> ## Lab 5: Support Vector Machines

> #Load libraries

> library(ggplot2)

> library("caret")

Loading required package: lattice

Warning message:

package ‘caret’ was built under R version 4.4.2

> library(e1071)

Warning message:

package ‘e1071’ was built under R version 4.4.2

> library(readr)

> library(ggfortify)

> library(e1071)

> library(class)

> library(psych)

Attaching package: ‘psych’

The following objects are masked from ‘package:ggplot2’:

%+%, alpha

Warning message:

package ‘psych’ was built under R version 4.4.2

> #Loading wine dataset

> wine <- read\_csv(unz("C:/Users/chaos/Downloads/wine.zip", "wine.data"), col\_names = FALSE)

**Rows:** 178 **Columns:** 14

── **Column specification** ────────────────────────────────────────────────────────────────────────────────

**Delimiter:** ","

dbl (14): X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14

ℹ Use `spec()` to retrieve the full column specification for this data.

ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

> names(wine) <- c("Type","Alcohol","Malic acid","Ash","Alcalinity of ash","Magnesium","Total phenols","Flavanoids","Nonflavanoid Phenols","Proanthocyanins","Color Intensity","Hue","Od280/od315 of diluted wines","Proline")

> head(wine)

# A tibble: 6 × 14

Type Alcohol `Malic acid` Ash `Alcalinity of ash` Magnesium `Total phenols` Flavanoids

*<dbl>* *<dbl>* *<dbl>* *<dbl>* *<dbl>* *<dbl>* *<dbl>* *<dbl>*

1 1 14.2 1.71 2.43 15.6 127 2.8 3.06

2 1 13.2 1.78 2.14 11.2 100 2.65 2.76

3 1 13.2 2.36 2.67 18.6 101 2.8 3.24

4 1 14.4 1.95 2.5 16.8 113 3.85 3.49

5 1 13.2 2.59 2.87 21 118 2.8 2.69

6 1 14.2 1.76 2.45 15.2 112 3.27 3.39

# ℹ 6 more variables: `Nonflavanoid Phenols` <dbl>, Proanthocyanins <dbl>, `Color Intensity` <dbl>,

# Hue <dbl>, `Od280/od315 of diluted wines` <dbl>, Proline <dbl>

> wine$Type <- as.factor(wine$Type)

> wine <- wine[,-c(4,5,10)]

> ## split train/test

> train.indexes <- sample(178,0.7\*178)

> train <- wine[train.indexes,]

> test <- wine[-train.indexes,]

> ## separate x (features) & y (class labels)

> x <- wine[,2:11]

> y <- wine[,1]

> ## feature boxplots

> boxplot(x, main="wine features")

> ## class label distributions

> plot(y)

> ## feature-class plots

> featurePlot(x = x, y = y, plot = "ellipse")

NULL

> featurePlot(x = x, y = y, plot = "box")

NULL

> scales <- list(x = list(relation = "free"), y = list(relation = "free"))

> featurePlot(x = x, y = y, plot = "density", scales = scales)

NULL

> ggplot(wine, aes(x = Flavanoids, y = `Nonflavanoid Phenols`, colour = Type)) +

+ geom\_point() +

+ theme\_minimal()

> ## train SVM model - linear kernel

> svm.mod0 <- svm(Type ~ ., data = train, kernel = 'linear')

> svm.mod0

Call:

svm(formula = Type ~ ., data = train, kernel = "linear")

Parameters:

SVM-Type: C-classification

SVM-Kernel: linear

cost: 1

Number of Support Vectors: 25

> train.pred <- predict(svm.mod0, train)

> cm = as.matrix(table(Actual = train$Type, Predicted = train.pred))

> cm

Predicted

Actual 1 2 3

1 40 0 0

2 0 53 0

3 0 0 31

> n = sum(cm) # number of instances

> nc = nrow(cm) # number of classes

> diag = diag(cm) # number of correctly classified instances per class

> rowsums = apply(cm, 1, sum) # number of instances per class

> colsums = apply(cm, 2, sum) # number of predictions per class

> p = rowsums / n # distribution of instances over the actual classes

> q = colsums / n # distribution of instances over the predicted

> recall = diag / rowsums

> precision = diag / colsums

> f1 = 2 \* precision \* recall / (precision + recall)

> data.frame(precision, recall, f1)

precision recall f1

1 1 1 1

2 1 1 1

3 1 1 1

> ## train SVM model - polynomial kernel

> svm.mod1 <- svm(Type ~ ., data = train, kernel = 'polynomial')

> svm.mod1

Call:

svm(formula = Type ~ ., data = train, kernel = "polynomial")

Parameters:

SVM-Type: C-classification

SVM-Kernel: polynomial

cost: 1

degree: 3

coef.0: 0

Number of Support Vectors: 62

> train.pred <- predict(svm.mod1, train)

> cm = as.matrix(table(Actual = train$Type, Predicted = train.pred))

> cm

Predicted

Actual 1 2 3

1 36 4 0

2 0 53 0

3 0 1 30

> n = sum(cm) # number of instances

> nc = nrow(cm) # number of classes

> diag = diag(cm) # number of correctly classified instances per class

> rowsums = apply(cm, 1, sum) # number of instances per class

> colsums = apply(cm, 2, sum) # number of predictions per class

> p = rowsums / n # distribution of instances over the actual classes

> q = colsums / n # distribution of instances over the predicted

> recall = diag / rowsums

> precision = diag / colsums

> f1 = 2 \* precision \* recall / (precision + recall)

> data.frame(precision, recall, f1)

precision recall f1

1 1.0000000 0.9000000 0.9473684

2 0.9137931 1.0000000 0.9549550

3 1.0000000 0.9677419 0.9836066

> #Use tune.svmto find the optimum C and gamma values.

> ## Tuned SVM - polynomial

> tuned.svm <- tune.svm(Type~., data = train, kernel = 'polynomial',gamma = seq(1/2^nrow(wine),1, .01), cost = 2^seq(-6, 4, 2))

> tuned.svm

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:

gamma cost

0.52 0.015625

- best performance: 0.03269231

> svm.mod2 <- svm(Type ~ ., data = train, kernel = 'polynomial', gamma = 0.69, cost = .25)

> svm.mod2

Call:

svm(formula = Type ~ ., data = train, kernel = "polynomial", gamma = 0.69, cost = 0.25)

Parameters:

SVM-Type: C-classification

SVM-Kernel: polynomial

cost: 0.25

degree: 3

coef.0: 0

Number of Support Vectors: 37

> train.pred <- predict(svm.mod2, train)

> cm = as.matrix(table(Actual = train$Type, Predicted = train.pred))

> cm

Predicted

Actual 1 2 3

1 40 0 0

2 0 53 0

3 0 0 31

> n = sum(cm) # number of instances

> nc = nrow(cm) # number of classes

> diag = diag(cm) # number of correctly classified instances per class

> rowsums = apply(cm, 1, sum) # number of instances per class

> colsums = apply(cm, 2, sum) # number of predictions per class

> p = rowsums / n # distribution of instances over the actual classes

> q = colsums / n # distribution of instances over the predicted

> recall = diag / rowsums

> precision = diag / colsums

> f1 = 2 \* precision \* recall / (precision + recall)

> data.frame(precision, recall, f1)

precision recall f1

1 1 1 1

2 1 1 1

3 1 1 1

> #Choose another classification method (kNN, NaiveBayes, etc.) and train a classifier

> #based on the same features.

> #kNN classifier.

> # Train a classifier model to predict wine type using the 11 attributes.

> set.seed(10)

> # Sample ~70% of the dataset

> sample\_index <- sample(nrow(wine), 0.7 \* nrow(wine))

> wine.train <- wine[sample\_index, ]

> wine.test <- wine[-sample\_index, ]

> # Extract predictor variables and target variable for training and testing

> train\_x <- wine.train[, -1]

> train\_y <- wine.train$Type

> test\_x <- wine.test[, -1]

> test\_y <- wine.test$Type

> # Calculate the approximate square root

> sqrt\_train\_size <- sqrt(nrow(wine.train))

> k <- round(sqrt\_train\_size)

> # Choosing the number of neighbors (k) for the model

> accuracy <- c()

> ks <- c(3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99)

> # Looping through different k values to evaluate accuracy

> for (k in ks) {

+ # Train and predict using k-NN

+ KNNpred <- knn(train = train\_x, test = test\_x, cl = train\_y, k = k)

+

+ # Create confusion matrix

+ contingency.table <- table(Predicted = KNNpred, Actual = test\_y)

+ contingency.matrix <- as.matrix(contingency.table)

+

+ # Calculate accuracy for each k

+ accuracy <- c(accuracy, sum(diag(contingency.matrix)) / length(test\_y))

+ }

> # Plot accuracy against k values

> plot(ks, accuracy, type = "b", xlab = "Number of Neighbors (k)", ylab = "Accuracy", main = "k-NN Accuracy for Different k Values", ylim = c(min(accuracy), max(accuracy)))

> # Selecting the best k (based on highest accuracy) and print metrics

> best\_k <- ks[which.max(accuracy)]

> cat("Best k:", best\_k, "\n")

Best k: 30

> # Train and predict using the best k

> KNNpred <- knn(train = train\_x, test = test\_x, cl = train\_y, k = best\_k)

> # Printing

> knn\_confusion\_matrix <- table(Predicted = KNNpred, Actual = test\_y)

> print("Confusion Matrix for best k:")

[1] "Confusion Matrix for best k:"

> print(knn\_confusion\_matrix)

Actual

Predicted 1 2 3

1 15 3 0

2 0 18 4

3 4 4 6

> # Computing precision, recall, and F1 score

> compute\_metrics <- function(conf\_matrix) {

+ diag <- diag(conf\_matrix)

+ rowsums <- apply(conf\_matrix, 1, sum)

+ colsums <- apply(conf\_matrix, 2, sum)

+ precision <- diag / colsums

+ recall <- diag / rowsums

+ f1 <- 2 \* (precision \* recall) / (precision + recall)

+ data.frame(Precision = precision, Recall = recall, F1 = f1)

+ }

> # Print

> print("k-NN Metrics for best k:")

[1] "k-NN Metrics for best k:"

> knn\_metrics <- compute\_metrics(knn\_confusion\_matrix)

> print(knn\_metrics)

Precision Recall F1

1 0.7894737 0.8333333 0.8108108

2 0.7200000 0.8181818 0.7659574

3 0.6000000 0.4285714 0.5000000

> #Using the NY housing dataset:

> #Loading dataset

> nyhousedataset <- read.csv("C:/Users/chaos/Downloads/NY-House-Dataset.csv")

> n <- nrow(nyhousedataset) # Number of rows

> colnames(nyhousedataset)

[1] "BROKERTITLE" "TYPE" "PRICE"

[4] "BEDS" "BATH" "PROPERTYSQFT"

[7] "ADDRESS" "STATE" "MAIN\_ADDRESS"

[10] "ADMINISTRATIVE\_AREA\_LEVEL\_2" "LOCALITY" "SUBLOCALITY"

[13] "STREET\_NAME" "LONG\_NAME" "FORMATTED\_ADDRESS"

[16] "LATITUDE" "LONGITUDE"

> # Ensure columns are numeric

> nyhousedataset$PRICE <- as.numeric(nyhousedataset$PRICE)

> nyhousedataset$PROPERTYSQFT <- as.numeric(nyhousedataset$PROPERTYSQFT)

> # Remove missing values

> nyhousedataset <- nyhousedataset[!is.na(nyhousedataset$PRICE) & !is.na(nyhousedataset$PROPERTYSQFT), ]

> # Set a price cap (e.g., $2,000,000)

> price\_cap <- 2000000

> # Filter dataset to exclude rows where PRICE exceeds the cap

> nyhousedataset <- nyhousedataset[nyhousedataset$PRICE <= price\_cap, ]

> # Split train/test

> set.seed(123)

> n <- nrow(nyhousedataset)

> train.indexes <- sample(1:n, size = 0.7 \* n)

> train <- nyhousedataset[train.indexes, ]

> test <- nyhousedataset[-train.indexes, ]

> # Train SVM model

> svm\_model <- svm(PRICE ~ PROPERTYSQFT, data = train, type = "eps-regression", kernel = "radial")

> print(svm\_model)

Call:

svm(formula = PRICE ~ PROPERTYSQFT, data = train, type = "eps-regression", kernel = "radial")

Parameters:

SVM-Type: eps-regression

SVM-Kernel: radial

cost: 1

gamma: 1

epsilon: 0.1

Number of Support Vectors: 2517

> # Predict on test set

> predicted\_prices <- predict(svm\_model, newdata = test)

> print(predicted\_prices)

4 9 17 25 49 53 57 61 63 65

330075.9 410123.3 844390.0 735510.6 454229.5 488582.0 1067588.8 752356.2 453165.1 661976.5

66 69 71 73 75 76 81 99 101 102

650897.0 1273096.3 778476.9 611792.3 746962.2 752356.2 582716.4 875090.1 545201.8 791052.7

109 113 118 122 126 128 131 132 133 134

752356.2 1342382.3 752356.2 752356.2 1236957.7 525760.4 360799.8 752356.2 387530.9 1600963.3

135 136 137 138 143 147 148 150 152 153

1267872.1 768052.7 650897.0 840218.1 847078.3 329986.2 752356.2 752356.2 752356.2 621706.9

154 155 159 176 177 180 189 190 196 205

772610.9 1259574.5 775428.3 765956.2 1221003.4 752356.2 386563.3 752356.2 689239.0 430795.8

206 211 215 225 227 228 229 235 244 247

752356.2 774595.4 752356.2 488582.0 752356.2 1315989.5 831312.9 458505.8 752356.2 712107.0

248 249 252 253 259 261 262 266 268 269

772136.4 752356.2 752356.2 461193.0 355973.0 792841.4 752356.2 387530.9 545201.8 434875.7

273 278 281 282 285 287 291 295 301 302

811436.5 339420.2 434875.7 488582.0 1348857.2 752356.2 1375693.6 802199.1 752356.2 647395.0

306 313 315 318 323 324 339 340 350 355

1277374.9 821829.3 410123.3 752356.2 837322.8 752356.2 791052.7 752356.2 378844.5 754454.2

357 358 362 363 366 368 371 378 382 390

632213.9 752356.2 1360971.7 546349.9 369962.6 889977.4 752356.2 752356.2 752356.2 752356.2

392 395 400 403 409 424 428 432 433 434

840218.1 774105.6 752356.2 752356.2 752356.2 752356.2 752356.2 752356.2 605383.4 746440.6

441 445 451 455 460 461 464 470 472 474

765956.2 532604.5 461193.0 813696.3 337557.3 791668.5 434875.7 1734993.2 371457.0 751457.0

475 476 477 478 483 484 485 487 489 490

669543.3 751653.6 752356.2 1282143.7 841815.6 752356.2 842312.5 332092.9 590261.3 603054.4

497 501 502 504 509 512 513 515 516 518

833070.4 772112.9 752356.2 752356.2 488582.0 752356.2 752356.2 752356.2 843006.9 631046.2

521 524 526 530 544 548 558 561 567 570

836118.0 447872.2 752356.2 752356.2 803249.1 1002101.1 1262329.5 746336.9 831406.0 748413.4

574 580 581 586 590 592 593 594 595 596

603054.4 752356.2 434875.7 752356.2 389243.4 603054.4 752976.0 1062719.0 661393.9 752356.2

598 605 606 607 613 616 618 621 626 629

659062.7 756338.4 351504.9 855370.8 752356.2 752356.2 620540.0 752356.2 765956.2 647395.0

637 638 642 651 655 657 658 662 663 665

752356.2 752356.2 793285.4 1692540.3 841983.2 959693.9 752356.2 943714.4 461193.0 746336.9

671 672 673 674 676 691 692 693 694 695

516669.4 752356.2 752356.2 339420.2 769901.3 759338.8 752356.2 752356.2 337236.0 749003.9

696 697 701 707 713 715 721 737 738 741

752356.2 752356.2 333635.6 865530.3 847235.6 752356.2 732767.9 752356.2 807963.2 752356.2

744 746 747 748 755 763 764 770 773 774

791668.5 518370.7 434875.7 831312.9 541187.1 410123.3 945976.0 661393.9 752356.2 1281941.1

790 793 802 806 808 815 816 820 826 831

815526.7 410123.3 626375.9 461193.0 785858.1 752356.2 752356.2 781817.4 752356.2 407756.3

832 834 835 842 844 849 855 856 857 859

579238.3 791069.7 571711.9 752356.2 387957.4 752356.2 679991.4 752356.2 752356.2 752356.2

860 865 867 870 872 883 884 894 895 896

752356.2 752356.2 823843.9 752356.2 752356.2 785568.1 752356.2 752356.2 748942.1 752356.2

897 902 903 912 924 927 930 932 933 941

968969.7 752356.2 582716.4 638053.1 752356.2 743114.7 752356.2 752356.2 488582.0 709840.8

944 945 950 955 956 962 964 981 985 987

661393.9 574026.2 381681.5 661393.9 830796.5 846457.4 488582.0 689239.0 1233034.8 1491265.5

994 995 1000 1001 1005 1007 1012 1014 1025 1030

752356.2 752356.2 560160.6 752356.2 751457.0 718870.9 779924.4 353842.2 778706.5 752356.2

1035 1039 1040 1043 1045 1046 1048 1050 1056 1057

752356.2 752356.2 752356.2 841983.2 752356.2 339420.2 661393.9 734415.1 379246.1 842373.2

1058 1066 1067 1071 1077 1080 1082 1083 1089 1090

1259574.5 752356.2 632213.9 752356.2 752356.2 748093.1 831312.9 752356.2 488582.0 789821.0

1091 1095 1096 1097 1098 1100 1106 1112 1117 1123

752356.2 841691.1 542333.5 542333.5 542333.5 488582.0 603054.4 464972.8 502556.9 752356.2

1128 1136 1137 1160 1162 1166 1176 1181 1182 1183

474779.7 410123.3 1298704.2 752356.2 351504.9 831312.9 848459.6 791052.7 1305376.3 332389.2

1187 1192 1196 1200 1208 1217 1218 1221 1224 1231

752356.2 632213.9 752356.2 809797.0 1304122.9 736603.8 635717.4 732217.8 842633.8 789821.0

1234 1239 1249 1251 1256 1262 1269 1270 1278 1281

760902.7 1178328.7 787559.8 644476.1 1373316.8 846358.0 533747.1 434875.7 701869.6 752356.2

1283 1289 1294 1301 1306 1314 1322 1323 1325 1331

603054.4 367762.2 387530.9 363169.9 1750594.2 752356.2 848507.7 1265445.6 900245.5 752356.2

1338 1344 1366 1376 1377 1378 1382 1385 1403 1405

752356.2 666052.7 847704.6 842538.3 626375.9 752356.2 834580.4 568242.8 814795.4 839650.7

1413 1414 1417 1425 1426 1427 1429 1430 1434 1435

752356.2 752356.2 752356.2 752356.2 836996.6 337208.0 752356.2 752356.2 752356.2 752356.2

1436 1440 1445 1451 1452 1453 1456 1457 1475 1480

752356.2 752356.2 725577.3 594328.6 675352.7 387530.9 753374.9 660228.4 752356.2 847704.6

1485 1488 1489 1491 1502 1503 1505 1509 1512 1519

784743.5 815526.7 831312.9 355973.0 1310441.3 685198.6 752356.2 351504.9 364208.0 821765.1

1522 1527 1528 1530 1533 1535 1541 1542 1551 1553

1332987.1 730564.3 461193.0 736603.8 752356.2 752356.2 752356.2 1297758.5 752356.2 718870.9

1558 1559 1564 1565 1570 1573 1575 1585 1588 1594

752356.2 791668.5 772136.4 752356.2 1367921.4 799737.1 752356.2 745719.8 745734.2 661393.9

1599 1612 1613 1614 1631 1634 1635 1636 1649 1656

779724.7 516669.4 1281941.1 752356.2 752356.2 752356.2 1256939.8 395374.5 434875.7 1257383.6

1657 1658 1659 1661 1664 1668 1680 1681 1684 1685

380864.8 756338.4 545201.8 1372203.3 743114.7 410123.3 434875.7 803249.1 752356.2 778706.5

1689 1690 1692 1694 1695 1700 1702 1705 1706 1709

526330.0 752356.2 1320199.0 752356.2 760007.7 752356.2 752356.2 752356.2 725020.8 752356.2

1712 1714 1715 1719 1721 1727 1730 1731 1742 1744

1062719.0 488582.0 752356.2 603054.4 831312.9 366502.2 468228.3 752356.2 1721216.8 762065.6

1748 1750 1751 1756 1763 1767 1773 1779 1780 1784

810135.6 752356.2 1478696.4 845155.9 752356.2 434875.7 752356.2 1297656.8 752356.2 499749.9

1785 1789 1790 1791 1792 1794 1795 1801 1802 1810

458505.8 842373.2 752356.2 339420.2 334018.7 752356.2 827425.6 752356.2 451042.0 371834.0

1818 1831 1839 1840 1843 1846 1853 1858 1861 1868

848409.5 833799.1 752356.2 752356.2 752356.2 396267.2 752356.2 752356.2 746440.6 752356.2

1870 1872 1874 1875 1876 1879 1883 1902 1904 1905

752356.2 765956.2 527469.5 752356.2 1750594.2 661393.9 749662.9 848203.4 468772.2 1274428.0

1911 1913 1916 1921 1942 1955 1961 1966 1972 1974

752356.2 835953.9 752356.2 847885.0 351504.9 752356.2 756183.0 339420.2 663723.9 574026.2

1979 1982 1985 1987 1988 1995 1997 2002 2006 2016

718870.9 783701.9 752356.2 752356.2 752356.2 762254.0 751653.6 889977.4 1002101.1 633381.7

2025 2028 2029 2033 2039 2042 2048 2051 2052 2056

828669.3 478624.6 825605.2 986604.6 329986.2 1248452.2 889977.4 1356849.3 411553.8 1364788.3

2057 2061 2064 2070 2071 2073 2076 2078 2079 2083

758726.2 752356.2 351504.9 834510.8 752356.2 752356.2 575183.9 434875.7 1256630.0 730012.1

2086 2089 2094 2097 2107 2115 2116 2117 2134 2135

752356.2 752356.2 545201.8 530891.5 1261735.1 752356.2 803249.1 341689.4 752356.2 848459.6

2137 2140 2144 2152 2156 2160 2163 2167 2168 2169

716622.3 1339384.4 915048.0 752356.2 752356.2 752356.2 1002101.1 377151.8 752356.2 339420.2

2174 2175 2186 2191 2193 2195 2197 2204 2205 2207

752356.2 752356.2 405411.5 752356.2 1323162.5 752356.2 810135.6 637469.2 461193.0 791052.7

2214 2220 2225 2227 2230 2234 2235 2253 2270 2274

1299152.3 339420.2 661393.9 463350.4 519505.8 649729.8 752356.2 752356.2 1267034.3 798082.7

2284 2287 2291 2293 2296 2297 2298 2301 2307 2310

752356.2 488582.0 545201.8 752356.2 603054.4 752356.2 387530.9 752356.2 845834.1 845834.1

2322 2325 2326 2329 2330 2332 2335 2340 2342 2346

398064.8 752356.2 384577.6 752356.2 752356.2 752356.2 968969.7 579817.8 1152283.2 752356.2

2351 2354 2357 2360 2364 2367 2368 2371 2381 2383

1277755.1 752356.2 711540.9 488582.0 842373.2 700726.3 1291155.8 752356.2 752356.2 754711.7

2384 2385 2392 2395 2402 2406 2409 2423 2425 2427

589680.5 351504.9 451572.0 1750594.2 334621.7 339420.2 752356.2 752356.2 752356.2 752356.2

2428 2431 2434 2443 2454 2456 2460 2461 2466 2467

752356.2 430288.7 430288.7 393160.9 765956.2 351504.9 752356.2 752356.2 752356.2 752356.2

2471 2482 2483 2487 2488 2491 2492 2495 2502 2505

689815.4 545201.8 770812.6 752356.2 752356.2 752356.2 434875.7 334420.1 828017.3 339420.2

2509 2511 2516 2518 2523 2525 2527 2530 2533 2536

827425.6 1256939.8 752356.2 333038.1 752356.2 1520739.9 410123.3 752356.2 752356.2 753224.6

2541 2543 2545 2546 2550 2554 2559 2577 2583 2584

820521.5 752356.2 793488.4 1344177.1 752356.2 1212772.3 1273576.7 754200.6 678832.5 831312.9

2585 2588 2597 2599 2604 2608 2611 2616 2617 2618

842373.2 488582.0 752356.2 598399.0 752356.2 334284.2 411553.8 461193.0 752356.2 752356.2

2627 2634 2642 2643 2647 2649 2653 2655 2657 2659

752356.2 752356.2 831134.3 516669.4 752356.2 752356.2 459042.4 398969.6 752356.2 653814.3

2661 2662 2667 2678 2679 2683 2690 2691 2694 2695

410123.3 464972.8 488582.0 752356.2 614707.2 752356.2 752356.2 847375.6 768052.7 1375736.1

2697 2705 2709 2710 2719 2725 2726 2728 2730 2731

752356.2 1302101.1 746440.6 716622.3 434875.7 752356.2 657313.7 387530.9 752356.2 752356.2

2738 2742 2745 2753 2758 2760 2772 2775 2779 2780

453165.1 752356.2 752356.2 752356.2 668961.8 752356.2 752356.2 752356.2 752356.2 747159.6

2784 2786 2789 2791 2793 2794 2796 2798 2799 2804

712107.0 608878.4 367762.2 416863.4 842633.8 831312.9 488582.0 589680.5 752356.2 752356.2

2810 2813 2824 2826 2828 2829 2838 2842 2845 2850

752356.2 792653.7 752356.2 414921.2 367400.4 752356.2 707569.5 568242.8 752356.2 752356.2

2852 2859 2861 2868 2870 2871 2873 2874 2877 2880

752356.2 831312.9 371457.0 389674.4 752356.2 752356.2 516669.4 334420.1 545201.8 424755.6

2886 2890 2891 2892 2894 2898 2901 2902 2915 2917

367762.2 424755.6 1328425.4 752356.2 752356.2 752356.2 752356.2 752356.2 772136.4 847927.7

2924 2930 2934 2938 2939 2941 2942 2951 2953 2956

344902.1 750595.8 752356.2 752356.2 752356.2 828601.4 675352.7 752356.2 516669.4 752356.2

2957 2959 2960 2961 2963 2966 2967 2971 2979 2983

752356.2 752356.2 416863.4 505369.4 815526.7 434875.7 752356.2 752356.2 572868.9 696142.5

2989 2994 2999 3000 3005 3012 3017 3018 3020 3030

488582.0 707569.5 1305376.3 752356.2 351504.9 752356.2 834013.4 752356.2 707569.5 752356.2

3031 3034 3041 3043 3049 3053 3056 3058 3072 3075

815526.7 752356.2 784203.2 752356.2 1356899.1 740954.0 752356.2 1062719.0 332092.9 752356.2

3077 3078 3080 3081 3084 3087 3089 3091 3095 3099

1369196.2 752356.2 752356.2 752356.2 716622.3 1375693.6 812952.7 747838.0 752356.2 1026222.0

3107 3108 3116 3123 3124 3130 3133 3142 3143 3145

410123.3 1370939.7 396715.1 1233034.8 434875.7 410123.3 506495.9 684620.6 390106.4 411076.1

3146 3150 3155 3157 3167 3168 3171 3174 3175 3177

494153.3 1328105.9 614707.2 583876.4 845155.9 752356.2 632797.8 631046.2 752356.2 643892.2

3178 3181 3187 3188 3190 3194 3195 3203 3209 3211

752356.2 752356.2 410123.3 446819.8 375677.0 759338.8 752356.2 752356.2 1355827.9 685776.3

3212 3218 3220 3221 3224 3233 3234 3236 3242 3243

424755.6 752356.2 752356.2 603054.4 752356.2 545201.8 396715.1 408700.5 461193.0 367762.2

3254 3255 3262 3266 3269 3270 3274 3281 3282 3284

659645.6 816961.1 384160.3 559007.5 434875.7 752356.2 752356.2 350376.2 748093.1 424755.6

3295 3305 3313 3316 3317 3321 3327 3334 3337 3343

752356.2 448399.3 574026.2 377250.6 1424433.3 574026.2 752356.2 752356.2 745713.4 826448.1

3348 3353 3354 3355 3381 3382 3391 3398 3401 3402

752356.2 815526.7 375286.8 1436804.2 752356.2 752356.2 476974.8 545201.8 1123027.2 516669.4

3405 3410 3413 3414 3420 3433 3439 3442 3444 3449

367400.4 461193.0 461193.0 831312.9 752356.2 752356.2 752356.2 612958.1 748413.4 752356.2

3451 3458 3463 3466 3468 3471 3475 3480 3481 3483

752356.2 587358.0 752356.2 417839.3 752356.2 545201.8 1581488.4 752356.2 752356.2 746440.6

3484 3486 3488 3494 3502 3503 3504 3506 3509 3512

842373.2 517236.3 367762.2 704723.2 752356.2 752356.2 661393.9 765956.2 752356.2 775428.3

3514 3519 3520 3522 3526 3531 3535 3542 3544 3547

752356.2 752356.2 373355.4 545201.8 398516.7 557854.8 1745337.4 330322.5 488582.0 752356.2

3548 3550 3551 3553 3567 3568 3573 3578 3581 3594

1265544.8 752356.2 601308.2 842947.0 752356.2 752356.2 752356.2 752356.2 823843.9 752356.2

3596 3610 3612 3618 3621 3623 3627 3630 3631 3632

752356.2 494153.3 848194.9 696142.5 752356.2 752356.2 752356.2 395820.4 461193.0 330221.3

3641 3647 3651 3652 3663 3665 3666 3673 3679 3696

661393.9 752356.2 1154584.5 331443.5 765956.2 752356.2 842373.2 752356.2 632213.9 367762.2

3699 3706 3707 3713 3718 3721 3730 3731 3732 3734

429782.3 330075.9 752356.2 577500.2 1259574.5 752356.2 468228.3 436930.7 699009.2 614707.2

3739 3741 3743 3759 3763 3764 3765 3767 3773 3774

752356.2 752356.2 436930.7 752356.2 387530.9 752356.2 752356.2 752356.2 847927.7 785298.8

3777 3780 3782 3784 3785 3792 3802 3803 3806 3807

752356.2 522346.4 752356.2 752356.2 752356.2 752356.2 363514.5 1650641.6 752356.2 785298.8

3809 3828 3829 3838 3843 3844 3845 3846 3850 3853

752356.2 477524.4 1366107.6 752356.2 701869.6 752356.2 756338.4 752356.2 719992.9 499749.9

3857 3867 3868 3875 3882 3887 3891 3896 3900 3902

752356.2 497508.4 752356.2 752356.2 516669.4 331788.9 330211.6 707569.5 772136.4 716059.2

3904 3908 3912 3918 3920 3925 3929 3930 3939 3942

394043.2 545201.8 773401.1 745670.7 752356.2 752356.2 774051.4 752356.2 846645.2 752356.2

3945 3949 3958 3961 3972 3974 3975 3977 3980 3985

752356.2 332092.9 730012.1 752356.2 448399.3 752356.2 534890.3 410123.3 341259.9 752356.2

3989 3991 4001 4004 4006 4012 4025 4028 4032 4050

752356.2 359806.6 511575.0 752356.2 750109.7 1291433.7 758332.9 339420.2 752356.2 831406.0

[ reached getOption("max.print") -- omitted 186 entries ]

> # Plot predicted vs actual prices

> plot(test$PRICE, predicted\_prices,

+ xlab = "Actual Price",

+ ylab = "Predicted Price",

+ main = "Predicted vs. Actual Price",

+ col = "blue", pch = 16)

> abline(0, 1, col = "red")

> # Feature boxplot

> boxplot(predicted\_prices, main = "Boxplot of Predicted Prices")

> # Feature-class plots

> featurePlot(x = train[, "PROPERTYSQFT", drop = FALSE], y = train$PRICE, plot = "ellipse")

NULL

> featurePlot(x = train[, "PROPERTYSQFT", drop = FALSE], y = train$PRICE, plot = "box")

NULL

> scales <- list(x = list(relation = "free"), y = list(relation = "free"))

> featurePlot(x = train[, "PROPERTYSQFT", drop = FALSE], y = train$PRICE, plot = "density", scales = scales)

NULL

> # ggplot scatter plot

> ggplot(nyhousedataset, aes(x = PROPERTYSQFT, y = PRICE, colour = TYPE)) +

+ geom\_point() +

+ theme\_minimal() +

+ labs(title = "Scatter Plot of Price vs Property Square Footage",

+ x = "Property Square Footage",

+ y = "Price")

> # Linear model

> # Train

> lm\_model <- lm(PRICE ~ PROPERTYSQFT, data = train)

> summary(lm\_model)

Call:

lm(formula = PRICE ~ PROPERTYSQFT, data = train)

Residuals:

Min 1Q Median 3Q Max

-890735 -278250 -55960 222786 1354656

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 3.612e+05 1.933e+04 18.68 <2e-16 \*\*\*

PROPERTYSQFT 2.439e+02 9.979e+00 24.45 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 413800 on 2765 degrees of freedom

Multiple R-squared: 0.1777, Adjusted R-squared: 0.1774

F-statistic: 597.5 on 1 and 2765 DF, p-value: < 2.2e-16

> # Predict on test set

> lm\_predicted <- predict(lm\_model, newdata = test)

> # Plot predicted vs actual prices for linear model

> plot(test$PRICE, lm\_predicted,

+ xlab = "Actual Price",

+ ylab = "Predicted Price",

+ main = "Linear Model: Predicted vs. Actual Price",

+ col = "black", pch = 16)

> abline(0, 1, col = "magenta")

>