor(nb_pred_class, levels = c("0", "1"))
nb_confusion <- confusionMatrix(as.factor(nb_pred_class), as.factor(test$Class))
nb_accuracy <- nb_confusion$overall[1]
nb_sensitivity <- nb_confusion$byClass[1]
nb_specificity <- nb_confusion$byClass[2]
nb_precision <- nb_confusion$byClass[3]
nb_f1_score <- nb_confusion$byClass[4]
if (is.nan(nb_f1_score)) {
  nb_f1_score <- 0
}

# Print metrics
print(paste0("Logistic Regression Accuracy: ", round(log_accuracy, 3)))
```

```
## [1] "Logistic Regression Accuracy: 0.999"
```

```
print(paste0("Logistic Regression Sensitivity: ", round(log_sensitivity, 3)))
```

```
## [1] "Logistic Regression Sensitivity: 1"
```

```
print(paste0("Logistic Regression Specificity: ", round(log_specificity, 3)))
```

```
## [1] "Logistic Regression Specificity: 0.579"
```

```
print(paste0("Logistic Regression Precision: ", round(log_precision, 3)))
```

```
## [1] "Logistic Regression Precision: 0.999"
```

```
print(paste0("Logistic Regression F1 Score: ", round(log_f1_score, 3)))
```

```
## [1] "Logistic Regression F1 Score: 0.861"
```

```
print(paste0("Naive Bayes Accuracy: ", round(nb_accuracy, 3)))
```

```
## [1] "Naive Bayes Accuracy: 0.932"
```

```
print(paste0("Naive Bayes Sensitivity: ", round(nb_sensitivity, 3)))
```

```
## [1] "Naive Bayes Sensitivity: 1"
```

```
print(paste0("Naive Bayes Specificity: ", round(nb_specificity, 3)))
```

```
## [1] "Naive Bayes Specificity: 0"
```

```
print(paste0("Naive Bayes Precision: ", round(nb_precision, 3)))
```

```
## [1] "Naive Bayes Precision: 0.932"
```

```
print(paste0("Naive Bayes F1 Score: ", round(nb_f1_score, 3)))
```

```
## [1] "Naive Bayes F1 Score: 0"
```

#Part G: Strengths and Weaknesses of Naïve Bayes and Logistic Regression In machine learning, two common algorithms for classification problems are Nave Bayes and Logistic Regression.

Nave Bayes is a probabilistic method that performs well with large datasets and requires less training data than other algorithms. It is simple and efficient, making it an excellent choice for real-time applications. Nevertheless, in real-world datasets, Nave Bayes presupposes that all characteristics are independent of one another. When dealing with associated characteristics, this might lead to erroneous forecasts.

Logistic Regression is a linear approach that is effective for binary classification issues. It is simple to use and understand, making it a popular choice for both beginners and professionals. Logistic Regression also generates probabilistic outputs that can be used to make decisions. Nevertheless, when dealing with complicated datasets, Logistic Regression presupposes a linear relationship between the independent factors and the log-odds of the dependent variable, which can lead to underfitting. Moreover, Logistic Regression might struggle with multicollinear datasets, which include linked independent variables.

#Part H: Benefits and Drawbacks of the classification metrics used Accuracy: Benefit: It is simple and easy to grasp, and it gauges the overall accuracy of the model predictions. Drawback: If the classes are uneven, great accuracy can be obtained simply by forecasting the majority class.

#Sensitivity: Benefit: Measures the model's ability to properly identify positive situations, which is useful in applications where the cost of false negatives is large (e.g. medical diagnoses) Drawback: May be less essential in cases where the cost of false positives is considerable.

#Specificity: Benefit: Indicates the model's ability to properly identify negative situations, which is useful in applications where the cost of false positives is large (e.g. fraud detection) Drawback: May be less essential in applications where the cost of false negatives is considerable.

#Precision: Benefit: Determines the fraction of real positive predictions among all positive predictions, which is useful in situations where the cost of false positives is large (e.g. spam email detection) Drawback: May be less essential in applications where the cost of false negatives is considerable.

#F1 Score: Benefit: Combines accuracy and recall into a single statistic, which can be advantageous for optimizing for both metrics simultaneously. Drawback: Because it is a weighted average of the two measurements, it may be less interpretable
than the individual metrics.