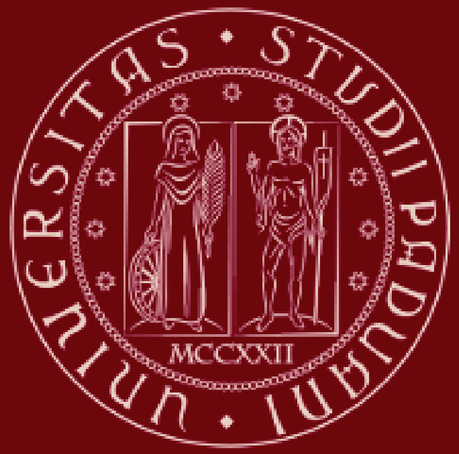
Mobile Phones Price: a classification problem



Statistical Learning course

Anthony Palmieri

Pietro Maria Sanguin

Angelica Giangiacomi

Academic year : 2021/2022

Overview

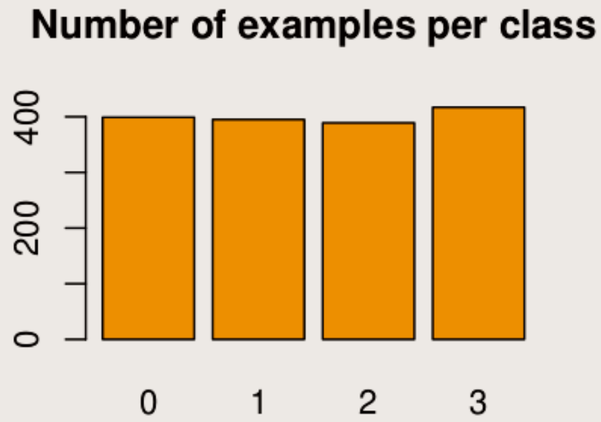
- **Dataset**: Mobile Price Train
- **Preprocessing and Data Cleaning**
- **Exploratory Data Analysis**
- **Training Phase**
  - Multinomial Logistic Regression
  - Linear Discriminant Analysis
  - Quadratic Discriminant Analysis
  - K-Nearest Neighbors
- **Conclusions**

Dataset: Mobile Price Train

- Multiclass Classification Problem
  - 4 labels: low-cost(0), medium-cost(1), high-cost(2), very-high cost(3)
- 2000 examples
- 20 features
  - Categorical: `blue, dual_sim, four_g, three_g, touch_screen, wifi`
  - Numerical: `battery_power, clock_speed, fc, int_memory, m_depth, mobile_wt, n_cores, pc, px_height, px_width, ram, sc_h, sc_w, talk_time`

Preprocessing and Data Cleaning

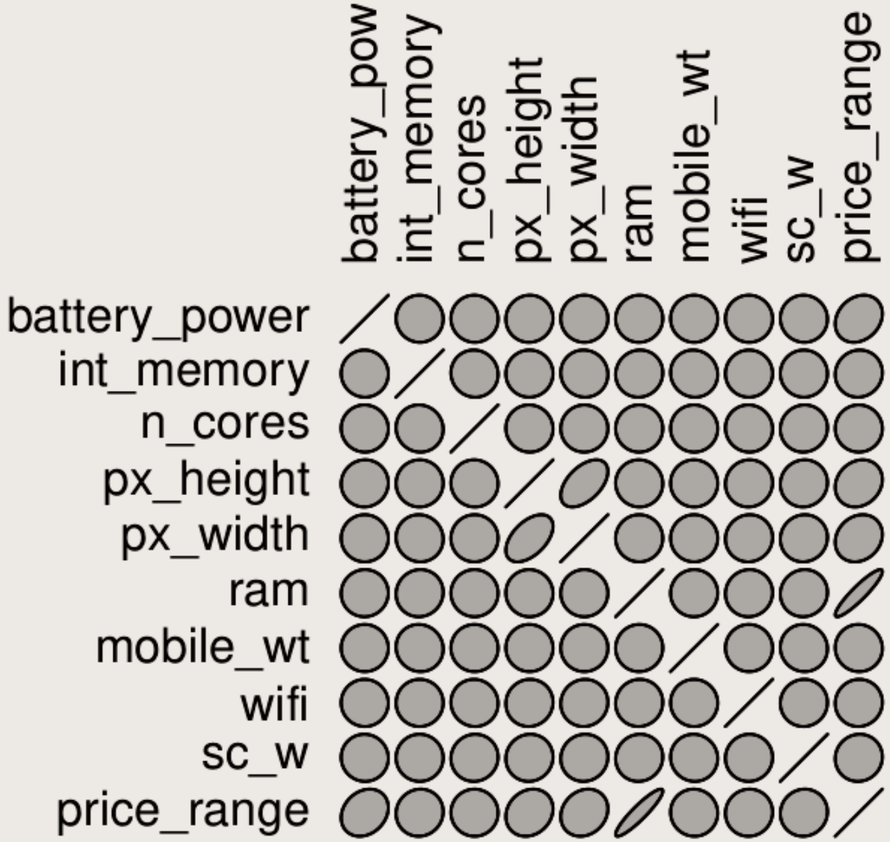
- → Not Available or Not a Number values? ✗
- → Split dataset into Training and Test sets
- → Check for Class Balance



- → Feature Scaling: MinMax Scaler

Exploratory Data Analysis (I)

→ Correlation Ellipses

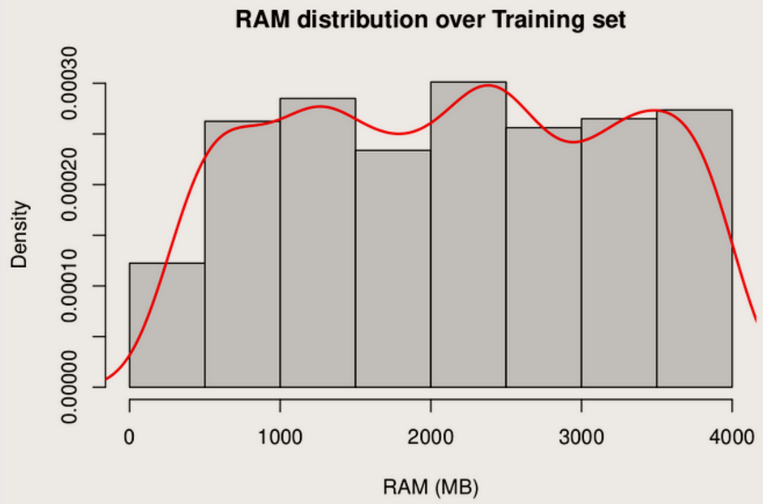


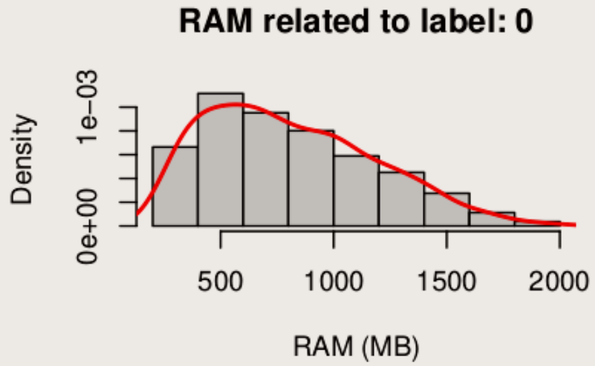
Highest correlation value:
$\rho = 0.91$

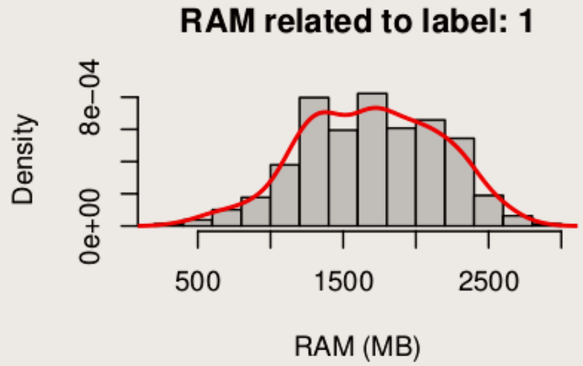
Exploratory Data Analysis (II)

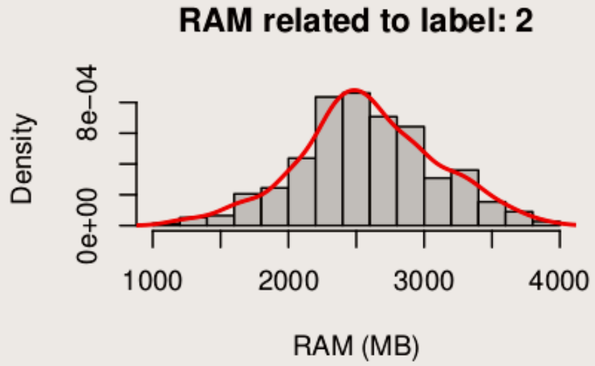
RAM Distribution per class

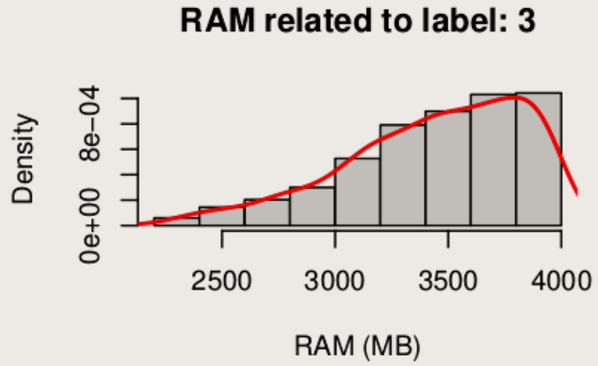
RAM Distribution over Training set











Exploratory Data Analysis (III)

→ Check for Multicollinearity using Variance Inflation Factor

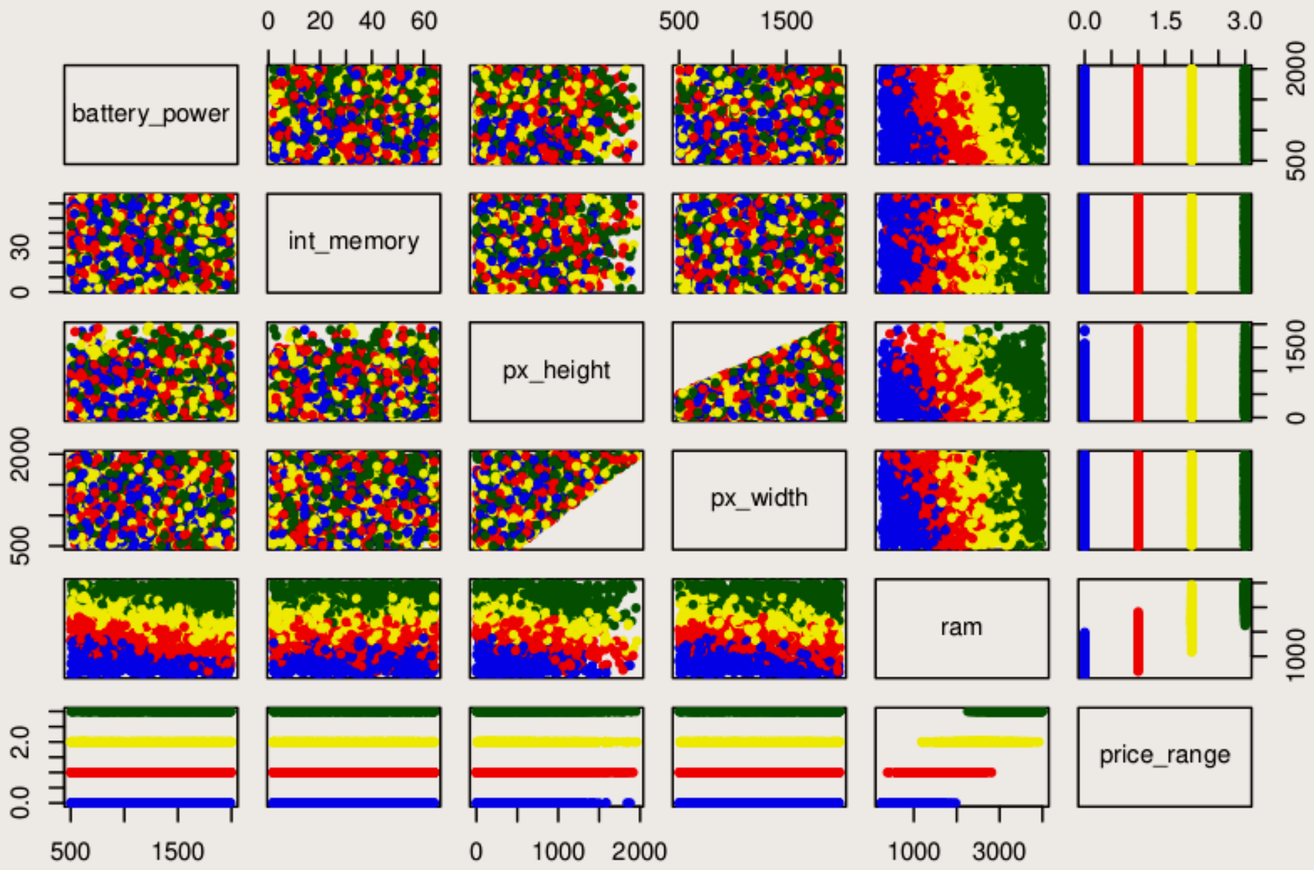
```
# print all VIFs values
vifs
```

```
##  [1] 1.011139 1.002891 1.730141 1.010976 1.007034 1.003561 1.006990 1.736878
##  [9] 1.353688 1.352578 1.006079 1.399044 1.396769 1.006836
```

All values close to 1 ⇝ no important multicollinear relationships

Exploratory Data Analysis (IV)

→ Check for Non-Linear Relationships



- Low cost
- Medium cost
- High cost
- Very high cost

↳ no important Non-Linear relationships

Training Phase

→ Multinomial Logistic Regression

⇝ a classification method that generalizes Logistic Regression to multiclass problems

→ Linear Discriminant Analysis

⇝ a classification method that under normality assumptions uses Bayes theorem to derive discrimination scores

→ Quadratic Discriminant Analysis

⇝ a classification method just like **LDA** but does not assume equal variance among the various groups

→ K - Nearest Neighbors

⇝ a classification method that uses a majority scheme on the K nearest examples to classify each point

Multinomial Logistic Regression (Full Dataset)

→ Train the model keeping all features

```r
# original model with all predictors

# create formula to fit the model
features = paste(names(X_train), collapse='+')
formula = paste(c('y_train', features), collapse='~')

# Fit the Multinomial Logistic Regression model
mlr_full <- nnet::multinom(formula, data = X_train_scaled)
```

```r
# Make predictions on Training set
y_train_pred <- predict(mlr_full, X_train_scaled)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred == y_train),"\n")
```

```
## Training set accuracy:  0.995
```

```r
# Make predictions on Test set
y_test_pred <- predict(mlr_full, X_test_scaled)

# Model accuracy on Test set
cat("Test set accuracy: ", mean(y_test_pred == y_test))
```

```
## Test set accuracy:  0.975
```

Multinomial Logistic Regression (BWS)

→ Train the model using p-values to apply Backward Selection

```r
# Final model after BACKWARD SELECTION

# create formula to fit the model
features = paste(names(X_train[,-c(19, 18, 2, 8, 6, 15, 17, 4, 16, 5, 3, 11, 20)]), collapse='+')
formula = paste(c('y_train', features), collapse='~')

# Fit the Multinomial Logistic Regression model
mlr_BS <- nnet::multinom(formula, data = X_train_scaled)
```

Features kept:

battery_power, int_memory, mobile_wt, n_cores, px_height, px_width, ram

```r
# Make predictions on Training set
y_train_pred <- predict(mlr_BS, X_train_scaled)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred == y_train),"\n")
```

```
## Training set accuracy:  0.985
```

```r
# Make predictions on Test set
y_test_pred <- predict(mlr_BS, X_test_scaled)

# Model accuracy on Test set
cat("Test set accuracy: ", mean(y_test_pred == y_test))
```

```
## Test set accuracy:  0.9775
```

Multinomial Logistic Regression (FWS)

→ Train the model using Cross-Validation error to apply Forward Selection

```
# final model after FORWARD SELECTION

# create formula to fit the model
features = paste(names(X_train[, c(14, 12, 1, 13, 9)]), collapse='+')
formula = paste(c('y_train', features), collapse='~')

# Fit the Multinomial Logistic Regression model
mlr_FS <- nnet::multinom(formula, data = X_train_scaled)
```

Features selected:

battery_power, mobile_wt, px_height, px_width, ram

```
# Make predictions on Training set
y_train_pred <- predict(mlr_FS, X_train_scaled)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred == y_train),"\n")
```

```
## Training set accuracy:  0.983125
```

```
# Make predictions on Test set
y_test_pred <- predict(mlr_FS, X_test_scaled)

# Model accuracy on Test set
cat("Test set accuracy: ", mean(y_test_pred == y_test))
```

```
## Test set accuracy:  0.9675
```

Multinomial Logistic Regression (RAM only)

→ Train the model using RAM as the only feature

```
# Fit the model
mlr_ram <- nnet::multinom(y_train ~ ram, data = X_train_scaled)
```

```
# Make predictions on Training set
y_train_pred <- predict(mlr_ram, X_train_scaled)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred == y_train),"\n")
```

```
## Training set accuracy:  0.749375
```

```
# Make predictions on Test set
y_test_pred <- predict(mlr_ram, X_test_scaled)

# Model accuracy on Test set
cat("Test set accuracy: ", mean(y_test_pred == y_test))
```

```
## Test set accuracy:  0.7875
```

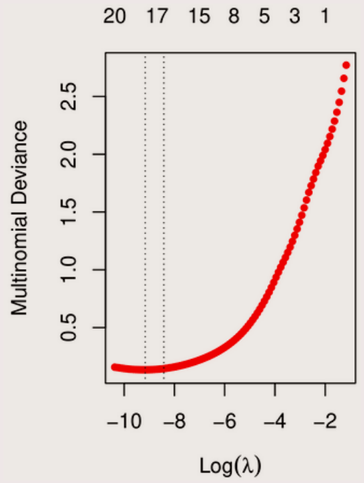
ram is not the only influential feature

Multinomial Logistic Regression (LASSO)

→ Train the model using LASSO Regularization

```
# Lasso Regression
mlr_lasso = cv.glmnet(as.matrix(X_train_scaled), y_train, alpha=1, family="multinomial")
```

log(λ) vs Multinomial Deviance




```
# LASSO predictions

# Make predictions on Training set
y_train_pred <- predict(mlr_lasso, newx = as.matrix(X_train_scaled), s = "lambda.min",  type = "class")

# Model accuracy on Training set
cat("Training set accuracy with LASSO: ", mean(y_train_pred == y_train),"\n")
```

```
## Training set accuracy with LASSO:  0.989375
```

```
# Make predictions on Test set
y_test_pred <- predict(mlr_lasso, newx = as.matrix(X_test_scaled), s = "lambda.min",  type = "class")

# Model accuracy on Test set
cat("Test set accuracy with LASSO: ", mean(y_test_pred == y_test))
```

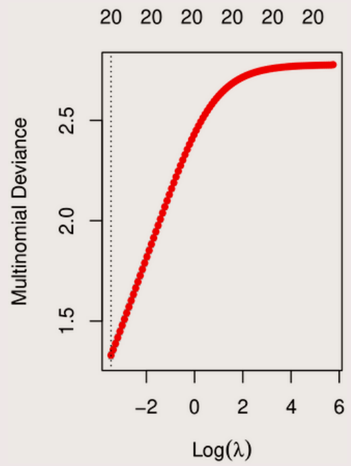
```
## Test set accuracy with LASSO:  0.975
```

Multinomial Logistic Regression (RIDGE)

→ Train the model using RIDGE Regularization

```
# Ridge Regression
mlr_ridge = cv.glmnet(as.matrix(X_train_scaled), y_train, alpha=0, family="multinomial")
```

log(λ) vs Multinomial Deviance




```
# RIDGE predictions

# Make predictions on Training set
y_train_pred <- predict(mlr_ridge , newx = as.matrix(X_train_scaled), s = "lambda.min",  type = "class"

# Model accuracy on Training set
cat("Training set accuracy with RIDGE: ", mean(y_train_pred == y_train),"\n")
```

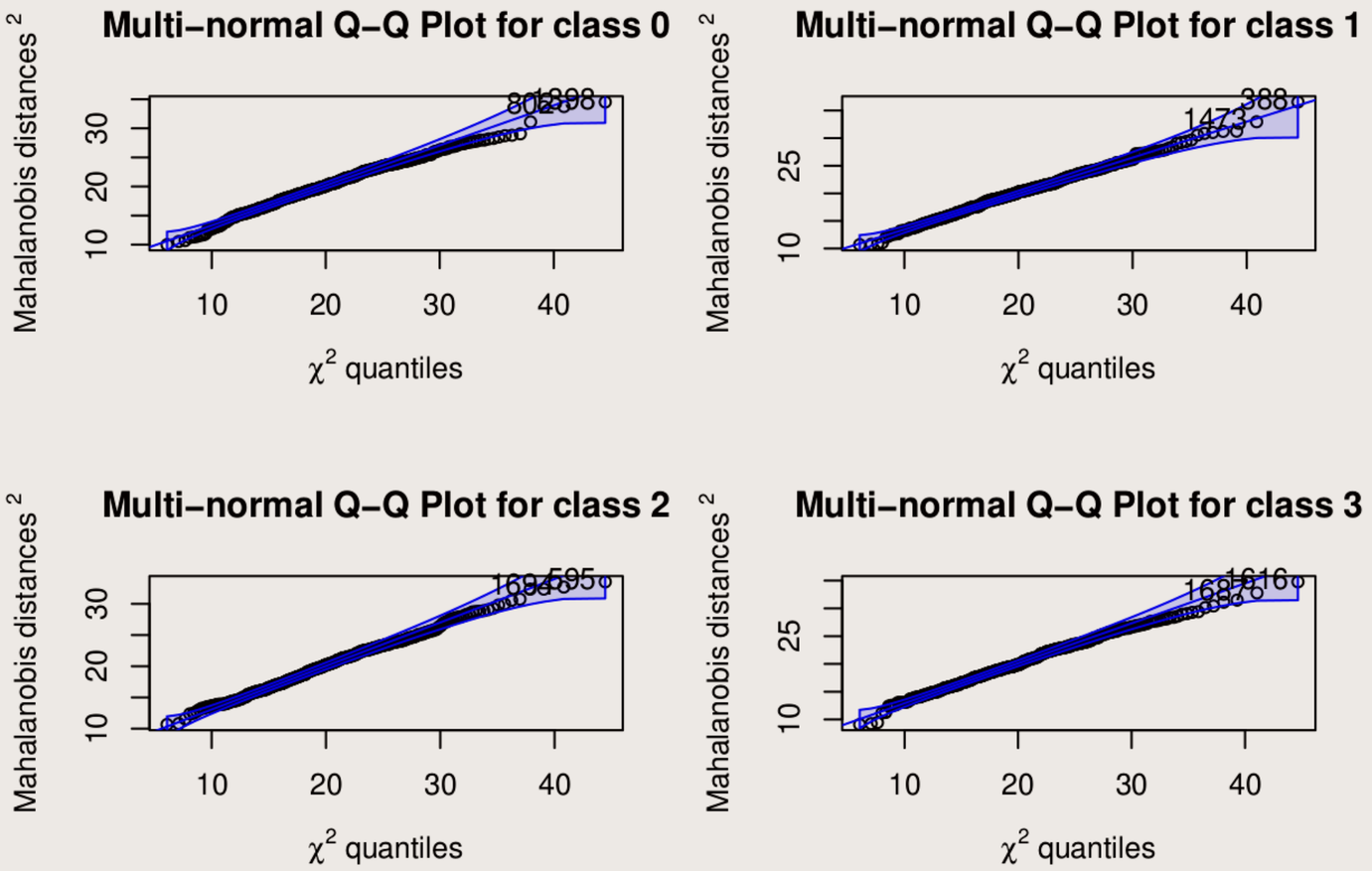
```
## Training set accuracy with RIDGE:  0.875625
```

```
# Make predictions on Test set
y_test_pred <- predict(mlr_ridge, newx = as.matrix(X_test_scaled), s = "lambda.min",  type = "class")

# Model accuracy on Test set
cat("Test set accuracy with RIDGE: ", mean(y_test_pred == y_test))
```

```
## Test set accuracy with RIDGE:  0.8525
```

Multivariate QQ-Plots

→ Check for Normal distribution within each class



Linear Discriminant Analysis (Full Dataset)

→ Train the model keeping all features

```
# original model considering all features
LDA <- lda(y_train~. , data = X_train_scaled)
```

```
CM_train <- table(y_train_pred$class, y_train)
CM_train
```

```
##    y_train
##        0   1   2   3
##   0 388   9   0   0
##   1  11 375  25   0
##   2   0  11 361  17
##   3   0   0   3 400
```

```
CM_test <- table(y_test_pred$class, y_test)
CM_test
```

```
##    y_test
##        0   1   2   3
##   0  99   1   0   0
##   1   2 103   6   0
##   2   0   1 104   4
##   3   0   0   1  79
```

```
# Make predictions on Training set
y_train_pred <- predict(LDA, X_train_scaled)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred$class == y_train),"\n")
```

```
## Training set accuracy:  0.9525
```

```
# Make predictions on Test set
y_test_pred <- predict(LDA, X_test_scaled)

# Model accuracy on Test set
cat("Test set accuracy: ", mean(y_test_pred$class == y_test),'\n')
```

```
## Test set accuracy:  0.9625
```

Linear Discriminant Analysis (FWS Multinom.)

→ Train the model with the same features used in the Multinomial Logistic Regression (FWS)

```
# Linear Discriminant Analysis with the same features got for the Multinomial Logistic Regression model

# Fit the Linear Discriminant Analysis model
LDA <- lda(y_train~ram+px_height+battery_power+px_width+mobile_wt, data = X_train_scaled)
```

```
CM_train <- table(y_train_pred$class, y_train)
CM_train
```

```
##    y_train
##       0   1   2   3
##   0 391   7   0   0
##   1   8 378  26   0
##   2   0  10 359  19
##   3   0   0   4 398
```

```
CM_test <- table(y_test_pred$class, y_test)
CM_test
```

```
##    y_test
##       0   1   2   3
##   0  98   0   0   0
##   1   3 105   7   0
##   2   0   0 104   5
##   3   0   0   0  78
```

```
# Make predictions on Training set
y_train_pred <- predict(LDA, X_train_scaled)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred$class == y_train),"\n")
```

```
## Training set accuracy:  0.95375
```

```
# Make predictions on Test set
y_test_pred <- predict(LDA, X_test_scaled)

# Model accuracy on Test set
cat("Test set accuracy: ", mean(y_test_pred$class == y_test),'\n')
```

```
## Test set accuracy:  0.9625
```

Linear Discriminant Analysis (BWS Multinom.)

→ Train the model with the same features used in the Multinomial Logistic Regression (BWS)

```
# Linear Discriminant Analysis with the same features got for the Multinomial Logistic Regression model

# Fit the Linear Discriminant Analysis model
LDA <- lda(y_train~ram+battery_power+int_memory+mobile_wt+n_cores+px_height+px_width, data = X_train_sc
```

```
CM_train <- table(y_train_pred$class, y_train)
CM_train
```

```
##    y_train
##       0   1   2   3
##   0 391   7   0   0
##   1   8 377  25   0
##   2   0  11 359  24
##   3   0   0   5 393
```

```
CM_test <- table(y_test_pred$class, y_test)
CM_test
```

```
##    y_test
##       0   1   2   3
##   0  99   0   0   0
##   1   2 104   7   0
##   2   0   1 104   5
##   3   0   0   0  78
```

```
# Make predictions on Training set
y_train_pred <- predict(LDA, X_train_scaled)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred$class == y_train),"\n")
```

```
## Training set accuracy:  0.95
```

```
# Make predictions on Test set
y_test_pred <- predict(LDA, X_test_scaled)

# Model accuracy on Test set
cat("Test set accuracy: ", mean(y_test_pred$class == y_test),'\n')
```

```
## Test set accuracy:  0.9625
```

Linear Discriminant Analysis (FWS)

→ Train the model using Cross-Validation error to apply Forward Selection

```
# final model after FORWARD SELECTION

# Fit the Linear Discriminant Analysis model
LDA <- lda(y_train~ram+battery_power+px_height+px_width+mobile_wt+wifi+sc_w, data = X_train_scaled)
```

```
CM_train <- table(y_train_pred$class, y_train)
CM_train
```

```
##    y_train
##        0   1   2   3
##   0 392   8   0   0
##   1   7 377  21   0
##   2   0  10 367  19
##   3   0   0   1 398
```

```
CM_test <- table(y_test_pred$class, y_test)
CM_test
```

```
##    y_test
##        0   1   2   3
##   0 101   0   0   0
##   1   0 105   5   0
##   2   0   0 106   5
##   3   0   0   0  78
```

```
# Make predictions on Training set
y_train_pred <- predict(LDA, X_train_scaled)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred$class == y_train),"\n")
```

```
## Training set accuracy:  0.95875
```

```
# Make predictions on Test set
y_test_pred <- predict(LDA, X_test_scaled)

# Model accuracy on Test set
cat("Test set accuracy: ", mean(y_test_pred$class == y_test),'\n')
```

```
## Test set accuracy:  0.975
```

Features selected:

battery_power, int_memory, mobile_wt, n_cores, px_height, px_width, ram

Quadratic Discriminant Analysis (Full Dataset)

→ Train the model keeping all features

```
# original model considering all features
QDA <- qda(y_train~. , data = X_train_scaled)
```

```
CM_train <- table(y_train_pred$class, y_train)
CM_train
```

```
##    y_train
##        0   1   2   3
##   0 393  12   0   0
##   1   6 377  13   0
##   2   0   6 363   6
##   3   0   0  13 411
```

```
CM_test <- table(y_test_pred$class, y_test)
CM_test
```

```
##    y_test
##       0   1   2   3
##   0  98   3   0   0
##   1   3 100   5   0
##   2   0   2 100   5
##   3   0   0   6  78
```

```
# Make predictions on Training set
y_train_pred <- predict(QDA, X_train_scaled)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred$class == y_train),"\n")
```

```
## Training set accuracy:  0.965
```

```
# Make predictions on Test set
y_test_pred <- predict(QDA, X_test_scaled)

# Model accuracy on Test set
cat("Test set accuracy: ", mean(y_test_pred$class == y_test),'\n')
```

```
## Test set accuracy:  0.94
```

Quadratic Discriminant Analysis (FWS Multinom.)

→ Train the model with the same features used in the Multinomial Logistic Regression (FWS)

```
# Quadratic Discriminant Analysis with the same features got for the Multinomial Logistic Regression

# Fit the Linear Discriminant Analysis model
QDA <- qda(y_train~ram+px_height+battery_power+px_width+mobile_wt, data = X_train_scaled)
```

```
CM_train <- table(y_train_pred$class, y_train)
CM_train
```

```
##    y_train
##       0   1   2   3
##   0 395  12   0   0
##   1   4 374  10   0
##   2   0   9 370   4
##   3   0   0   9 413
```

```
CM_test <- table(y_test_pred$class, y_test)
CM_test
```

```
##    y_test
##       0   1   2   3
##   0 100   2   0   0
##   1   1 101   4   0
##   2   0   2 105   4
##   3   0   0   2  79
```

```
# Make predictions on Training set
y_train_pred <- predict(QDA, X_train_scaled)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred$class == y_train),"\n")
```

```
## Training set accuracy:  0.97
```

```
# Make predictions on Test set
y_test_pred <- predict(QDA, X_test_scaled)

# Model accuracy on Test set
cat("Test set accuracy: ", mean(y_test_pred$class == y_test),'\n')
```

```
## Test set accuracy:  0.9625
```

Quadratic Discriminant Analysis (BWS Multinom.)

→ Train the model with the same features used in the Multinomial Logistic Regression (BWS)

```
# Quadratic Discriminant Analysis with the same features got for the Multinomial Logistic Regression mo

# Fit the Linear Discriminant Analysis model
QDA <- qda(y_train~ram+battery_power+int_memory+mobile_wt+n_cores+px_height+px_width, data = X_train_sc
```

```
CM_train <- table(y_train_pred$class, y_train)
CM_train
```

```
##    y_train
##       0   1   2   3
##   0 393  15   0   0
##   1   6 373  11   0
##   2   0   7 366   5
##   3   0   0  12 412
```

```
CM_test <- table(y_test_pred$class, y_test)
CM_test
```

```
##    y_test
##       0   1   2   3
##   0  99   1   0   0
##   1   2 102   4   0
##   2   0   2 103   2
##   3   0   0   4  81
```

```
# Make predictions on Training set
y_train_pred <- predict(QDA, X_train_scaled)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred$class == y_train),"\n")
```

```
## Training set accuracy:  0.965
```

```
# Make predictions on Test set
y_test_pred <- predict(QDA, X_test_scaled)

# Model accuracy on Test set
cat("Test set accuracy: ", mean(y_test_pred$class == y_test),'\n')
```

```
## Test set accuracy:  0.9625
```

Quadratic Discriminant Analysis (FWS LDA)

→ Train the model with the same features used in the Linear Discriminant Analysis (FWS)

```
# Quadratic Discriminant Analysis with the same features got for LDA model, through Forward Selection

# Fit the Linear Discriminant Analysis model
QDA <- qda(y_train~ram+battery_power+px_height+px_width+mobile_wt+wifi+sc_w, data = X_train_scaled)
```

```
CM_train <- table(y_train_pred$class, y_train)
CM_train
```

```
##    y_train
##       0   1   2   3
##   0 395  17   0   0
##   1   4 369   7   0
##   2   0   9 370   5
##   3   0   0  12 412
```

```
CM_test <- table(y_test_pred$class, y_test)
CM_test
```

```
##    y_test
##       0   1   2   3
##   0 100   3   0   0
##   1   1 101   4   0
##   2   0   1 103   3
##   3   0   0   4  80
```

```
# Make predictions on Training set
y_train_pred <- predict(QDA, X_train_scaled)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred$class == y_train),"\n")
```

```
## Training set accuracy:  0.96625
```

```
# Make predictions on Test set
y_test_pred <- predict(QDA, X_test_scaled)

# Model accuracy on Test set
cat("Test set accuracy: ", mean(y_test_pred$class == y_test),'\n')
```

```
## Test set accuracy:  0.96
```

Quadratic Discriminant Analysis (FWS)

→ Train the model using Cross-Validation error to apply Forward Selection

```
# final model after FORWARD SELECTION

# Fit the Linear Discriminant Analysis model
QDA <- qda(y_train ~ ram+battery_power+clock_speed+mobile_wt+px_height+px_width+fc, data = X_train_sc
```

```
CM_train <- table(y_train_pred$class, y_train)
CM_train
```

```
##    y_train
##       0   1   2   3
##   0 393  11   0   0
##   1   6 376  10   0
##   2   0   8 366   5
##   3   0   0  13 412
```

```
CM_test <- table(y_test_pred$class, y_test)
CM_test
```

```
##    y_test
##       0   1   2   3
##   0 100   4   0   0
##   1   1  99   4   0
##   2   0   2 105   4
##   3   0   0   2  79
```

```
# Make predictions on Training set
y_train_pred <- predict(QDA, X_train_scaled)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred$class == y_train),"\n")
```

```
## Training set accuracy:  0.966875
```

```
# Make predictions on Test set
y_test_pred <- predict(QDA, X_test_scaled)

# Model accuracy on Test set
cat("Test set accuracy: ", mean(y_test_pred$class == y_test),'\n')
```

```
## Test set accuracy:  0.9575
```

Features selected:

battery_power, clock_speed, mobile_wt, px_height, px_width, fc, ram

K - Nearest Neighbors (Not Scaled)

→ Train the model using original data

```
# train the K-Nearest Neighbors with the original Dataset

set.seed(1)
ctrl <- trainControl(method="repeatedcv", repeats = 5)
knn <- train(as.factor(price_range) ~ ., data = Training_set, method = "knn",
             trControl = ctrl, tuneLength = 20)
```

```
table(y_train_pred, as.factor(y_train))
```

```
##
## y_train_pred   0   1   2   3
##            0 391  18   0   0
##            1   8 367  20   0
##            2   0  10 354  16
##            3   0   0  15 401
```

```
table(y_test_pred, as.factor(y_test))
```

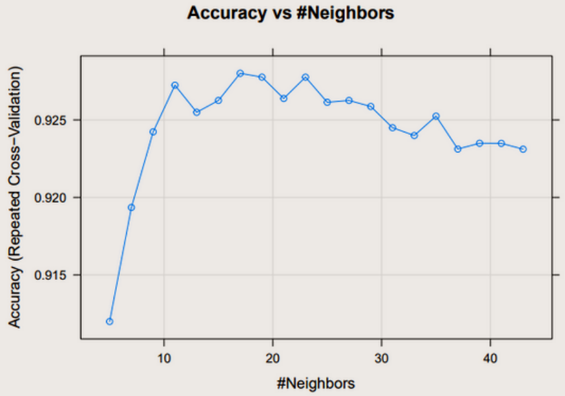
```
##
## y_test_pred   0   1   2   3
##           0  98   2   0   0
##           1   3 102   4   0
##           2   0   1 101   5
##           3   0   0   6  78
```

```
y_train_pred <- predict(knn,newdata = Training_set)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred == as.factor(y_train)), '\n')
```

```
## Training set accuracy:  0.945625
```

```
y_test_pred <- predict(knn,newdata = Test_set)

# Model accuracy on Test set
cat("\nTest set accuracy: ", mean(y_test_pred == as.factor(y_test)), '\n')
```

```
##
## Test set accuracy:  0.9475
```



Final value
K =17

K - Nearest Neighbors (Scaled)

→ Train the model using scaled data

```
# train the K-Nearest Neighbors with scaled data

set.seed(1)
ctrl <- trainControl(method="repeatedcv", repeats = 5)
knn <- train(as.factor(price_range) ~ ., data = Training_set, method = "knn", trControl = ctrl,
             tuneLength = 20, preProcess = c("center","scale"))
```

```
table(y_train_pred, as.factor(y_train))
```

```
##
## y_train_pred   0   1   2   3
##            0 329  93   7   0
##            1  65 194  67   6
##            2   5  95 242  81
##            3   0  13  73 330
```

```
table(y_test_pred, as.factor(y_test))
```

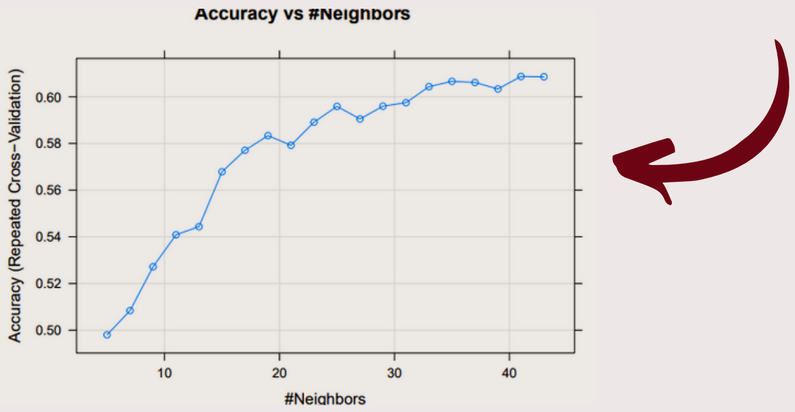
```
##
## y_test_pred  0  1  2  3
##           0 84 28  1  0
##           1 14 54 27  0
##           2  3 19 62 16
##           3  0  4 21 67
```

```
y_train_pred <- predict(knn,newdata = Training_set)

# Model accuracy on Training set
cat("Training set accuracy: ", mean(y_train_pred == as.factor(y_train)), '\n')
```

```
## Training set accuracy:  0.684375
```

```
y_test_pred <- predict(knn,newdata = Test_set)

# Model accuracy on Test set
cat("\nTest set accuracy: ", mean(y_test_pred == as.factor(y_test)), '\n')
```

```
##
## Test set accuracy:  0.6675
```



Final value
K = 41

Conclusion

→ Summary of trained models

Model	**Train set accuracy** (%)	**Test set accuracy** (%)
Multinomial Logistic Regression (Full Dataset)	99.50	97.50
Multinomial Logistic Regression (BWS)	98.50	97.75
Multinomial Logistic Regression (FWS)	98.31	96.75
Multinomial Logistic Regression (ram only)	74.94	78.75
Multinomial Logistic Regression (Lasso)	98.94	97.50
Multinomial Logistic Regression (Ridge)	87.56	85.25
Linear Discriminant Analysis (Full Dataset)	95.25	96.25
Linear Discriminant Analysis (FWS Multinom.)	95.38	96.25
Linear Discriminant Analysis (BWS Multinom.)	95.00	96.25
Linear Discriminant Analysis (FWS)	95.88	97.50
Quadratic Discriminant Analysis (Full Dataset)	96.50	94.00
Quadratic Discriminant Analysis (FWS Multinom.)	97.00	96.25
Quadratic Discriminant Analysis (BWS Multinom.)	96.50	96.25
Quadratic Discriminant Analysis (FWS LDA)	96.63	96.00
Quadratic Discriminant Analysis (FWS)	96.69	95.75
K-NN (not scaled)	94.56	94.75
K-NN (scaled)	68.44	66.75

Conclusion

→ Summary of trained models

Model	**Train set accuracy** (%)	**Test set accuracy** (%)
Multinomial Logistic Regression (Full Dataset)	99.50	97.50
Multinomial Logistic Regression (BWS)	98.50	97.75
Multinomial Logistic Regression (FWS)	98.31	96.75
Multinomial Logistic Regression (ram only)	74.94	78.75
Multinomial Logistic Regression (Lasso)	98.94	97.50
Multinomial Logistic Regression (Ridge)	87.56	85.25
Linear Discriminant Analysis (Full Dataset)	95.25	96.25
Linear Discriminant Analysis (FWS Multinom.)	95.38	96.25
Linear Discriminant Analysis (BWS Multinom.)	95.00	96.25
Linear Discriminant Analysis (FWS)	95.88	97.50
Quadratic Discriminant Analysis (Full Dataset)	96.50	94.00
Quadratic Discriminant Analysis (FWS Multinom.)	97.00	96.25
Quadratic Discriminant Analysis (BWS Multinom.)	96.50	96.25
Quadratic Discriminant Analysis (FWS LDA)	96.63	96.00
Quadratic Discriminant Analysis (FWS)	96.69	95.75
K-NN (not scaled)	94.56	94.75
K-NN (scaled)	68.44	66.75

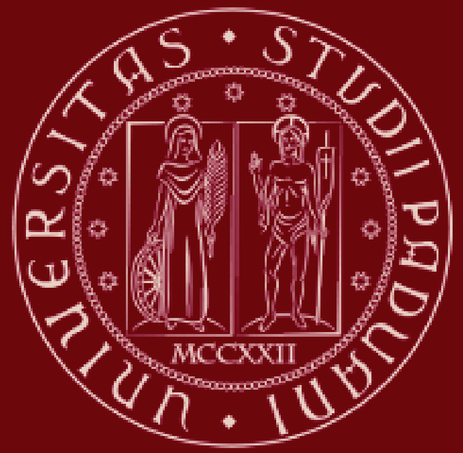
Highest
Test accuracy

Conclusion

→ Summary of trained models

Model	**Train set accuracy** (%)	**Test set accuracy** (%)
Multinomial Logistic Regression (Full Dataset)	99.50	97.50
Multinomial Logistic Regression (BWS)	98.50	97.75
Multinomial Logistic Regression (FWS)	98.31	96.75
Multinomial Logistic Regression (ram only)	74.94	78.75
Multinomial Logistic Regression (Lasso)	98.94	97.50
Multinomial Logistic Regression (Ridge)	87.56	85.25
Linear Discriminant Analysis (Full Dataset)	95.25	96.25
Linear Discriminant Analysis (FWS Multinom.)	95.38	96.25
Linear Discriminant Analysis (BWS Multinom.)	95.00	96.25
Linear Discriminant Analysis (FWS)	95.88	97.50
Quadratic Discriminant Analysis (Full Dataset)	96.50	94.00
Quadratic Discriminant Analysis (FWS Multinom.)	97.00	96.25
Quadratic Discriminant Analysis (BWS Multinom.)	96.50	96.25
Quadratic Discriminant Analysis (FWS LDA)	96.63	96.00
Quadratic Discriminant Analysis (FWS)	96.69	95.75
K-NN (not scaled)	94.56	94.75
K-NN (scaled)	68.44	66.75

Other relevant values



Thanks for the attention!