# K-Mean Clusterig

if (!require("ggplot2")) install.packages("ggplot2")

if (!require("gridExtra")) install.packages("gridExtra") library(ggplot2)

library(gridExtra)

grade\_input <- read.csv("D:\\grades\_km\_input.csv")

kmdata <- as.matrix(grade\_input[, c("English", "Math", "Science")])

wss <- sapply(1:15, function(k) sum(kmeans(kmdata, k, nstart = 25)$withinss)) dev.new()

plot(1:15, wss, type = "b", xlab = "Clusters", ylab = "WSS", main = "Elbow Method") readline("Press [Enter] to show cluster plots...")

km <- kmeans(kmdata, 3, nstart = 25) df <- as.data.frame(kmdata) df$cluster <- factor(km$cluster) centers <- as.data.frame(km$centers)

g1 <- ggplot(df, aes(English, Math, color = cluster)) +

geom\_point() + geom\_point(data = centers, aes(English, Math), color = "black", size = 5, shape = 8) +

labs(title = "English vs Math")

g2 <- ggplot(df, aes(English, Science, color = cluster)) +

geom\_point() + geom\_point(data = centers, aes(English, Science), color = "black", size = 5, shape = 8) +

labs(title = "English vs Science")

g3 <- ggplot(df, aes(Math, Science, color = cluster)) +

geom\_point() + geom\_point(data = centers, aes(Math, Science), color = "black", size = 5, shape = 8) +

labs(title = "Math vs Science")

dev.new()

grid.arrange(g1, g2, g3, ncol = 1, top = "High School Student Cluster Analysis") readline("Press [Enter] to exit...")

# Apriori

# Install required packages install.packages("arules") install.packages("arulesViz") install.packages("RColorBrewer")

# Load libraries library(arules) library(arulesViz) library(RColorBrewer)

# Load the Groceries dataset data("Groceries")

# Summary and class of the dataset summary(Groceries) class(Groceries)

# Generate association rules

rules <- apriori(Groceries, parameter = list(supp = 0.02, conf = 0.2))

# Summary and inspection of rules summary(rules)

inspect(rules[1:10])

# Plot top 20 item frequencies (relative) itemFrequencyPlot(Groceries,

topN = 20,

col = brewer.pal(8, 'Pastel2'),

main = 'Relative Item Frequency Plot', type = "relative",

ylab = "Item Frequency (Relative)")

# Generate frequent itemsets of length 2 itemsets\_2 <- apriori(Groceries,

parameter = list(minlen = 2, maxlen = 2, support = 0.02, target = "frequent

itemsets"))

summary(itemsets\_2) inspect(itemsets\_2[1:10])

# Generate frequent itemsets of length 3 itemsets\_3 <- apriori(Groceries,

parameter = list(minlen = 3, maxlen = 3, support = 0.02, target = "frequent

itemsets"))

summary(itemsets\_3) inspect(itemsets\_3)

# Linear Regression

# Define data vectors years\_of\_exp = c(7, 5, 1, 3)

salary\_in\_lakhs = c(21, 13, 6, 8)

# Create a data frame

employee.data = data.frame(years\_of\_exp, salary\_in\_lakhs)

# View the dataset print(employee.data)

# Build the linear regression model

model = lm(salary\_in\_lakhs ~ years\_of\_exp, data = employee.data)

# Show model summary summary(model)

# Plot the data with regression line

plot(salary\_in\_lakhs ~ years\_of\_exp, data = employee.data, main = "Salary vs Years of Experience",

xlab = "Years of Experience", ylab = "Salary (in Lakhs)",

col = "blue", pch = 16)

# Add regression line abline(model, col = "red", lwd = 2)

# Logistic Regression

# Install and load packages install.packages("ISLR") install.packages("InformationValue") install.packages("caret")

library(ISLR) library(InformationValue) library(caret)

# Load data

data <- ISLR::Default print(head(data)) summary(data) nrow(data)

# Split data into train and test set.seed(1)

sample <- sample(c(TRUE, FALSE), nrow(data), replace = TRUE, prob = c(0.7, 0.3)) train <- data[sample, ]

test <- data[!sample, ] nrow(train)

nrow(test)

# Fit logistic regression model

model <- glm(default ~ student + balance + income, family = "binomial", data = train)

summary(model)

# Predict probabilities on test set

predicted\_probs <- predict(model, test, type = "response")

# Convert probabilities to class labels using 0.5 cutoff predicted\_classes <- ifelse(predicted\_probs > 0.5, "Yes", "No")

# Confusion matrix

cm <- confusionMatrix(factor(predicted\_classes), factor(test$default), positive = "Yes") print(cm)

# Decision Tree

# Load dataset

dataset <- read.csv("C:\\Users\\lenovo\\Downloads\\Social\_Network\_Ads.csv") dataset <- dataset[3:5]

dataset$Purchased <- factor(dataset$Purchased, levels = c(0, 1)) print(dataset)

# Install and load packages install.packages("caTools") install.packages("rpart") install.packages("ElemStatLearn")

library(caTools) library(rpart) library(ElemStatLearn)

# Split data set.seed(123)

split <- sample.split(dataset$Purchased, SplitRatio = 0.75) training\_set <- subset(dataset, split == TRUE)

test\_set <- subset(dataset, split == FALSE)

# Feature scaling

training\_set[, -3] <- scale(training\_set[, -3])

test\_set[, -3] <- scale(test\_set[, -3])

# Train Decision Tree

classifier <- rpart(formula = Purchased ~ ., data = training\_set)

# Predict

y\_pred <- predict(classifier, newdata = test\_set[, -3], type = "class")

# Confusion matrix

cm <- table(test\_set[, 3], y\_pred) print(cm)

# Visualization — Training set set <- training\_set

X1 <- seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 <- seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01) grid\_set <- expand.grid(X1, X2)

colnames(grid\_set) <- c("Age", "EstimatedSalary")

y\_grid <- predict(classifier, newdata = grid\_set, type = "class")

# Force new plot window for training set dev.new()

plot(set[, -3],

main = "Decision Tree Classification (Training set)", xlab = "Age", ylab = "Estimated Salary",

xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE) points(grid\_set, pch = ".", col = ifelse(y\_grid == 1, "springgreen3", "tomato")) points(set, pch = 21, bg = ifelse(set[, 3] == 1, "green4", "red3"))

# Visualization — Test set set <- test\_set

X1 <- seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 <- seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01) grid\_set <- expand.grid(X1, X2)

colnames(grid\_set) <- c("Age", "EstimatedSalary")

y\_grid <- predict(classifier, newdata = grid\_set, type = "class")

# Force new plot window for test set dev.new()

plot(set[, -3],

main = "Decision Tree Classification (Test set)", xlab = "Age", ylab = "Estimated Salary",

xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE) points(grid\_set, pch = ".", col = ifelse(y\_grid == 1, "springgreen3", "tomato")) points(set, pch = 21, bg = ifelse(set[, 3] == 1, "green4", "red3"))

# Plot Decision Tree dev.new() plot(classifier) text(classifier)

# Naive Bayes Classification

dataset <- read.csv("D:/socialnetworking.csv")

dataset <- dataset[3:5] # Keep Age, EstimatedSalary, Purchased dataset$Purchased <- factor(dataset$Purchased, levels = c(0, 1))

install.packages("caTools") library(caTools) set.seed(123)

split <- sample.split(dataset$Purchased, SplitRatio = 0.75) training\_set <- subset(dataset, split == TRUE)

test\_set <- subset(dataset, split == FALSE)

training\_set[, 1:2] <- scale(training\_set[, 1:2])

test\_set[, 1:2] <- scale(test\_set[, 1:2])

install.packages("e1071") library(e1071)

classifier <- naiveBayes(x = training\_set[, 1:2], y = training\_set$Purchased) y\_pred <- predict(classifier, newdata = test\_set[, 1:2])

cm <- table(Actual = test\_set$Purchased, Predicted = y\_pred) print("Confusion Matrix:")

print(cm)

set <- training\_set

X1 <- seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 <- seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01) grid\_set <- expand.grid(X1, X2)

colnames(grid\_set) <- c("Age", "EstimatedSalary") y\_grid <- predict(classifier, newdata = grid\_set)

dev.new() # Open new window plot(set[, -3],

main = "Naive Bayes (Training Set)", xlab = "Age", ylab = "Estimated Salary", xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE) points(grid\_set, pch = ".", col = ifelse(y\_grid == 1, "springgreen3", "tomato")) points(set, pch = 21, bg = ifelse(set[, 3] == 1, "green4", "red3"))

set <- test\_set

X1 <- seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 <- seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01) grid\_set <- expand.grid(X1, X2)

colnames(grid\_set) <- c("Age", "EstimatedSalary") y\_grid <- predict(classifier, newdata = grid\_set)

dev.new() # Open new window plot(set[, -3],

main = "Naive Bayes (Test Set)",

xlab = "Age", ylab = "Estimated Salary", xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE) points(grid\_set, pch = ".", col = ifelse(y\_grid == 1, "springgreen3", "tomato")) points(set, pch = 21, bg = ifelse(set[, 3] == 1, "green4", "red3"))

# Text Analysis

# Read dataset

dataset\_original <- read.delim('D:\\2020\\Big Data Analytics\\Practical\\P6 NLP\\Restaurant\_Reviews.tsv',

quote = '', stringsAsFactors = FALSE)

# Install and load text mining packages install.packages('tm') install.packages('SnowballC') library(tm)

library(SnowballC)

# Text preprocessing

corpus <- VCorpus(VectorSource(dataset\_original$Review)) corpus <- tm\_map(corpus, content\_transformer(tolower)) corpus <- tm\_map(corpus, removeNumbers)

corpus <- tm\_map(corpus, removePunctuation) corpus <- tm\_map(corpus, removeWords, stopwords()) corpus <- tm\_map(corpus, stemDocument)

corpus <- tm\_map(corpus, stripWhitespace)

# Create Document-Term Matrix

dtm <- DocumentTermMatrix(corpus) dtm <- removeSparseTerms(dtm, 0.999)

# Convert to data frame

dataset <- as.data.frame(as.matrix(dtm)) dataset$Liked <- dataset\_original$Liked

# Encode target as factor

dataset$Liked <- factor(dataset$Liked, levels = c(0, 1))

# Install and load caTools install.packages('caTools') library(caTools)

# Split dataset set.seed(123)

split <- sample.split(dataset$Liked, SplitRatio = 0.8) training\_set <- subset(dataset, split == TRUE) test\_set <- subset(dataset, split == FALSE)

# Install and load randomForest install.packages('randomForest') library(randomForest)

# Train Random Forest

classifier <- randomForest(x = training\_set[, -ncol(training\_set)], y = training\_set$Liked,

ntree = 10)

# Predict

y\_pred <- predict(classifier, newdata = test\_set[, -ncol(test\_set)])

# Confusion Matrix

cm <- table(Actual = test\_set$Liked, Predicted = y\_pred) print("Confusion Matrix:")

print(cm)