Title: ": Searching for Similarity: Classification" Authors: Gaurang Goel (GXG190015) Date: 03/22/2023

#Data Source https://www.kaggle.com/datasets/camnugent/california-housing-prices

```
# Load necessary packages
library(caret)
```

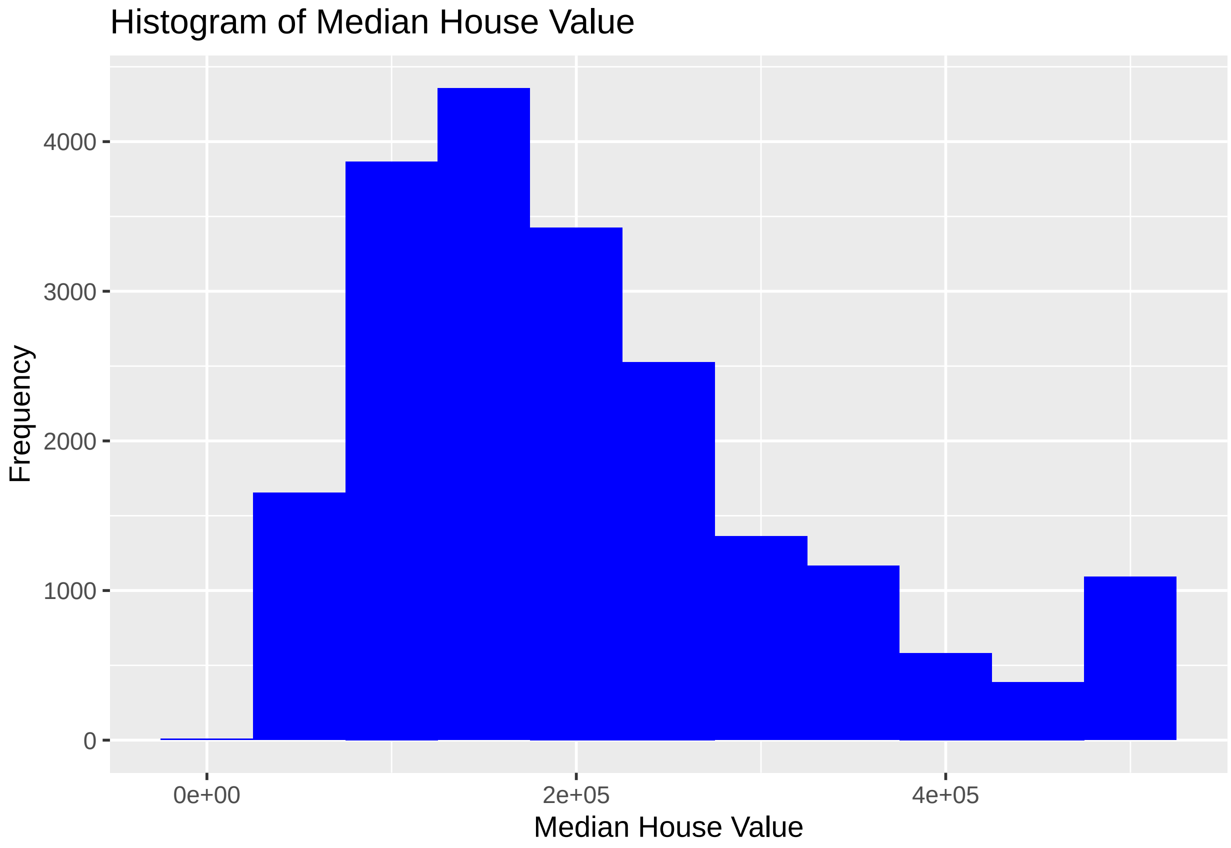
```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

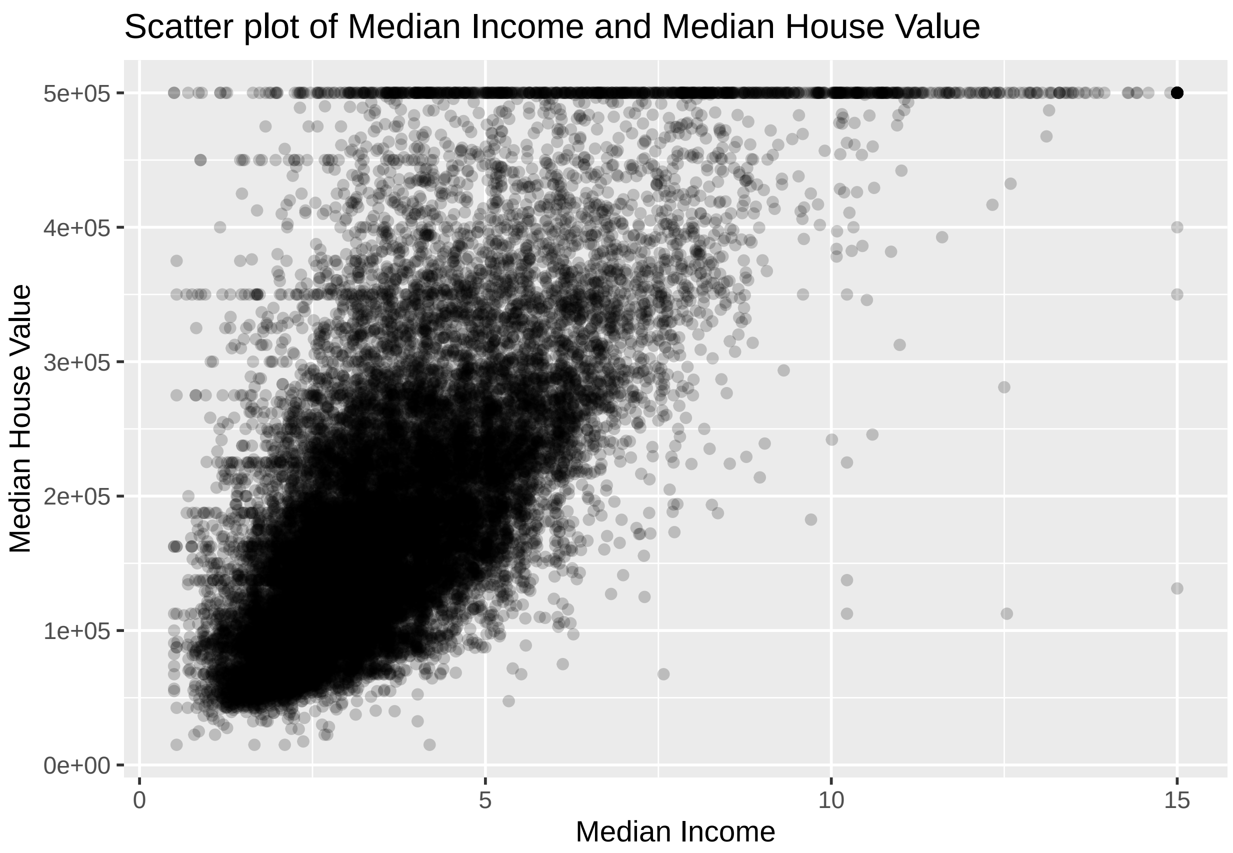
```
library(e1071)
library(rpart)
library(ggplot2)

# Load data
data <- read.csv("/Users/gauranggoel/Downloads/housing.csv")

# Convert median_house_value to binary outcome
data$above_median <- ifelse(data$median_house_value >= median(data$median_house_value),
                            "Yes", "No")

# Remove rows with missing values
data <- na.omit(data)

# Summary statistics
summary(data)
```

```
##    longitude        latitude     housing_median_age  total_rooms   
##  Min.   :-124.3   Min.   :32.54   Min.   : 1.00      Min.   :    2  
##  1st Qu.:-121.8   1st Qu.:33.93   1st Qu.:18.00      1st Qu.: 1450  
##  Median :-118.5   Median :34.26   Median :29.00      Median : 2127  
##  Mean   :-119.6   Mean   :35.63   Mean   :28.63      Mean   : 2636  
##  3rd Qu.:-118.0   3rd Qu.:37.72   3rd Qu.:37.00      3rd Qu.: 3143  
##  Max.   :-114.3   Max.   :41.95   Max.   :52.00      Max.   :39320  
##  total_bedrooms     population      households     median_income    
##  Min.   :   1.0   Min.   :    3   Min.   :   1.0   Min.   : 0.4999  
##  1st Qu.: 296.0   1st Qu.:  787   1st Qu.: 280.0   1st Qu.: 2.5637  
##  Median : 435.0   Median : 1166   Median : 409.0   Median : 3.5365  
##  Mean   : 537.9   Mean   : 1425   Mean   : 499.4   Mean   : 3.8712  
##  3rd Qu.: 647.0   3rd Qu.: 1722   3rd Qu.: 604.0   3rd Qu.: 4.7440  
##  Max.   :6445.0   Max.   :35682   Max.   :6082.0   Max.   :15.0001  
##  median_house_value ocean_proximity    above_median      
##  Min.   : 14999     Length:20433       Length:20433      
##  1st Qu.:119500     Class :character   Class :character  
##  Median :179700     Mode  :character   Mode  :character  
##  Mean   :206864                                          
##  3rd Qu.:264700                                          
##  Max.   :500001                                          
```

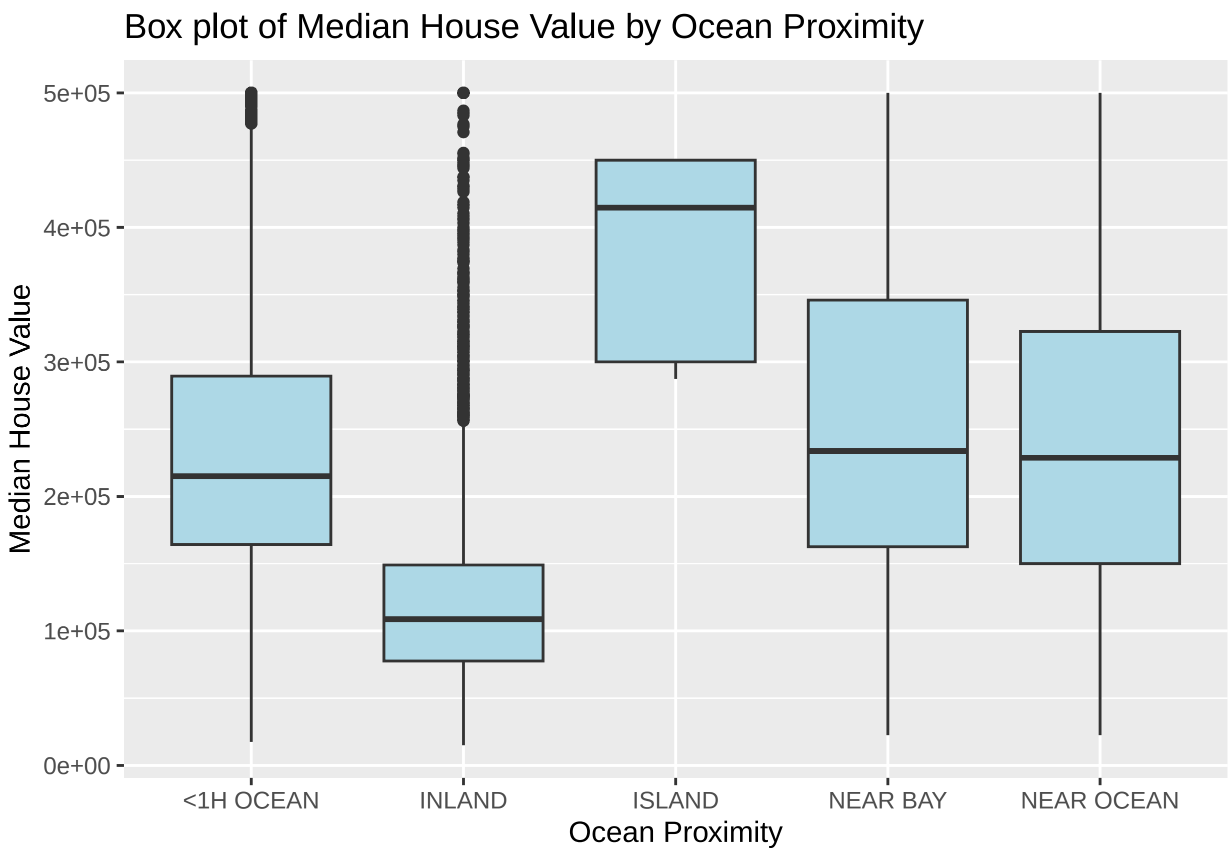
```r
# Histogram of median_house_value
ggplot(data, aes(median_house_value)) +
  geom_histogram(binwidth = 50000, fill = "blue") +
  labs(title = "Histogram of Median House Value", x = "Median House Value", y = "Frequency")
```



```r
# Scatter plot of median_income and median_house_value
ggplot(data, aes(median_income, median_house_value)) +
  geom_point(alpha = 0.2, color = "black") +
  labs(title = "Scatter plot of Median Income and Median House Value",
       x = "Median Income", y = "Median House Value")
```



```
# Box plot of median_house_value by ocean_proximity
ggplot(data, aes(ocean_proximity, median_house_value)) +
  geom_boxplot(fill = "lightblue") +
  labs(title = "Box plot of Median House Value by Ocean Proximity",
       x = "Ocean Proximity", y = "Median House Value")
```



```r
# Split data into train and test sets
set.seed(123)
trainIndex <- createDataPartition(data$above_median, p = .8, list = FALSE, times = 1)
training <- data[trainIndex, ]
testing <- data[-trainIndex, ]

# Logistic regression
lrModel <- train(above_median ~ ., data = training, method = "glm", family = "binomial",
                 control = glm.control(maxit = 100))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
lrPredictions <- predict(lrModel, newdata = testing)

# Convert testing$above_median to factor with same levels as lrPredictions
testing$above_median <- factor(testing$above_median, levels = levels(lrPredictions))

lrAccuracy <- confusionMatrix(lrPredictions, testing$above_median)$overall['Accuracy']
print(paste("Logistic regression accuracy: ", lrAccuracy))
```

```
## [1] "Logistic regression accuracy:  0.99828641370869"
```

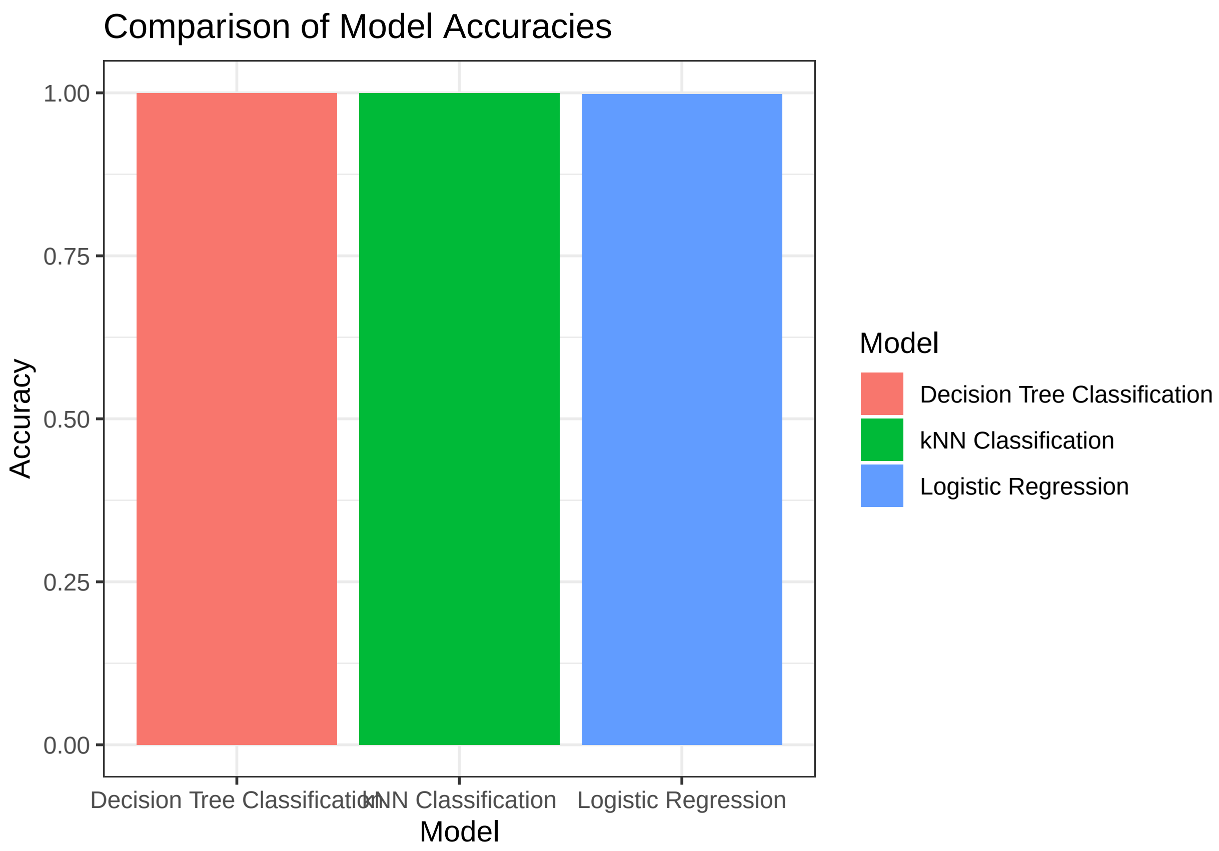
```r
# kNN classification
knnModel <- train(above_median ~ ., data = training, method = "knn")
knnPredictions <- predict(knnModel, newdata = testing)
knnAccuracy <- confusionMatrix(knnPredictions, testing$above_median)$overall['Accuracy']
print(paste("kNN classification accuracy: ", knnAccuracy))
```

```
## [1] "kNN classification accuracy:  1"
```

```r
# Decision tree classification
dtModel <- train(above_median ~ ., data = training, method = "rpart")
dtPredictions <- predict(dtModel, newdata = testing)
dtAccuracy <- confusionMatrix(dtPredictions, testing$above_median)$overall['Accuracy']
print(paste("Decision tree classification accuracy: ", dtAccuracy))
```

```
## [1] "Decision tree classification accuracy:  1"
```

```r
# Compare Results
results <- data.frame(Model = c("Logistic Regression", "kNN Classification",
                                "Decision Tree Classification"),
                      Accuracy = c(lrAccuracy, knnAccuracy, dtAccuracy))

# Plot a bar chart of the accuracy for each model
ggplot(results, aes(x = Model, y = Accuracy, fill = Model)) +
  geom_bar(stat = "identity") +
  ggtitle("Comparison of Model Accuracies") +
  xlab("Model") +
  ylab("Accuracy") +
  theme_bw()
```



#Analysis

Logistic regression is a popular binary classification approach. It uses a logistic function, which is an S-shaped curve that translates any input to a number between 0 and 1. The dependent variable in this code is "above median," a binary variable that indicates if a house's median price is greater than the median price of all houses in the dataset. Location, housing characteristics, and demography are some of the independent factors. Since it is a simple and efficient technique that works well with huge datasets, logistic regression obtained excellent accuracy.

The k-Nearest Neighbors (kNN) method is a non-parametric classification and regression algorithm. It operates by determining the k nearest neighbors in the training set to a given observation in the test set and then classifying the observation based on the majority class among its k nearest neighbors. I used the train() function from the caret package in this code to fit a kNN model to the training data. The function employs the default k value of 5. kNN is a straightforward technique that works well with datasets that have well-defined clusters or borders. It is, however, susceptible to noisy or irrelevant information and can be computationally costly for big datasets.

Decision trees are supervised learning algorithms that may be utilized for classification and regression applications. They operate by recursively dividing the feature space into smaller parts depending on the predictor variables' values, until a halting requirement is reached. With this code, I utilized the caret package's train() function to fit a decision tree model to the training data using the rpart() method. The Gini index is used as the splitting criteria in the procedure, which assesses a node's impurity based on the proportion of observations in each class. Decision trees can manage non-linear correlations between predictor factors and result variables and are interpretable.