



AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)
FACULTY OF SCIENCE AND TECHNOLOGY

FINAL TERM PROJECT

INTRODUCTION TO DATA SCIENCE

Summer 2022-2023

Section: B

Submitted By

Name: Md. Nasifur Rahman

ID: 20-43651-2

Submitted To

Touhedul Islam

Assistant Professor

Department of Computer Science

Date of Submission: August 16,2023

Source: <https://www.kaggle.com/datasets/uciml/mushroom-classification>

Introduction:

This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family Mushroom drawn from The Audubon Society Field Guide to North American Mushrooms. Each species is identified as definitely edible, poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like "leaflets three, let it be" for Poisonous Oak and Ivy.

Attribute Information:

Class: To define if it is edible or poisonous.

Cap Shape: The shape of the mushroom's cap (e.g., convex, flat, bell, etc.).

Cap Surface: The texture of the mushroom's cap (e.g., fibrous, smooth, scaly, etc.).

Cap Color: The color of the mushroom's cap (e.g., brown, yellow, red, etc.).

Bruises: Presence or absence of bruises on the mushroom (bruises, no bruises).

Odor: The odor of the mushroom (e.g., almond, foul, anise, etc.).

Gill Attachment: How the gills are attached to the cap (e.g., free, attached).

Gill Spacing: The spacing between gills (e.g., close, crowded).

Gill Size: The size of the gills (e.g., broad, narrow).

Gill Color: The color of the gills (e.g., black, pink, white, etc.).

Stalk Shape: The shape of the mushroom's stalk (e.g., enlarging, tapering).

Stalk Root: The root type of the stalk (e.g., bulbous, club, equal, rooted).

Stalk Surface Above Ring: The texture of the stalk above the ring (e.g., fibrous, smooth).

Stalk Surface Below Ring: The texture of the stalk below the ring (e.g., fibrous, smooth).

Stalk Color Above Ring: The color of the stalk above the ring (e.g., brown, white).

Stalk Color Below Ring: The color of the stalk below the ring (e.g., brown, white).

Veil Type: The type of veil on the mushroom (partial, universal).

Veil Color: The color of the veil (e.g., brown, white).

Ring Number: The number of rings on the stalk (e.g., none, one, two).

Ring Type: The type of ring on the stalk (e.g., cobwebby, evanescent).

Spore Print Color: The color of the spore print (e.g., black, brown, white, etc.).

Population: The abundance of mushrooms in a specific population (e.g., abundant, clustered).

Habitat: The habitat where the mushrooms are typically found (e.g., grasses, woods, meadows, etc.).

Import Dataset:

```
1 library(caret)
2 library(ggplot2)
3 library(lattice)
4 library(readr)
5 library(class)
6
7 data<- read.csv("E:/mushrooms.csv",header = TRUE,sep=',')
8 str(data)
```

Output:

```
> data<- read.csv("E:/mushrooms.csv",header = TRUE,sep=',')
> str(data)
'data.frame': 8124 obs. of 23 variables:
 $ class      : chr  "p" "e" "e" "p" ...
 $ cap.shape  : chr  "x" "x" "b" "x" ...
 $ cap.surface: chr  "s" "s" "s" "y" ...
 $ cap.color  : chr  "n" "y" "w" "w" ...
 $ bruises   : chr  "t" "t" "t" "t" ...
 $ odor       : chr  "p" "a" "l" "p" ...
 $ gill.attachment: chr  "f" "f" "f" "f" ...
 $ gill.spacing: chr  "c" "c" "c" "c" ...
 $ gill.size  : chr  "n" "b" "b" "n" ...
 $ gill.color : chr  "k" "k" "n" "n" ...
 $ stalk.shape: chr  "e" "e" "e" "e" ...
 $ stalk.root : chr  "e" "c" "c" "e" ...
 $ stalk.surface.above.ring: chr  "s" "s" "s" "s" ...
 $ stalk.surface.below.ring: chr  "s" "s" "s" "s" ...
 $ stalk.color.above.ring : chr  "w" "w" "w" "w" ...
 $ stalk.color.below.ring : chr  "w" "w" "w" "w" ...
 $ veil.type  : chr  "p" "p" "p" "p" ...
 $ veil.color : chr  "w" "w" "w" "w" ...
 $ ring.number: chr  "o" "o" "o" "o" ...
 $ ring.type  : chr  "p" "p" "p" "p" ...
 $ spore.print.color : chr  "k" "n" "n" "k" ...
 $ population: chr  "s" "n" "n" "s" ...
 $ habitat    : chr  "u" "g" "m" "u" ...
```

Explanation:

- **caret:** A comprehensive package for training and evaluating machine learning models.
- **ggplot2:** A popular package for creating data visualizations.
- **lattice:** A package for creating lattice plots, which are similar to traditional scatterplots but offer more flexibility.
- **readr:** A package for reading and parsing data efficiently.
- **class:** A package that provides various classification algorithms, including the k-nearest neighbors (k-NN) algorithm.
- **str():** The str() function is used to display the structure of the data object. It provides information about the data frame's structure, including the names of its columns, data types of each column, and a preview of the data.

Data Pre-processing:

- There is no missing value.

```
> missing_values <- c("", "NA", "N/A", "NULL", "?")
> missing_counts <- sapply(data, function(col) sum(col %in% missing_values))
> print(missing_counts)
```

	class	cap.shape	cap.surface	cap.color
0	0	0	0	0
bruises	0	odor	gill.attachment	gill.spacing
0	0	0	0	0
gill.size	gill.color	stalk.shape	stalk.root	
0	0	0	2480	
stalk.surface.above.ring	stalk.surface.below.ring	stalk.color.above.ring	stalk.color.below.ring	
0	0	0	0	
veil.type	veil.color	ring.number	ring.type	
0	0	0	0	
spore.print.color	population	habitat		
0	0	0		

- Converted all values in numeric.

```
class classe classp cap.shapex cap.shapexf cap.shapex cap.shapex cap.shapex cap.surfaceg cap.surfacef cap.surfacef
1 0 1 0 0 0 0 0 0 0 1 0 0 0
2 1 0 0 0 0 0 0 0 0 1 0 0 0
3 1 0 0 0 0 0 0 0 0 0 1 0 0
4 0 1 0 0 0 0 0 0 0 0 0 1 0
5 1 0 0 0 0 0 0 0 0 1 0 0 0
cap.colorc cap.colore cap.colorg cap.colorn cap.colorp cap.colorr cap.coloru cap.colow cap.colory bruiseit odorc
1 0 0 0 0 1 0 0 0 0 0 0 1 0
2 0 0 0 0 0 0 0 0 0 0 0 1 0
3 0 0 0 0 0 0 0 0 0 0 1 0 0
4 0 0 0 0 0 0 0 0 0 0 1 0 0
5 0 0 0 1 0 0 0 0 0 0 0 0 0
odorf odorl odorm odorn odorp odors odory gill.attachmentf gill.spacingw gill.sizen gill.colore gill.colory
1 0 0 0 0 1 0 0 1 0 0 0 0
2 0 0 0 0 0 0 0 1 0 0 0 0
3 0 1 0 0 0 0 0 1 0 0 0 0
4 0 0 0 0 1 0 0 1 0 0 0 0
5 0 0 0 1 0 0 0 1 1 0 0 0
gill.colordh gill.colork gill.colorm gill.colono gill.colorp gill.colorr gill.coloru gill.colow gill.colory
1 0 1 0 0 0 0 0 0 0 0 0 0
2 0 1 0 0 0 0 0 0 0 0 0 0
3 0 0 1 0 0 0 0 0 0 0 0 0
4 0 0 1 0 0 0 0 0 0 0 0 0
5 0 0 1 0 0 0 0 0 0 0 0 0
stalk.shapet stalk.rootb stalk.rootc stalk.rootd stalk.roote stalk.rootf stalk.surface.above.ringk stalk.surface.above.rings
1 0 0 0 1 0 0 0 0 1
2 0 0 0 1 0 0 0 0 1
3 0 0 0 1 0 0 0 0 1
4 0 0 0 1 0 0 0 0 1
5 1 0 0 0 1 0 0 0 1
stalk.surface.above.ringy stalk.surface.below.ringk stalk.surface.below.rings stalk.surface.below.ringy
1 0 0 1 0
```

- Calculated Z-Score normalization

```
> zscore_normalized_data <- scale(encoded_data)
> head(zscore_normalized_data)
```

	classe	classp	cap.shapex	cap.shapexf	cap.shapex	cap.shapex	cap.shapex	cap.surfaceg	cap.surfacef	cap.surfacef
1	-1.0365489	1.0365489	-0.02219347	-0.7961608	-0.3368573	-0.06288113	1.1054186	-0.02219347	1.4758508	
2	0.9646211	-0.9646211	-0.02219347	-0.7961608	-0.3368573	-0.06288113	1.1054186	-0.02219347	1.4758508	
3	0.9646211	-0.9646211	-0.02219347	-0.7961608	-0.3368573	-0.06288113	-0.9045233	-0.02219347	1.4758508	
4	-1.0365489	1.0365489	-0.02219347	-0.7961608	-0.3368573	-0.06288113	1.1054186	-0.02219347	-0.6774919	
5	0.9646211	-0.9646211	-0.02219347	-0.7961608	-0.3368573	-0.06288113	1.1054186	-0.02219347	1.4758508	
6	0.9646211	-0.9646211	-0.02219347	-0.7961608	-0.3368573	-0.06288113	1.1054186	-0.02219347	-0.6774919	
	cap.surfacef	cap.colorc	cap.colore	cap.colorg	cap.colorn	cap.colorp	cap.colorr	cap.coloru	cap.colow	cap.colory
1	-0.8152744	-0.07378939	-0.4758376	-0.5410833	1.5989381	-0.1343238	-0.04441978	-0.04441978	-0.383134	-0.3898652
2	-0.8152744	-0.07378939	-0.4758376	-0.5410833	-0.6253381	-0.1343238	-0.04441978	-0.04441978	-0.383134	2.5646732
3	-0.8152744	-0.07378939	-0.4758376	-0.5410833	-0.6253381	-0.1343238	-0.04441978	-0.04441978	2.609732	-0.3898652
4	1.2264300	-0.07378939	-0.4758376	-0.5410833	-0.6253381	-0.1343238	-0.04441978	-0.04441978	2.609732	-0.3898652
5	-0.8152744	-0.07378939	-0.4758376	1.8479169	-0.6253381	-0.1343238	-0.04441978	-0.04441978	-0.383134	-0.3898652
6	1.2264300	-0.07378939	-0.4758376	-0.5410833	-0.6253381	-0.1343238	-0.04441978	-0.04441978	-0.383134	2.5646732
	bruiseit	odorc	odorf	odorl	odorm	odorn	odorp	odors	odory	gill.attachmentf
1	1.1858436	-0.1555724	-0.6017711	-0.2275528	-0.066712	-0.8760876	5.5435180	-0.2762286	-0.2762286	0.1628864
2	1.1858436	-0.1555724	-0.6017711	-0.2275528	-0.066712	-0.8760876	-0.1803687	-0.2762286	-0.2762286	0.1628864
3	1.1858436	-0.1555724	-0.6017711	4.3940440	-0.066712	-0.8760876	-0.1803687	-0.2762286	-0.2762286	0.1628864
4	1.1858436	-0.1555724	-0.6017711	-0.2275528	-0.066712	-0.8760876	5.5435180	-0.2762286	-0.2762286	0.1628864
5	-0.8431777	-0.1555724	-0.6017711	-0.2275528	-0.066712	1.1412978	-0.1803687	-0.2762286	-0.2762286	0.1628864
6	1.1858436	-0.1555724	-0.6017711	-0.2275528	-0.066712	-0.8760876	-0.1803687	-0.2762286	-0.2762286	0.1628864
	gill.spacingw	gill.sizen	gill.colore	gill.colorg	gill.colorn	gill.colorp	gill.colorr	gill.coloru	gill.colow	gill.colory
1	-0.4388366	1.4945907	-0.1093466	-0.3193666	-0.3146646	4.3484982	-0.3848221	-0.0891037	-0.4742807	
2	-0.4388366	-0.6689971	-0.1093466	-0.3193666	-0.3146646	4.3484982	-0.3848221	-0.0891037	-0.4742807	
3	-0.4388366	-0.6689971	-0.1093466	-0.3193666	-0.3146646	-0.2299361	2.5982835	-0.0891037	-0.4742807	
4	-0.4388366	1.4945907	-0.1093466	-0.3193666	-0.3146646	-0.2299361	2.5982835	-0.0891037	-0.4742807	
5	2.2784719	-0.6689971	-0.1093466	-0.3193666	-0.3146646	4.3484982	-0.3848221	-0.0891037	-0.4742807	
6	-0.4388366	-0.6689971	-0.1093466	-0.3193666	-0.3146646	-0.2299361	2.5982835	-0.0891037	-0.4742807	
	gill.colorr	gill.coloru	gill.colow	gill.colory	stalk.shapet	stalk.rootb	stalk.rootc	stalk.rootd	stalk.roote	stalk.roote
1	-0.05442976	-0.2538848	-0.4166868	-0.1034305	-1.1447353	-0.9318467	-0.2710317	2.5005603	-0.1555724	

Correlation:

```
> round(cor(encoded_data), digits = 2 )
```

	classe	classp	cap.shapec	cap.shapef	cap.shapek	cap.shapes	cap.shapex	cap.surfaceg	
classe	1.00	-1.00	-0.02	-0.02	-0.16	0.06	0.03	-0.02	
classp	-1.00	1.00	0.02	0.02	0.16	-0.06	-0.03	0.02	
cap.shapec	-0.02	0.02	1.00	-0.02	-0.01	0.00	-0.02	0.25	
cap.shapef	-0.02	0.02	-0.02	1.00	-0.27	-0.05	-0.72	-0.01	
cap.shapek	-0.16	0.16	-0.01	-0.27	1.00	-0.02	-0.30	0.01	
cap.shapes	0.06	-0.06	0.00	-0.05	-0.02	1.00	-0.06	0.00	
cap.shapex	0.03	-0.03	-0.02	-0.72	-0.30	-0.06	1.00	-0.02	
cap.surfaceg	-0.02	0.02	0.25	-0.01	0.01	0.00	-0.02	1.00	
cap.surfaces	-0.10	0.10	-0.02	-0.09	0.14	-0.04	-0.04	-0.02	
cap.surfacey	-0.09	0.09	0.02	0.03	0.02	-0.05	-0.02	-0.02	
cap.surfaces	-0.10	-0.09	0.03	-0.10	0.05	0.04	-0.03	0.04	
cap.surfacey	0.10	0.09	-0.03	0.10	-0.05	-0.04	0.03	-0.04	
cap.shapec	-0.02	0.02	0.00	-0.01	-0.01	-0.01	0.00	0.00	
cap.shapef	-0.09	0.03	0.00	0.01	0.06	0.02	-0.05	0.01	
cap.shapek	0.14	0.02	0.04	0.16	-0.14	0.13	0.00	-0.01	
cap.shapes	-0.04	-0.05	0.00	-0.03	0.04	0.03	-0.01	0.00	
cap.shapex	-0.04	-0.02	-0.01	-0.05	0.05	-0.06	0.04	0.00	
cap.surfaceg	-0.02	-0.02	0.00	-0.01	-0.01	-0.01	0.00	0.00	
cap.surfaces	1.00	-0.55	-0.04	-0.01	-0.16	0.08	0.05	-0.03	
cap.surfacey	-0.55	1.00	0.04	0.10	-0.07	0.02	-0.03	0.05	
cap.coloru	0.04	0.13	-0.11	0.50	-0.16	-0.62	0.22	-0.07	0.79
cap.colorw	-0.04	-0.13	0.11	-0.50	0.16	0.62	-0.22	0.07	-0.79
cap.colorx	0.00	0.02	0.02	0.00	0.00	-0.01	-0.01	0.00	0.03
cap.shapef	0.01	-0.09	-0.01	0.05	-0.12	0.08	-0.10	-0.01	0.07
cap.shapek	-0.01	-0.07	-0.12	-0.23	-0.05	-0.03	-0.08	0.05	-0.10
cap.shapes	0.00	-0.02	-0.02	-0.05	-0.01	-0.04	-0.01	0.00	0.07
cap.surfaces	-0.19	-0.29	-0.01	0.19	0.29	0.29	0.29	0.29	0.29
cap.surfacey	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
cap.surfaces	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03
cap.surfacey	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03

Explanation:

- The `cor()` function is used to compute the correlation matrix of the data frame.
- `round()` function is applied to the correlation matrix calculated in the previous step. The `digits` parameter is set to 2, indicating that the correlation coefficients should be rounded to two decimal places.

KNN:

Spited data into training and test sets and defined target column index.

```
23 set.seed(123)
24 train_indices <- sample(1:nrow(encoded_data), 0.7 * nrow(encoded_data))
25 training_data <- encoded_data[train_indices, ]
26 test_data <- encoded_data[-train_indices, ]
27 target_column_index <- 1
```

Explanation:

- The `set.seed()` function is used to set the random number generator's seed. This ensures that the random sampling and other random operations performed later in the code will produce the same results each time the code is run.
- The `sample()` function is used to randomly sample indices from 1 to the total number of rows in the `encoded_data` data frame. The second argument `0.7 * nrow(encoded_data)` specifies that approximately 70% of the rows will be sampled as training data. The sampled indices are stored in the `train_indices` variable.
- `training_data` will be used to train a machine learning model and `test_data` will be used to evaluate the performance of the trained model.

Pearson's correlation:

```
> correlation_matrix <- cor(training_data[, -target_column_index], training_data[, target_column_index])
> important_attributes <- colnames(training_data)[abs(correlation_matrix) > 0.1]
> correlation_matrix <- cor(training_data[, -target_column_index], training_data[, target_column_index])
> print(correlation_matrix)
```

```
      [,1]
classp    -1.000000000
cap.shapec -0.019286750
cap.shapef -0.016268908
cap.shapek -0.170760033
cap.shapes  0.061982566
cap.shapex  0.031250830
cap.surfaceg -0.019286750
cap.surfaces -0.096512452
cap.surfacey -0.084785007
cap.colorc  0.030118100
cap.colore -0.097945624
cap.colorg  0.044894826
cap.colorn  0.044972971
cap.colorp -0.035714883
cap.colorr  0.040823263
cap.coloru  0.040823263
cap.colorw  0.137216698
cap.colory -0.112370785
bruiseest  0.502135130
odorc      -0.163956964
odorf      -0.619386698
```

Explanation:

- Calculate coefficient and select important attributes
- `abs(correlation_matrix) > 0.1`: This creates a logical matrix where each element is TRUE if the absolute value of the corresponding correlation coefficient is greater than 0.1, and FALSE otherwise.

KNN model:

```
41 k <- 5
42 knn_model <- knn(train = training_dataset[, -target_column_index],
43                  test = test_dataset[, -target_column_index],
44                      cl = training_dataset[, target_column_index],
45                      k = k)
```

Accuracy test:

```
> accuracy_test <- sum(knn_model == test_dataset[, target_column_index]) / nrow(test_dataset)
> cat("Accuracy (Test Set):", accuracy_test, "\n")
Accuracy (Test Set): 0.9983593
> |
```

10-fold cross validation:

```
> num_folds <- 10
> cv_folds <- createFolds(training_dataset[, target_column_index], k = num_folds)
> cv_accuracy <- numeric(num_folds)
> best_k <- 5
> for (fold in 1:num_folds) {
+   train_indices <- cv_folds[[fold]]
+   train_data <- training_dataset[train_indices, ]
+   test_data <- training_dataset[-train_indices, ]
+
+   knn_model <- knn(
+     train = train_data[, -target_column_index],
+     test = test_data[, -target_column_index],
+     cl = train_data[, target_column_index],
+     k = best_k
+   )
+   accuracy <- sum(knn_model == test_data[, target_column_index]) / nrow(test_data)
+   cv_accuracy[fold] <- accuracy
+ }
> cat("Accuracy (", num_folds, "-fold CV):\n")
Accuracy ( 10 -fold CV):
> print(cv_accuracy)
[1] 0.9953107 0.9892515 0.9984369 0.9906195 0.9931601 0.9972646 0.9972640 0.9992183 0.9986320 0.9878859
> average_accuracy <- mean(cv_accuracy)
> cat("Average Accuracy:", average_accuracy, "\n")
Average Accuracy: 0.9947043
```

Confusion matrix:

```
xtab <- table(knn_model, test_dataset[, target_column_index])
cm <- caret::confusionMatrix(xtab)
xtab <- table(knn_model, test_dataset[, target_column_index])
cm <- caret::confusionMatrix(xtab)
print(cm)
```

Confusion Matrix and Statistics

```
nn_model    0    1
           0 1148    0
           1    4 1286

              Accuracy : 0.9984
              95% CI   : (0.9958, 0.9996)
    No Information Rate : 0.5275
    P-Value [Acc > NIR] : <2e-16

              Kappa : 0.9967

McNemar's Test P-Value : 0.1336

    Sensitivity : 0.9965
    Specificity : 1.0000
    Pos Pred Value : 1.0000
    Neg Pred Value : 0.9969
    Prevalence : 0.4725
    Detection Rate : 0.4709
    Detection Prevalence : 0.4709
    Balanced Accuracy : 0.9983

    'Positive' Class : 0
```

Explanation:

- Confusion matrix calculated using the table() function. xtab is a table containing the counts of true positive, true negative, false positive, and false negative predictions.
- confusionMatrix() function from the "caret" package used to create a confusion matrix object that contains various metrics for evaluating the model's performance.

Recall and Precision:

```
> library(Metrics)
> Metrics::recall(knn_model, test_dataset[, target_column_index])
[1] 0.9968992
>
> precision_value <- Metrics::precision(predicted_labels, true_labels)
> print(precision_value)
[1] 1
```

Explanation:

- The Metrics package in R provides functions for computing various metrics commonly used in evaluating the performance of machine learning models. The Metrics::recall() function specifically calculates the recall metric for evaluating the performance of a binary classification model.
- The Metrics::precision() function calculates the precision as the ratio of true positive predictions to the total number of instances that were predicted as positive by the model.