

Practical 1

```
# Install and load dataset
```

```
install.packages("ISLR")
```

```
library(ISLR)
```

```
data(Smarket)
```

```
summary(Smarket)
```

```
# Logistic Regression Model
```

```
logit_model <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,  
                   data = Smarket, family = binomial)
```

```
summary(logit_model)
```

```
logit_probs <- predict(logit_model, type = "response")
```

```
logit_pred <- ifelse(logit_probs > 0.5, "Up", "Down")
```

```
logit_cm <- table(Predicted = logit_pred, Actual = Smarket$Direction)
```

```
logit_cm
```

```
logit_accuracy <- mean(logit_pred == Smarket$Direction)
```

```
logit_accuracy
```

Practical 2

```
install.packages("MASS")
install.packages("glmnet")
install.packages("caret")

library(MASS)
library(glmnet)
library(caret)

data("Boston")
set.seed(123)

train_index <- createDataPartition(Boston$medv, p = 0.7, list = FALSE)
train_data <- Boston[train_index, ]
test_data <- Boston[-train_index, ]

x_train <- model.matrix(medv ~ ., data = train_data)[, -1]
y_train <- train_data$medv
x_test <- model.matrix(medv ~ ., data = test_data)[, -1]
y_test <- test_data$medv

# Linear Regression
lm_model <- lm(medv ~ ., data = train_data)
lm_preds <- predict(lm_model, newdata = test_data)
lm_mse <- mean((lm_preds - y_test)^2)

# Ridge
ridge_cv <- cv.glmnet(x_train, y_train, alpha = 0)
ridge_lambda <- ridge_cv$lambda.min
ridge_model <- glmnet(x_train, y_train, alpha = 0, lambda = ridge_lambda)
ridge_preds <- predict(ridge_model, s = ridge_lambda, newx = x_test)
ridge_mse <- mean((ridge_preds - y_test)^2)

# Lasso
```

```
lasso_cv <- cv.glmnet(x_train, y_train, alpha = 1)
lasso_lambda <- lasso_cv$lambda.min
lasso_model <- glmnet(x_train, y_train, alpha = 1, lambda = lasso_lambda)
lasso_preds <- predict(lasso_model, s = lasso_lambda, newx = x_test)
lasso_mse <- mean((lasso_preds - y_test)^2)

cat("Linear MSE:", lm_mse, "\n")
cat("Ridge MSE :", ridge_mse, "\n")
cat("Lasso MSE :", lasso_mse, "\n")
```

Practical 3

```
library(ISLR)

library(ggplot2)

data(Wage)

fit_poly <- lm(wage ~ poly(age, 4), data = Wage)

summary(fit_poly)

ggplot(Wage, aes(x = age, y = wage)) +
  geom_point(alpha = 0.3, color = "blue") +
  stat_smooth(method = "lm", formula = y ~ poly(x, 4),
             color = "red", size = 1.2) +
  labs(title = "4th Degree Polynomial Regression of Wage on Age",
       x = "Age", y = "Wage") +
  theme_minimal()
```

Practical 4

```
library(ISLR)
```

```
library(rpart)
```

```
library(rpart.plot)
```

```
data(Carseats)
```

```
Carseats$High <- as.factor(ifelse(Carseats$Sales > 8, "Yes", "No"))
```

```
tree_model <- rpart(High ~ . - Sales, data = Carseats, method = "class")
```

```
rpart.plot(tree_model, type = 2, extra = 104,  
           fallen.leaves = TRUE, cex = 0.7)
```

Practical 5

```
library(MASS)

library(gbm)

library(randomForest)

library(caret)


set.seed(123)

data(Boston)


train_index <- createDataPartition(Boston$medv, p = 0.7, list = FALSE)

train_data <- Boston[train_index, ]

test_data <- Boston[-train_index, ]


boost_model <- gbm(
  medv ~ ., data = train_data,
  distribution = "gaussian",
  n.trees = 5000,
  interaction.depth = 4,
  shrinkage = 0.01,
  cv.folds = 5,
  verbose = FALSE
)


boost_pred <- predict(boost_model, newdata = test_data, n.trees = 5000)

boost_rmse <- sqrt(mean((test_data$medv - boost_pred)^2))

cat("Boosting RMSE:", boost_rmse, "\n")


bag_model <- randomForest(
  medv ~ ., data = train_data,
  mtry = ncol(train_data) - 1,
  ntree = 500
)


bag_pred <- predict(bag_model, newdata = test_data)
```

```
bag_rmse <- sqrt(mean((test_data$medv - bag_pred)^2))
```

```
cat("Bagging RMSE:", bag_rmse, "\n")
```

```
cat("Boosting RMSE:", boost_rmse, "\n")
```

```
cat("Bagging RMSE :", bag_rmse, "\n")
```

Practical 6

```
library(ISLR)

library(e1071)

data("Carseats")

Carseats$High <- ifelse(Carseats$Sales > 8, "Yes", "No")
Carseats$High <- as.factor(Carseats$High)
Carseats <- Carseats[, -which(names(Carseats) == "Sales")]

set.seed(123)
index <- sample(1:nrow(Carseats), nrow(Carseats) * 0.7)
train_data <- Carseats[index, ]
test_data <- Carseats[-index, ]

svm_model <- svm(High ~ ., data = train_data, kernel = "linear", cost = 1, scale = TRUE)
summary(svm_model)

pred <- predict(svm_model, test_data)
confusion_matrix <- table(Predicted = pred, Actual = test_data$High)
confusion_matrix

accuracy <- mean(pred == test_data$High)
cat("Accuracy: ", accuracy, "\n")

svm_plot <- svm(High ~ Price + Income, data = train_data, kernel = "linear", cost = 1)
plot(svm_plot, train_data[, c("Price", "Income", "High")])
```


Practical 7

```
data("USArrests")

USArrests_scaled <- scale(USArrests)

pca_result <- prcomp(USArrests_scaled, center = TRUE, scale. = TRUE)

summary(pca_result)
pca_result$rotation
pca_result$x

plot(pca_result, type = "l", main = "Scree Plot")
biplot(pca_result, scale = 0)

library(factoextra)
fviz_pca_biplot(pca_result,
  repel = TRUE,
  col.var = "red",
  col.ind = "blue",
  labelsize = 3,
  title = "PCA Biplot - USArrests")
```