Assignment1

Vishal Arora

10/10/2020

Assignment 1 - Data 622

Data Loading and data visualization

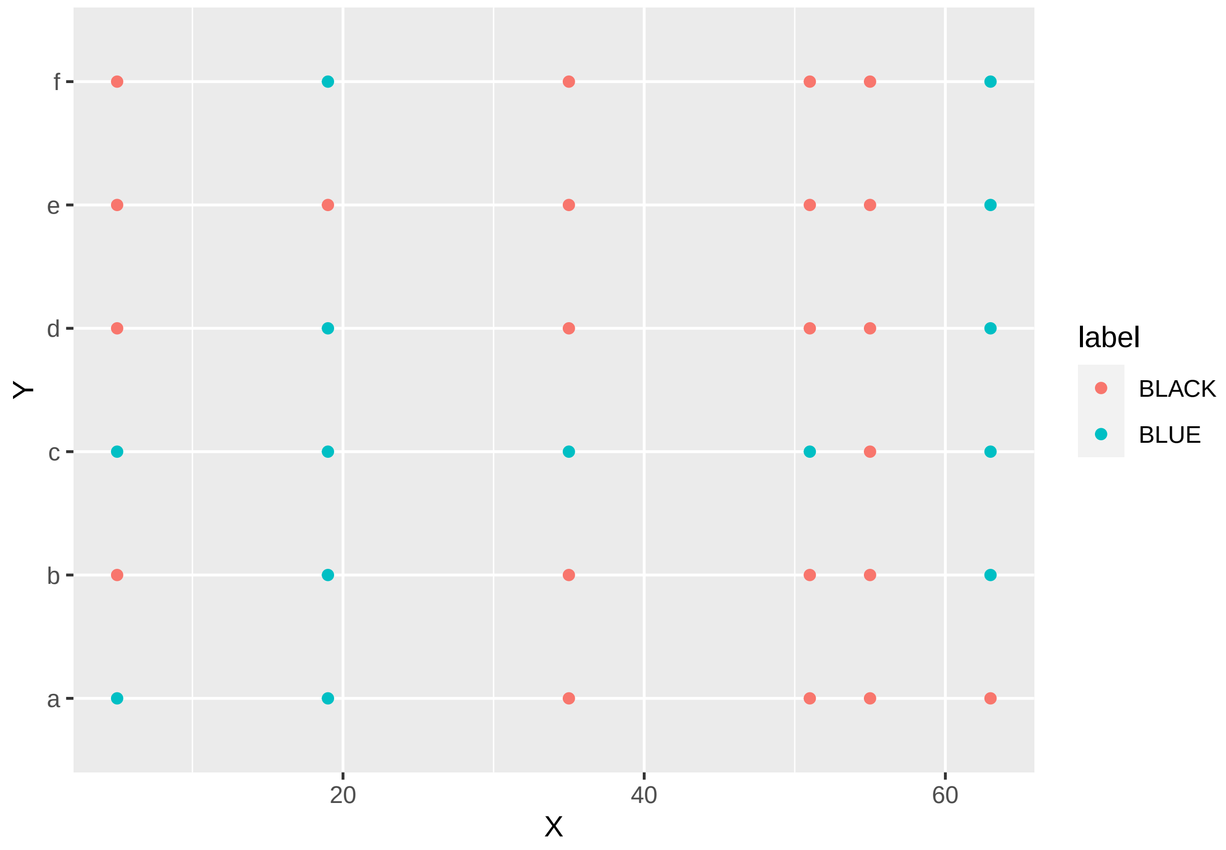
```
df <- read.csv("data_hw1_622.csv", header = TRUE, sep =",")
# checking data structure
str(df)
```

```
## 'data.frame':    36 obs. of  3 variables:
##  $ X    : int  5 5 5 5 5 5 19 19 19 19 ...
##  $ Y    : chr  "a" "b" "c" "d" ...
##  $ label: chr  "BLUE" "BLACK" "BLUE" "BLACK" ...
```

```
# printing the top 5 rows of the data frame.
kable(head(df)) %>%
  kable_styling(bootstrap_options = c("striped","hover","condensed","responsive"),full_width   = F,posi
  row_spec(0, background ="gray")
```

X	Y	label
5	a	BLUE
5	b	BLACK
5	c	BLUE
5	d	BLACK
5	e	BLACK
5	f	BLACK

```
ggplot(df,aes(y=Y,x=X,color=label)) + geom_point()
```



The data has 36 rows and 3 columns , out of which columns 'Y' and 'label' are Character and column 'X' is int, but all the columns are categorical in nature and hence can be converted to factors to be consistent.

Conversion & summarizing data

```
df$X = as.factor(df$X)
df$Y = as.factor(df$Y)
df$label = as.factor(df$label)
#df[sapply(df, is.character)] <- lapply(df[sapply(df, is.character)], as.factor)

# summary statistics of the columns
summary(df)
```

```
##    X      Y        label
##  5 :6   a:6   BLACK:22
##  19:6   b:6   BLUE :14
##  35:6   c:6
##  51:6   d:6
##  55:6   e:6
##  63:6   f:6
```

```
str(df)
```

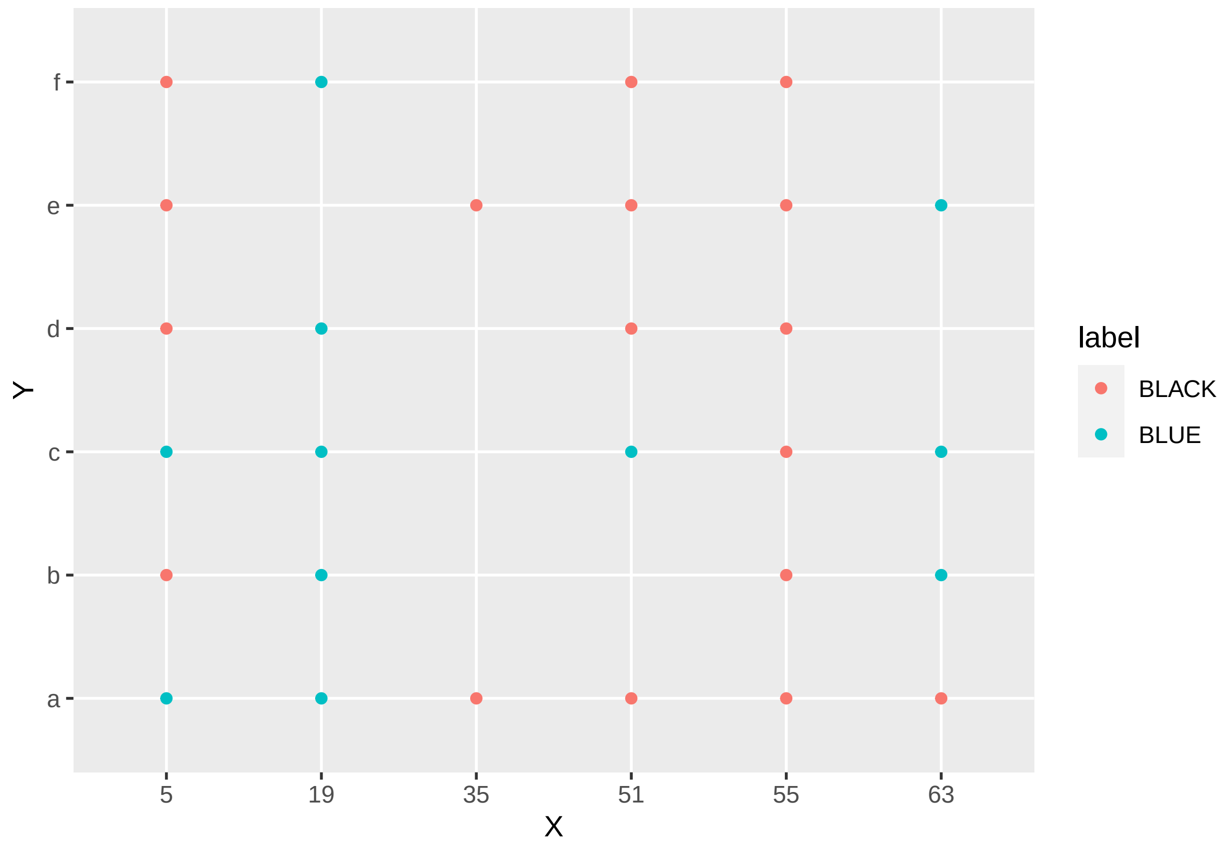
```
## 'data.frame':    36 obs. of  3 variables:
##  $ X    : Factor w/ 6 levels "5","19","35",..: 1 1 1 1 1 1 2 2 2 2 ...
##  $ Y    : Factor w/ 6 levels "a","b","c","d",..: 1 2 3 4 5 6 1 2 3 4 ...
##  $ label: Factor w/ 2 levels "BLACK","BLUE": 2 1 2 1 1 1 2 2 2 2 ...
```

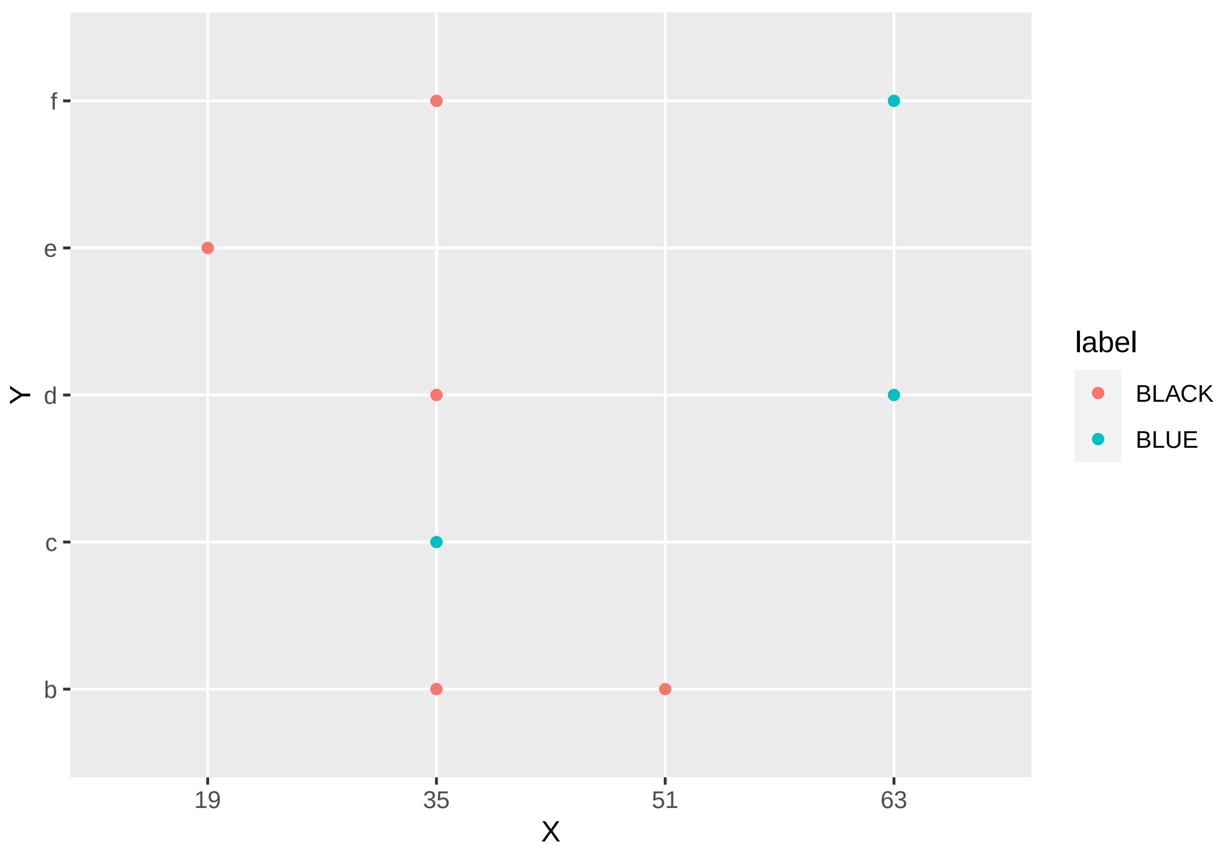
The data is moderately imbalanced BLACK:BLUE ratio is 60:40.

Splitting Data & Models

```
##
##    BLACK     BLUE
## 60.71429 39.28571
```

```
##
## BLACK  BLUE
##  62.5  37.5
```





```r
# Logistic Regression
fit1 <- glm(label ~ ., data = train_set, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(fit1)
```

```
##
## Call:
## glm(formula = label ~ ., family = "binomial", data = train_set)
##
## Deviance Residuals:
##      Min         1Q     Median         3Q        Max
## -1.48230  -0.00005   0.00000   0.00000   1.48230
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.931e-01  1.732e+00  -0.400    0.689
## X19          4.180e+01  2.162e+04   0.002    0.998
## X35         -2.187e+01  3.406e+04  -0.001    0.999
## X51         -1.943e+01  1.007e+04  -0.002    0.998
## X55         -5.931e+01  2.114e+04  -0.003    0.998
## X63          1.386e+00  1.732e+00   0.800    0.423
## Yb           4.627e-15  2.121e+00   0.000    1.000
## Yc           3.982e+01  1.526e+04   0.003    0.998
## Yd          -1.985e+01  1.602e+04  -0.001    0.999
## Ye           1.852e-17  2.121e+00   0.000    1.000
## Yf          -1.985e+01  1.602e+04  -0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 37.5205  on 27  degrees of freedom
## Residual deviance:  7.6382  on 17  degrees of freedom
## AIC: 29.638
##
## Number of Fisher Scoring iterations: 21
```

```
pred_glm <- predict(fit1, test_set, type = 'response')
pred_glm_label <- ifelse(pred_glm > 0.5,1,2)
confuMatrix_LR = table(pred_glm_label, test_set$label)

# Calculating  the ACC,TPR,FPR,TNR & FNR from confusion matrix
acc_LR <- sum(diag(confuMatrix_LR)) / sum(confuMatrix_LR)
tpr_LR <- confuMatrix_LR[1,1]/sum(confuMatrix_LR[1,1], confuMatrix_LR[2,1])
fpr_LR <- confuMatrix_LR[1,2]/sum(confuMatrix_LR[1,2], confuMatrix_LR[2,2])
tnr_LR <- confuMatrix_LR[2,2]/sum(confuMatrix_LR[2,2], confuMatrix_LR[1,2])
fnr_LR <- confuMatrix_LR[2,1]/sum(confuMatrix_LR[2,1], confuMatrix_LR[1,1])
ROCRpred <- prediction(pred_glm, test_set$label)
ROCRperf <- performance(ROCRpred, 'tpr', 'fpr')
ROCRperfauc <- performance(ROCRpred, 'auc')
auc_LR <- ROCRperfauc@y.values[[1]]

# Putting all the values for Logistic regression into a row.
LRrow <- c("LR ",round(auc_LR,2), round(acc_LR,2),round(tpr_LR,2),round(fpr_LR,2), round(tnr_LR,2),round
```

```
## $X
##        X
## Y                5          19          35          51          55          63
##   BLACK 0.23529412 0.00000000 0.11764706 0.23529412 0.35294118 0.05882353
##   BLUE  0.18181818 0.45454545 0.00000000 0.09090909 0.00000000 0.27272727
##
## $Y
##        Y
## Y                a           b           c           d           e           f
##   BLACK 0.23529412 0.11764706 0.05882353 0.17647059 0.23529412 0.17647059
##   BLUE  0.18181818 0.18181818 0.36363636 0.09090909 0.09090909 0.09090909
```

```
## [1] BLUE  BLACK BLUE  BLACK BLACK BLACK BLUE  BLUE
## attr(,"prob")
## [1] 1.0000000 0.6666667 0.5000000 0.7500000 0.7500000 0.6666667 1.0000000
## [8] 1.0000000
## Levels: BLACK BLUE
```

```
# Results matrix

resMatrix <- data.frame(matrix(ncol = 6, nrow = 0))
resMatrix <- rbind(resMatrix,LRrow,NBrow,KNN3row,KNN5row)
colnames(resMatrix) <- c("ALGO", "AUC","ACC", "TPR", "FPR", "TNR ", "FNR")

kable(resMatrix) %>%
  kable_styling(bootstrap_options = c("striped","hover","condensed","responsive"),full_width   = F,posi
  row_spec(0, background ="gray")
```

ALGO	AUC	ACC	TPR	FPR	TNR	FNR
LR	0.8	0.38	0.2	0.33	0.67	0.8
NAIVEB	0.73	0.75	0.8	0.33	0.67	0.2
KNN3	0.9	0.88	0.8	0	1	0.2
KNN5	0.8	0.75	0.6	0	1	0.4

Commentary

TPR - True positive rate (sensitivity) tries to find the percentage of true positives where predictions were correctly identified.

FPR - False positives rate (1 - specificity(True Negative)) tries to find percentage of true positive where prediction is incorrectly identified (Type 1 Errors).

(Specificity and sensitivity are inversely proportional, while TPR and FPR are not.)

Accuracy - Finds the percentage of true positive and true negatives where predictions were correctly identified.

AUC - Area under the curve is the area under the ROC curve when comparing TPR to FPR for several models. The closer the curve is to the upper left corner the better the model classification.

LR Model - The GLM linear regression model shows that the accuracy of TP and TN is the lowest among all models at 0.36 while the AUC is the .81. This tells us the model did lower than average job at classfication.

NB Model - The NB model shows that the accuracy of TP and TN is 0.73 while the AUC is the .75 . This model has done average in terms of Accuracy & Area under Curve.

KNN3 Model - The KNN , has the highest AUC at .9 and also the accuracy is highest at .88 among all the models. Also the model is quiet good in predicting True Negative and has False Positive rate of 0 which is good. This model has exceeded expectations. This model also has good rate of predicting True Positive rate among all the models.

Knn5 Model - The KNN model where K=5 also has performed better than otger models but lesser than K=3 model, with AUC at .8 and accuracy at .75 .

And if we want to go further , then we can use F-Score to further push the case that NB & KNN-3 are the best models.

F score is the harmonic mean of precision and recall. It lies between 0 and 1. Higher the value, better the model. It is formulated as 2((precision*recall) / (precision+recall)).

Precision= (TP / TP + FP)

recall = TPR

As the dataset is too small, AUC would be preferred measure of classifier performance than accuracy for the following reasons:

- AUC does not bias on size of evaluation data
- Also accuracy depends on setting a probability cut-off (for balanced data this is fine, but in the imbalanced case the minority class probabilities may be all below 0.5, while AUC considers the ranking of positives and negative according to probability minority.)
- Metric like accuracy is calculated based on the class distribution of test dataset or cross-validation, but this ratio may change when we apply the classifier to real life data, because the underlying class distribution has been changed or unknown. On the other hand, TP rate and FP rate which are used to construct AUC will not be affected by class distribution shifting.

Classifier Performance differences

A quick look at the scattered plots for the entire dataset and the training and test sets reveals how many samples of different classes intertwined between each other.This makes it harder for several of these classifier to properly assign classes.

Also as the Dataset is small , thus as expected the performance of the Logistic Regression model is poor, and NB performance is good with small dataset.

The suprising part is that KNN (k=3) is top most performing model this is due to the small data set and KNN is a non-parametrised algorithm