Data Science Lab

Vasudevan T V

Matrix Operations

- Using Vectorisation
- Here various matrix operations are performed without using loops
- For this, we can use various functions in the built in package numpy

```
# Matrix Addition
>>> import numpy
>>> matrix1=numpy.matrix([[1,2],[3,4]])
>>> matrix2=numpy.matrix([[4,3],[2,1]])
>>> matrix3=numpy.add(matrix1,matrix2)
>>> print(matrix3)
[[5 5]
 [5 5]]
```

Matrix Operations

```
# Matrix Subtraction
>>> import numpy
>>> matrix1=numpy.matrix([[2,2],[2,2]])
>>> matrix2=numpy.matrix([[1,1],[1,1]])
>>> matrix3=numpy.subtract(matrix1,matrix2)
>>> print(matrix3)
[[1 1]
 [1 1]]
# Matrix Multiplication
>>> import numpy
>>> matrix1=numpy.matrix([[2,2],[2,2]])
>>> matrix2=numpy.matrix([[1,1],[1,1]])
>>> matrix3=numpy.matmul(matrix1,matrix2)
>>> print(matrix3)
[[4 4]
 [4 4]]
```

Matrix Operations

```
# Scalar Multiplication
>>> import numpy
>>> matrix1=numpy.matrix([[2,2],[2,2]])
>>> matrix2=2*matrix1
>>> print(matrix2)
[[4 4]
 [4 4]]
# Matrix Transpose
>>> import numpy
>>> matrix1=numpy.matrix([[1,2],[3,4]])
>>> print(matrix1)
[[1 2]
 [3 4]]
>>> matrix2=numpy.transpose(matrix1)
>>> print(matrix2)
[[1 3]
 [2 4]]
```

Matrix Transformations

- We can use matrices for performing various geometric transformations such as translation, rotation, scaling etc.
- Translation is the process of moving an object to a different position
- Rotation is the process of changing the angle of the object
- Scaling is the process of changing the size of objects

Matrix Transformations

- Translation Matrix

$$\begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Program

```
import numpy
def translationMatrix(tx=0, ty=0):
    return numpy.matrix([[1,0,tx],
                         [0,1,ty],
                         [0,0, 1]])
matrix=translationMatrix(1,1)
print(matrix)
```

Matrix Transformations

- Rotation Matrix

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Program

```
import numpy
def rotationMatrix(degree):
    theta = numpy.radians(degree)
    c,s=numpy.cos(theta),numpy.sin(theta)
    return numpy.matrix([[c, -s, 0],
                         [s,  c, 0]
                         [0,  0, 1]])
matrix=rotationMatrix(30)
print(matrix)
```

Matrix Transformations

- Scaling Matrix

$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Program

```
import numpy
def scalingMatrix(sx=0, sy=0):
    return numpy.matrix([[sx,0,0],
                         [0,sy,0],
                         [0, 0,1]])
matrix=scalingMatrix(2,2)
print(matrix)
```

Singular Value Decomposition (SVD)

- It is the process of decomposing a matrix into 3 components which are also matrices
- A matrix M is decomposed into 3 matrices U, S and V
- If M is a real matrix, U and V are orthogonal matrices and S is a diagonal matrix
- The advantage of such a decomposition is that we can do the subsequent matrix operations faster
- Applications - solving homogeneous linear equations, pattern recognition, natural language processing, weather prediction, machine learning etc.

Singular Value Decomposition (SVD)

- Program

```
# Imports matrix, matmul and diag functions only
from numpy import matrix
from numpy import matmul
from numpy import diag
# Imports svd fn from linalg(linear algebra) submodule of
# scipy module
from scipy.linalg import svd
# define a matrix
A = matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(A)
# Singular-value decomposition
# A is decomposed into 3 matrices U, a diagonal matrix
# and V
# Here S contains only the diagonal elements of the
# diagonal matrix
U, S, V = svd(A)
```

Singular Value Decomposition (SVD)

- Program - continued

```
print(U)
print(S)
print(V)
# create diagonal matrix from diagonal elements
Sigma = diag(S)
print(Sigma)
# reconstruct matrix
B = matmul(U,matmul(Sigma,V))
print(B)
```

Singular Value Decomposition (SVD)

- Output

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[-0.21483724  0.88723069  0.40824829]
 [-0.52058739  0.24964395 -0.81649658]
 [-0.82633754 -0.38794278  0.40824829]]
[1.68481034e+01 1.06836951e+00 4.41842475e-16]
[[-0.47967118 -0.57236779 -0.66506441]
 [-0.77669099 -0.07568647  0.62531805]
 [-0.40824829  0.81649658 -0.40824829]]
[[1.68481034e+01 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.06836951e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 4.41842475e-16]]
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
```

Histogram

- Write a python program to plot a histogram of marks obtained by students in a class
- Marks - 22,87,5,43,56,73,55,54,11,20,51,5,79,31,27

```
# imports pyplot, a module used in the package matplotlib
# to plot various figures
from matplotlib import pyplot
# imports array() from numpy package
from numpy import array
# subplots() specify the number of plots in the figure
# first argument is number of rows
# second argument is number of columns
# This function returns a tuple containing figure and axes
# objects
# These objects are assigned to fig and ax
# They are needed for changing figure level and axes level
# attributes
fig,ax = pyplot.subplots(1,1)
```

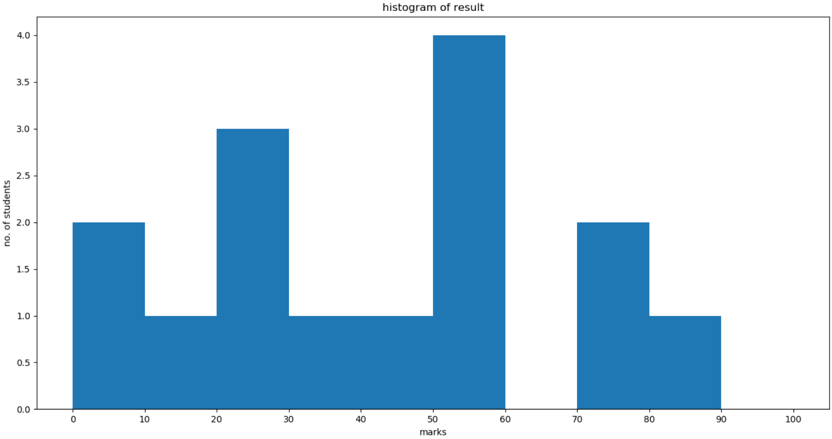
Histogram

- Program - continued

```
a = array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])
# Draws a histogram, first argument is the array of
# numbers, second argument bins are intervals of values
ax.hist(a,bins=[0, 10, 20, 30, 40, 50, 60, 70, 80,90,100])
ax.set_title("histogram of result")
ax.set_xticks([0, 10, 20, 30, 40, 50, 60, 70, 80, 90,100])
ax.set_xlabel('marks')
ax.set_ylabel('no. of students')
# Shows the plot
pyplot.show()
```

Histogram

- Output



Histogram

- Write a python program to draw a histogram of petal length in the iris data set
- Program

```
from matplotlib import pyplot
# imports pandas package, used for data analysis
import pandas
# reads the csv file into a data frame
# A data frame is a table with rows and columns
df = pandas.read_csv('iris.csv')
fig,ax = pyplot.subplots(1,1)
```

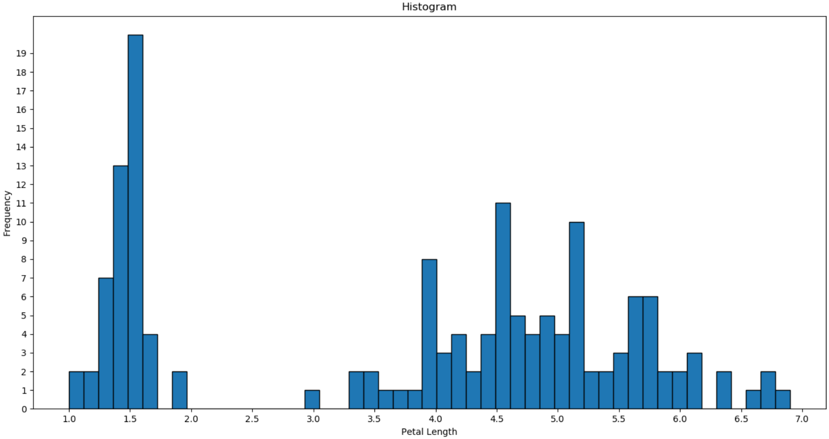
Histogram

- Write a python program to draw a histogram of petal length in the iris data set
- Program - continued

```
# plots the histogram of petal length attribute
# By default bins = 10
df['petal.length'].plot(kind='hist', edgecolor="black",
bins=49)
ax.set_title("Histogram")
ax.set_xticks([1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0,5.5,
6.0,6.5,7.0])
ax.set_yticks([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,
17,18,19])
ax.set_xlabel('Petal Length')
pyplot.show()
```

Histogram

- Output



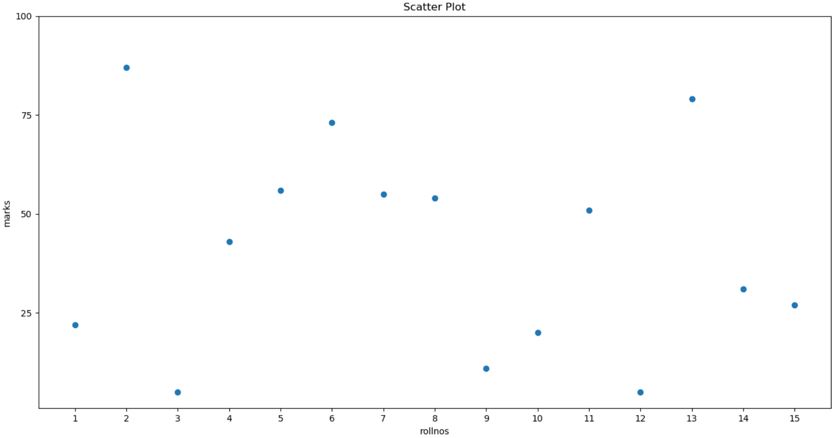
Scatter Plot

- Write a python program to draw a scatterplot that shows the relationship between rollnos and marks of students in a class
- rollnos = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
- marks = [22,87,5,43,56,73,55,54,11,20,51,5,79,31,27]

```
from matplotlib import pyplot
rollnos = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
marks = [22,87,5,43,56,73,55,54,11,20,51,5,79,31,27]
fig,ax = pyplot.subplots(1,1)
# Draws a scatterplot, first argument is x axis values,
# second argument is y axis values
ax.scatter(rollnos, marks)
ax.set_title("Scatter Plot")
ax.set_xticks([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15])
ax.set_yticks([25,50,75,100])
ax.set_xlabel('rollnos')
ax.set_ylabel('marks')
pyplot.show()
```

Scatter Plot

- Output



Scatter Plot

- Write a python program to draw a scatterplot that shows the relationship between petal length and petal width in the iris data set
- Program

```
from matplotlib import pyplot
import pandas
df = pandas.read_csv('iris.csv')
fig, ax = pyplot.subplots(1,1)
# Creates a dictionary of colour values of each species
colors = {'Setosa':'red', 'Versicolor':'green',
'Virginica':'blue'}
```

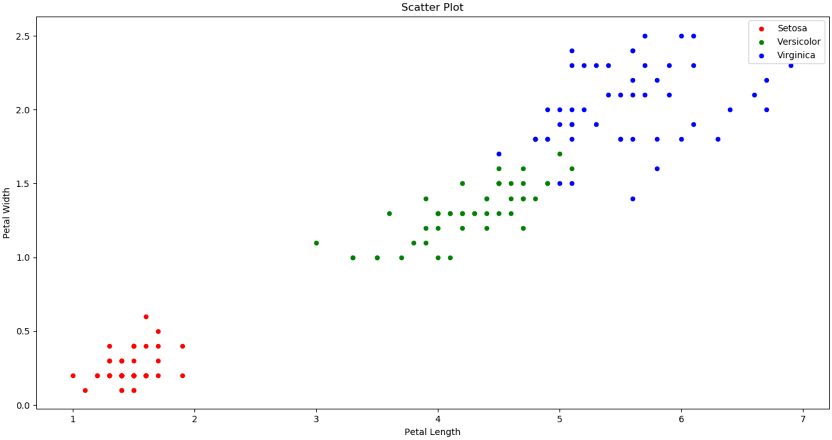
Scatter Plot

- Write a python program to draw a scatterplot that shows the relationship between petal length and petal width in the iris data set
- Prgram - continued

```
# Groups the data based on species values
grouped = df.groupby('species')
# group represents the grouped data frame
# draws the scatter plot for each group
for key, group in grouped:
    group.plot(ax=ax, kind='scatter', x='petal.length',
    y='petal.width', label=key, color=colors[key])
ax.set_title("Scatter Plot")
ax.set_xlabel('Petal Length')
ax.set_ylabel('Petal Width')
pyplot.show()
```

Scatter Plot

- Output



Classification Using kNN Algorithm

- Given a data set of 15 food items (food.csv) having 4 features - ingredient, sweetness, crunchiness and food type. Write a R program to predict the food type of tomato using kNN algorithm.

```
$ R
R version 3.3.3 (2017-03-06) -- "Another Canoe"
...............................................
# Read the csv file into a data frame
> food=read.csv("food.csv")
```

Classification Using kNN Algorithm

```
# Prints food data frame
> food
    Ingredient Sweetness Crunchiness  FoodType
1        apple        10           9     fruit
2        bacon         1           4   protein
3       banana        10           1     fruit
4       carrot         7          10 vegetable
5       celery         3          10 vegetable
6       cheese         1           1   protein
7     cucumber         2           8 vegetable
8         fish         3           1   protein
9        grape         8           5     fruit
10  green bean         3           7 vegetable
11     lettuce         1           9 vegetable
12        nuts         3           6   protein
13      orange         7           3     fruit
14        pear        10           7     fruit
15      shrimp         2           3   protein
```

Classification Using kNN Algorithm

```
# Creates a data frame of food item tomato
> tomato=data.frame(ingredient="tomato",sweetness=6,
crunchiness=4)
# prints the tomato data frame
> tomato
    Ingredient Sweetness Crunchiness
1      tomato         6           4
# Create a data frame of second and third columns of food
> food1=food[,2:3]
```

Classification Using kNN Algorithm

```
> food1
   Sweetness Crunchiness
1         10           9
2          1           4
3         10           1
4          7          10
5          3          10
6          1           1
7          2           8
8          3           1
9          8           5
10         3           7
11         1           9
12         3           6
13         7           3
14        10           7
15         2           3
```

Classification Using kNN Algorithm

```
# Create a data frame of second and third columns of
# tomato
> tomato1=tomato[,2:3]
> tomato1
  Sweetness Crunchiness
1         6           4
# Load package class which contains knn()
> library(class)
# Use knn() and store the prediction in pred
# argument 1 is the data frame containing training data
# argument 2 is the data frame containing test data
# argument 3 is a vector that show the class of each item
# in the training data, argument 4 is the value of k
> pred=knn(food1,tomato1,food$FoodType,k=1)
> pred
[1] fruit
Levels: fruit protein vegetable
```

Classification Using kNN Algorithm

- Diagnosing Breast Cancer With The kNN Algorithm
- The data includes 569 examples of cancer biopsies, each with 32 features
- One feature is an identification number, another is the cancer diagnosis, and 30 are numeric-valued laboratory measurements
- The diagnosis is coded as "M" to indicate malignant or "B" to indicate benign
- The other 30 numeric measurements comprise the mean, standard error, and worst(that is, largest) value for 10 different characteristics of the digitized cell nuclei
- These include Radius, Texture, Perimeter, Area etc.

Classification Using kNN Algorithm

- Diagnosing Breast Cancer With The kNN Algorithm

```
$ R
R version 3.3.3 (2017-03-06) -- "Another Canoe"
..............................................
# Loads class packge containing knn()
> library(class)
# Loads gmodels packge containing CrossTable()
> library(gmodels)
# Read the csv file into a data frame
> wbcd = read.csv("wisc_bc_data.csv")
# Define normalize fn for performing min max normalisation
# This will transform the values of all features to a
# range between 0 and 1
> normalize <- function(x)
{
 return ((x - min(x)) / (max(x) - min(x)))
}
```

Classification Using kNN Algorithm

- Diagnosing Breast Cancer With The kNN Algorithm

```
# Apply this function to our data frame
> wbcd_n = as.data.frame(lapply(wbcd[3:31], normalize))
# Training Data
> wbcd_train = wbcd_n[1:469, ]
# Test data
> wbcd_test = wbcd_n[470:569, ]
# Training Labels
> wbcd_train_labels = wbcd[1:469, 2]
# Test Labels
> wbcd_test_labels = wbcd[470:569, 2]
```

Classification Using kNN Algorithm

- Diagnosing Breast Cancer With The kNN Algorithm

```
# Prediction of Breast Cancer using knn()
> wbcd_test_pred = knn(wbcd_train, wbcd_test,
wbcd_train_labels,21)
# Print the prediction results
> wbcd_test_pred
 [1] B B B B B B B B B B M B B B B B B B M B B B B M B B..
[38] B B B M B B M B B B M M B B B M B B B B B B B B B B..
[75] B B B B B B B B B B B B B B B B B B B M M M M M M B
Levels: B M
```

Classification Using kNN Algorithm

- Diagnosing Breast Cancer With The kNN Algorithm

```
# Analysis of Prediction
> CrossTable(x = wbcd_test_labels, y = wbcd_test_pred,
prop.chisq=FALSE)
Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|
```

Classification Using kNN Algorithm

- Diagnosing Breast Cancer With The kNN Algorithm

```
Total Observations in Table:  100
                 | wbcd_test_pred
wbcd_test_labels |         B |         M | Row Total |
-----------------|-----------|-----------|-----------|
               B |        77 |         0 |        77 |
                 |     1.000 |     0.000 |     0.770 |
                 |     0.975 |     0.000 |           |
                 |     0.770 |     0.000 |           |
-----------------|-----------|-----------|-----------|
               M |         2 |        21 |        23 |
                 |     0.087 |     0.913 |     0.230 |
                 |     0.025 |     1.000 |           |
                 |     0.020 |     0.210 |           |
-----------------|-----------|-----------|-----------|
    Column Total |        79 |        21 |       100 |
                 |     0.790 |     0.210 |           |
-----------------|-----------|-----------|-----------|
```

Classification Using Naive Bayes Algorithm

- Write a R program to predict the species of iris data set using Naive Bayes algorithm and evaluate its performance

```
# Loads e1071 package containing naiveBayes
library(e1071)
# Loads caTools package containing sample.split()
library(caTools)
# Loads gmodels packge containing CrossTable()
library(gmodels)

# Read the csv file into a data frame
iris = read.csv("iris.csv")

# Splitting data into train
# and test data
# set.seed() is used for generating the same sample
# in every execution
# We specify a seed number
set.seed(100)
```

Classification Using Naive Bayes Algorithm

- Program

```
split <- sample.split(iris$species, SplitRatio = 0.7)
iris1 <- subset(iris, split == "TRUE")
iris2 <- subset(iris, split == "FALSE")

iris_train = iris1[,1:4]
iris_test = iris2[,1:4]

iris_train_labels = iris1[,5]
iris_test_labels = iris2[,5]

classifier_cl <- naiveBayes(iris_train,iris_train_labels )
classifier_cl

# Predicting on test data'
iris_test_pred <- predict(classifier_cl, iris_test)
iris_test_pred
```

Classification Using Naive Bayes Algorithm

- Program

```
# Analysis of Prediction
# prop.chisq=FALSE will remove unnecessary chi square
# values
CrossTable(iris_test_labels, iris_test_pred,
prop.chisq=FALSE)
```

- Output

```
    Naive Bayes Classifier for Discrete Predictors
Call:
naiveBayes.default(x = iris_train, y = iris_train_labels)
A-priori probabilities:
iris_train_labels
    Setosa Versicolor  Virginica
 0.3333333  0.3333333  0.3333333
```

Classification Using Naive Bayes Algorithm

- Output

```
   Conditional probabilities:
                sepal.length
iris_train_labels     [,1]      [,2]
       Setosa     5.025714 0.3266072
       Versicolor 5.894286 0.5455396
       Virginica  6.625714 0.5907836
                sepal.width
iris_train_labels     [,1]      [,2]
       Setosa     3.445714 0.3567359
       Versicolor 2.782857 0.3468223
       Virginica  2.985714 0.2658426
                petal.length
iris_train_labels     [,1]      [,2]
       Setosa     1.471429 0.1808012
       Versicolor 4.191429 0.4859021
       Virginica  5.608571 0.5083835
```

Classification Using Naive Bayes Algorithm

- Output

```
                 petal.width
iris_train_labels      [,1]       [,2]
       Setosa     0.2285714 0.08934872
       Versicolor 1.3228571 0.20448747
       Virginica  2.0485714 0.28218833

# For numerical values, conditional probabilities display
# their mean and standard deviation
```

```
 [1] Setosa     Setosa     Setosa     Setosa     Setosa     Setosa    
 [7] Setosa     Setosa     Setosa     Setosa     Setosa     Setosa    
[13] Setosa     Setosa     Setosa     Versicolor Versicolor Versicolor
[19] Versicolor Versicolor Versicolor Virginica  Versicolor Versicolor
[25] Versicolor Versicolor Versicolor Versicolor Versicolor Versicolor
[31] Versicolor Virginica  Virginica  Virginica  Virginica  Virginica 
[37] Virginica  Versicolor Virginica  Virginica  Virginica  Virginica 
[43] Virginica  Virginica  Virginica 
Levels: Setosa Versicolor Virginica
```

Classification Using Naive Bayes Algorithm

- Output

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  45


                 | iris_test_pred
iris_test_labels |     Setosa | Versicolor |  Virginica |  Row Total |
-----------------|------------|------------|------------|------------|
          Setosa |         15 |          0 |          0 |         15 |
                 |      1.000 |      0.000 |      0.000 |      0.333 |
                 |      1.000 |      0.000 |      0.000 |            |
                 |      0.333 |      0.000 |      0.000 |            |
-----------------|------------|------------|------------|------------|
      Versicolor |          0 |         14 |          1 |         15 |
                 |      0.000 |      0.933 |      0.067 |      0.333 |
                 |      0.000 |      0.875 |      0.071 |            |
                 |      0.000 |      0.311 |      0.022 |            |
-----------------|------------|------------|------------|------------|
       Virginica |          0 |          2 |         13 |         15 |
                 |      0.000 |      0.133 |      0.867 |      0.333 |
                 |      0.000 |      0.125 |      0.929 |            |
                 |      0.000 |      0.044 |      0.289 |            |
-----------------|------------|------------|------------|------------|
    Column Total |         15 |         16 |         14 |         45 |
                 |      0.333 |      0.356 |      0.311 |            |
-----------------|------------|------------|------------|------------|
```

Classification Using C5.0 Decision Tree Algorithm

- Write a R program to identify risky bank loans using C5.0 Decision Tree Algorithm and evaluate its performance

```
# Use C5.0 Decision Tree algorithm to identify risky bank loans
# Also evaluate the performance of the algorithm
# Given credit.csv data set containing 1000 bank loan records
# Loads C50 package containg C5.0()
library(C50)
# Loads gmodels packge containing CrossTable()
library(gmodels)
# Read the csv file into a data frame
credit <- read.csv("credit.csv")
# Training Data, 17th column default is omitted
credit_train <- credit[1:900,-17 ]
#Test Data, 17th column default is omitted
credit_test <- credit[901:1000,-17 ]
# Training Labels, containing values of 17th column default
credit_train_labels = credit[1:900, 17]
# Test Labels, containing values of 17th column default
credit_test_labels = credit[901:1000, 17]
```

Classification Using C5.0 Decision Tree Algorithm

- Program

```r
# C5.0() returns a C5.0 model object and stores it in credit_model
# credit_train is a data frame containing training data
# credit_train_labels is converted into a factor containing categorical values
credit_model <- C5.0(credit_train, as.factor(credit_train_labels))
# Prints basic data about the decision tree
credit_model
# Shows the decision tree and some other information
summary(credit_model)
# Predicting on test data
credit_pred <- predict(credit_model, credit_test)
credit_pred
# Analysis of Prediction
# prop.chisq=FALSE will remove unnecessary chi square values
CrossTable(credit_test_labels, credit_pred, prop.chisq=FALSE )
```

Classification Using C5.0 Decision Tree Algorithm

- Output

```
Call:
C5.0.default(x = credit_train, y = as.factor(credit_train_labels))

Classification Tree
Number of samples: 900
Number of predictors: 16

Tree size: 63

Non-standard options: attempt to group attributes


Call:
C5.0.default(x = credit_train, y = as.factor(credit_train_labels))


C5.0 [Release 2.07 GPL Edition]         Sun Jan 30 12:54:58 2022
-------------------------------

Class specified by attribute `outcome'

Read 900 cases (17 attributes) from undefined.data

Decision tree:

checking_balance in {unknown,> 200 DM}: no (414/53)
checking_balance in {< 0 DM,1 - 200 DM}:
:...months_loan_duration <= 11:
    :...credit_history in {critical,good,poor,perfect}: no (71/11)
    :   credit_history = very good: yes (6/1)
```

Classification Using C5.0 Decision Tree Algorithm

- Output

```
Time: 0.0 secs

  [1] yes no  no  no  no  no  yes no  no  no  no  no  no  no  yes yes no  no
 [19] no  yes no  no  no  no  no  yes yes yes no  yes no  no  no  no  no  yes
 [37] no  no  yes no  no  no  no  no  no  no  no  no  no  no  yes no  no  no
 [55] yes no  no  no  no  yes no  no  no  no  no  no  no  no  no  no  yes no
 [73] no  yes no  no  no  no  no  no  yes no  no  yes no  no  no  no  yes yes
 [91] no  no  yes yes no  no  yes no  yes yes
Levels: no yes


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  100


                   | credit_pred
credit_test_labels |        no |       yes | Row Total |
-------------------|-----------|-----------|-----------|
                no |        55 |        13 |        68 |
                   |     0.809 |     0.191 |     0.680 |
                   |     0.733 |     0.520 |           |
                   |     0.550 |     0.130 |           |
-------------------|-----------|-----------|-----------|
               yes |        20 |        12 |        32 |
                   |     0.625 |     0.375 |     0.320 |
                   |     0.267 |     0.480 |           |
                   |     0.200 |     0.120 |           |
-------------------|-----------|-----------|-----------|
      Column Total |        75 |        25 |       100 |
                   |     0.750 |     0.250 |           |
-------------------|-----------|-----------|-----------|
```

Prediction Using Multiple Linear Regression

- Write a R program to predict medical expenses using multiple linear regression technique and evaluate its performance

```r
# Predict Medical Expenses using Multiple Linear Regression Technique
# Also evaluate its performance
# Given insurance.csv data set containing 1338 data items
# Our model's dependent variable is expenses , which measures the medical costs
# each person charged to the insurance plan for the year
# Read the csv file into a data frame
insurance <- read.csv("insurance.csv")
# Training Data
insurance_train <- insurance[1:1000,]
#Test Data
insurance_test <- insurance[1001:1338,]
# lm() returns a multiple linear regression model object
# the dependent variable expenses goes to the left of the tilde
# the independent variables go to the right, separated by + sign
# data specifies the data frame in which these variables can be found
# lm() is contained in stats package, which is loaded by default
insurance_model <- lm(expenses ~ age + sex + bmi + children + smoker + region, data = insurance_train)
# Prints estimated regression coefficients
insurance_model
# Evaluate Model Performance
summary(insurance_model)
# Predicting on test data
insurance_pred <- predict(insurance_model, insurance_test)
insurance_pred
```

Prediction Using Multiple Linear Regression

- Output

```
Call:
lm(formula = expenses ~ age + sex + bmi + children + smoker +
    region, data = insurance_train)

Coefficients:
    (Intercept)              age          sexmale              bmi
       -12083.3            264.3           -288.5            339.9
       children        smokeryes  regionnorthwest  regionsoutheast
          410.2          23832.4           -439.9          -1291.3
regionsouthwest
        -1263.1


Call:
lm(formula = expenses ~ age + sex + bmi + children + smoker +
    region, data = insurance_train)

Residuals:
     Min       1Q   Median       3Q      Max
-11070.0  -2783.7   -926.3   1255.7  25270.9

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     -12083.26    1135.37 -10.643  < 2e-16 ***
age                264.26      13.40  19.721  < 2e-16 ***
sexmale           -288.53     377.41  -0.765  0.44474
bmi                339.91      32.57  10.436  < 2e-16 ***
children           410.24     156.89   2.615  0.00906 **
smokeryes        23832.38     475.70  50.099  < 2e-16 ***
regionnorthwest   -439.90     543.63  -0.809  0.41861
regionsoutheast  -1291.29     534.51  -2.416  0.01588 *
regionsouthwest  -1263.15     537.30  -2.351  0.01892 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5934 on 991 degrees of freedom
Multiple R-squared:  0.7569,    Adjusted R-squared:  0.7549
F-statistic: 385.7 on 8 and 991 DF,  p-value: < 2.2e-16
```

Prediction Using Multiple Linear Regression

- Output

```
      1001       1002       1003       1004       1005       1006       1007 
27583.4127 27654.6512  1476.9030  9110.7582  6981.4349  6457.5775  5793.6412 
      1008       1009       1010       1011       1012       1013       1014 
34262.0651  3547.7564 10944.6876  7087.9132 29195.2402 15715.3142 11261.9993 
      1015       1016       1017       1018       1019       1020       1021 
 6076.8905 11433.6305  1271.4416  5969.6590 15151.3352  4954.9571 12418.8576 
      1022       1023       1024       1025       1026       1027       1028 
28046.1886 35263.5807  -569.5267 14860.4921  3963.8578 25299.6560  -372.3569 
      1029       1030       1031       1032       1033       1034       1035 
11376.1261  4391.7943 31915.8844 36956.8261  5338.1408 23547.3657 16353.7672 
      1036       1037       1038       1039       1040       1041       1042 
 9972.1544 29403.8658 33976.6012  3258.4527  2297.3010 30084.2063   231.6483 
      1043       1044       1045       1046       1047       1048       1049 
27175.8495  2822.4706 14552.7358 31888.5500  7804.7607 34265.6128  2146.5830 
      1050       1051       1052       1053       1054       1055       1056 
33649.2764 12075.7742 13517.7324 11126.5799 33977.5689  1909.6591 11119.2972 
```

Prediction Using Multiple Linear Regression

- Output - insurance_model
- The Intercept is the predicted value of expenses when the independent variables are equal to zero
- The other Coefficients indicate the estimated increase in expenses for an increase of one in each of the features, assuming all other values are held constant
- For each additional year of age, medical expenses will be increased by 264.3, when all other features remain constant
- For each additional child, medical expenses will be increased by 410.2, when all other features remain constant

Prediction Using Multiple Linear Regression

- Output - summary(insurance_model)
- The Residuals section provides summary statistics for the errors in our prediction
- The Coefficients section provides statistics for the errors associated with regression coefficients
- The multiple R-squared value indicates the variation in the dependent variable, which is nearly 75 percent

k means Clustering Algorithm

- Program

```
# Write a program to partition the iris data set(given) into different clusters using k-means clustering algorithm.
# Choose k as 3.Check clustering result against species class label.

# set.seed() is used for generating the same sample in every execution
# We specify a seed number
set.seed(100)

# Reads the iris data set into the iris data frame
iris <- read.csv("iris.csv")

# Make a copy of iris data
iris2 <- iris

# Remove the species class label
iris2$species <- NULL

# Clustering with kmeans is performed by kmeans()
# kmeans() is contained within stats package which is loaded by default
# The kmeans() function requires a data frame containing only numeric data and a parameter specifying the desired number of clusters
# This function will return a cluster object that stores cluster information
# The cluster information includes cluster sizes, cluster means, vector of cluster assignments etc.
iris_clusters <- kmeans(iris2, 3)
print(iris_clusters)

# Check clustering result against species class label
# iris_clusters$cluster is a vector of cluster assignments from the kmeans()
# table() performs a tabulation of categorical variable and gives its frequency as output
table(iris$species, iris_clusters$cluster)
```

k means Clustering Algorithm

- Output

```
MES13s-Mac-mini:Data Science Lab mes13$ Rscript 33KMC.R
K-means clustering with 3 clusters of sizes 33, 21, 96

Cluster means:
  sepal.length sepal.width petal.length petal.width
1     5.175758    3.624242     1.472727   0.2727273
2     4.738095    2.904762     1.790476   0.3523810
3     6.314583    2.895833     4.973958   1.7031250

Clustering vector:
  [1] 1 2 2 2 1 1 1 1 2 2 1 1 2 2 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 2 1 1 1 2 1 1
 [38] 1 2 1 1 2 2 1 1 2 1 2 1 1 3 3 3 3 3 3 3 2 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3
 [75] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3
[112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[149] 3 3

Within cluster sum of squares by cluster:
[1]   6.432121  17.669524 118.651875
 (between_SS / total_SS =  79.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"

             1  2  3
  Setosa     33 17  0
  Versicolor  0  4 46
  Virginica   0  0 50
```