**Deepankar Sharma**

Roll No: 233512013

Program: MCA

Year: 2025

Semester: 4

Data science laboratory

Term Work

**Exercise 1: Data Cleaning and Visualization (Air Quality)**

**Experiment No.: 1**  
**Date:**

**Problem Definition:**

Identify and handle missing values in the Air Quality dataset and visualize pollution trends.

**Theory Background:**

* **Missing Values Handling:** Removal or imputation (mean/median).
* **Visualization:** Line charts, histograms, and boxplots.

**R Program:**

data(airquality)

summary(airquality)

airquality$Ozone[is.na(airquality$Ozone)] <- mean(airquality$Ozone, *na.rm* = TRUE)

airquality$Solar.R[is.na(airquality$Solar.R)] <- mean(airquality$Solar.R, *na.rm* = TRUE)

library(ggplot2)

ggplot(airquality, aes(*x* = factor(Month), *y* = Ozone)) +

  geom\_boxplot() +

  labs(*title* = "Ozone Levels by Month", *x* = "Month", *y* = "Ozone")

**OUTPUT :**

*A screenshot of a computer

AI-generated content may be incorrect.*

**Exercise 2: Data Cleaning and Visualization (Titanic Dataset)**

**Experiment No.: 2**  
**Date:**

**Problem Definition:**

Handle missing values in the Titanic dataset and visualize survival patterns.

**Theory Background:**

* **Missing Values Handling:** Imputation, dropping missing data.
* **Visualization:** Bar plots, pie charts, histograms.

**R Program:**

# install.packages("titanic")

library(titanic)

data <- titanic\_train

# Check missing values

summary(data)

# Impute missing age with mean

data$Age[is.na(data$Age)] <- mean(data$Age, *na.rm*=TRUE)

# Drop unnecessary columns

data <- subset(data, *select* = -c(Cabin))

# Visualize survival

library(ggplot2)

ggplot(data, aes(*x* = factor(Survived))) + geom\_bar() +

  labs(*title*="Survival Count", *x*="Survived", *y*="Count")

ggplot(data, aes(*x* = factor(Pclass), *fill* = factor(Survived))) +

  geom\_bar(*position* = "dodge") + labs(*title*="Survival by Passenger Class")

**OUTPUT**

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 3: Naïve Bayes Classifier (Titanic)**

**Experiment No.: 3**  
**Date:**

**Problem Definition:**

Predict survival using Naïve Bayes Classifier.

**Theory Background:**

* Naïve Bayes algorithm.
* Use of the e1071 package.

**R Program:**

library(e1071)

library(titanic)

# Load Titanic dataset

data <- titanic\_train

# Remove missing values

data <- na.omit(data)

# Convert variables to factors

data$Survived <- factor(data$Survived)

data$Sex <- factor(data$Sex)

data$Pclass <- factor(data$Pclass)

# Train-test split (70% train, 30% test)

set.seed(123)

train\_idx <- sample(1:nrow(data), 0.7 \* nrow(data))

train <- data[train\_idx, ]

test <- data[-train\_idx, ]

# Train Naive Bayes model

model <- naiveBayes(Survived ~ Pclass + Sex + Age, *data* = train)

# Make predictions

pred <- predict(model, test)

# Confusion Matrix

conf\_matrix <- table(*Predicted* = pred, *Actual* = test$Survived)

print(conf\_matrix)

# Accuracy

accuracy <- sum(diag(conf\_matrix)) / sum(conf\_matrix)

cat("Accuracy:", round(accuracy \* 100, 2), "%\n")

# Predict class probabilities

prob\_pred <- predict(model, test, *type* = "raw")  # returns matrix with prob for 0 and 1

test$prob\_survived <- prob\_pred[, "1"]

test$Sex <- factor(test$Sex)

# Plot: Probability of Survival by Age and Sex

library(ggplot2)

ggplot(test, aes(*x* = Age, *y* = prob\_survived, *color* = Sex)) +

  geom\_point(*alpha* = 0.5) +

  geom\_smooth(*method* = "loess", *se* = FALSE) +

  labs(*title* = "Predicted Probability of Survival by Age and Sex",

*x* = "Age", *y* = "Predicted Probability (Survived = 1)") +

  theme\_minimal()

test$pred\_class <- pred

# Compute accuracy by Pclass

library(dplyr)

accuracy\_by\_pclass <- test %>%

  group\_by(Pclass) %>%

  summarise(*accuracy* = mean(pred\_class == Survived))

# Plot

ggplot(accuracy\_by\_pclass, aes(*x* = Pclass, *y* = accuracy, *fill* = Pclass)) +

  geom\_col() +

  labs(*title* = "Model Accuracy by Passenger Class",

*x* = "Passenger Class", *y* = "Accuracy") +

  scale\_y\_continuous(*labels* = scales::percent) +

  theme\_minimal()

# Convert confusion matrix to data frame

conf\_df <- as.data.frame(conf\_matrix)

colnames(conf\_df) <- c("Predicted", "Actual", "Freq")

ggplot(conf\_df, aes(*x* = Actual, *y* = Predicted, *fill* = Freq)) +

  geom\_tile(*color* = "white") +

  geom\_text(aes(*label* = Freq), *color* = "black", *size* = 5) +

  scale\_fill\_gradient(*low* = "white", *high* = "steelblue") +

  labs(*title* = "Confusion Matrix Heatmap") +

  theme\_minimal()

**OUTPUT**

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 4: Support Vector Machine (SVM) (mtcars)**

**Experiment No.: 4**  
**Date:**

**Problem Definition:**

Train SVM model to classify cars based on automatic/manual transmission.

**Theory Background:**

* SVM theory.
* e1071 package implementation.

**R Program:**

# Load packages

library(e1071)

library(ggplot2)

# Prepare data

data(mtcars)

mtcars$am <- factor(mtcars$am)

# Train SVM model

svm\_model <- svm(am ~ mpg + hp + wt, *data* = mtcars, *kernel* = "linear")  # using linear kernel for interpretability

# Predictions

pred\_svm <- predict(svm\_model, mtcars)

# Confusion matrix

conf\_matrix <- table(*Predicted* = pred\_svm, *Actual* = mtcars$am)

print(conf\_matrix)

# Create prediction grid

mpg\_seq <- seq(min(mtcars$mpg), max(mtcars$mpg), *length* = 100)

wt\_seq <- seq(min(mtcars$wt), max(mtcars$wt), *length* = 100)

grid <- expand.grid(*mpg* = mpg\_seq, *wt* = wt\_seq)

grid$hp <- median(mtcars$hp)  # fix hp at median

# Predict on grid

grid$pred <- predict(svm\_model, *newdata* = grid)

# Plot decision boundary

ggplot() +

  geom\_tile(*data* = grid, aes(*x* = mpg, *y* = wt, *fill* = pred), *alpha* = 0.3) +

  geom\_point(*data* = mtcars, aes(*x* = mpg, *y* = wt, *color* = am), *size* = 3) +

  labs(*title* = "SVM Classification: Transmission (am)",

*x* = "Miles per Gallon (mpg)", *y* = "Weight (wt)",

*fill* = "Predicted", *color* = "Actual") +

  theme\_minimal()

# Extract support vectors

support\_vectors <- mtcars[svm\_model$index, ]

ggplot(mtcars, aes(*x* = mpg, *y* = wt, *color* = am)) +

  geom\_point(*size* = 3) +

  geom\_point(*data* = support\_vectors, aes(*x* = mpg, *y* = wt), *shape* = 8, *size* = 4, *color* = "black") +

  labs(*title* = "Support Vectors Highlighted",

*subtitle* = "Black stars are support vectors",

*x* = "mpg", *y* = "wt") +

  theme\_minimal()

**OUTPUT A screenshot of a computer

AI-generated content may be incorrect.**

A screen shot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 5: k-Nearest Neighbors (MNIST)**

**Experiment No.: 5**  
**Date:**

**Problem Definition:**

Classify handwritten digits using k-NN classifier.

**Theory Background:**

* k-NN algorithm.
* class package.

**R Program:**

# Load necessary libraries

library(class)

# Load MNIST data (make sure files are in your working directory)

train <- read.csv("data/archive/mnist\_train.csv")

test <- read.csv("data/archive/mnist\_test.csv")

# Convert labels to factors

train$label <- as.factor(train$label)

test$label <- as.factor(test$label)

# Use smaller subsets for faster testing (k-NN is slow on large data)

train\_small <- train[1:2000, ]   # 2,000 training samples

test\_small <- test[1:300, ]      # 300 test samples

# Apply k-NN with k = 5

pred <- knn(*train* = train\_small[, -1],

*test* = test\_small[, -1],

*cl* = train\_small$label,

*k* = 5)

# Confusion Matrix

conf\_matrix <- table(*Predicted* = pred, *Actual* = test\_small$label)

print(conf\_matrix)

# Accuracy

accuracy <- mean(pred == test\_small$label)

cat("Accuracy:", round(accuracy \* 100, 2), "%\n")

# Remove zero-variance columns before PCA

pixel\_data <- train\_small[, -1]  # exclude label column

# Keep only columns with non-zero variance

pixel\_data <- pixel\_data[, apply(pixel\_data, 2, var) != 0]

# Apply PCA

pca <- prcomp(pixel\_data, *center* = TRUE, *scale.* = TRUE)

# Create a dataframe with the first 2 principal components and labels

pca\_data <- data.frame(pca$x[, 1:2], *label* = train\_small$label)

# Plot clusters

library(ggplot2)

ggplot(pca\_data, aes(*x* = PC1, *y* = PC2, *color* = label)) +

  geom\_point(*alpha* = 0.6) +

  labs(*title* = "PCA of MNIST (Train Set - First 2 PCs)",

*x* = "Principal Component 1", *y* = "Principal Component 2") +

  theme\_minimal()

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.** **A screen shot of a computer

AI-generated content may be incorrect.**

**Exercise 6: Multiple Linear Regression (Boston Housing)**

**Experiment No.: 6**  
**Date:**

**Problem Definition:**

Predict house prices using multiple linear regression.

**Theory Background:**

* Multiple regression theory.
* lm() function.

**R Program:**

# Load necessary library

library(MASS)

# Load dataset

data(Boston)

# Fit multiple linear regression model

model\_lm <- lm(medv ~ ., *data* = Boston)

# Model summary

summary(model\_lm)

# Predict on training data

predicted\_medv <- predict(model\_lm, *newdata* = Boston)

# Plot

plot(Boston$medv, predicted\_medv,

*main* = "Actual vs Predicted House Prices",

*xlab* = "Actual medv",

*ylab* = "Predicted medv",

*col* = "blue", *pch* = 20)

abline(*a* = 0, *b* = 1, *col* = "red", *lwd* = 2)

plot(model\_lm, *which* = 1)  # Residuals vs Fitted

**OUTPUT :**

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

**Exercise 7: Logistic Regression (Titanic)**

**Experiment No.: 7**  
**Date:**

**Problem Definition:**

Predict survival using logistic regression.

**Theory Background:**

* Logistic regression theory.

**R Program:**

# Load necessary package

library(titanic)

library(ggplot2)

library(dplyr)

# Load dataset

data <- titanic\_train

# Data Cleaning: Remove NAs

data <- na.omit(data)

# Ensure proper types

data$Survived <- factor(data$Survived)

data$Sex <- factor(data$Sex)

data$Pclass <- factor(data$Pclass)

# Train logistic regression model

model <- glm(Survived ~ Pclass + Sex + Age, *data* = data, *family* = binomial)

# Model Summary

summary(model)

# Predict probabilities

data$predicted\_prob <- predict(model, *type* = "response")

# Classify as 0 or 1 using 0.5 threshold

data$predicted\_class <- ifelse(data$predicted\_prob > 0.5, 1, 0)

# Confusion Matrix

conf\_matrix <- table(*Predicted* = data$predicted\_class, *Actual* = data$Survived)

print(conf\_matrix)

# Accuracy

accuracy <- mean(data$predicted\_class == as.numeric(as.character(data$Survived)))

cat("Accuracy:", round(accuracy \* 100, 2), "%\n")

ggplot(data, aes(*x* = Age, *y* = predicted\_prob, *color* = Sex)) +

  geom\_point(*alpha* = 0.6) +

  geom\_smooth(*method* = "loess") +

  labs(*title* = "Predicted Survival Probability by Age and Sex",

*x* = "Age", *y* = "Predicted Probability") +

  theme\_minimal()

avg\_pred <- data %>%

  group\_by(Pclass) %>%

  summarise(*Average\_Predicted\_Survival* = mean(predicted\_prob))

ggplot(avg\_pred, aes(*x* = Pclass, *y* = Average\_Predicted\_Survival, *fill* = Pclass)) +

  geom\_col() +

  labs(*title* = "Average Predicted Survival by Passenger Class",

*x* = "Pclass", *y* = "Predicted Survival Probability") +

  theme\_minimal()

**OUTPUT :**

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 8: Polynomial Regression (Synthetic Data)**

**Experiment No.: 8**  
**Date:**

**Problem Definition:**

Fit polynomial regression on synthetic nonlinear data.

**Theory Background:**

* Polynomial regression theory.

**R Program:**

set.seed(100)

x <- 1:100

y <- 5 + 2\*x - 0.05\*x^2 + rnorm(100,0,10)

data\_poly <- data.frame(x, y)

poly\_model <- lm(y ~ poly(x, 2), *data* = data\_poly)

summary(poly\_model)

plot(x, y)

lines(x, predict(poly\_model), *col* = "red", *lwd* = 2)

**OUTPUT (Screenshots from RStudio):**

*A screenshot of a computer program

AI-generated content may be incorrect.*

A computer screen shot of a computer screen

AI-generated content may be incorrect.

**Exercise 9: Hierarchical Clustering (mtcars)**

**Experiment No.: 9**  
**Date:**

**Problem Definition:**

Perform hierarchical clustering on mtcars dataset.

**Theory Background:**

* Clustering theory.
* Dendrograms.

**R Program:**

data(mtcars)

d <- dist(mtcars)

hc <- hclust(d)

plot(hc, *main* = "Hierarchical Clustering Dendrogram")

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 10: DBSCAN Clustering (Noisy Data)**

**Experiment No.: 10**  
**Date:**

**Problem Definition:**

Cluster data using DBSCAN algorithm.

**Theory Background:**

* Density-based clustering.

**R Program:**

library(dbscan)

set.seed(123)

data <- matrix(rnorm(200), *ncol*=2)

data[51:100, ] <- data[51:100, ] + 3

db <- dbscan(data, *eps* = 0.5, *minPts* = 5)

plot(data, *col* = db$cluster + 1L, *main* = "DBSCAN Clustering")

**OUTPUT :**

A screenshot of a computer

AI-generated content may be incorrect.