data=read.csv("C:/Users/prach/Downloads/Advertising\_Data.csv")

data

anova(model)

# Perform exploratory data analysis (EDA)

summary(data)

head(data)

str(data)

# Fit a linear regression model

model <- lm( Product\_Sold ~TV+ Billboards+Google\_Ads+ Social\_Media+Influencer\_Marketing+ Affiliate\_Marketing, data = data)

summary(model)

# Check assumptions of linear regression

# Extract residuals from the model

residuals <- residuals(model)

# Visual inspection

# Histogram

hist(residuals, main = "Histogram of Residuals")

# Q-Q plot

qqnorm(residuals, main = "Q-Q Plot of Residuals")

qqline(residuals)

# Statistical test

shapiro.test(residuals)

# Assessing the impact of different marketing channels on product sales

cor(data[, c("TV", "Billboards", "Google\_Ads", "Social\_Media", "Influencer\_Marketing", "Affiliate\_Marketing")], data$Product\_Sold)

#predictions <- predict(model,data = new\_data)

# View the predictions

#predictions

tv=(data$TV)

tv

subset(-1)

product\_sold=data$Product\_Sold

product\_sold

t1=t.test(tv,product\_sold)

billboards=data$Billboards

billboards

t2=t.test(billboards,product\_sold)

google\_ads=data$Google\_Ads

t3=t.test(google\_ads,product\_sold)

t4=t.test(data$Social\_Media,product\_sold)

t5=t.test(data$Influencer\_Marketing,product\_sold)

t6=t.test(data$Affiliate\_Marketing,product\_sold)

rs=summary(model)$r.squared

d1=data$TV

d2=data$Product\_Sold

model=lm(Product\_Sold ~TV,data=data)

s=summary(model)

rs=summary(model)$r.squared

d2=data$Billboards

model=lm(Product\_Sold ~Billboards,data=data)

s=summary(model1)

rs=s$r.squared

d3=lm(Product\_Sold ~Google\_Ads,data=data)

s=summary(m3)

rs=s$r.squared

m1=lm(Product\_Sold~Social\_Media,data=data)

summary(m1)

r1=summary(m1)$r.squared

m2=lm(Product\_Sold~Influencer\_Marketing,data=data)

summary(m2)

r2=summary(m2)$r.squared

m3=lm(Product\_Sold~data$Affiliate\_Marketing,data=data)

summary(m3)

r3=summary(m3)$r.squared

#subset selection method

library(stats)

library(leaps)

regfit.full=regsubsets(Product\_Sold~.,data=data)

summary(regfit.full)

#multicolaniarity

##v=vif(model)

#vif=(1/(1-rs))

library(olsrr)

o=ols\_vif\_tol(model)

library(car)

v=vif(model)

b=barplot(v,main="VIF values",horiz=TRUE,col="red")

reg=data.frame(data)

reg

#finding correlation plot

corre=cor(reg)

corre

library(corrplot)

corrplot(corre,method="circle",bg="grey")

# Extract residuals and fitted values from the model

residuals <- residuals(model)

fitted\_values <- fitted(model)

# Plot residuals vs. fitted values

plot(fitted\_values, residuals,

xlab = "Fitted Values", ylab = "Residuals",

main = "Residuals vs. Fitted Values Plot")

abline(h = 0, col = "red", lty = 2) # Add a horizontal reference line at 0

# Extract independent variables from the model

independent\_variables <- model$model[, -1] # Exclude the intercept column

# Plot residuals vs. each independent variable

par(mfrow = c(2, 2)) # Setting up a 2x2 plot layout

for (i in 1:ncol(independent\_variables)) {

plot(independent\_variables[, i], residuals,

xlab = colnames(independent\_variables)[i], ylab = "Residuals",

main = paste("Residuals vs.", colnames(independent\_variables)[i]))

abline(h = 0, col = "red", lty = 2) # Add a horizontal reference line at 0

}

library(corrplot)

corrplot(cor(model))

# Load necessary libraries

library(readr)

library(caret)

library(ggplot2)

# Load the dataset

#data <- read\_csv("your\_dataset.csv")

# Splitting the dataset into features (X) and target variable (y)

X <- subset(data, select = -Product\_Sold) # Features

y <- data$Product\_Sold # Target variable

# Splitting the dataset into training and testing sets

set.seed(42) # for reproducibility

trainIndex <- createDataPartition(y, p = .8, list = FALSE)

X\_train <- X[trainIndex, ]

X\_test <- X[-trainIndex, ]

y\_train <- y[trainIndex]

y\_test <- y[-trainIndex]

length(y)

# Training the model

model <-lm(y ~x,cbind(y\_train, X\_train))

# Predicting the values

y\_pred <- predict(model, newdata = X\_test)

# Visualizing the predictions

ggplot() +

geom\_point(aes(x = y\_test, y = y\_pred), color = "blue") +

geom\_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +

labs(x = "Actual Product Sold", y = "Predicted Product Sold") +

ggtitle("Actual vs Predicted Product Sold")

y=(data$Product\_Sold)

model2=lm(y~.,data=data)

library(olsrr)

ols\_step\_all\_possible(model2)

k=ols\_step\_best\_subset(model2)

k$metrics

p=plot(k)

# Load required libraries

library(caret)

library(randomForest)

# Load the provided dataset

#data <- read.csv("product\_ad\_data.csv")

# Set seed for reproducibility

set.seed(123)

# Split the data into training and testing sets

train\_index <- createDataPartition(data$Product\_Sold, p = 0.8, list = FALSE)

train\_data <- data[train\_index, ]

test\_data <- data[-train\_index, ]

# Train a random forest regression model

model <- train(Product\_Sold ~ ., data = train\_data, method = "rf")

# Train the KNN model using the training data

# Assuming your target variable is "Product\_Sold" and other columns are features

library(class)

k <- 5 # choose the value of k

predicted <-knn(train = train\_data[, -ncol(train\_data)], test = test\_data[, -ncol(test\_data)], cl = train\_data[, ncol(train\_data)], k = k)

# Evaluate the model using the testing data

accuracy <- sum(predicted == test\_data[, ncol(test\_data)]) / nrow(test\_data)

print(paste("Accuracy:", accuracy))

library(glmnet)

# Train a logistic regression model using the training data

model <- glm(Product\_Sold ~ ., data = train\_data, family = binomial)

# Make predictions using the logistic regression model

predicted\_logreg <- predict(model, newdata = test\_data, type = "response") > 0.5

# Calculate accuracy of logistic regression model

accuracy\_logreg <- sum(predicted\_logreg == test\_data[, "Product\_Sold"]) / nrow(test\_data)

print(paste("Accuracy of Logistic Regression model:", accuracy\_logreg))

# Compare accuracies of both models

if (accuracy\_knn > accuracy\_logreg) {

print("KNN model has higher accuracy.")

} else if (accuracy\_knn < accuracy\_logreg) {

print("Logistic Regression model has higher accuracy.")

} else {

print("Both models have the same accuracy.")

}

# Function to predict regressors for given input data

predict\_regressors <- function(input\_data) {

predicted\_regressors <- predict(model, newdata = 500,600)

return(predicted\_regressors)

}

# Example usage:

new\_data <- data.frame(

TV = c(500, 600),

Billboards = c(200, 300),

Google\_Ads = c(400, 500),

Social\_Media = c(300, 400),

Influencer\_Marketing = c(100, 200),

Affiliate\_Marketing = c(200, 300)

)

predicted\_amounts <- predict\_regressors(new\_data)

print(predicted\_amounts)

# Load required libraries

library(caret) # for data preprocessing and model evaluation

library(ggplot2) # for visualization

# Load your dataset

#data <- read.csv()

# Preprocess your data (e.g., handle missing values, encode categorical variables, normalize data)

# For example, if you have missing values:

# data <- na.omit(data)

# Split your dataset into training and testing sets (e.g., 70% training, 30% testing)

set.seed(123) # for reproducibility

train\_index <- createDataPartition(data$Product\_Sold, p = 0.7, list = FALSE)

train\_data <- data[train\_index, ]

test\_data <- data[-train\_index, ]

# Train the KNN model using the training data

k <- 5 # choose the value of k

knn\_model <- train(Product\_Sold ~ ., data = train\_data, method = "knn", trControl = trainControl(method = "cv", number = 10), tuneGrid = expand.grid(k = k))

accuracy <- mean(knn\_model == test\_data$Product\_Sold)

# Train the multiple linear regression model using the training data

lm\_model <- train(Product\_Sold ~ ., data = train\_data, method = "lm", trControl = trainControl(method = "cv", number = 10))

# Compare model performances using resampling

compare\_models <- resamples(list(KNN = knn\_model, MultipleRegression = lm\_model))

# Summarize the results

summary(compare\_models)

# Visualize the results

dotplot(compare\_models)

# Rename columns in new\_data1 to match the original dataset

n\_new\_data1<- c("TV", "Billboards", "Google\_Ads", "Social\_Media", "Influencer\_Marketing", "Affiliate\_Marketing")

# Convert new\_data1 to a data frame if it's not already

new\_data1 <- data.frame(n\_new\_data1)

# Now try to predict using the model

predictions <- predict(model, newdata = new\_data1)

# Now try to predict using the model

predictions <- predict(model, newdata =n\_new\_data1)

new\_data1=data.frame(product\_sold=c(890,350,650,230,500,150))

predict(model,newdata=new\_data1)

predict(model,newdata=new\_data1,interval="confidence")

predict(model,newdata=new\_data1,interval="prediction",level=0.95)

pred\_int=predict(model,interval="prediction",level=0.95)

# Assuming you have already loaded your dataset into a variable called 'data'

# Split the data into training and test sets (e.g., using the 'caret' package)

library(caret)

set.seed(123) # for reproducibility

train\_index <- createDataPartition(data$target\_variable, p = 0.8, list = FALSE)

train\_data <- data[train\_index, ]

test\_data <- data[-train\_index, ]

# Train the k-NN model

library(class) # for k-NN algorithm

k <- 5 # number of neighbors

knn\_model <- knn(train = train\_data[, predictors], test = test\_data[, predictors], cl = train\_data$target\_variable, k = k)

# Evaluate accuracy

accuracy <- mean(knn\_model == test\_data$target\_variable)

print(paste("Accuracy of k-NN model:", round(accuracy \* 100, 2), "%"))

length(knn\_model)

length(test\_data$Product\_Sold)

# Subset or reshape knn\_model to match the length of test\_data$Product\_Sold

knn\_model\_subset <- knn\_model[1:length(test\_data$Product\_Sold)]