**Assignment -3**

**Stat Learning: Classification Lab**

**Name: Sowmya Patlolla**

install.packages("ISLR2")

library (ISLR2)

names (Smarket)

dim (Smarket)

Summary (Smarket)

ls ()

**A screenshot of a computer

Description automatically generated**

View (Smarket)

pairs (Smarket)

**A screenshot of a computer

Description automatically generated**

cor(Smarket)

cor(Smarket[, -9])

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Logistic Regression**

**glm.fits <- glm(**

**Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,**

**data = Smarket, family = binomial**

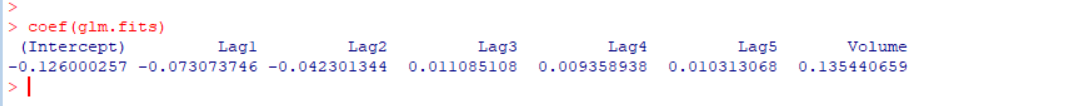
**)**

**summary(glm.fits)**

**A screenshot of a computer

Description automatically generated**

**coef(glm.fits)**

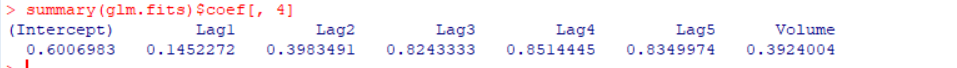
****

**summary(glm.fits)$coef**

**A computer error code with numbers

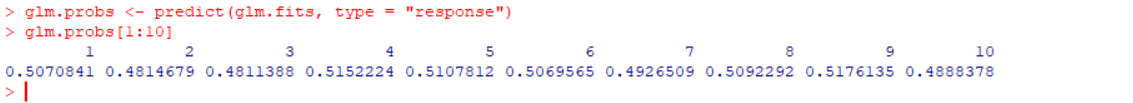
Description automatically generated with medium confidence**

summary(glm.fits)$coef[, 4]

****

glm.probs <- predict(glm.fits, type = "response")

glm.probs[1:10]

****

**contrasts (Direction)**

**glm.pred <- rep("Down", 1250)**

**glm.pred[glm.probs > .5] = "Up"**

**table (glm.pred, Direction)**

**(507 + 145) / 1250**

**mean (glm.pred == Direction)**

**train <- (Year < 2005)**

**Smarket.2005 <- Smarket[!train, ]**

**dim (Smarket.2005)**

**A red and white text

Description automatically generated with medium confidence**

**Direction.2005 <- Direction [! train]**

**glm.fits <- glm(**

**Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,**

**data = Smarket, family = binomial, subset = train**

**)**

**glm.probs <- predict(glm.fits, Smarket.2005,**

**type = "response")**

**A white background with red text

Description automatically generated**

**glm.pred <- rep("Down", 252)**

**glm.pred[glm.probs > .5] <- "Up"**

**table (glm.pred, Direction.2005)**

**mean (glm.pred == Direction.2005)**

**mean (glm.pred != Direction.2005)**

**A computer code with red and blue text

Description automatically generated**

**glm.fits <- glm(Direction ~ Lag1 + Lag2, data = Smarket, family = binomial, subset = train)**

**glm.probs <- predict(glm.fits, Smarket.2005, type = "response")**

**glm.pred <- rep("Down", 252)**

**glm.pred[glm.probs > .5] <- "Up"**

**table (glm.pred, Direction.2005)**

**A close-up of a computer screen

Description automatically generated**

**mean (glm.pred == Direction.2005)**

**106 / (106 + 76)**

**predict (glm.fits,**

**newdata =**

**data.frame(Lag1 = c(1.2, 1.5), Lag2 = c(1.1, -0.8)),**

**type = "response"**

**A white background with red text

Description automatically generated**

**TYPICAL USE:**

The one of the uses of the logistic regression is to estimate the binary event’s probability which is either zero or one, the classification problems can also be identified by the logistic regression. some of the examples in which the logistic regression is helpful is email spam detection, finding out credit card transaction is genuine or fraud, and logistic regression can also be used in making predictions like person buying a product or not, or a student finishing his course within due date or not.

**OBSERVATIONS ON THE OUTPUT:**

In the current scenario the lowest one is the p-value for lag1. If the market return was good yesterday, then there is a less chance that it will grow today, as per the negative coefficient of the predictor. Because even with a value of 0.15 the p value is still large. And there is no relationship between the direction and lag 1.

**MOST SIGNIFICANT FINDING OF THE EXERCISE:**

Whichever data set is present it needs to be supplied to the method called predict (), if the data set is not supplied then the resultant values obtained for the training data will be used to fit the logistic regression model.

**LINEAR DISCRIMINANT ANALYSIS**

**library (MASS)**

**lda.fit <- lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)**

**lda.fit**

**A computer screen shot of a code

Description automatically generated**

**plot(lda.fit)**

**A screenshot of a computer

Description automatically generated**

**lda.pred <- predict(lda.fit, Smarket.2005)**

**names(lda.pred)**

**lda.class <- lda.pred$class**

**table (lda.class, Direction.2005)**

**mean (lda.class == Direction.2005)**

**sum (lda.pred$posterior[, 1] >= .5)**

**sum (lda.pred$posterior[, 1] < .5)**

**lda.pred$posterior[1:20, 1]**

**lda.class[1:20]**

**sum (lda.pred$posterior[, 1] > .9)**

**A computer screen shot of a computer code

Description automatically generated**

**TYPICAL USE:**

Data visualization, dimension reduction, and classification all require linear discriminant analysis. The LDA provides clear, reputable, and trustworthy results. It serves to represent group distinctions by dividing two or more classes. Supervised classification problems can be widely identified using the LDA.

**OBSERVATIONS ON THE OUTPUT:**

The LDA result shows that 49.2% of the training observations, or π^1=0.492and π^2=0.508, are correlated to days when the market decreased. It also provides the group means, which LDA employs to calculate k. The average of each predictor within each class makes up the group means. They reveal a tendency for returns from the previous two days to be positive on market down days and a tendency for them to be negative on market up days.

**MOST SIGNIFICANT FINDING OF THE EXERCISE:**

The lda() function is used in order to fit an LDA model, and it is a part of MASS library. The lda decision rule can be created from the linear combination of lag1 and lag2 that is used to create the lda decision rule by the coefficients of linear discriminants.

**QUADRATIC DISCRIMINANT ANALYSIS**

qda.fit <- qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

qda.fit

qda.class <- predict (qda.fit, Smarket.2005) $class

table (qda.class, Direction.2005)

mean (qda.class == Direction.2005)

**A screenshot of a computer code

Description automatically generated**

**TYPICAL USE:**

A discriminant analysis fits the data better than the DLA although the quadratic discriminant analysis has more parameters to determine. The freedom for the covariance matrix gets increased in the QDA which leads to the increase of the parameters because each class will have a unique covariance matrix.

**OBSERVATIONS ON THE OUTPUT:**

The group means are produced using the QDA. The QDA classifier utilizes the quadratic function of the predictors instead of the linear one, hence it does not incorporate the coefficients of the linear discriminants.

**MOST SIGNIFICANT FINDING OF THE EXERCISE:**

With the help of the QDA () function we fit an LDA model. Compared to the linear forms assumed by the LDA and logistic regression, the quadratic from presented by QDA may more correctly capture the underlying relationship. It is better to verify the strategy’s efficacy on a bit larger test set before assuming that this method would always perform better in the market.

**NAIVE BAYES**

**library(e1071)**

**nb.fit <- naiveBayes(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)**

**nb.fit**

**A screenshot of a computer program

Description automatically generated**

**mean (Lag1[train][Direction[train] == "Down"])**

**sd(Lag1[train][Direction[train] == "Down"])**

**A white background with red text

Description automatically generated**

nb.class <- predict (nb.fit, Smarket.2005)

table (nb.class, Direction.2005)

mean (nb.class == Direction.2005)

nb.preds <- predict(nb.fit, Smarket.2005, type = "raw")

nb.preds[1:5, ]

**A screenshot of a computer code

Description automatically generated**

**TYPICAL USE**

The naïve bayes is used to actively predict the class of a test data set which operates quickly. The multi class prediction issues can be resolved using the naïve bayes because it works good enough with them. Even with the less data provided the naïve bayes can perform better than the other models if the concept of independence is true for naïve bayes.

**OBSERVATIONS ON THE OUTPUT**

So, on the provided data, Naïve bayes performs very well, more than 50% of time the naïve bayes makes the predictions correctly. This performs moderately worse than QDA, but much better than LDA.

**MOST SIGNIFICANT FINDING OF THE EXERCISE**

The naïve bayes is implemented using R’s naive Bayes () function. Each quantitative feature is modelled by the default using a gaussian distribution in this simplistic implementation of bayes classifier. Estimates of the likelihood that each observation belongs to a particular class can be generated using the function which is predict ().

**K-Nearest Neighbours**

**library(class)**

**train.X <- cbind(Lag1, Lag2)[train, ]**

**test.X <- cbind(Lag1, Lag2)[!train, ]**

**train.Direction <- Direction[train]**

**set.seed(1)**

**knn.pred <- knn(train.X, test.X, train.Direction, k = 1)**

**table (knn.pred, Direction.2005)**

**A computer code with red text

Description automatically generated**

**knn.pred <- knn(train.X, test.X, train.Direction, k = 3)**

**table (knn.pred, Direction.2005)**

**mean (knn.pred == Direction.2005)**

**dim (Caravan)**

**attach (Caravan)**

**summary (Purchase)**

**348 / 5822**

**A computer screen shot of a computer code

Description automatically generated**

**standardized.X <- scale (Caravan [, -86])**

**var (Caravan [, 1])**

**var (Caravan [, 2])**

**var (standardized.X[, 1])**

**var (standardized.X[, 2])**

**A white background with red text

Description automatically generated**

**test <- 1:1000**

**train.X <- standardized.X[-test, ]**

**test.X <- standardized.X[test, ]**

**train.Y <- Purchase[-test]**

**test.Y <- Purchase[test]**

**set.seed(1)**

**knn.pred <- knn(train.X, test.X, train.Y, k = 1)**

**mean (test.Y != knn.pred)**

**mean (test.Y != "No")**

**table (knn.pred, test.Y)**

**9 / (68 + 9)**

**A screenshot of a computer code

Description automatically generated**

**knn.pred <- knn(train.X, test.X, train.Y, k = 3)**

**table (knn.pred, test.Y)**

**5 / 26**

**knn.pred <- knn(train.X, test.X, train.Y, k = 5)**

**table (knn.pred, test.Y)**

**4 / 15**

**glm.fits <- glm(Purchase ~ ., data = Caravan, family = binomial, subset = -test)**

**A screenshot of a computer code

Description automatically generated**

**glm.probs <- predict(glm.fits, Caravan[test, ],**

**type = "response")**

**glm.pred <- rep("No", 1000)**

**glm.pred[glm.probs > .5] <- "Yes" table(glm.pred, test.Y)**

**glm.pred <- rep("No", 1000)**

**glm.pred[glm.probs > .25] <- "Yes"**

**table (glm.pred, test.Y)**

**11 / (22 + 11)**

**A screenshot of a computer code

Description automatically generated**

**TYPICAL USE**

The applications used in KNN method need not to be understood by humans but must and should have great accuracy, only those applications can be used in KNN method. The measurement of the distance plays the key factor which affects how accurate the obtained predictions are.

**OBSERVATIONS ON THE OUTPUT**

The results are not that great because when k=1 only 50% of the data are correctly predicted, we can observe a slight improvement in the results when we change the value of k which is k=3. After observing the results by changing the value of k we see that after increasing the k value there is no more gain in the results. So, we can conclude that QDA is quite better at producing the best results for this data.

**MOST SIGNIFICANT FINDING OF THE EXERCISE**

One of the functions which is a part of the class library is KNN () function which is used to perform KNN. the observations which are closely related to the test observations are determined by the KNN classifier. The variables scale is very important here. The distance between the variables plays a crucial role as in terms of the KNN classifier will be affected more with the large-scale variables compared to the small-scale variables.

**1.Did your model perform better/worse/same relative to the method (logistic) in part 1? Why do you believe this is the case?**

Our Naive Bayes model's performance in comparison to the part 1 logistic regression model can differ. When the data match this assumption of feature independence, naive Bayes performs well. Naive Bayes can perform similarly, if not better, if the data really do show such independence. Naive Bayes may fall short, nonetheless, in scenarios with intricate feature dependencies. Intricate relationships are accommodated by logistic regression, which does not assume independence. Logistic regression might succeed if these are present. On the other hand, data that agree with Naive Bayes' assumptions may perform similarly because they are straightforward and have a lower chance of overfitting.

**2.Compared with part 2 (LDA) do you feel your method is more interpretable? {This means that your model is easier to relate back to the original values/plots you did in part 1}.**

In section 2, Naive Bayes often provides easier interpretation than LDA. When determining class probabilities based on individual predictors, Naive Bayes uses simple probability. Predictor importance can thus be more easily interpreted as a result. LDA, in comparison, contains intricate linear combinations of predictors, making it harder to relate to part 1's original data and graphs. Although LDA offers insight into data separation, its coefficients might not be as easily understood as Naive Bayes' probability.

**3.What properties or limitations of your dataset (or selected predictors) do you feel most contributed to the shortcomings in model performance (if your models are very low accuracy, be sure to emphasize this response as it may help your score to be able to explain why your efforts didn't generate good\* results).**

Independence Assumption: When feature independence is violated, poor results can occur.

Imbalanced data: Predictions that are not balanced may be biased against minority classes.

Non-Normal Data: Non-normal data may cast doubt on the assumptions made by Naive Bayes.

Feature Engineering: Poor feature selection or preprocessing can reduce model accuracy.

Missing Data: Unaddressed missing values can bias parameter estimates.

Outliers: Extreme values can disrupt mean and variance estimates, impacting model accuracy.