DR. D.Y. PATIL INSTITUTE OF TECHNOLOGY

Pimpri, Pune-411018

DEPARTMENT
Of
COMPUTER ENGINEERING

Lab Manual

Laboratory Practice III

VISION OF THE INSTITUTE:

"Empowerment through knowledge"

MISSION OF THE INSTITUTE:

"Developing human potential to serve the Nation by

- **Dedicated efforts for quality education**
- **Yearning to promote research and development**
- **Persistent endeavor to imbibe moral and professional ethics**
- **Inculcating the concept of emotional intelligence**
- **Emphasizing extension work to reach out to the**

with knowledge and power of innovation"

- **Treading the path to meet the future challenges**

VISION OF THE DEPARTMENT:

"To produce globally competitive computer professionals, enriched

MISSION OF THE DEPARTMENT:

- **"Imparting quality education using state-of-art facilities to meet the global challenges.**
- **Enhancing the potential of aspiring students and faculty for higher education and lifelong learning.**
- **Imbibing ethical values and developing leadership skills those lead to professionals with strong commitments."**

PROGRAMME EDUCATIONAL OBJECTIVES

PEO 1:

Have strong fundamental concepts in mathematics, science and engineering to address technological challenges.

PEO 2:

Possess knowledge and skills in the field of Computer Engineering for analyzing, designing and implementing novel software products in a dynamic environment for successful career and pursue higher studies.

PEO 3:

Demonstrate multidisciplinary approach and leadership skills that augment their professional competency.

PEO 4:

Exhibit commitment to ethical practices, societal contributions and lifelong learning.

Graduate Attributes and Program Outcomes

Graduate Attributes	Program Outcomes
1. Engineering Knowledge	**a.** An ability to apply knowledge of computing, mathematics, science and engineering fundamentals appropriate to Computer Engineering.
2. Problem Analysis	**b.** An ability to define the problems and provide solutions by designing and conducting experiments, interpreting and analyzing data.
3. Design & Development of	**c.** An ability to design, implement and evaluate a system, process, component and programme to meet desired needs within realistic constraints.

Solutions	
4. Investigation of Complex Problem	**d.** An ability to investigate, formulate, analyze and provide appropriate solution to the engineering problems.
5. Modern Tools Usage	**e.** An ability to use modern engineering tools and technologies necessary for engineering practice.
6. Engineer and Society	**f.** An ability to analyze the local and global impact of computing on individuals, organizations and society.
7. Environment & Sustainability	g. An ability to understand the environmental issues and provide the sustainable system
8. Ethics	**h.** An ability to understand professional and ethical responsibility.
9. Individual & Team work	**i.** An ability to function effectively as an individual or as a team member to accomplish the goal.
10. Communication	**j.** An ability to communicate effectively at different levels.
11. Lifelong Learning	**k.** An ability to keep abreast with contemporary technologies through lifelong learning.
12. Project management & Finance	**l.** An ability to understand engineering, management, financial aspects, performance, optimizations and time complexity necessary for professional practice.

A. SYLLABUS STRUCTURE

Savitribai Phule Pune University
Fourth Year of Computer Engineering (2015 Course)
(with effect from 2018-19)

Semester II

Course Code	Course	Teaching Scheme Hours / Week		Examination Scheme and Marks						Credit	
		Theory	Practical	In-Sem	End-Sem	TW	PR	OR/ *PRE	Total	TH/ TUT	PR
410250	Machine Learning	03	--	30	70	--	--	--	100	03	--
410251	Information and Cyber Security	03	--	30	70	--	--	--	100	03	--
410252	Elective III	03	--	30	70	--	--	--	100	03	--
410253	Elective IV	03	--	30	70	--	--	--	100	03	--
410254	Laboratory Practice III	--	04	--	--	50	50	--	100	--	02
410255	Laboratory Practice IV	--	04	--	--	50	--	*50	100	--	02
410256	Project Work Stage II	--	06	--	--	100	--	*50	150	--	06
									Total Credit	12	10
Total		12	14	120	280	200	50	100	750	22	
410257	Audit Course 6									Grade	

Elective III	Elective IV
410252 (A) Advanced Digital Signal Processing	410253 (A) Software Defined Networks
410252 (B) Compilers	410253 (B) Human Computer Interface
410252 (C) Embedded and Real Time Operating System	410253 (C) Cloud Computing
410252 (D) Soft Computing and Optimization Algorithms	410253 (D) Open Elective

410259-Audit Course 6 (AC6) Options:

AC6-I: Business Intelligence
AC6-II: Gamification
AC6-III: Quantum Computing
AC6-IV: Usability Engineering
AC6-V: Conversational Interfaces
AC6-VI: MOOC- Learn New Skills

Abbreviations:

TW: Term Work **TH**: Theory **OR**: Oral **PR**: Practical
Sem: Semester **PRE**: Project/ Mini-Project Presentation

Savitribai Phule Pune University
Fourth Year of Computer Engineering (2015 Course)
410254:Laboratory Practice III

Teaching Scheme:	Credit	Examination Scheme:
Practical : 04 Hours/Week	02	Term Work: 50 Marks Practical: 50 Marks

Companion Courses: 410250 and 410251

Course Objectives and Outcomes: Practical hands on is the absolute necessity as far as employability of the learner is concerned. The presented course is solely intended to enhance the competency by undertaking the laboratory assignments of the core courses.

About

Laboratory Practice III is for practical hands on for core courses Machine Learning and Information & Cyber Security.

Guidelines for Laboratory Conduction

- List of recommended programming assignments and sample mini-projects is provided for reference.
- Referring these, Course Teacher or Lab Instructor may frame the assignments/mini-project by understanding the prerequisites, technological aspects, utility and recent trends related to the respective courses.
- Preferably there should be multiple sets of assignments/mini-project and distribute among batches of students.
- Real world problems/application based assignments/mini-projects create interest among learners serving as foundation for future research or startup of business projects.
- Mini-project can be completed in group of 2 to 3 students.
- Software Engineering approach with proper documentation is to be strictly followed.
- Use of open source software is to be encouraged.
- Instructor may also set one assignment or mini-project that is suitable to respective course beyond the scope of syllabus.

Operating System recommended :- 64-bit Open source Linux or its derivative

Programming Languages: C++/JAVA/PYTHON/R

Programming tools recommended: Front End: Java/Perl/PHP/Python/Ruby/.net, Backend : MongoDB/MYSQL/Oracle, Database Connectivity : ODBC/JDBC, Additional Tools: Octave, Matlab, WEKA.

Guidelines for Student Journal

The laboratory assignments are to be submitted by student in the form of journal. Journal may consists of prologue, Certificate, table of contents, and **handwritten write-up** of each assignment (Title, Objectives, Problem Statement, Outcomes, software and Hardware requirements, Date of Completion, Assessment grade/marks and assessor's sign, Theory- Concept in brief, Algorithm/Database design, test cases, conclusion/analysis). **Program codes with sample output of all performed assignments are to be submitted as softcopy.**

As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and program listing to journal may be avoided. Use of digital storage media/DVD containing students programs maintained by lab In-charge is highly encouraged. For reference one or two journals may be maintained with program prints at Laboratory.

Continuous assessment of laboratory work is to be done based on overall performance and lab assignments performance of student. Each lab assignment assessment will assign grade/marks based on parameters with appropriate weightage. Suggested parameters for overall assessment as well as each lab assignment assessment include- timely completion, performance, innovation, efficient codes, punctuality and neatness **reserving weightage for successful mini-project completion and related documentation.**

Guidelines for Practical Examination

- Both internal and external examiners should jointly frame suitable problem statements for practical examination based on the term work completed.
- During practical assessment, the expert evaluator should give the maximum weightage to the satisfactory implementation of the problem statement.
- The supplementary and relevant questions may be asked at the time of evaluation to test the student's for advanced learning, understanding of the fundamentals, effective and efficient implementation.
- Encouraging efforts, transparent evaluation and fair approach of the evaluator will not create any uncertainty or doubt in the minds of the students. So adhering to these principles will consummate our team efforts to the promising boost to the student's academics.

Guidelines for Instructor's Manual

The instructor's manual is to be developed as a hands-on resource and as ready reference. The instructor's manual need to include prologue (about University/program/ institute/ department/foreword/ preface etc), University syllabus, conduction and Assessment guidelines, topics under consideration-concept, objectives, outcomes, set of typical applications/assignments/ guidelines, references among others.

Suggested List of Laboratory Assignments& Mini Projects

(any 04 assignments Machine Learning and Information & Cyber Security AND Mini-project per course)

410250: Machine Learning

1. **Assignment on Linear Regression:**
The following table shows the results of a recently conducted study on the correlation of the number of hours spent driving with the risk of developing acute backache. Find the equation of the best fit line for this data.

Number of hours spent driving (x)	Risk score on a scale of 0-100 (y)
10	95
9	80
2	10
15	50
10	45
16	98
11	38
16	93

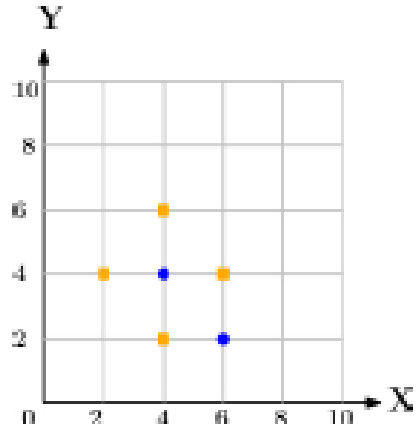
2. **Assignment on Decision Tree Classifier:**
A dataset collected in a cosmetics shop showing details of customers and whether or not they responded to a special offer to buy a new lip-stick is shown in table below. Use this dataset to

build a decision tree, with Buys as the target variable, to help in buying lip-sticks in the future. Find the root node of decision tree. According to the decision tree you have made from previous training data set, what is the decision for the test data: [Age < 21, Income = Low, Gender = Female, Marital Status = Married]?

ID	Age	Income	Gender	Marital Status	Buys
1	< 21	High	Male	Single	No
2	< 21	High	Male	Married	No
3	21-35	High	Male	Single	Yes
4	>35	Medium	Male	Single	Yes
5	>35	Low	Female	Single	Yes
6	>35	Low	Female	Married	No
7	21-35	Low	Female	Married	Yes
8	< 21	Medium	Male	Single	No
9	<21	Low	Female	Married	Yes
10	> 35	Medium	Female	Single	Yes
11	< 21	Medium	Female	Married	Yes
12	21-35	Medium	Male	Married	Yes
13	21-35	High	Female	Single	Yes
14	> 35	Medium	Male	Married	No

3. **Assignment on k-NN Classification:**

In the following diagram let blue circles indicate positive examples and orange squares indicate negative examples. We want to use k-NN algorithm for classifying the points. If k=3, find the class of the point (6,6). Extend the same example for Distance-Weighted k-NN and Locally weighted Averaging



4. **Assignment on K-Means Clustering:**

We have given a collection of 8 points. P1=[0.1,0.6] P2=[0.15,0.71] P3=[0.08,0.9] P4=[0.16, 0.85] P5=[0.2,0.3] P6=[0.25,0.5] P7=[0.24,0.1] P8=[0.3,0.2]. Perform the k-mean clustering with initial centroids as m1=P1 =Cluster#1=C1 and m2=P8=cluster#2=C2. Answer the following

1] Which cluster does P6 belongs to?
2] What is the population of cluster around m2?
3] What is updated value of m1 and m2?

6.	**Mini-Project 2 on SVM:** Apply the Support vector machine for classification on a dataset obtained from UCI ML repository. For Example: Fruits Classification or Soil Classification or Leaf Disease Classification
7.	**Mini-Project 3 on PCA:** Apply the Principal Component Analysis for feature reduction on any Company Stock Market Dataset
	410251:: : Information and Cyber Security
1.	Implementation of S-DES
2.	Implementation of S-AES
3.	Implementation of Diffie-Hellman key exchange
4.	Implementation of RSA.
5.	Implementation of BCC algorithm.
6.	**Mini Project 1:** SQL Injection attacks and Cross -Site Scripting attacks are the two most common attacks on web application. Develop a new policy based Proxy Agent, which classifies the request as a scripted request or query based request, and then, detects the respective type of attack, if any in the request. It should detect both SQL injection attack as well as the Cross-Site Scripting attacks.
7.	**Mini Project 2:**This task is to demonstrate insecure and secured website. Develop a web site and demonstrate how the contents of the site can be changed by the attackers if it is http based and not secured. You can also add payment gateway and demonstrate how money transactions can be hacked by the hackers. Then support your website having https with SSL and demonstrate how secured website is.

Course Objective:

To understand human learning aspects and relate it with machine learning concepts.
To understand nature of the problem and apply machine learning algorithm.
To find optimized solution for given problem.

To offer an understanding of principle concepts, central topics and basic approaches in information and cyber security.
To know the basics of cryptography.
To acquire knowledge of standard algorithms and protocols employed to provide confidentiality, integrity and authenticity.
To enhance awareness about Personally Identifiable Information (PII), Information Management, cyber forensics.

Course Outcome:

On completion of the course, student will be able to–

Distinguish different learning based applications
Apply different preprocessing methods to prepare training data set for machine learning. Design and implement supervised and unsupervised machine learning algorithm.
Implement different learning models Learn Meta classifiers and deep learning concept

Gauge the security protections and limitations provided by today's technology.
Identify information security and cyber security threats.
Analyze threats in order to protect or defend it in cyberspace from cyber-attacks.
Build appropriate security solutions against cyber-attacks.

ML Assignment No. 1

1.1 Title

Assignment based on Linear Regression.

1.2 Problem Definition:

The following table shows the results of a recently conducted study on the correlation of the number of hours spent driving with the risk of developing acute backache. Find the equation of the best fit line for this data.

1.3 Prerequisite:

Basic of Python, Data Mining Algorithm

1.4 Software Requirements:

Anaconda with Python 3.7

1.5 Hardware Requirement:

PIV, 2GB RAM, 500 GB HDD, Lenovo A13-4089Model.

1.6 Learning Objectives:

Learn Linear Regression for given different Dataset

1.7 Outcomes:

After completion of this assignment students are able to understand the How to find the correlation between to Two variable, How to Calculate Accuracy of the Linear Model and how to plot graph using matplotlib.

1.8 Theory Concepts:

1.8.1 Linear Regression

Regression analysis is used in stats to find trends in data. For example, you might guess that there's a connection between how much you eat and how much you weight; regression analysis can help you quantify that.

What is Linear Regression?

In a cause and effect relationship, the **independent variable** is the cause, and the **dependent variable** is the effect. **Least squares linear regression** is a method for predicting the value of a dependent variable *Y*, based on the value of an independent variable *X*.

Prerequisites for Regression

Simple linear regression is appropriate when the following conditions are satisfied.

- The dependent variable *Y* has a linear relationship to the independent variable *X*. To check this, make sure that the XY scatterplot is linear and that the residual plot shows a random pattern. For each value of X, the probability distribution of Y has the same standard deviation σ.
- When this condition is satisfied, the variability of the residuals will be relatively constant across all values of X, which is easily checked in a residual plot.
- For any given value of X,
    - The Y values are independent, as indicated by a random pattern on the residual plot.
    - The Y values are roughly normally distributed (i.e., symmetric and unimodal). A little skewness is ok if the sample size is large. A histogram or a dotplot will show the shape of the distribution.

The Least Squares Regression Line

Linear regression finds the straight line, called the **least squares regression line** or LSRL, that best represents observations in a bivariate data set. Suppose *Y* is a dependent variable, and *X* is an independent variable. The population regression line is:

$$Y = B_0 + B_1 X$$

where B_0 is a constant, B_1 is the regression coefficient, X is the value of the independent variable, and Y is the value of the dependent variable.

Given a random sample of observations, the population regression line is estimated by:

$$\hat{y} = b_0 + b_1 x$$

where b_0 is a constant, b_1 is the regression coefficient, x is the value of the independent variable, and $\hat{y}$ is the *predicted* value of the dependent variable.

How to Define a Regression Line

Normally, you will use a computational tool - a software package (e.g., Excel) or a graphing calculator - to find b_0 and b_1. You enter the *X* and *Y* values into your program or calculator, and the tool solves for each parameter.

In the unlikely event that you find yourself on a desert island without a computer or a graphing calculator, you can solve for b_0 and b_1 "by hand". Here are the equations.

$$b_1 = \Sigma [(x_i-x)(y_i-y)] / \Sigma [(x_i-x)^2]$$

$$b_1 = r * (s_y / s_x)$$

$$b_0 = y - b_1 * x$$

where b_0 is the constant in the regression equation, b_1 is the regression coefficient, r is the correlation between x and y, x_i is the *X* value of observation *i*, y_i is the *Y* value of observation *i*, x is the mean of *X*, y is the mean of *Y*, s_x is the standard deviation of *X*, and s_y is the standard deviation of *Y*.

Properties of the Regression Line

When the regression parameters (b_0 and b_1) are defined as described above, the regression line has the following properties.

- The line minimizes the sum of squared differences between observed values (the *y* values) and predicted values (the $\hat{y}$ values computed from the regression equation).
- The regression line passes through the mean of the *X* values (x) and through the mean of the *Y* values (y).
- The regression constant (b_0) is equal to the y intercept of the regression line.
- The regression coefficient (b_1) is the average change in the dependent variable (*Y*) for a 1-unit change in the independent variable (*X*). It is the slope of the regression line.

The least squares regression line is the only straight line that has all of these properties.

The Coefficient of Determination

The **coefficient of determination** (denoted by R^2) is a key output of regression analysis. It is interpreted as the proportion of the variance in the dependent variable that is predictable from the independent variable.

- The coefficient of determination ranges from 0 to 1.
- An R^2 of 0 means that the dependent variable cannot be predicted from the independent variable.
- An R^2 of 1 means the dependent variable can be predicted without error from the independent variable.
- An R^2 between 0 and 1 indicates the extent to which the dependent variable is predictable. An R^2 of 0.10 means that 10 percent of the variance in *Y* is predictable from *X*; an R^2 of 0.20 means that 20 percent is predictable; and so on.

The formula for computing the coefficient of determination for a linear regression model with one independent variable is given below.

Coefficient of determination. The coefficient of determination (R^2) for a linear regression model with one independent variable is:

$$R^2 = \{ (1 / N) * \Sigma [(x_i - x) * (y_i - y)] / (\sigma_x * \sigma_y) \}^2$$

where N is the number of observations used to fit the model, Σ is the summation symbol, x_i is the x value for observation i, x is the mean x value, y_i is the y value for observation i, y is the mean y value, σ_x is the standard deviation of x, and σ_y is the standard deviation of y.

If you know the linear correlation (r) between two variables, then the coefficient of determination (R^2) is easily computed using the following formula: $R^2 = r^2$.

Standard Error

The **standard error** about the regression line (often denoted by SE) is a measure of the average amount that the regression equation over- or under-predicts. The higher the coefficient of determination, the lower the standard error; and the more accurate predictions are likely to be.

Example

Last year, five randomly selected students took a math aptitude test before they began their statistics course. The Statistics Department has three questions.

- What linear regression equation best predicts statistics performance, based on math aptitude scores?
- If a student made an 80 on the aptitude test, what grade would we expect her to make in statistics?
- How well does the regression equation fit the data?

How to Find the Regression Equation

In the table below, the x_i column shows scores on the aptitude test. Similarly, the y_i column shows statistics grades. The last two columns show deviations scores - the difference between the student's score and the average score on each test. The last two rows show sums and mean scores that we will use to conduct the regression analysis.

Student	x_i	y_i	(x_i-x)	(y_i-y)
1	95	85	17	8
2	85	95	7	18
3	80	70	2	-7
4	70	65	-8	-12
5	60	70	-18	-7
Sum	390	385		
Mean	78	77		

And for each student, we also need to compute the squares of the deviation scores (the last two columns in the table below).

Student	x_i	y_i	$(x_i-x)^2$	$(y_i-y)^2$
1	95	85	289	64
2	85	95	49	324
3	80	70	4	49
4	70	65	64	144
5	60	70	324	49
Sum	390	385	730	630
Mean	78	77		

And finally, for each student, we need to compute the product of the deviation scores.

Student	x_i	y_i	$(x_i-x)(y_i-y)$
1	95	85	136
2	85	95	126
3	80	70	-14
4	70	65	96
5	60	70	126
Sum	390	385	470
Mean	78	77	

The regression equation is a linear equation of the form: $\hat{y} = b_0 + b_1x$. To conduct a regression analysis, we need to solve for b_0 and b_1. Computations are shown below. Notice that all of our inputs for the regression analysis come from the above three tables.

First, we solve for the regression coefficient (b_1):

$$b_1 = \Sigma [(x_i-x)(y_i-y)] / \Sigma [(x_i-x)^2]$$

$$b_1 = 470/730$$

$$b_1 = 0.644$$

Once we know the value of the regression coefficient (b_1), we can solve for the regression slope (b_0):

$$b_0 = y - b_1 * x$$

$$b_0 = 77 - (0.644)(78)$$

$$b_0 = 26.768$$

Therefore, the regression equation is: $\hat{y} = 26.768 + 0.644x$.

How to Use the Regression Equation

Once you have the regression equation, using it is a snap. Choose a value for the independent variable (*x*), perform the computation, and you have an estimated value ($\hat{y}$) for the dependent variable.

In our example, the independent variable is the student's score on the aptitude test. The dependent variable is the student's statistics grade. If a student made an 80 on the aptitude test, the estimated statistics grade ($\hat{y}$) would be:

$$\hat{y} = b_0 + b_1 x$$

$$\hat{y} = 26.768 + 0.644x = 26.768 + 0.644 * 80$$

$$\hat{y} = 26.768 + 51.52 = 78.288$$

Warning: When you use a regression equation, do not use values for the independent variable that are outside the range of values used to create the equation. That is called **extrapolation**, and it can produce unreasonable estimates.

In this example, the aptitude test scores used to create the regression equation ranged from 60 to 95. Therefore, only use values inside that range to estimate statistics grades. Using values outside that range (less than 60 or greater than 95) is problematic.

How to Find the Coefficient of Determination

Whenever you use a regression equation, you should ask how well the equation fits the data. One way to assess fit is to check the coefficient of determination, which can be computed from the following formula.

$$R^2 = \{ (1 / N) * \Sigma [(x_i - x) * (y_i - y)] / (\sigma_x * \sigma_y) \}^2$$

where N is the number of observations used to fit the model, Σ is the summation symbol, x_i is the x value for observation i, x is the mean x value, y_i is the y value for observation i, y is the mean y value, σ_x is the standard deviation of x, and σ_y is the standard deviation of y.

Computations for the sample problem of this lesson are shown below. We begin by computing the standard deviation of x (σ_x):

$$\sigma_x = \text{sqrt} [\Sigma (x_i - x)^2 / N]$$

$$\sigma_x = \text{sqrt}(730/5) = \text{sqrt}(146) = 12.083$$

Next, we find the standard deviation of y, (σ_y):

$$\sigma_y = \text{sqrt} [\Sigma (y_i - y)^2 / N]$$

$$\sigma_y = \text{sqrt}(630/5) = \text{sqrt}(126) = 11.225$$

And finally, we compute the coefficient of determination (R^2):

$$R^2 = \{ (1 / N) * \Sigma [(x_i - x) * (y_i - y)] / (\sigma_x * \sigma_y) \}^2$$

$$R^2 = [(1/5) * 470 / (12.083 * 11.225)]^2$$

$$R^2 = (94 / 135.632)^2 = (0.693)^2 = 0.48$$

A coefficient of determination equal to 0.48 indicates that about 48% of the variation in statistics grades (thedependent variable) can be explained by the relationship to math aptitude scores (the independent variable). This would be considered a good fit to the data, in the sense that it would substantially improve an educator's ability to predict student performance in statistics class.

Residuals

The difference between the observed value of the dependent variable (y) and the predicted value ($\hat{y}$) is called the **residual** (e). Each data point has one residual.

$$\text{Residual} = \text{Observed value} - \text{Predicted value}$$
$$e = y - \hat{y}$$

Both the sum and the mean of the residuals are equal to zero. That is, $\Sigma e = 0$ and $e = 0$.

What is Best Fit Line- A line of best fit (or "trend" line) is a straight line that best represents the data on a scatter plot. This line may pass through some of the points, none of the points, or all of the points.

Given Dataset in Our Definition-

Number of hours spent driving (x)	Risk score on a scale of 0-100 (y)
10	95
9	80
2	10
15	50
10	45
16	98
11	38
16	93

Algorithm

Import the Required Packages
Read Given Dataset

1. Import the Linear Regression and Create object of it

Find the Accuracy of Model using Score Function

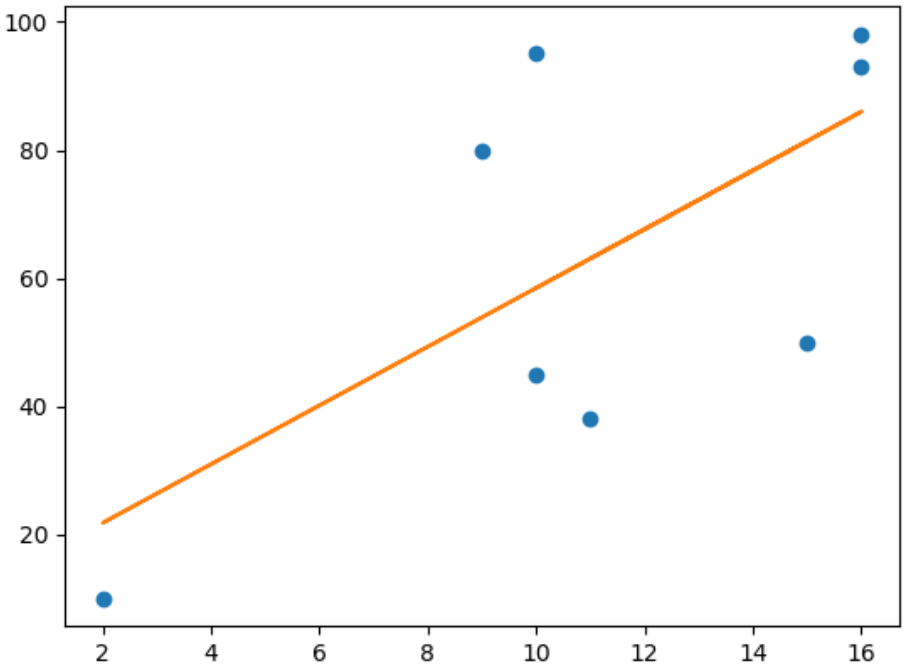
2. Predict the value using Regressor Object

Take input from user.
Calculate the value of y
Draw Scatter Plot

Important Function Used for Linear Regression

coef_ it is used to calculate slope in ML Model
Intercept_ it is used to calculate intercept in ML Model
fit - it shows the relationship between two varraible
score – it display accuracy score of model

Scatter Plot generated after Implementation of Code



1.9 Conclusion

Thus we learn that to how to find the trend of data using X as Independent Variable and Y is and Dependent Variable by using Linear Regression.

1.10 Assignment Questions

1. What is Linear Regression?
2. What is Sigma?
3. What is Mu?
4. What is Probability of Distribution in term of Binominal and Frequency Distribution?
5. What is Standard Deviation?
6. What is Coefficient of Determination?

References:-

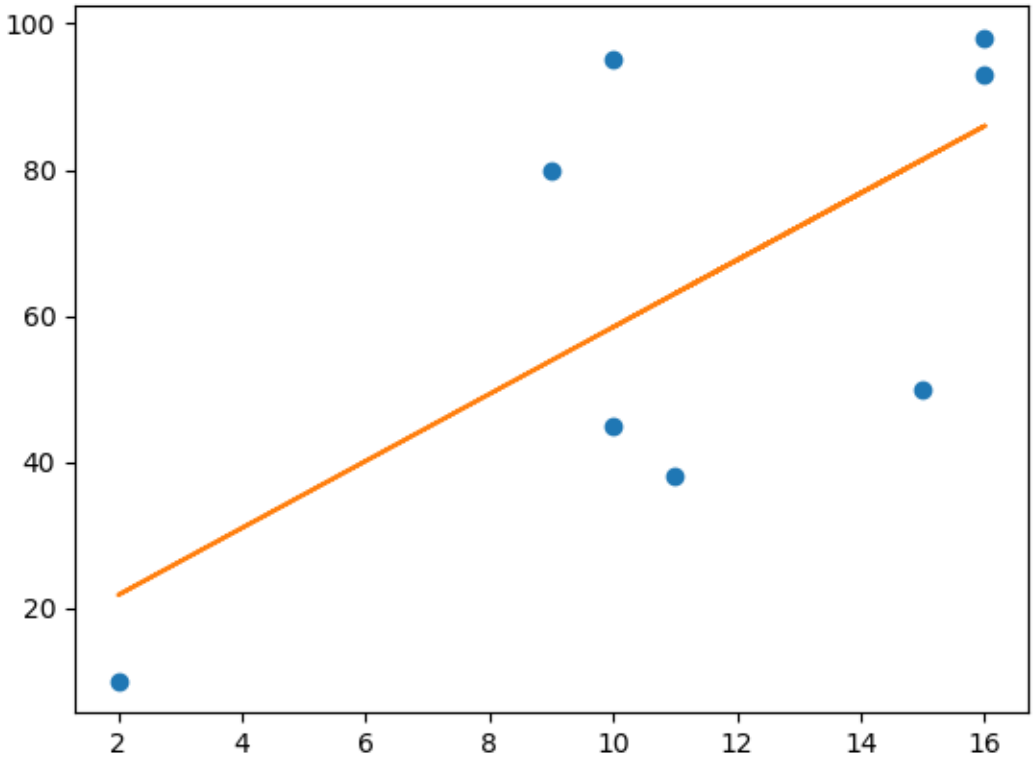
1. https://stattrek.com/regression/linear-regression.aspx?Tutorial=AP
2. Mittu Skillologies Youtube Channel

```
import matplotlib.pyplot as plt
import pandas as pd

# Read Dataset
dataset=pd.read_csv("hours.csv")
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,1].values

# Import the Linear Regression and Create object of it
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X,y)
Accuracy=regressor.score(X, y)*100
print("Accuracy :")
print(Accuracy)

# Predict the value using Regressor Object
y_pred=regressor.predict([[10]])
print(y_pred)

# Take user input
hours=int(input('Enter the no of hours'))

#calculate the value of y
eq=regressor.coef_*hours+regressor.intercept_
y='%f*%f+%f' %(regressor.coef_,hours,regressor.intercept_)
print("y :")
print(y)
print("Risk Score : ", eq[0])
plt.plot(X,y,'o')
plt.plot(X,regressor.predict(X));
plt.show()
```

Output
Accuracy :
43.709481451010035
[58.46361406]

Enter the no of hours 10
y :
4.587899*10.000000+12.584628
Risk Score : 58.4636140637776

Graph



ML Assignment No. 2

2.1 Title

Assignment based on Decision Tree Classfier

2.2 Problem Definition:

A dataset collected in a cosmetics shop showing details of customers and whether or not they responded to a special offer to buy a new lip-stick is shown in table below. Use this dataset to build a decision tree, with Buys as the target variable, to help in buying lip-sticks in the future. Find the root node of decision tree. According to the decision tree you have made from previous training data set, what is the decision for the test data: [Age < 21, Income = Low, Gender = Female, Marital Status = Married]?

2.3 Prerequisite:

Basic of Python, Data Mining Algorithm, Concept of Decision Tree Classifier

2.4 Software Requirements:

Anaconda with Python 3.7

2.5 Hardware Requirement:

PIV, 2GB RAM, 500 GB HDD, Lenovo A13-4089Model.

2.6 Learning Objectives:

Learn How to Apply Decision Tree Classifier to find the root node of decision tree. According to the decision tree you have made from previous training data set.

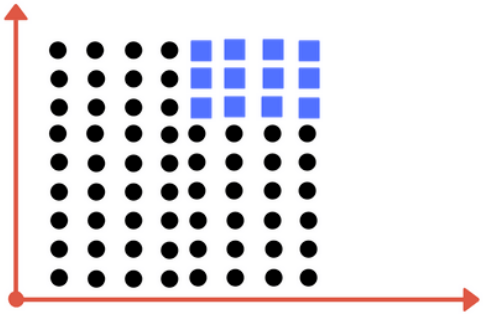
2.7 Outcomes:

After completion of this assignment students are able Implement code for Create Decision tree for given dataset and find the root node for same based on the given condition.

2.8 Theory Concepts:

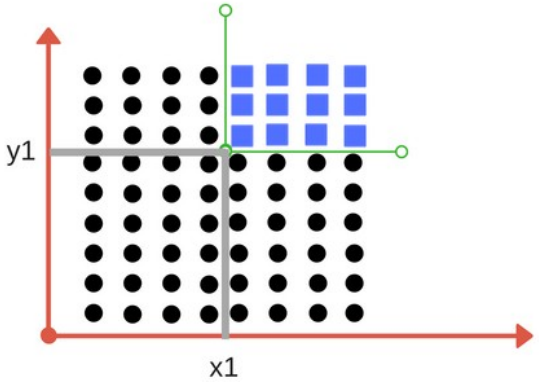
2.8.1 Motivation

Suppose we have following plot for two classes represented by black circle and blue squares. Is it possible to draw a single separation line ? Perhaps no.



Can you draw single division line for theseclasses?

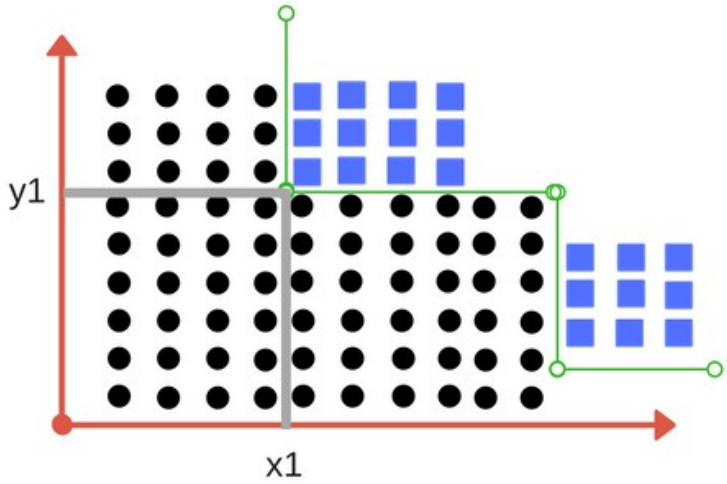
We will need more than one line, to divide into classes. Something similar to following image:



We need two lines one for threshold of x and threshold fory.

We need two lines here one separating according to threshold value of ***x*** and other for threshold value of ***y.***

Decision Tree Classifier, repetitively divides the working area(plot) into sub part by identifying lines. (repetitively because there may be two distant regions of same class divided by other as shown in image below).



So when does it terminate?

Either it has divided into classes that are pure (only containing members of single class)

Some criteria of classifier attributes are met.

Impurity-

In above division, we had clear separation of classes. But what if we had following case?

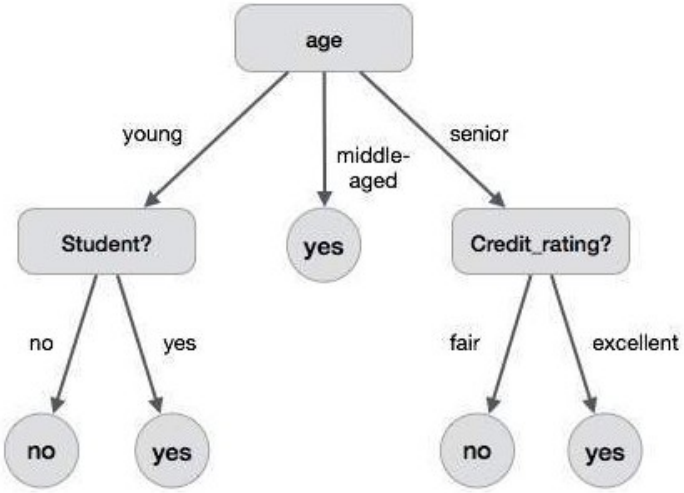
Impurity is when we have a traces of one class division into other. This can arise due to following reason

We run out of available features to divide the class upon.

We tolerate some percentage of impurity (we stop further division) for faster performance. (There is always trade off between accuracy and performance).

For example in second case we may stop our division when we have x number of fewer number

of elements left. This is also known as *gini impurity.*



Division based on some features.

2. Entropy

Entropy is degree of randomness of elements or in other words it is measure of impurity. Mathematically, it can be calculated with the help of probability of the items as:

$$H = -\sum p(x) \log p(x)$$

p(x) is probability of item x.

It is negative summation of probability times the log of probability of item x.

For example,

if we have items as number of dice face occurrence in a throw event as 1123,

the entropy is

p(1) = 0.5

p(2) = 0.25

p(3) = 0.25

***entropy** = - (0.5 * log(0.5)) - (0.25 * log(0.25)) -(0.25 * log(0.25)*

*= **0.45***

3. Information Gain

Suppose we have multiple features to divide the current working set. What feature should we select for division? Perhaps one that gives us less impurity.

Suppose we divide the classes into multiple branches as follows, the information gain at any node is defined as

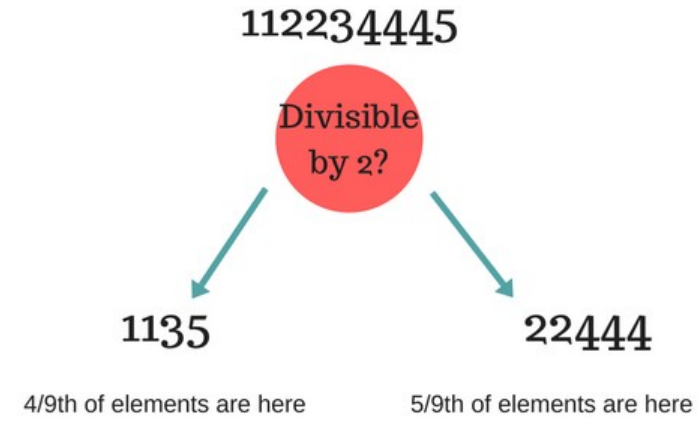
Information Gain (n) = Entropy(x) — ([weighted average] * entropy(children for feature))

This need a bit explanation!

Suppose we have following class to work with intially

112234445

Suppose we divide them based on property: divisible by 2



Entropy at root level : 0.66

Entropy of left child : 0.45 , weighted value = (4/9) * 0.45 = 0.2

Entropy of right child: 0.29 , weighted value = (5/9) * 0.29 = 0.16

Information Gain = 0.66 - [0.2 + 0.16] = ***0.3***

Check what information gain we get if we take decision as **prime number instead of divide by 2.** Which one is better for this case?

Decision tree at every stage selects the one that gives best information gain.

When information gain is 0 means the feature does not divide the working set at all.

Given Data set in Our Definition

ID	Age	Income	Gender	Marital Status	Buys
1	< 21	High	Male	Single	No
2	< 21	High	Male	Married	No
3	21-35	High	Male	Single	Yes
4	>35	Medium	Male	Single	Yes
5	>35	Low	Female	Single	Yes
6	>35	Low	Female	Married	No
7	21-35	Low	Female	Married	Yes
8	< 21	Medium	Male	Single	No
9	<21	Low	Female	Married	Yes
10	> 35	Medium	Female	Single	Yes
11	< 21	Medium	Female	Married	Yes
12	21-35	Medium	Male	Married	Yes
13	21-35	High	Female	Single	Yes
14	> 35	Medium	Male	Married	No

What is the decision for the test data: [Age < 21, Income = Low, Gender = Female, Marital Status = Married]?

Answer is Whether Yes or No??

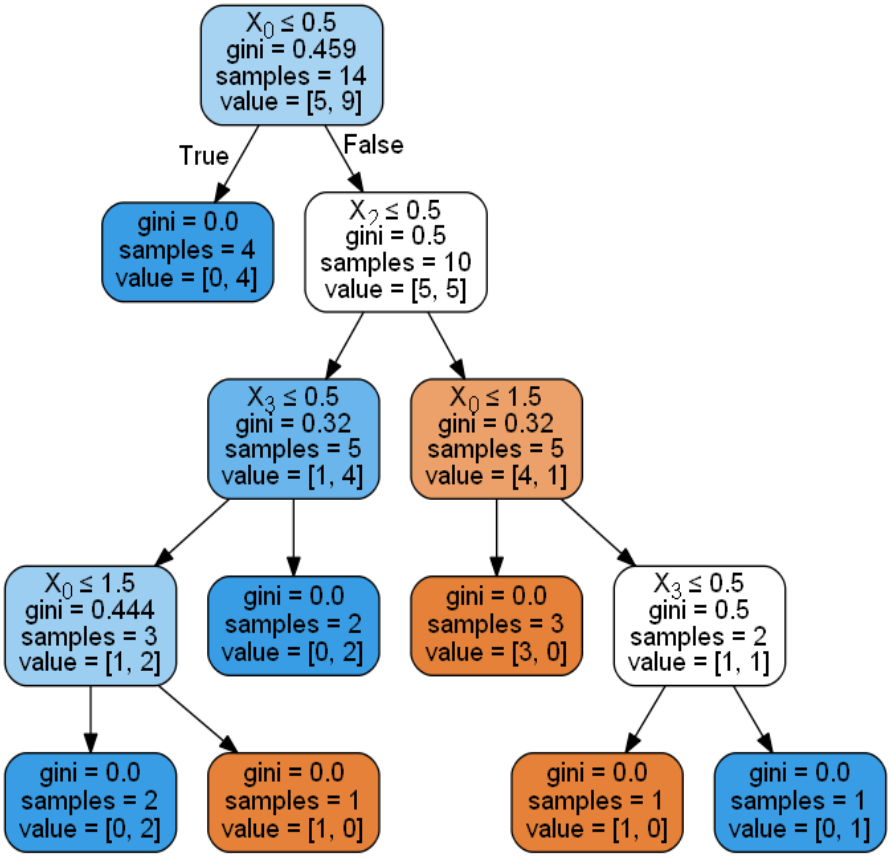
1. Which of the attributes would be the root node.
A. Age
B. Income
C. Gender
D. Marital Status
Solution: C
2. What is the decision for the test data [Age < 21, Income = Low, Gender = Female, Marital Status = Married]?
A. Yes
B. No
Solution: A
[Hints: construct the decision tree to answer these questions]

Algorithm

Import the Required Packages
Read Given Dataset

3. Perform the label Encoding Mean Convert String value into Numerical values
4. Import and Apply Decision Tree Classifier
5. Predict value for the given Expression like [Age < 21, Income = Low, Gender = Female, Marital Status = Married]? In encoding Values [1,1,0,0]
6. Import the packages for Create Decision Tree.
7. Check the Decision Tree Created based on Expression.

Decision Tree Generated after Implementation of Code



2.12 Conclusion

In this way we learn that to how to create Decision Tree based on given decision, Find the Root Node of the tree using Decision tree Classifier.

2.13 Assignment Questions

7. What is gini index? What is the Formula for Calculate same?
8. What is label Encoder? How it differ from One Hot Encoder?

9. What is Overfitting Problem while building a decision tree model?
10. What is Pre-pruning and Post pruning Approach in decision tree model?
11. What are different advantages and disadvantages of decision tree algorithm?

References:-

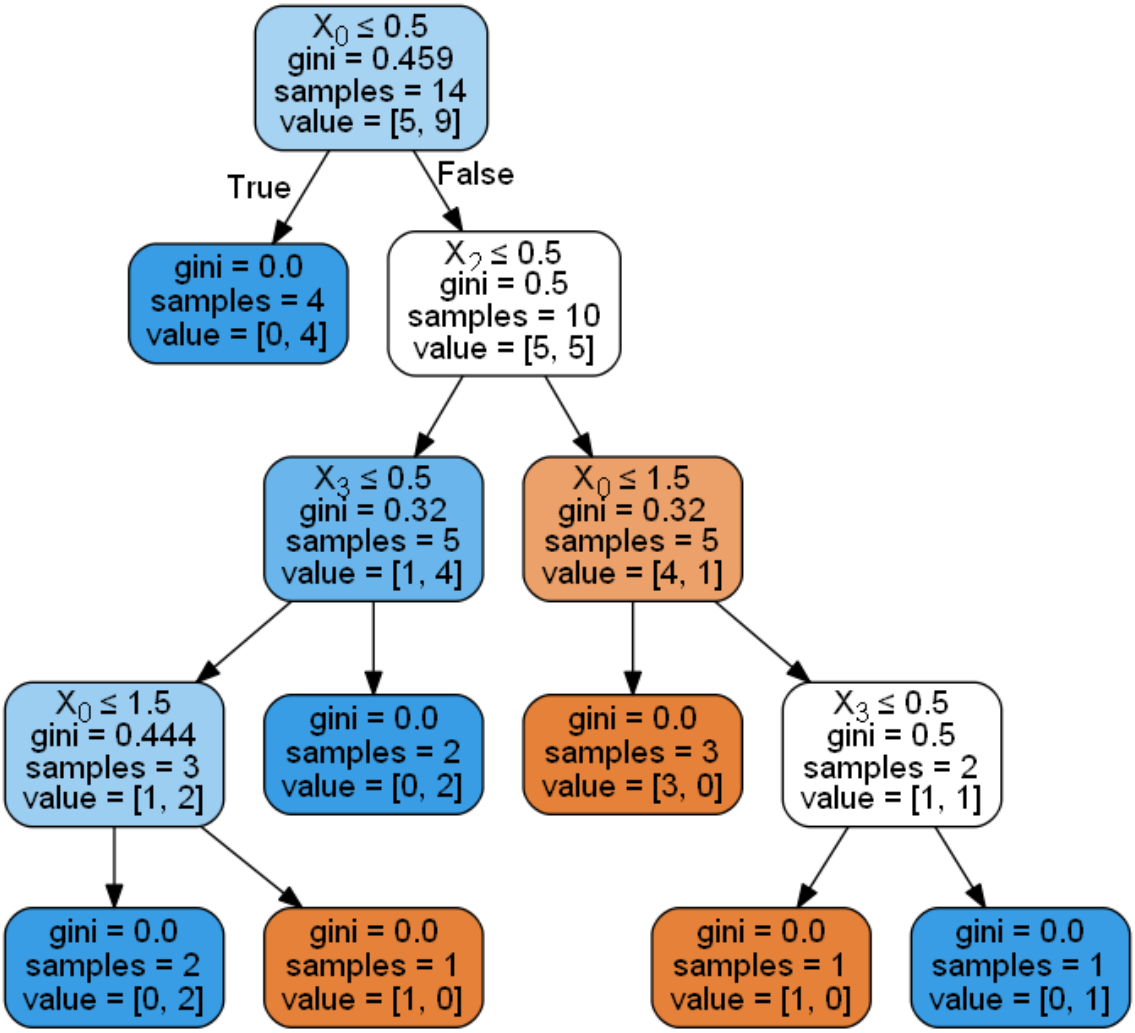
3. https://medium.com/machine-learning-101/chapter-3-decision-trees-theory-e7398adac567
4. Mittu Skillologies Youtube Channel
5. http://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/


```
import pandas as pd
import numpy as np

#reading Dataset
dataset=pd.read_csv("data.csv")
X=dataset.iloc[:,:-1]
y=dataset.iloc[:,5]

#Perform Label encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

X=X.apply(le.fit_transform)
print("X")

from sklearn.tree import DecisionTreeClassifier
regressor=DecisionTreeClassifier()
regressor.fit(X.iloc[:,1:5],y)

#Predict value for the given Expression
X_in=np.array([1,1,0,0])
y_pred=regressor.predict([X_in])
print("Prediction:", y_pred)
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
```

```
dot_data=StringIO()

export_graphviz(regressor,out_file=dot_data,filled=True,rounded=True,special_characters=True)
graph=pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('tree.png')
```

Output
X
Prediction: ['Yes']

Decision Tree generated-


$X_0 \leq 0.5$
gini = 0.459
samples = 14
value = [5, 9]
True
False
gini = 0.0
samples = 4
value = [0, 4]
$X_2 \leq 0.5$
gini = 0.5
samples = 10
value = [5, 5]
$X_3 \leq 0.5$
gini = 0.32
samples = 5
value = [1, 4]
$X_0 \leq 1.5$
gini = 0.32
samples = 5
value = [4, 1]
$X_0 \leq 1.5$
gini = 0.444
samples = 3
value = [1, 2]
gini = 0.0
samples = 2
value = [0, 2]
gini = 0.0
samples = 3
value = [3, 0]
$X_3 \leq 0.5$
gini = 0.5
samples = 2
value = [1, 1]
gini = 0.0
samples = 2
value = [0, 2]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 1
value = [0, 1]

ML Assignment No. 3

3.1 Title

Assignment based on k-NN Classification

3.2 Problem Definition:

In the following diagram let blue circles indicate positive examples and orange squares indicate negative examples. We want to use k-NN algorithm for classifying the points. If k=3, find-the class of the point (6,6). Extend the same example for Distance-Weighted k-NN and Locally weighted Averaging.

3.3 Prerequisite:

Basic of Python, Data Mining Algorithm, Concept of KNN Classification

3.4 Software Requirements:

Anaconda with Python 3.7

3.5 Hardware Requirement:

PIV, 2GB RAM, 500 GB HDD, Lenovo A13-4089Model.

3.6 Learning Objectives:

Learn How to Apply KNN Classification for Classify Positive and Negative Points in given example.

3.7 Outcomes:

After completion of this assignment students are able Implement code for KNN Classification for Classify Positive and Negative Points in given example also Extend the same example for Distance-Weighted k-NN and locally weighted Averaging

3.8 Theory Concepts:

3.8.1 Motivation

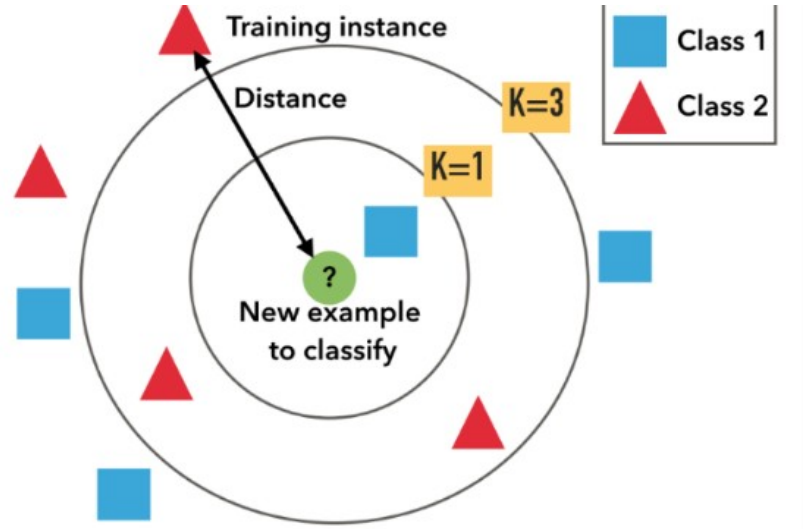
K-Nearest Neighbors (kNN) Algorithm-

KNN is an *non parametric lazy learning* algorithm. That is a pretty concise statement. When you say a technique is non parametric , it means that it does not make any assumptions on the underlying data distribution. This is pretty useful , as in the real world , most of the practical data does not obey the typical theoretical assumptions made (eg gaussian mixtures, linearly separable etc) . Non parametric algorithms like KNN come to the rescue here.

It is also a lazy algorithm. What this means is that it does not use the training data points to do any *generalization*. In other words, there is *no explicit training phase* or it is very minimal. This means the training phase is pretty fast . Lack of generalization means that KNN keeps all the training data. More exactly, all the training data is needed during the testing phase. (Well this is an exaggeration, but not far from truth). This is in contrast to other techniques like SVM where you can discard all non support vectors without any problem. Most of the lazy algorithms – especially KNN – makes decision based on the entire training data set (in the best case a subset of them).

The dichotomy is pretty obvious here – There is a non existent or minimal training phase but a costly testing phase. The cost is in terms of both time and memory. More time might be needed as in the worst case, all data points might take point in decision. More memory is needed as we need to store all training data.

KNN Algorithm is based on **feature similarity**: How closely out-of-sample features resemble our training set determines how we classify a given data point:



Example ofk-NN classification. The test sample (inside circle) should be classified either to the first class of blue squares or to the second class of red triangles. Ifk = 3(outside circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If, for examplek = 5it is assigned to the first class (3 squares vs. 2 triangles outside the outercircle).

KNN can be used for **classification**—the output is a class membership (predicts a class—a discrete value). An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. It can also be used for **regression**—output is the value for the object (predicts continuous values). This value is the average (or median) of the values of its k nearest neighbors.

Assumptions in KNN

Before using KNN, let us revisit some of the assumptions in KNN.

KNN assumes that the data is in a *feature space*. More exactly, the data points are in a metric space. The data can be scalars or possibly even multidimensional vectors. Since the points are in feature space, they have a notion of distance – This need not necessarily be Euclidean distance although it is the one commonly used.

Each of the training data consists of a set of vectors and class label associated with each vector. In the simplest case , it will be either + or – (for positive or negative classes). But KNN , can work equally well with arbitrary number of classes.

We are also given a single number "k" . This number decides how many neighbors (where neighbors is defined based on the distance metric) influence the classification. This is usually a odd number if the number of classes is 2. If k=1 , then the algorithm is simply called the nearest neighbor algorithm.

KNN for Classification

Lets see how to use KNN for classification. In this case, we are given some data points for training and also a new unlabelled data for testing. Our aim is to find the class label for the new point. The algorithm has different behavior based on k.

Case 1 : k = 1 or Nearest Neighbor Rule

This is the simplest scenario. Let x be the point to be labeled . Find the point closest to x . Let it be y. Now nearest neighbor rule asks to assign the label of y to x. This seems too simplistic and some times even counter intuitive. If you feel that this procedure will result a huge error , you are right – but there is a catch. This reasoning holds only when the number of data points is not very large.

If the number of data points is very large, then there is a very high chance that label of x and y are same. An example might help – Lets say you have a (potentially) biased coin. You toss it for 1 million time and you have got head 900,000 times. Then most likely your next call will be head. We can use a similar argument here.

Let me try an informal argument here - Assume all points are in a D dimensional plane . The number of points is reasonably large. This means that the density of the plane at any point is fairly high. In other words , within any subspace there is adequate number of points. Consider a point x in the subspace which also has a lot of neighbors. Now let y be the nearest neighbor. If x and y are sufficiently close, then we can assume that probability that x and y belong to same class is fairly same – Then by decision theory, x and y have the same class.

The book "Pattern Classification" by Duda and Hart has an excellent discussion about this Nearest Neighbor rule. One of their striking results is to obtain a fairly tight error bound to the Nearest Neighbor rule. The bound is

$$P^* \leq P \leq P^*(2 - \frac{c}{c-1}P^*)$$

Where P^* is the Bayes error rate, c is the number of classes and P is the error rate of Nearest Neighbor. The result is indeed very striking (atleast to me) because it says that if the number of

points is fairly large then the error rate of Nearest Neighbor is less that twice the Bayes error rate. Pretty cool for a simple algorithm like KNN.

Case 2 : k = K or k-Nearest Neighbor Rule

This is a straightforward extension of 1NN. Basically what we do is that we try to find the k nearest neighbor and do a majority voting. Typically k is odd when the number of classes is 2. Lets say k = 5 and there are 3 instances of C1 and 2 instances of C2. In this case , KNN says that new point has to labeled as C1 as it forms the majority. We follow a similar argument when there are multiple classes.

One of the straight forward extension is not to give 1 vote to all the neighbors. A very common thing to do is *weighted kNN* where each point has a weight which is typically calculated using its distance. For eg under inverse distance weighting, each point has a weight equal to the inverse of its distance to the point to be classified. This means that neighboring points have a higher vote than the farther points.

It is quite obvious that the accuracy *might* increase when you increase k but the computation cost also increases.

Some Basic Observations

1. If we assume that the points are d-dimensional, then the straight forward implementation of finding k Nearest Neighbor takes O(dn) time.
2. We can think of KNN in two ways – One way is that KNN tries to estimate the posterior probability of the point to be labeled (and apply bayesian decision theory based on the posterior probability). An alternate way is that KNN calculates the decision surface (either implicitly or explicitly) and then uses it to decide on the class of the new points.
3. There are many possible ways to apply weights for KNN – One popular example is the Shephard's method.
4. Even though the naive method takes O(dn) time, it is very hard to do better unless we make some other assumptions. There are some efficient data structures like **KD-Tree** which can reduce the time complexity but they do it at the cost of increased training time and complexity.
5. In KNN, k is usually chosen as an odd number if the number of classes is 2.
6. Choice of k is very critical – A small value of k means that noise will have a higher influence on

the result. A large value make it computationally expensive and kinda defeats the basic philosophy behind KNN (that points that are near might have similar densities or classes) .A simple approach to select k is set $k = \sqrt{n}$

7. There are some interesting data structures and algorithms when you apply KNN on graphs – See **Euclidean minimum spanning tree** and **Nearest neighbor graph** .

8. There are also some nice techniques like condensing, search tree and partial distance that try to reduce the time taken to find the k nearest neighbor.

Applications of KNN

KNN is a versatile algorithm and is used in a huge number of fields. Let us take a look at few uncommon and non trivial applications.

1. Nearest Neighbor based Content Retrieval

This is one the fascinating applications of KNN – Basically we can use it in Computer Vision for many cases – You can consider handwriting detection as a rudimentary nearest neighbor problem. The problem becomes more fascinating if the content is a video – given a video find the video closest to the query from the database – Although this looks abstract, it has lot of practical applications – Eg : Consider **ASL** (American Sign Language) . Here the communication is done using hand gestures.

So lets say if we want to prepare a dictionary for ASL so that user can query it doing a gesture. Now the problem reduces to find the (possibly k) closest gesture(s) stored in the database and show to user. In its heart it is nothing but a KNN problem. One of the professors from my dept , Vassilis Athitsos , does research in this interesting topic – See **Nearest Neighbor Retrieval and Classification** for more details.

2. Gene Expression

This is another cool area where many a time, KNN performs better than other state of the art techniques . In fact a combination of KNN-SVM is one of the most popular techniques there. This is a huge topic on its own and hence I will refrain from talking much more about it.

3. Protein-Protein interaction and 3D structure prediction

Graph based KNN is used in protein interaction prediction. Similarly KNN is used in structure prediction.

4. Credit ratings—collecting financial characteristics vs. comparing people with similar financial

features to a database. By the very nature of a credit rating, people who have similar financial details would be given similar credit ratings. Therefore, they would like to be able to use this existing database to predict a new customer's credit rating, without having to perform all the calculations.

Should the bank give a loan to an individual? Would an individual default on his or her loan? Is that person closer in characteristics to people who defaulted or did not default on their loans?

In political science—classing a potential voter to a "will vote" or "will not vote", or to "vote Democrat" or "vote Republican".

5. More advance examples could include handwriting detection (like OCR), image recognition and even video recognition.

Pros:

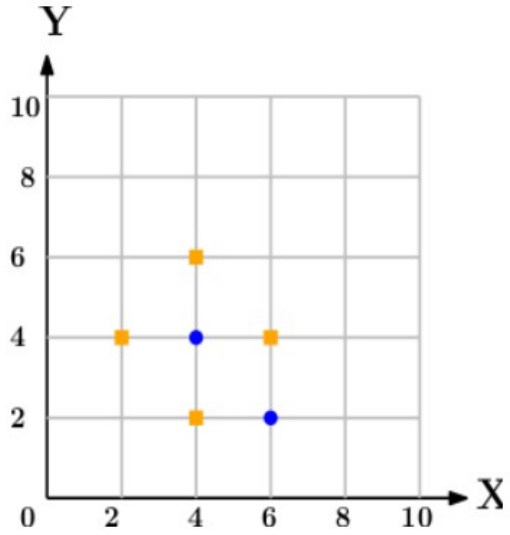
- No assumptions about data—useful, for example, for nonlinear data
- Simple algorithm—to explain and understand/interpret
- High accuracy (relatively)—it is pretty high but not competitive in comparison to better supervised learning models
- Versatile—useful for classification or regression

Cons:

- Computationally expensive—because the algorithm stores all of the training data
- High memory requirement
- Stores all (or almost all) of the training data
- Prediction stage might be slow (with big N)
- Sensitive to irrelevant features and the scale of the data

Given Diagram Represent Positive and Negative Point with Color

In the following diagram let blue circles indicate positive examples and orange squares indicate negative examples. We want to use k-NN algorithm for classifying the points. If k=3, find-the class of the point (6,6). Extend the same example for Distance-Weighted k-NN and Locally weighted Averaging



Algorithm

Import the Required Packages
Read Given Dataset
Import KNeighborshood Classifier and create object of it.
Predict the class for the point(6,6) w.r.t to General KNN.
Predict the class for the point(6,6) w.r.t to Distance Weighted KNN.

3.11 Conclusion

In this way we learn KNN Classification to predict the General and Distance Weighted KNN for Given data point in term of Positive or Negative.

3.12 Assignment Questions

12. How KNN Different from Kmean Clustering Algorithm?
13. What is the Formula for Euclidean Distance?
14. What is Hamming Distance?
15. What is formula for Manhatten and Minkowski Distnace?
16. What are the application of KNN?

References:-

6. https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7
7. Mittu Skillologies Youtube Channel
8. https://www.saedsayad.com/k_nearest_neighbors.htm

```
#import the packages
import pandas as pd
import numpy as np

#Read dataset
dataset=pd.read_csv("kdata.csv")
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,2].values

#import KNeighborshood Classifier and create object of it
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=3)
classifier.fit(X,y)

#predict the class for the point(6,6)
X_test=np.array([6,6])
y_pred=classifier.predict([X_test])
print('General KNN',y_pred)

classifier=KNeighborsClassifier(n_neighbors=3,weights='distance')
classifier.fit(X,y)

#predict the class for the point(6,6)
X_test=np.array([6,2])
y_pred=classifier.predict([X_test])
print('Distance Weighted KNN',y_pred)
```

Output

General KNN ['negative']
Distance Weighted KNN ['positive']

ML Assignment No. 4

4.1 Title

Assignment based on k-mean Clustering

4.2 Problem Definition:

We have given a collection of 8 points. P1=[0.1,0.6] P2=[0.15,0.71] P3=[0.08,0.9] P4=[0.16,0.85] P5=[0.2,0.3] P6=[0.25,0.5] P7=[0.24,0.1] P8=[0.3,0.2]. Perform the k-mean clustering with initial centroids as m1=P1 =Cluster#1=C1 and m2=P8=cluster#2=C2. Answer the following
1] Which cluster does P6 belongs to?
2] What is the population of cluster around m2?
3] What is updated value of m1 and m2?

4.3 Prerequisite:

Basic of Python, Data Mining Algorithm, Concept of K-mean Clustering

4.4 Software Requirements:

Anaconda with Python 3.7

4.5 Hardware Requirement:

PIV, 2GB RAM, 500 GB HDD, Lenovo A13-4089Model.

4.6 Learning Objectives:

Learn How to Apply Kmean Clustering for given datapoints

4.7 Outcomes:

After completion of this assignment students are able Implement code for the k-mean clustering with initial centroids

4.8 Theory Concepts:

4.8.1 Motivation

A Hospital Care chain wants to open a series of Emergency-Care wards within a region. We assume that the hospital knows the location of all the maximum accident-prone areas in the region. They have to decide the number of the Emergency Units to be opened and the location of these Emergency Units, so that all the accident-prone areas are covered in the vicinity of these Emergency Units.

The challenge is to decide the location of these Emergency Units so that the whole region is covered. Here is when K-means Clustering comes to rescue!

A cluster refers to a small group of objects. Clustering is grouping those objects into clusters. In order to learn clustering, it is important to understand the scenarios that lead to cluster different objects. Let us identify a few of them.

What is Clustering?

Clustering is dividing data points into homogeneous classes or clusters:

Points in the same group are as similar as possible

Points in different group are as dissimilar as possible

When a collection of objects is given, we put objects into group based on similarity.

Application of Clustering:

Clustering is used in almost all the fields. You can infer some ideas from Example 1 to come up with lot of clustering applications that you would have come across.

Listed here are few more applications, which would add to what you have learnt.

- ✓ Clustering helps marketers improve their customer base and work on the target areas. It helps group people (according to different criteria's such as willingness, purchasing power etc.) based on their similarity in many ways related to the product under consideration.
- ✓ Clustering helps in identification of groups of houses on the basis of their value, type and geographical locations.
- ✓ Clustering is used to study earth-quake. Based on the areas hit by an earthquake in a region, clustering can help analyse the next probable location where earthquake can occur.

Clustering Algorithms:

A Clustering Algorithm tries to analyse natural groups of data on the basis of some similarity. It locates the centroid of the group of data points. To carry out effective clustering, the algorithm evaluates the distance between each point from the centroid of the cluster.

The goal of clustering is to determine the intrinsic grouping in a set of unlabelled data.

What is K-means Clustering?

K-means (Macqueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining.

K-means Clustering – Example 1:

A pizza chain wants to open its delivery centres across a city. What do you think would be the possible challenges?

- They need to analyse the areas from where the pizza is being ordered frequently.
- They need to understand as to how many pizza stores has to be opened to cover delivery in the area.
- They need to figure out the locations for the pizza stores within all these areas in order to keep the distance between the store and delivery points minimum.

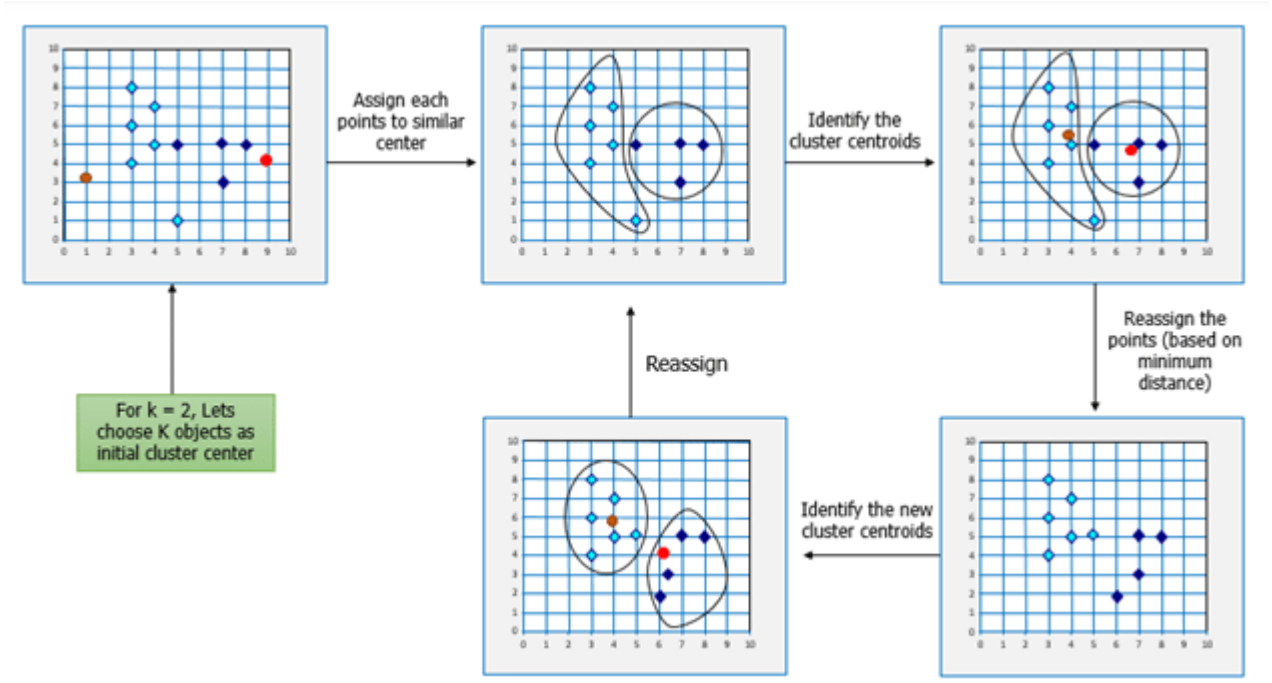
Resolving these challenges includes a lot of analysis and mathematics. We would now learn about how clustering can provide a meaningful and easy method of sorting out such real life challenges. Before that let's see what clustering is.

K-means Clustering Method:

If k is given, the K-means algorithm can be executed in the following steps:

- Partition of objects into k non-empty subsets
- Identifying the cluster centroids (mean point) of the current partition.
- Assigning each point to a specific cluster
- Compute the distances from each point and allot points to the cluster where the distance from the centroid is minimum.
- After re-allotting the points, find the centroid of the new cluster formed.

The step by step process:



Now, let's consider the problem in Example 1 and see how we can help the pizza chain to come up with centres based on K-means algorithm.

Similarly, for opening Hospital Care Wards:

K-means Clustering will group these locations of maximum prone areas into clusters and define a cluster center for each cluster, which will be the locations where the Emergency Units will open. These Clusters centers are the centroids of each cluster and are at a minimum distance from all the points of a particular cluster, henceforth, the Emergency Units will be at minimum distance from all the accident prone areas within a cluster.

Here is another example for you, try and come up with the solution based on your understanding of K-means clustering.

K-means Clustering – Example 2:

Let's consider the data on drug-related crimes in Canada. The data consists of crimes due to various drugs that include, Heroin, Cocaine to prescription drugs, especially by underage people. The crimes resulted due to these substance abuse can be brought down by starting de-addiction centres in areas most afflicted by this kind of crime. With the available data, different objectives can be set. They are:

- Classify the crimes based on the abuse substance to detect prominent cause.
- Classify the crimes based on age groups.
- Analyze the data to determine what kinds of de-addiction centre is required.
- Find out how many de-addiction centres need to be setup to reduce drug related crime rate.

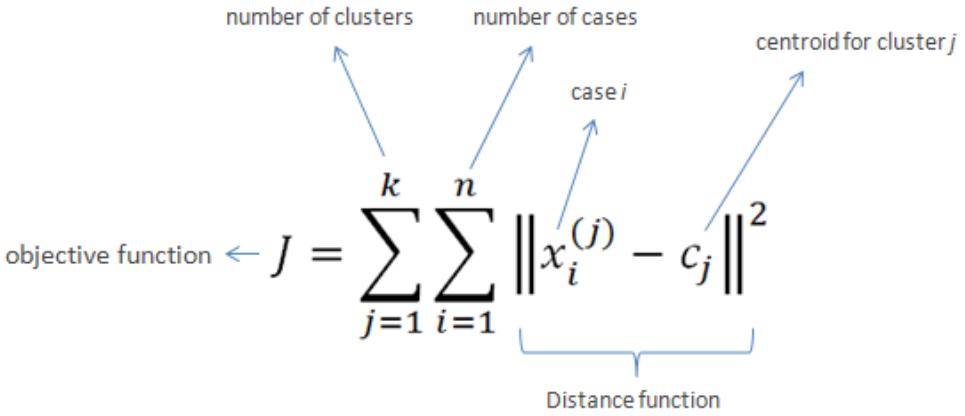
The K-means algorithm can be used to determine any of the above scenarios by analyzing the

available data.

Following the K-means Clustering method used in the previous example, we can start off with a given k, following by the execution of the K-means algorithm.

Mathematical Formulation for K-means Algorithm:

K-Means clustering intends to partition *n* objects into *k* clusters in which each object belongs to the cluster with th mean. This method produces exactly *k* different clusters of greatest possible distinction. The best number of clusters *k* leading to the greatest separation (distance) is not known as a priori and must be computed from the data objective of K-Means clustering is to minimize total intra-cluster variance, or, the squared error function:



4.9 Algorithm

Import the Required Packages
Create dataset using DataFrame
Find centroid points
plot the given points
for i in centroids():
plot given elements with centroid elements
import KMeans class and create object of it
using labels find population around centroid
Find new centroids

4.10 Conclusion

In this way we learn Kmean Clustering Algorithm.

4.11 Assignment Questions

Why does k-means clustering algorithm use only Euclidean distance metric?
How to decide on the correct number of clusters?

1. What are the Different Application of K Mean Clustering?
2. What is K-medoids?
3. What is *k*-medians clustering?

References:-

9. https://www.saedsayad.com/k_nearest_neighbors.htm
10. https://www.edureka.co/blog/k-means-clustering/

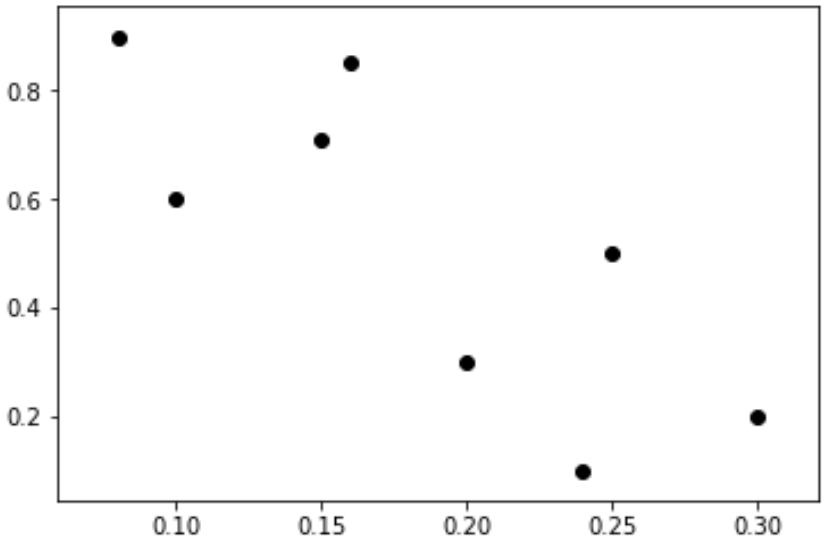

```
#import packages
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#create dataset using DataFrame
df=pd.DataFrame({'X':[0.1,0.15,0.08,0.16,0.2,0.25,0.24,0.3],
                 'y':[0.6,0.71,0.9,0.85,0.3,0.5,0.1,0.2]})
f1 = df['X'].values
f2 = df['y'].values
X = np.array(list(zip(f1, f2)))
print(X)

#centroid points
C_x=np.array([0.1,0.3])
C_y=np.array([0.6,0.2])
centroids=C_x,C_y
```

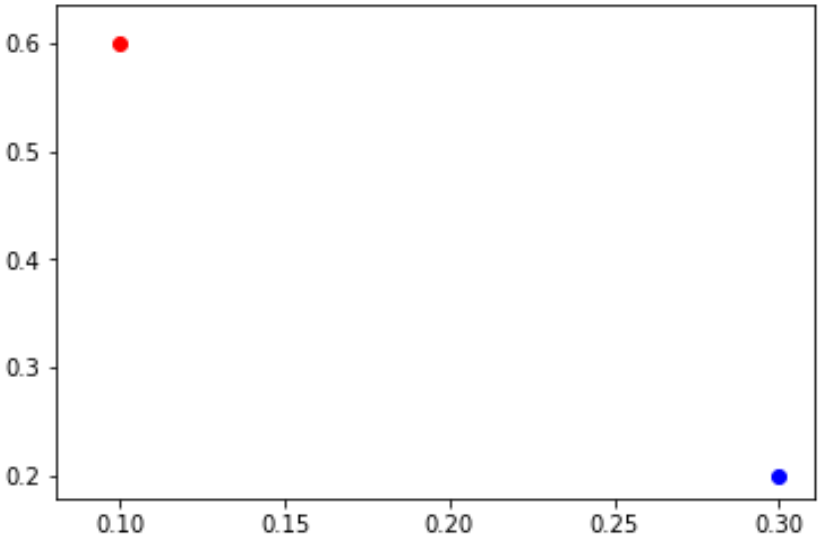
```
#plot the given points
colmap = {1: 'r', 2: 'b'}
plt.scatter(f1, f2, color='k')
plt.show()

#for i in centroids():
plt.scatter(C_x[0],C_y[0], color=colmap[1])
plt.scatter(C_x[1],C_y[1], color=colmap[2])
plt.show()

C = np.array(list((C_x, C_y)), dtype=np.float32)
print (C)

#plot given elements with centroid elements
plt.scatter(f1, f2, c='#050505')
plt.scatter(C_x[0], C_y[0], marker='*', s=200, c='r')
plt.scatter(C_x[1], C_y[1], marker='*', s=200, c='b')
plt.show()


#import KMeans class and create object of it
from sklearn.cluster import KMeans
model=KMeans(n_clusters=2,random_state=0)
model.fit(X)
labels=model.labels_
print(labels)

#using labels find population around centroid
count=0
for i in range(len(labels)):
    if (labels[i]==1):
        count=count+1

print('No of population around cluster 2:',count-1)

#Find new centroids
new_centroids = model.cluster_centers_

print('Previous value of m1 and m2 is:')
```

```
print('M1==',centroids[0])
print('M1==',centroids[1])

print('updated value of m1 and m2 is:')
print('M1==',new_centroids[0])
print('M1==',new_centroids[1])
```

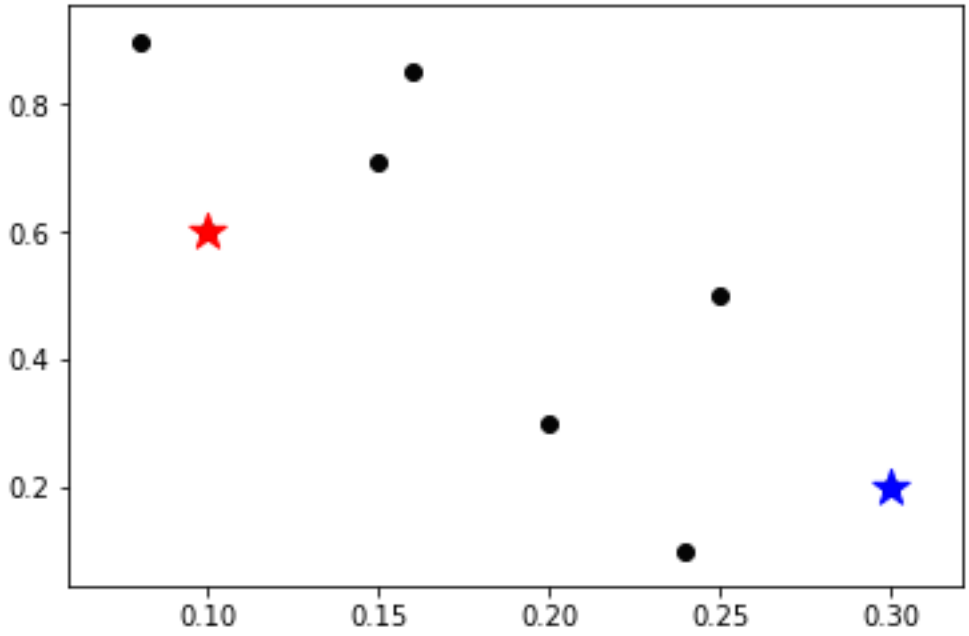
Output

```
[[0.1  0.6 ]
 [0.15 0.71]
 [0.08 0.9 ]
 [0.16 0.85]
 [0.2  0.3 ]
 [0.25 0.5 ]
 [0.24 0.1 ]
 [0.3  0.2 ]]
```





[[0.1 0.3]
[0.6 0.2]]



[1 1 1 1 0 0 0 0]
No of population around cluster 2: 3
Previous value of m1 and m2 is:
M1== [0.1 0.3]
M1== [0.6 0.2]
updated value of m1 and m2 is:

M1== [0.2475 0.275]
M1== [0.1225 0.765]

5.Mini-Project 1 on Genetic Algorithm:

Apply the Genetic Algorithm for optimization on a dataset obtained from UCI ML repository. For Example: IRIS Dataset or Travelling Salesman Problem or KDD Dataset

5.1 IRIS Dataset 5.2 Travelling Salesman Problem 5.3 KDD Dataset.

OR

Mini-Project 2 on SVM:

Apply the Support vector machine for classification on a dataset obtained from UCI ML repository. For Example: Fruits Classification or Soil Classification or Leaf Disease Classification.

Fruits Classification Soil Classification Leaf Disease Classification

OR

Mini-Project 3 on PCA:

Apply the Principal Component Analysis for feature reduction on any Company Stock Market Dataset.

II] Information and Cyber Security

Assignment No 1.

Title : Implementation of S-DES

Theory:

DES is the archetypal block cipher—an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another ciphertext bitstring of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key ostensibly consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits.

The key is nominally stored or transmitted as 8 bytes, each with odd parity. According to ANSI X3.92-1981 (Now, known as ANSI INCITS 92-1981), section 3.5:

One bit in each 8-bit byte of the *KEY* may be utilized for error detection in key generation, distribution, and storage. Bits 8, 16,..., 64 are for use in ensuring that each byte is of odd parity.

Like other block ciphers, DES by itself is not a secure means of encryption, but must instead be used in a mode of operation. FIPS-81 specifies several modes for use with DES.[24] Further comments on the usage of DES are contained in FIPS-74.[25]

Decryption uses the same structure as encryption, but with the keys used in reverse order.

S-DES or Simplified Data Encryption Standard

The process of encrypting a plan text into an encrypted message with the use of S-DES has been

divided into multi-steps which may help you to understand it as easily as possible.
Points should be remembered.

1. It is a block cipher.
2. It has 8-bits block size of plain text or cipher text.
3. It uses 10-bits key size for encryption.
4. It is a symmetric cipher.
5. It has Two Rounds.

Let's start the game!

Key Generation of S-DES or How to Generate the Key of Simplified DES

First and foremost, we need to generate a key. With the help of this key we will encrypt the message.

Now the interesting question is, how to generate the key, and where the key is to be used. Just follow the steps.

Step 1:

Just select a random key of 10-bits, which only should be shared between both parties which means sender and receiver.

As I selected below!

Select key:1010000010

Note:You can select any random number of 10-bits.

Step 2:

Put this key into P.10 Table and permute the bits.

P.10 Table:

Input	1	2	3	4	5	6	7	8	9	10
Output Should be	3	5	2	7	4	10	1	9	8	6

As I put key into P.10 Table.

Input	1	0	1	0	0	0	0	0	1	0
Output	1	0	0	0	0	0	1	1	0	0

Now the output will be:

Key: 1000001100

Step 3:

Divide the key into two halves, left half and right half;

{1 0 0 0 0} | {0 1 1 0 0}

Step 4:

Now apply the one bit Round shift on each half:

Before round shift: {10000} | {01100}

After round shift: {00001} | {11000}

The output will be:

{0 0 0 0 1} {1 1 0 0 0}

Step 5:

Now once again combine both halve of the bits, right and left. Put them into the P8 table. What you get, that will be the K1 or First key.

Combine: 0 0 0 0 1 1 1 0 0 0

Permute into 8bit table:

P8-Table

Input	1	2	3	4	5	6	7	8	9	10
Combine-bits	0	0	0	0	1	1	1	0	0	0
Output Should be	6	3	7	4	8	5	10	9		
Output bits	1	0	1	0	0	1	0	0		

See the table the 1 and 2 number of bits are removed and other are permuted, as 6 in place of one, 9 in place of 8 and so on.

The output and K1 or key One will be:

K1=1 0 1 0 0 1 0 0

Step6:

As we know S-DES has two round and for that **we also need two keys**, one key we generate in the above steps (step 1 to step 5). Now we need to **generate a second** bit and after that we will move to encrypt the plain text or message.

It is simple to generate the second key. Simply, go in **step 4** copy both halves, each one consists of 5 bits. But be careful on the taking of bits. Select those halves which are output of first round shift, don't take the bits which are not used in the first round. In simple words, take the output of first round shift in above **step 4.**

Which are: {00001} | {11000}

Step 7:

Now just apply two round shift circulate on each half of the bits, which means to change the position of two bits of each halves.

left half: 00001

Right half: 11000

After the two rounds shift on each half out-put of each half will be.

Left half: 00100

Right half: 00011

Combine both together: As: 0 0 1 0 0 – 0 0 0 1 1

Step 8:

Now put the bits into 8-P Table, what you get, that will be your second key. Table is also given in **step 5**.

But here the combinations of bits are changed because of two left round shift from step 5. Check it in depth.

Combine bits: 0 0 1 0 0 0 0 0 1 1

P.8 Table

Input	1	2	3	4	5	6	7	8	9	10
Combine-bits	0	0	1	0	0	0	0	0	1	1
Output Should be	6	3	7	4	8	5	10	9		
Output bits	0	1	0	0	0	0	1	1		

The output of the bits are your Second key or K2:
K2: 0 1 0 0 0 0 1 1

Finally we create both keys successfully:

K1: 1 0 1 0 0 1 0 0 (see in step 5)
K2: 0 1 0 0 0 0 1 1 (see in step 8)

How To Encrypt the Plain Text into Cipher Text in S-DES After Generating Keys.

Now, let's start Encryption of plain text into cipher text.

Encryption of Plain text into Cipher text in S-DES:

Come on do it, **step by step**.

Note: the size of input text is 8 bit and output also will be 8-bit. Or the block size is 8-bit/one byte always.

Step 1:

Suppose this is our plain text in binary which is 8-bit.

Plain text: 01110010

Step 2:

Put the plain text into IP-8(initial permutation) table and permute the bits.

IP-8 table

Bits number	1	2	3	4	5	6	7	8
Bits to be permuted	0	1	1	1	0	0	1	0
Permute the bits	2	6	3	1	4	8	5	7
Permuted bits	1	0	1	0	1	0	0	1

Output is: 1 0 1 0 1 0 0 1

Step 3:

Now break the bits into two halves, each half will consist of 4 bits. The halves will be right and

left.
Two Halves of the bits:

Left half {1 0 1 0} -right half {1 0 0 1}

Step 4:

Take the right 4 bits and put them into E.P (expand and per-mutate) Table.
Bits of right half: 1001

E.P Table

Right half bits	1	0	0	0				
Number	1	2	3	4	5	6	7	8
Expand bits	4	1	2	3	2	3	4	1
Output of bits	0	1	0	0	0	0	0	1

Output of right four bits will be 8 bits, after their expanding with the help of E.P table.

O-P: 0 1 0 0 0 0 0 1

Step 5: Now, just take the output and XOR it with First key Or K 1 (which we created in previous topic that is how to generate key.).

XOR (⊕) them:

O-p: 0 1 0 0 0 0 0 1
⊕
K1: 1 0 1 0 0 1 0 0
Output of XOR: **1 1 1 0 0101**

Step 6:

Once again split the output of XOR's bit into two halves and each half will consist of 4 bits.

Splitting them into two halves:

Left half :{ 1 1 1 0} right half :{ 0101}
Now put the each half into the **s-boxes**, there is only two s-boxes. **S-0 and S-1.**

S-0

Col	0	1	2	3
Rows				
0	01	00	11	10
1	11	10	01	00

2	00	10	01	11
3	11	01	11	10

S-1

Col	0	1	2	3
Rows				
0	00	01	10	11
1	10	00	01	11
2	11	00	01	00
3	10	01	00	11

Note: put the left half into S-0 box and put the right half into S-1 Box.
But how to put them in into S-Boxes?

Take any half, (but don't forget the above mentioned note..).
The most first and most last bit will be consider the row and other remaining, which are, 2 and 3, will be considered the columns.
See, here I'm taking the left half: which is 1 1 1 0.

Now I will take First and last bit which are: 1 and 0. These will be row.

And I also will take 2nd and 3rd bits which are: 11. These will be column number.
10 means=2rd **row**
11 means = 3rd **col**

See below how it will be the second row, and 3rd column. Please, remember the IP Addressing, such as 2^8 for 255.
$1^2\,0^1 = 2+0=2$
$1^2\,1^1 = 2+1=3$

Let's check the 2nd row and 3rd column.

For **left half** we check the in S-**0** . In which 2nd row and 3rd column.

The output is 00 for left half;

Now let's take the **right half** and find the output of the right of in **S-1 box**.
Right half: 0101
Row: $0\ 1 = 0^2\,1^1 = 0+1=1$
Col: $1\ 0 = 1^2\,0^1 = 2+1=3$

Which means the value will be in 1st row and 3rd column, let's in **check S-1.**
The output will be: 11 for left half.

Step 7:

Now combine these two halves together.

Left half: {00} and right half: {11}
It will be: 0 0 1 1

Step 8: Now take these 4 bits and put them in **P-4** (permutation 4) table and get the result.

P 4 Table

Numbers	1	2	3	4
Input	0	0	1	1
Output should be	2	4	3	1
Out-Put	0	1	1	0

Now the output is: 0 1 1 0

Step 9:

Now get XOR the output with left 4 bits of Initial Per-mutation. The left bits of initial per-mutation are in **step 3**, which are **1 0 1 0**.(please, in step 3).
Let them to be XOR.

0 1 1 0
⊕
1 0 1 0

Out-put will be: 1 1 0 0

Step 10:

Now get the right half of the initial permutation, which is **step 3,** and combine that with this out-put.

The out-put of XOR in **step 9:** 1 1 0 0
Right half of IP (initial permutation): 1 0 0 1
Let's combine both. 1 1 0 0 – 1 0 0 1= 1 1 0 0 1 0 0 1
Now the output is 8 bits.: **1 1 0 0 1 0 0 1**

Step 11: Now once again break the out-put into two halves, left and right;

Left: {1 1 0 0} right: {1 0 0 1}

Step 12:

Now swap both halves, which means put the left half in place of right and vice versa.

Result:
Left half: {1 0 0 1} right half: {1 1 0 0}

Step 13:

Now let's take these halves and once again start the same procedure from **step 2** or initial Permutation, **BUT** be careful on using key in this stage we use second key or K2 (not K1). And put that into IP^{-1} (IP inverse) Table. What you get will be your final cipher text.

Let me to do it in brief. You should check carefully what I did.
Ø 1 0 0 1 11 0 0

After initial permutation according to **IP-8 table (see step 2)**
Out-put will be: 0 1 0 1 1 0 1 0

Ø Now break it into two halves
Left {0 1 0 1} right {1 0 1 0}

Ø Now Take the right 4bits and put them into **EP table**, and get the result of 8 bits.
It will be: 0 1 0 1 0 1 0 1

Ø Let **XOR it with K2.**
K2: 0 1 0 0 0 0 1 1
⊕

Out-put of **EP**: 0 1 0 1 0 1 0 1
Out- Put: **0 0 0 1 0 1 1 0**

Ø Once again split the output of XOR's bit into two halves:
Left: {0 0 0 1} right: {0 1 1 0}

Ø Now put each half in S-Boxes, which are S-0 and S-1:

Note: put the left half into S-0 box and put the right half into S-1 Box.

Before that get Rows and column:

Left: 0 0 0 1
Row: 0 1= 1
Col: 0 0 = 0

Let find the row and column of left, in S-0, the value is row number one and col number Zero.

It will be. 1 1

Ø now check the right half: 0 1 1 0
Row: 0 0=0
Col: 1 1= 3

Let's find the row and column of right, in S-1, the value is row number zero and col number three.

It will be: 11
Ø Now combine both halves together.
It will be: 1 1 1 1

Ø Now take these 4 bits and put them in **P-4** (**Per-mutation** 4) table and get the result.

The out-put also will be: 1111 after permutation.

Ø Now get it XOR with left 4 bits of Initial Per-Mutation.
1 1 1 1
XOR
0 1 0 1

Out-put:1 0 1 0

Ø Now get the right half of the initial permutation and combine that with this out- put.
1 0 1 0 - 0 1 1 0

Ø Now once again break the it into two halves, left and right
Left: {1 0 1 0} right: {0 1 1 0}

Ø Now **swap both halves,** which means put the left half in place of right and vice versa.
0 1 1 0 1 0 1 0

Ø Now put it into IP^{-1} Table which is :

IP^{-1} Table

Numbers	1	2	3	4	5	6	7	8
input	0	1	1	0	1	0	1	0
Out-put to be	2	6	3	1	4	8	5	7
Out-Put	1	0	1	0	0	0	1	1

Out-Put is the cipher text. Which is: 1 0 1 0 0 0 1 1

Finally we Encrypted successfully our **plain text: 01110010** into **cipher text** which is:

1 0 1 0 0 0 1 1

S-DES PROGRAM:

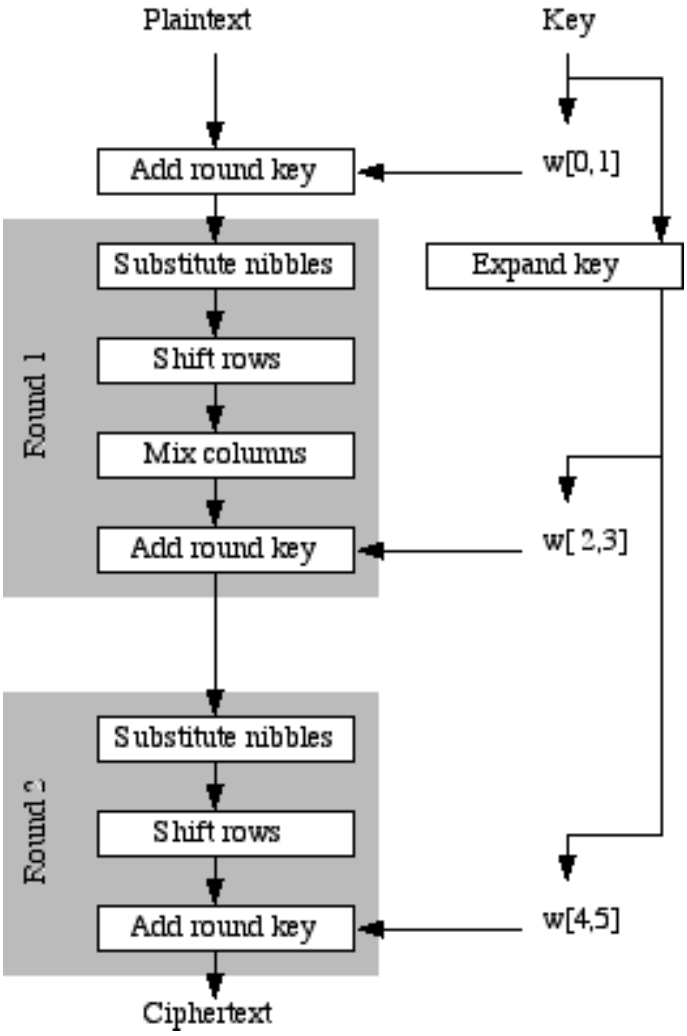
```
# Permutation boxes and S-boxes
InputPerm = [2, 6, 3, 1, 4, 8, 5, 7]
FinalPerm = [4, 1, 3, 5, 7, 2, 8, 6]

EPtable = [4, 1, 2, 3, 2, 3, 4, 1]
S0 = [[1, 0, 3, 2], [3, 2, 1, 0], [0, 2, 1, 3], [3, 1, 3, 2]]
S1 = [[0, 1, 2, 3], [2, 0, 1, 3], [3, 0, 1, 0], [2, 1, 0, 3]]
P4table = [2, 4, 3, 1]

P10Table = [3, 5, 2, 7, 4, 10, 1, 9, 8, 6]
P8Table = [6, 3, 7, 4, 8, 5, 10, 9]


def perm(B, table): # applies different permutations on text 'B'
    output = 0
    for i, e in enumerate(table):
        if i >= e:
            output |= (B & (128 >> (e - 1))) >> (i - (e - 1))
        else:
            output |= (B & (128 >> (e - 1))) << ((e - 1) - i)
    return output


def keyGen(key): # generates 2 sub-keys for 2 rounds
    def leftShift(pk):
        ls, rs = pk >> 5, pk & 0b0000011111
        ls = ((ls >> 4) & 1) | (ls << 1) & 0b0000011110
        rs = ((rs >> 4) & 1) | (rs << 1) & 0b0000011110
        return ((ls << 5) & 0b1111100000) | rs

    k = perm(key, P10Table)
    shiftOne = leftShift(k)
    shiftTwo = leftShift(leftShift(shiftOne))
    subkey1 = perm(shiftOne, P8Table)
    subkey2 = perm(shiftTwo, P8Table)
    return subkey1, subkey2


def InitialPermutation(B): # applies IP
    return perm(B, InputPerm)


def swapper(B): # swaps 2 nibbles in a byte
    return (B << 4 | B >> 4) & 0xff
```

```
def mixer(key, B): # applies round function (Feistel function)
    def f(skey, rNib):
        temp = skey ^ perm(swapper(rNib), EPtable)
        l, r = temp & 0xf0, temp & 0x0f
        lr, lc, rr, rc = ((l >> 2) & 0x2) | (l & 0x1), (l >> 1) & 0x3, ((r
>> 2) & 0x2) | (r & 0x1), (r >> 1) & 0x3
        sboxout = swapper((S0[lr][lc] << 2) + S1[rr][rc])
        return perm(sboxout, P4table)

    lNib, rNib = B & 0xf0, B & 0x0f
    return lNib ^ f(key, rNib) | rNib


def FinalPermutation(B): # applies FP (inverse of IP)
    return perm(B, FinalPerm)


def encrypt(text, key): # applies encryption algorithm
    t = mixer(keyGen(key)[0], InitialPermutation(text))
    return FinalPermutation(mixer(keyGen(key)[1], swapper(t)))


def decrypt(cipher, key): # applies decryption algorithm
    t = mixer(keyGen(key)[1], InitialPermutation(cipher))
    return FinalPermutation(mixer(keyGen(key)[0], swapper(t)))


if __name__ == "__main__":
    plaintext = input("Enter plaintext: ") # any length character input
    key = int(input("Enter Key: ")) # binary input of length 10
    encipher = ""
    decipher = ""
    for i in plaintext:
        encipher += chr(encrypt(ord(i), key))
    print("Cipher Text: " + encipher)
    for i in encipher:
        decipher += chr(decrypt(ord(i), key))
    print("Decipher Text: " + decipher)
```

Output:

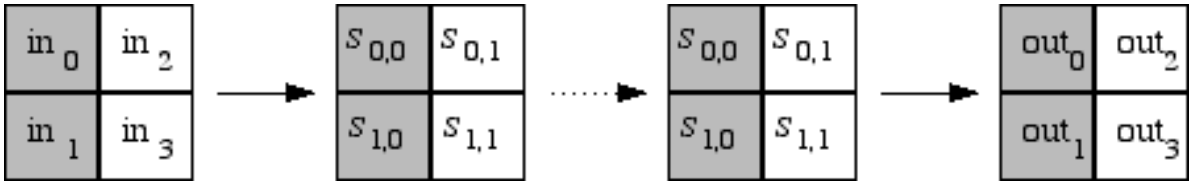
Assignment 2. Implementation of S-AES Algorithm

Overview S-AES is to AES as S-DES is to DES. In fact, the structure of S-AES is exactly the same as AES. The differences are in the key size (16 bits), the block size (16 bits) and the number of rounds (2 rounds). Here is an overview:



S-AES Encryption Overview

Substitute nibbles Instead of dividing the block into a four by four array of bytes, S-AES divides it into a two by two array of “nibbles”, which are four bits long. This is called the state array and is shown below.



S-AES state array

In the first stage of each encryption round, an S-box is used to translate each nibble into a new nibble. First we associate the nibble $b_0b_1b_2b_3$ with the polynomial $b_0x^3 + b_1x^2 + b_2x + b_3$. This polynomial is then inverted as an element of GF(16), with the “prime polynomial” used being $x^4 + x + 1$. Then we multiply by a matrix and add a vector as in AES.

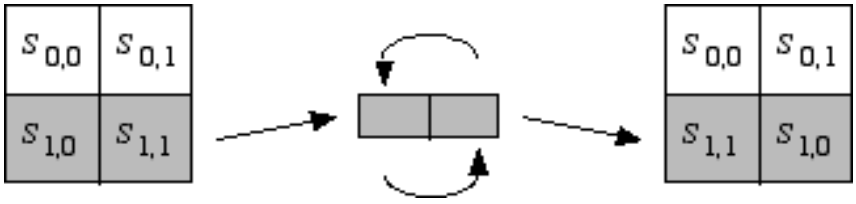
$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Remember that the addition and multiplication in the equation above are being done modulo 2 (with XOR), but not in GF(16).

Since a computer would do the S-box substitution using a table lookup, we give the full table for the S-box here.

nibble	S-box(nibble)	nibble	S-box(nibble)
0000	1001	1000	0110
0001	0100	1001	0010
0010	1010	1010	0000
0011	1011	1011	0011
0100	1101	1100	1100
0101	0001	1101	1110
0110	1000	1110	1111
0111	0101	1111	0111

Shift Rows The next stage is to shift the rows. In fact, the first row is left alone and the second row is shifted.



S-AES shift row transformation

Mix Columns After shifting the rows, we mix the columns. Each column is multiplied by the matrix

$$\begin{matrix} 1 & 4 \\ 4 & 1 \end{matrix}$$

.

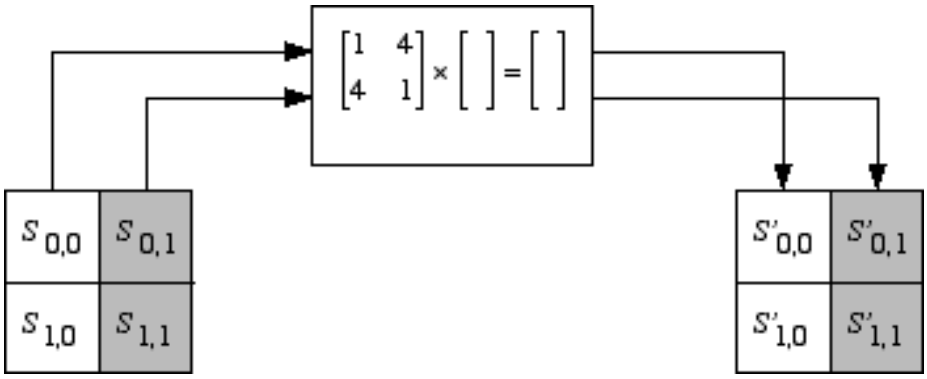
These operations are done in GF(16), so remember that the 1 corresponds to the polynomial 1 and the 4 corresponds to the polynomial x^2. Thus this matrix could also be written as

$$\begin{matrix} 1 & x^2 \end{matrix}$$

x^2 1 .

Don’t forget to reduce modulo $x^4 + x$ + 1.

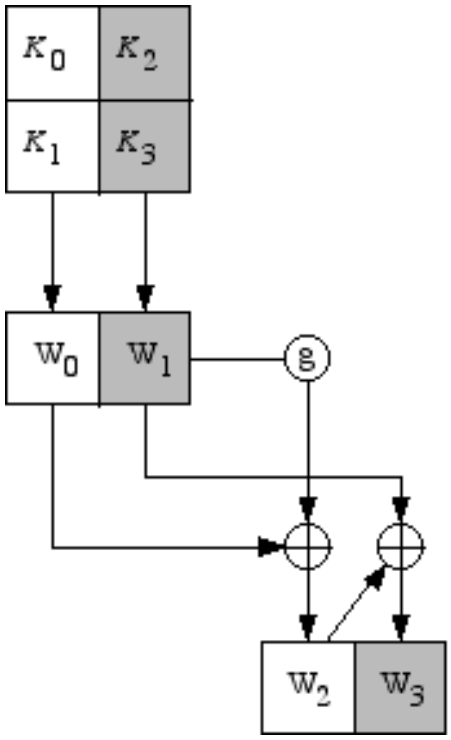
The mix column transformation is omitted in the last round, in order to simplify the decryption.



S-AES mix column transformation

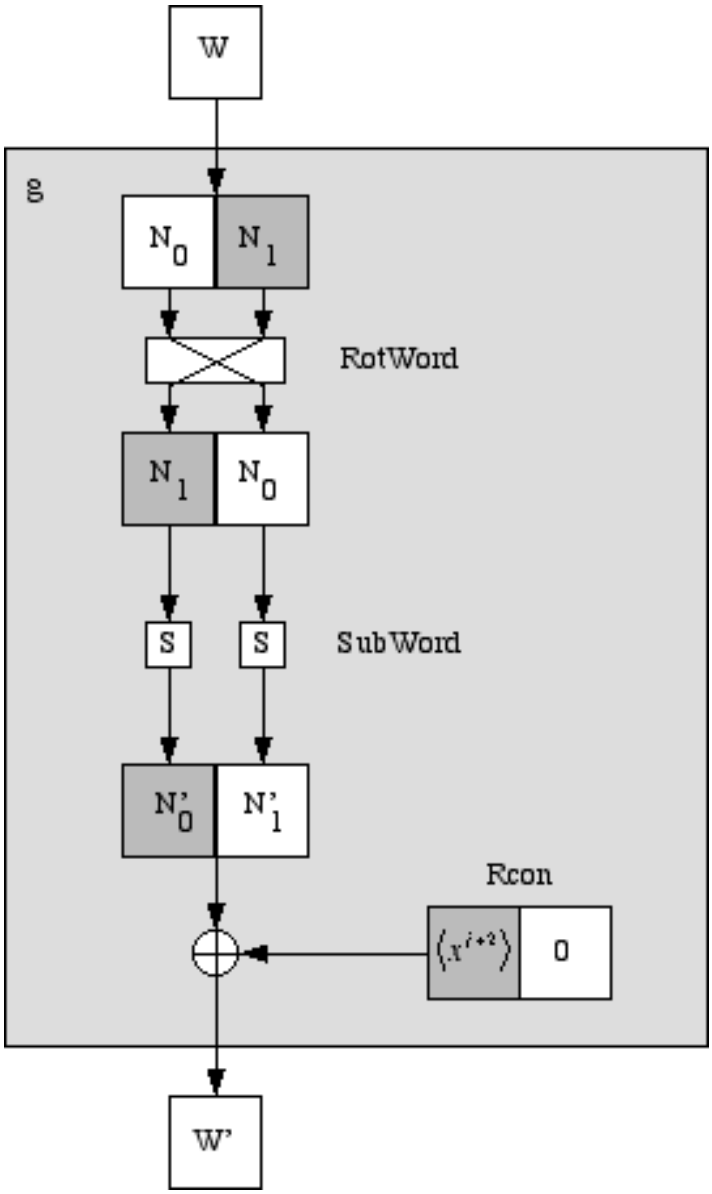
Add Round Key The last stage of each round of encryption is to add the round key. (In fact, this is also done before the first round.) Before the first round, the first two words (W_0 and W_1) of the expanded key are added. In the first round, W_2 and W_3 are added. In the last round, W_4 and W_5 are added. All additions are done modulo 2, that is, with XOR.

Key Expansion Key expansion is done very similarly to AES. The four nibbles in the key are grouped into two 8-bit “words”, which will be expanded into 6 words. The first part of the expansion, which produces the third and fourth words, is shown below. The rest of the expansion is done in exactly the same way, replacing W_0 and W_1 with W_2 and W_3, and replacing W_2 and W_3 with W_4 and W_5.



S-AES key expansion

The g function is shown in the next diagram. It is very similar to AES, first rotating the nibbles and then putting them through the S-boxes. The main difference is that the round constant is produced using x^{j+2}, where j is the number of the round of expansion. That is, the first time you expand the key you use a round constant of x^3 = 1000 for the first nibble and 0000 for the second nibble. The second time you use x^4 = 0011 for the first nibble and 0000 for the second nibble.



S-AES g function

Exercise Use the key 1010 0111 0011 1011 to encrypt the plaintext “ok” as expressed in ASCII, that is 0110 1111 0110 1011. The designers of S-AES got the ciphertext 0000 0111 0011 1000. Do you?

References

[1] Mohammad Musa, Edward Schaefer, and Stephen Wedig, A simplified AES algorithm and its linear and differential cryptanalyses, Cryptologia 27 (April 2003), no. 2, 148–177.

S-AES Program:

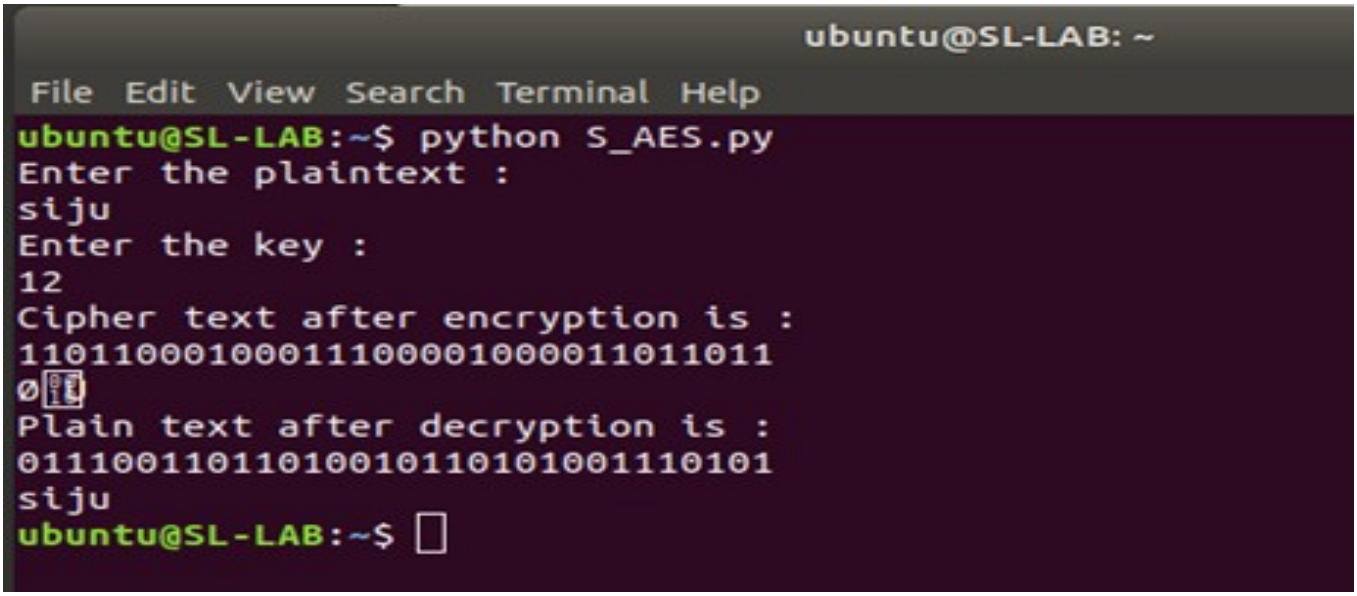
```
# S boxe
sbox = [9, 4, 10, 11, 13, 1, 8, 5, 6, 2, 0, 3, 12, 14, 15, 7]


def convertNumToAsciiBit(x): # coverts decimal to binary
    y = ""
    for i in range(len(x)):
        val = ord(x[i])
        j = 7
        ans = ""
        while j >= 0:
            w = val // (pow(2, j))
            ans += str(w)
            val = val % (pow(2, j))
            j -= 1
        y += ans
    return y


def convertAsciiToChar(x): # converts ASCII value to char
    y = ""
    for i in range(0, len(x), 8):
        ans = 0
        for j in range(8):
            ans += int(x[i + j]) * pow(2, 7 - j)
        if i == len(x) - 8:
            if chr(ans) != '#':
                y += chr(ans)
        else:
            y += chr(ans)
    return y


def keyExpansion(key): # generates 2 round keys
    x = [key[:4], key[4:8], key[8:12], key[12:16]]
    for i in range(4): # binary to decimal for each nible
        x[i] = list(map(int, x[i]))
        x[i] = x[i][0] * 8 + x[i][1] * 4 + x[i][2] * 2 + x[i][3]
    keylist = [x[0], x[1], x[2], x[3]]
    for i in range(2):
        w2 = [0, 0, 0, 0]
        if i == 0:
            val = 8 # rcon for first round
        else:
            val = 3 # rcon for 2nd round
        w2[0] = keylist[4 * i] ^ val ^ (sbox[keylist[4 * i + 3]])
        w2[1] = keylist[4 * i + 1] ^ 0 ^ (sbox[keylist[4 * i + 2]])
        w2[2] = w2[0] ^ keylist[4 * i + 2]
        w2[3] = w2[1] ^ keylist[4 * i + 3]
        keylist.append(w2[0])
        keylist.append(w2[1])
        keylist.append(w2[2])
        keylist.append(w2[3])
    return keylist # has all 3 sub-keys


def getByteFromBit(x):# converts binary to bytes
    y = []
    i = 0
    while i < (len(x)):
        y.append(8 * x[i] + 4 * x[i + 1] + 2 * x[i + 2] + x[i + 3])
```

```
        i += 4
    return y


def mixCols(y): # applies Mix-Columns
    w = []
    for i in range(len(y)):
        val = y[i] * 4
        if val >= 32:
            val ^= 38
        if val >= 16:
            val ^= 19
        w.append(val)
    ans = [0, 0, 0, 0]
    ans[0] = y[0] ^ w[1]
    ans[1] = y[1] ^ w[0]
    ans[2] = y[2] ^ w[3]
    ans[3] = y[3] ^ w[2]
    return ans


def convertByteToBit(y): # converts byte value to binary
    cipher = []
    for i in range(len(y)):
        val = y[i]
        val1 = val // 8
        cipher.append(val1)
        val = val % 8
        val1 = val // 4
        cipher.append(val1)
        val = val % 4
        val1 = val // 2
        cipher.append(val1)
        val1 = val % 2
        cipher.append(val1)
    cipher = list(map(str, cipher))
    return "".join(cipher)


def mult(x, y):
    val = x * y
    if y == 2:
        if val >= 32:
            val ^= 38
        if val >= 16:
            val ^= 19
    else:
        val = x * 8
        if val >= 64:
            val ^= 76
        if val >= 32:
            val ^= 38
        if val >= 16:
            val ^= 19
        val ^= x
    return val


def inverseMixCols(y): # applies inverse Mix-Columns
    w = [0, 0, 0, 0]
    w[0] = mult(y[0], 9) ^ mult(y[1], 2)
    w[1] = mult(y[1], 9) ^ mult(y[0], 2)
    w[2] = mult(y[2], 9) ^ mult(y[3], 2)
    w[3] = mult(y[3], 9) ^ mult(y[2], 2)
    return w
```

```
def aesDecrypt(y, keylist): # applies Decryption Algorithm
    j = 2
    for i in range(len(y)):
        y[i] ^= keylist[4 * j + i]
    j = 1
    while j >= 0:
        y[1], y[3] = y[3], y[1]
        for i in range(len(y)):
            y[i] = sbox.index(y[i])
        for i in range(len(y)):
            y[i] ^= keylist[4 * j + i]
        if j != 0:
            y = inverseMixCols(y)
        j -= 1
    return convertByteToBit(y)


def aesEncrypt(y, keylist): # applies Encryption Algorithm
    for i in range(len(y)):
        y[i] ^= keylist[i % 4]
    for i in range(1, 3):
        for j in range(len(y)):
            y[j] = sbox[y[j]]
        y[1], y[3] = y[3], y[1]
        if i != 2:
            y = mixCols(y)
        for j in range(len(y)):
            y[j] = y[j] ^ keylist[4 * i + j]
    return convertByteToBit(y)


if __name__ == "__main__":
    print("Enter the plaintext : ") # any length char input
    x = input()
    print("Enter the key : ") # char input of length 2
    key = input()
    if len(key) != 2:
        print("BAD KEY : Should be 16 bits")
        exit(0)
    key = convertNumToAsciiBit(key)
    keylist = keyExpansion(key)
    if len(x) % 2 != 0:
        x += '#'  # filler - #
    x = convertNumToAsciiBit(x)
    x = list(map(int, x))
    i = 0
    cipher = ""
    while i < len(x) - 1:
        y = getByteFromBit(x[i:i + 16])
        cipher += aesEncrypt(y, keylist)
        i += 16
    print("Cipher text after encryption is : ")
    print(cipher)
    print(convertAsciiToChar(cipher))
    x = list(map(int, cipher))
    i = 0
    plaintext = ""
    while i < len(x) - 1:
        y = getByteFromBit(x[i:i + 16])
        plaintext += aesDecrypt(y, keylist)
        i += 16
    print("Plain text after decryption is : ")
    print(plaintext)
```

```
print(convertAsciiToChar(plaintext))
```

Output:



Assignment No.3. Implementation of Diffie-Hellman Algorithm.

Title: Implementation of Diffie-Hellman key exchange
Theory:

Diffie-Hellman key exchange, also called exponential key exchange, is a method of digitalencryption that uses numbers raised to specific powers to produce decryption keys on the basis of components that are never directly transmitted, making the task of a would-be code breaker mathematically overwhelming.

To implement Diffie-Hellman, the two end users Alice and Bob, while communicating over a channel they know to be private, mutually agree on positive whole numbers p and q, such that p is a prime number and q is a generator of p. The generator q is a number that, when raised to positive whole-number powers less than p, never produces the same result for any two such whole numbers. The value of p may be large but the value of q is usually small.

Once Alice and Bob have agreed on p and q in private, they choose positive whole-number personal keys a and b, both less than the prime-number modulus p. Neither user divulges their personal key to anyone; ideally they memorize these numbers and do not write them down or store them anywhere. Next, Alice and Bob compute public keys a^* and b^* based on their personal keys according to the formulas

$$a^* = q^a \bmod p$$

and

$$b^* = q^b \bmod p$$

The two users can share their public keys a^* and b^* over a communications medium assumed to be insecure, such as the Internet or a corporate wide area network (WAN). From these public keys, a number x can be generated by either user on the basis of their own personal keys. Alice computes x using the formula

$$x = (b^*)^a \bmod p$$

Bob computes x using the formula

$$x = (a^*)^b \bmod p$$

The value of x turns out to be the same according to either of the above two formulas. However, the personal keys a and b, which are critical in the calculation of x, have not been transmitted over a public medium. Because it is a large and apparently random number, a potential hacker has almost no chance of correctly guessing x, even with the help of a powerful computer to conduct millions of trials. The two users can therefore, in theory, communicate privately over a public medium with an encryption method of their choice using the decryption key x.

The most serious limitation of Diffie-Hellman in its basic or "pure" form is the lack of authentication. Communications using Diffie-Hellman all by itself are vulnerable to man in the middle attacks. Ideally, Diffie-Hellman should be used in conjunction with a recognized authentication method such as digital signatures to verify the identities of the users over the public communications medium. Diffie-Hellman is well

suited for use in data communication but is less often used for data stored or archived over long periods of time.

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

- For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables one prime P and G (a primitive root of P) and two private values a and b.
- P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly, the opposite person received the key and from that generates a secret key after which they have the same secret key to encrypt.

Step by Step Explanation

ALICE	BOB
Public Keys available = P, G	Public Keys available = P, G
Private Key Selected = a	Private Key Selected = b
Key generated =	Key generated =
Exchange of generated keys takes place	
Key received = y	key received = x
Generated Secret Key =	Generated Secret Key =
Algebraically it can be shown that	

Example

```
Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and
        Bob selected a private key b = 3

Step 3: Alice and Bob compute public values
Alice:    x =(9^4 mod 23) = (6561 mod 23) = 6
        Bob:    y = (9^3 mod 23) = (729 mod 23)  = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key y =16 and
        Bob receives public key x = 6

Step 6: Alice and Bob compute symmetric keys
        Alice:  ka = y^a mod p = 65536 mod 23 = 9
        Bob:    kb = x^b mod p = 216 mod 23 = 9

Step 7: 9 is the shared secret.
```

Diffie Hellman Program:

```
#include<stdio.h>

long long int power(int a, int b, int mod)
{
long long int t;
if(b==1)
return a;
t=power(a,b/2,mod);
if(b%2==0)
return (t*t)%mod;
else
return (((t*t)%mod)*a)%mod;
}
long int calculateKey(int a, int x, int n)
{
return power(a,x,n);
}
void main()
{
int p,g,x,a,y,b;

printf("Enter the value of p and g : ");
scanf("%d%d",&p,&g);
printf("Enter the value of x for the first person : ");
scanf("%d",&x);
a=power(g,x,p);
printf("Enter the value of y for the second person : ");
scanf("%d",&y);
b=power(g,y,p);
printf("key for the first person is :%lld\n",power(b,x,p));
printf("key for the second person is :%lld\n",power(a,y,p));

}
```

Output:

```
diffie2.c:23:15: note: each undeclared identifier is reported only once for each function it appears in
ubuntu@SL-LAB:~/Downloads$ gcc diffie2.c
ubuntu@SL-LAB:~/Downloads$ ./a.out
Enter the value of p and g : 23
5
Enter the value of x for the first person : 4
Enter the value of y for the second person : 3
key for the first person is :18
key for the second person is :18
ubuntu@SL-LAB:~/Downloads$
```

Assignment No.4 Implementation of a RSA Algorithm

Theory:

RSA is a cryptosystem for public-key encryption, and is widely used for securing sensitive data, particularly when being sent over an insecure network such as the Internet.

RSA was first described in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman of the Massachusetts Institute of Technology. Public-key cryptography, also known as asymmetric cryptography, uses two different but mathematically linked keys, one public and one private. The public key can be shared with everyone, whereas the private key must be kept secret. In RSA cryptography, both the public and the private keys can encrypt a message; the opposite key from the one used to encrypt a message is used to decrypt it. This attribute is one reason why RSA has become the most widely used asymmetric algorithm: It provides a method of assuring the confidentiality, integrity, authenticity and non-reputability of electronic communications and data storage.

Many protocols like SSH, OpenPGP, S/MIME, and SSL/TLS rely on RSA for encryption and digital signature functions. It is also used in software programs -- browsers are an obvious example, which need to establish a secure connection over an insecure network like the Internet or validate a digital signature. RSA signature verification is one of the most commonly performed operations in IT.

Explaining RSA's popularity

RSA derives its security from the difficulty of factoring large integers that are the product of two large prime numbers. Multiplying these two numbers is easy, but determining the original prime numbers from the total -- factoring -- is considered infeasible due to the time it would take even using today's super computers.

The public and the private key-generation algorithm is the most complex part of RSA cryptography. Two large prime numbers, *p* and *q*, are generated using the Rabin-Miller primality test algorithm. A modulus *n* is calculated by multiplying *p* and *q*. This number is used by both the public and private keys and provides the link between them. Its length, usually expressed in bits, is called the key length. The public key consists of the modulus *n*,and a public exponent, *e*, which is normally set at 65537, as it's a prime number that is not too large. The *e* figure doesn't have to be a secretly selected prime number as the public key is shared with everyone. The private key consists of the modulus *n* and the private exponent *d*, which is calculated using the Extended Euclidean algorithm to find the multiplicative inverse with respect to the totient of *n*.

A simple, worked example

Alice generates her RSA keys by selecting two primes: p=11 and q=13. The modulus $n=p\times q=143$. The totient of n $\phi(n)=(p-1)x(q-1)=120$. She chooses 7 for her RSA public key *e*and calculates her RSA private key using the Extended Euclidean Algorithm which gives her 103.

Bob wants to send Alice an encrypted message *M* so he obtains her RSA public key (*n*, e) which in this example is (143, 7). His plaintext message is just the number 9 and is encrypted into ciphertext *C* as follows:

$M^e \bmod n = 9^7 \bmod 143 = 48 = C$

When Alice receives Bob's message she decrypts it by using her RSA private key (*d*, *n*) as follows:

$C^d \bmod n = 48^{103} \bmod 143 = 9 = M$

To use RSA keys to digitally sign a message, Alice would create a hash or message digest of her message to Bob, encrypt the hash value with her RSA private key and add it to the message. Bob can then verify that the message has been sent by Alice and has not been altered by decrypting the hash value with her public key. If this value matches the hash of the original message, then only Alice could have sent it (authentication and non-repudiation) and the message is exactly as she wrote it (integrity). Alice could, of course, encrypt her message with Bob's RSA public key (confidentiality) before sending it to Bob. A digital certificate contains information that identifies the certificate's owner and also contains the owner's public key. Certificates are signed by the certificate authority that issues them, and can simplify the process of obtaining public keys and verifying the owner.

Security of RSA

As discussed, the security of RSA relies on the computational difficulty of factoring large integers. As computing power increases and more efficient factoring algorithms are discovered, the ability to factor larger and larger numbers also increases. Encryption strength is directly tied to key size, and doubling key length delivers an exponential increase in strength, although it does impair performance. RSA keys are typically 1024- or 2048-bits long, but experts believe that 1024-bit keys could be broken in the near future, which is why government and industry are moving to a minimum key length of 2048-bits. Barring an unforeseen breakthrough in quantum computing, it should be many years before longer keys are required, but elliptic curve cryptography is gaining favor with many security experts as an alternative to RSA for implementing public-key cryptography. It can create faster, smaller and more efficient cryptographic keys. Much of today's hardware and software is ECC-ready and its popularity is likely to grow as it can deliver equivalent security with lower computing power and battery resource usage, making it more suitable for mobile apps than RSA. Finally, a team of researchers which included Adi Shamir, a co-inventor of RSA, has successfully determined a 4096-bit RSA key using acoustic cryptanalysis, however any encryption algorithm is vulnerable to this type of attack.

RSA algorithm is asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key.** As the name describes that the Public Key is given to everyone and Private key is kept private.

An example of asymmetric cryptography :

1. A client (for example browser) sends its public key to the server and requests for some data.
2. The server encrypts the data using client's public key and sends the encrypted data.
3. Client receives this data and decrypts it.

Since this is asymmetric, nobody else except browser can decrypt the data even if a third party has public key of browser.

The idea! The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private

key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024 bit keys could be broken in the near future. But till now it seems to be an infeasible task.

Let us learn the mechanism behind RSA algorithm :

>> Generating Public Key :

- Select two prime no's. Suppose **P = 53 and Q = 59**.

Now First part of the Public key : **n = P*Q = 3127**.

- We also need a small exponent say **e** :

But e Must be

- An integer.
- Not be a factor of n.
- **1 < e < Φ(n)** [Φ(n) is discussed below],
- Let us now consider it to be equal to 3.

- Our Public Key is made of n and e

>> Generating Private Key :

- We need to calculate Φ(n) :

Such that **Φ(n) = (P-1)(Q-1)**
so, Φ(n) = 3016

- Now calculate Private Key, **d** :

d = (k*Φ(n) + 1) / e for some integer k

For k = 2, value of d is 2011.

Now we are ready with our – Public Key (n = 3127 and e = 3) and Private Key(d = 2011)

Now we will encrypt **"HI"** :

- Convert letters to numbers : H = 8 and I = 9

- Thus **Encrypted Data c = 89^e mod n**.

Thus our Encrypted Data comes out to be 1394

Now we will decrypt **1394** :

- **Decrypted Data = c^d mod n**.

Thus our Encrypted Data comes out to be 89

8 = H and I = 9 i.e. "HI".

RSA Program :

```
def gcd(a, b): # calculates GCD of a and d
    while b != 0:
        c = a % b
        a = b
        b = c
    return a


def modinv(a, m): # calculates modulo inverse of a for mod m
    for x in range(1, m):
        if (a * x) % m == 1:
```

```
            return x
    return None


def coprimes(a): # calculates all possible co-prime numbers with a
    l = []
    for x in range(2, a):
        if gcd(a, x) == 1 and modinv(x, phi) != None:
            l.append(x)
    for x in l:
        if x == modinv(x, phi):
            l.remove(x)
    return l

def encrypt_block(m): # encrypts a single block
    c = m ** e % n
    return c


def decrypt_block(c): # decrypts a single block
    m = c ** d % n
    return m


def encrypt_string(s): # applies encryption
    return ''.join([chr(encrypt_block(ord(x))) for x in list(s)])


def decrypt_string(s): # applies decryption
    return ''.join([chr(decrypt_block(ord(x))) for x in list(s)])

if __name__ == "__main__":
    p = int(input('Enter prime p: '))
    q = int(input('Enter prime q: '))

    print("Choosen primes:\np=" + str(p) + ", q=" + str(q) + "\n")

    n = p * q
    print("n = p * q = " + str(n) + "\n")

    phi = (p - 1) * (q - 1)
    print("Euler's function (totient) [phi(n)]: " + str(phi) + "\n")

    print("Choose an e from a below coprimes array:\n")
    print(str(coprimes(phi)) + "\n")
    e = int(input())

    d = modinv(e, phi) # calculates the decryption key d

    print("\nYour public key is a pair of numbers (e=" + str(e) + ", n=" + str(n) +
").\n")
    print("Your private key is a pair of numbers (d=" + str(d) + ", n=" + str(n) +
").\n")

    s = input("Enter a message to encrypt: ")
    print("\nPlain message: " + s + "\n")
    enc = encrypt_string(s)
    print("Encrypted message: ", enc, "\n")
    dec = decrypt_string(enc)
    print("Decrypted message: " + dec + "\n")
```

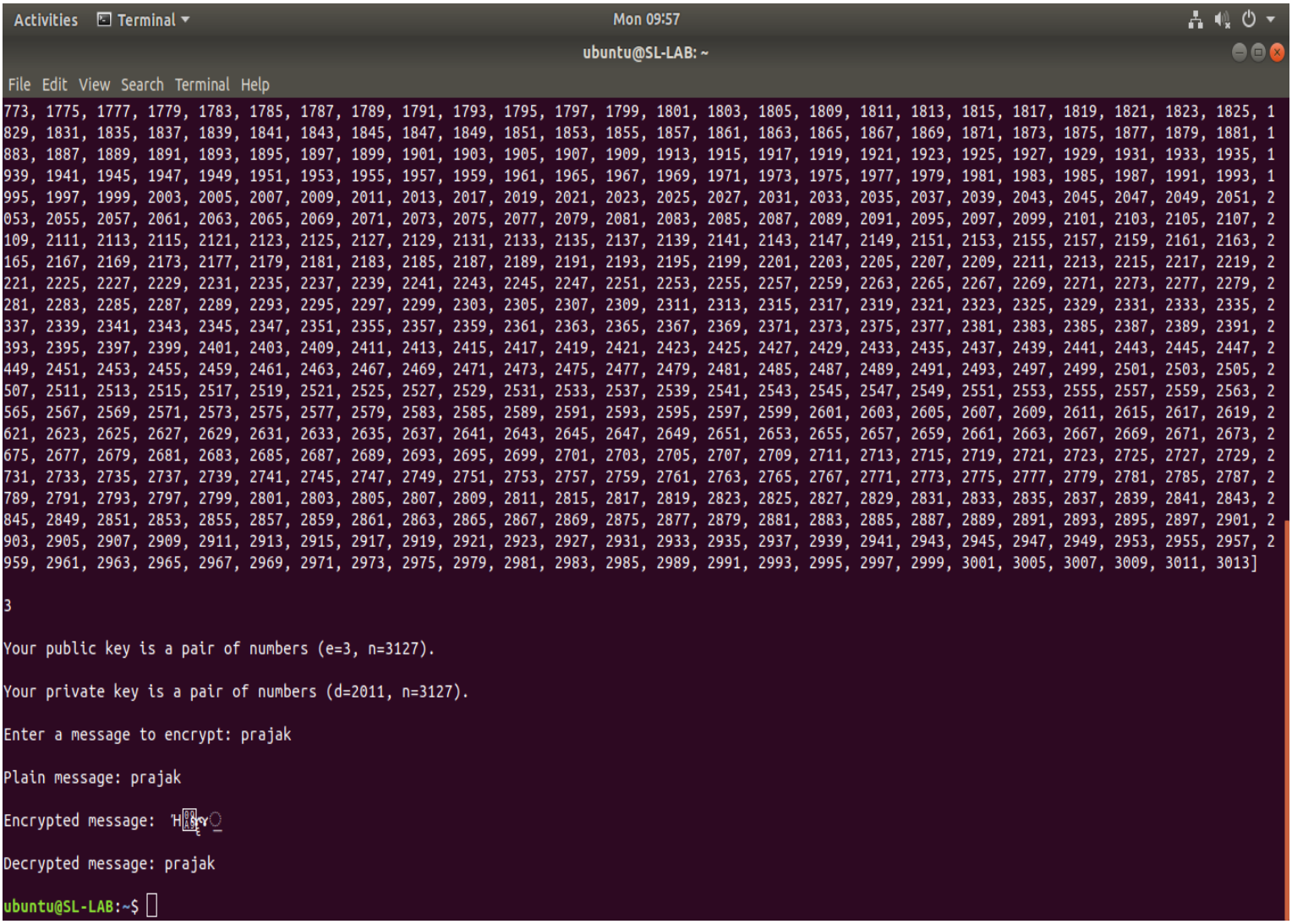
Output:

Activities Terminal Mon 09:57

ubuntu@SL-LAB: ~

File Edit View Search Terminal Help

```
ubuntu@SL-LAB:~$ python RSA.py
Enter prime p: 53
Enter prime q: 59
Choosen primes:
p=53, q=59

n = p * q = 3127

Euler's function (totient) [phi(n)]: 3016

Choose an e from a below coprimes array:

[3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25, 27, 31, 33, 35, 37, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 8
9, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 119, 121, 123, 125, 127, 129, 131, 133, 135, 137, 139, 141, 147, 149, 151, 153, 155, 157, 1
59, 161, 163, 165, 167, 171, 173, 175, 177, 179, 181, 183, 185, 187, 189, 191, 193, 197, 199, 201, 205, 207, 209, 211, 213, 215, 217, 219, 223, 225, 2
27, 229, 231, 235, 237, 239, 241, 243, 245, 249, 251, 253, 255, 257, 259, 263, 265, 267, 269, 271, 275, 277, 279, 281, 283, 285, 287, 289, 291, 293, 2
95, 297, 301, 303, 305, 307, 309, 311, 313, 315, 317, 321, 323, 327, 329, 331, 333, 335, 337, 339, 341, 343, 345, 347, 349, 353, 355, 357, 359, 361, 3
63, 365, 367, 369, 371, 373, 375, 379, 381, 383, 385, 387, 389, 391, 393, 395, 397, 399, 401, 405, 407, 409, 411, 413, 415, 417, 419, 421, 423, 425, 4
27, 431, 433, 437, 439, 441, 443, 445, 447, 449, 451, 453, 457, 459, 461, 463, 465, 467, 469, 471, 473, 475, 477, 479, 483, 485, 487, 489, 491, 495, 4
97, 499, 501, 503, 505, 509, 511, 513, 515, 517, 519, 523, 525, 527, 529, 531, 535, 537, 539, 541, 543, 545, 547, 549, 553, 555, 557, 561, 563, 565, 5
67, 569, 571, 573, 575, 577, 579, 581, 583, 587, 589, 591, 593, 595, 597, 599, 601, 603, 605, 607, 613, 615, 617, 619, 621, 623, 625, 627, 629, 631, 6
33, 635, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 665, 669, 671, 673, 675, 677, 679, 681, 683, 685, 687, 691, 693, 695, 697, 699, 7
01, 703, 705, 707, 709, 711, 713, 717, 719, 721, 723, 727, 729, 731, 733, 735, 737, 739, 743, 745, 747, 749, 751, 755, 757, 759, 761, 763, 765, 769, 7
71, 773, 775, 777, 779, 781, 785, 787, 789, 791, 795, 797, 799, 801, 803, 805, 807, 809, 811, 813, 815, 817, 821, 823, 825, 827, 829, 831, 833, 835, 8
37, 839, 843, 847, 849, 851, 853, 855, 857, 859, 861, 863, 865, 867, 869, 873, 875, 877, 879, 881, 883, 885, 887, 889, 891, 893, 895, 901, 903, 905, 9
07, 909, 911, 913, 915, 917, 919, 921, 925, 927, 929, 931, 933, 935, 937, 939, 941, 943, 945, 947, 951, 953, 955, 959, 961, 963, 965, 967, 969, 971, 9
73, 977, 979, 981, 983, 985, 989, 991, 993, 995, 997, 999, 1003, 1005, 1007, 1009, 1011, 1013, 1017, 1019, 1021, 1023, 1025, 1029, 1031, 1033, 1035, 1
037, 1039, 1041, 1043, 1045, 1047, 1049, 1051, 1055, 1057, 1059, 1061, 1063, 1065, 1067, 1069, 1071, 1075, 1077, 1081, 1083, 1085, 1087, 1089, 1091, 1
093, 1095, 1097, 1099, 1101, 1103, 1107, 1109, 1111, 1113, 1115, 1117, 1119, 1121, 1123, 1125, 1127, 1129, 1133, 1135, 1137, 1139, 1141, 1143, 1145, 1
147, 1149, 1151, 1153, 1155, 1159, 1161, 1163, 1165, 1167, 1169, 1171, 1173, 1175, 1177, 1179, 1181, 1185, 1187, 1191, 1193, 1195, 1197, 1199, 1201, 1
203, 1205, 1207, 1211, 1213, 1215, 1217, 1219, 1221, 1223, 1225, 1227, 1229, 1231, 1233, 1237, 1239, 1241, 1243, 1245, 1249, 1251, 1253, 1255, 1257, 1
259, 1263, 1265, 1267, 1269, 1271, 1273, 1277, 1279, 1281, 1283, 1285, 1289, 1291, 1293, 1295, 1297, 1299, 1301, 1303, 1307, 1309, 1311, 1315, 1317, 1
319, 1321, 1323, 1325, 1327, 1329, 1331, 1333, 1335, 1337, 1341, 1343, 1345, 1347, 1349, 1351, 1353, 1355, 1357, 1359, 1361, 1367, 1369, 1371, 1373, 1
375, 1377, 1379, 1381, 1383, 1385, 1387, 1389, 1393, 1395, 1397, 1399, 1401, 1403, 1405, 1407, 1409, 1411, 1413, 1415, 1419, 1423, 1425, 1427, 1429, 1
431, 1433, 1435, 1437, 1439, 1441, 1445, 1447, 1449, 1451, 1453, 1455, 1457, 1459, 1461, 1463, 1465, 1467, 1471, 1473, 1475, 1477, 1481, 1483, 1485, 1
487, 1489, 1491, 1493, 1497, 1499, 1501, 1503, 1505, 1509, 1511, 1513, 1515, 1517, 1519, 1523, 1525, 1527, 1529, 1531, 1533, 1535, 1539, 1541, 1543, 1
545, 1549, 1551, 1553, 1555, 1557, 1559, 1561, 1563, 1565, 1567, 1569, 1571, 1575, 1577, 1579, 1581, 1583, 1585, 1587, 1589, 1591, 1593, 1597, 1601, 1
603, 1605, 1607, 1609, 1611, 1613, 1615, 1617, 1619, 1621, 1623, 1627, 1629, 1631, 1633, 1635, 1637, 1639, 1641, 1643, 1645, 1647, 1649, 1655, 1657, 1
```



Assignment No.5. ECC Algorithm Implementation

Assignment No

Title: Implementation of ECC algorithm

Theory:

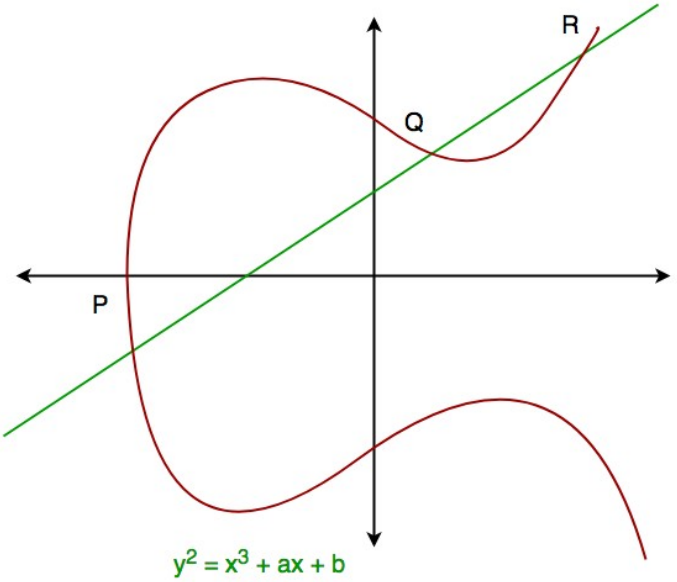
Elliptic Curve Cryptography (ECC) is an approach to public-key cryptography, based on the algebraic structure of elliptic curves over finite fields. ECC requires a smaller key as compared to non-ECC cryptography to provide equivalent security (a 256-bit ECC security have an equivalent security attained by 3072-bit RSA cryptography).

For a better understanding of Elliptic Curve Cryptography, it is very important to understand the basics of Elliptic Curve. An elliptic curve is a planar algebraic curve defined by an equation of the form

where ‘a’ is the co-efficient of x and ‘b’ is the constant of the equation

The curve is non-singular; that is its graph has no cusps or self-intersections (when the characteristic of the co-efficient field is equal to 2 or 3).

In general, an elliptic curve looks like as shown below. Elliptic curves could intersect atmost 3 points when a straight line is drawn intersecting the curve. As we can see that elliptic curve is symmetric about the x-axis, this property plays a key role in the algorithm.



This approach uses six tuple {P, a, b, G, n, h}

P = Field that the curve is define over

G = Generator point

a, b = Values define the curve

h = Co- factor

n = Prime order of G

ECC Encryption/Decryption

ECC Encryption/Decryption Several approaches to encryption/decryption using elliptic curves have been analyzed in the literature. This one is an analog of the ElGamal public-key encryption algorithm. The sender must first encode any message m as a point on the elliptic curve Pm (there are relatively straightforward techniques for this). Note that the ciphertext is a pair of points on the elliptic curve. The sender masks the message using random k, but also sends along a "clue" allowing the receiver who know the private-key to recover k and hence the message. For an attacker to recover the message, the attacker would have to compute k given G and kG, which is assumed hard.

- first encode any message m as a point on the elliptic curve

  P_m

- select suitable curve & point G as in D-H
- A & B select private keys $n_A<n$, $n_B<n$
- compute public keys: $P_A = n_A G$, $P_B = n_B G$
- A encrypts P_m : $C_m=\{kG, P_m + kP_B\}$,

  k: random positive integer

  P_B: B's public key

- B decrypts C_m compute:

$$P_m + kP_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$$

ECC Program:

1.

```
\*ECCKeyGeneration.java*\

import java.security.*;
import java.security.spec.*;

public class ECCKeyGeneration {
  public static void main(String[] args) throws Exception {
    KeyPairGenerator kpg;
    kpg = KeyPairGenerator.getInstance("EC","SunEC");
    ECGenParameterSpec ecsp;
    ecsp = new ECGenParameterSpec("secp192r1");
    kpg.initialize(ecsp);

    KeyPair kp = kpg.genKeyPair();
    PrivateKey privKey = kp.getPrivate();
    PublicKey pubKey = kp.getPublic();

    System.out.println(privKey.toString());
    System.out.println(pubKey.toString());
  }
}
```

2.

```
/*ECC Provider Test*/
import java.security.Provider;
import java.security.Provider.Service;
import java.security.Security;
import sun.security.ec.SunEC;

public class ECCProviderTest {

    /**
     * @param args the command line arguments
     */
    public static void main(final String[] args) {
        Provider sunEC = new SunEC();
        Security.addProvider(sunEC);
        for(Service service : sunEC.getServices()) {
            System.out.println(service.getType() + ": "
                    + service.getAlgorithm());
        }
    }

}
```

3.

```
/* ECC Signature */
import java.math.BigInteger;
import java.security.*;
import java.security.spec.*;

public class ECCSignature {
  public static void main(String[] args) throws Exception {
    KeyPairGenerator kpg;
    kpg = KeyPairGenerator.getInstance("EC","SunEC");

    ECGenParameterSpec ecsp;
    ecsp = new ECGenParameterSpec("sect163k1");
```

```
    kpg.initialize(ecsp);

    KeyPair kp = kpg.genKeyPair();
    PrivateKey privKey = kp.getPrivate();
    PublicKey pubKey = kp.getPublic();
    System.out.println(privKey.toString());
    System.out.println(pubKey.toString());

    Signature ecdsa;
    ecdsa = Signature.getInstance("SHA1withECDSA","SunEC");
    ecdsa.initSign(privKey);

    String text = "In teaching others we teach ourselves";
    System.out.println("Text: " + text);
    byte[] baText = text.getBytes("UTF-8");

    ecdsa.update(baText);
    byte[] baSignature = ecdsa.sign();
    System.out.println("Signature: 0x" + (new BigInteger(1,
baSignature).toString(16)).toUpperCase());

    Signature signature;
    signature = Signature.getInstance("SHA1withECDSA","SunEC");
    signature.initVerify(pubKey);
    signature.update(baText);
    boolean result = signature.verify(baSignature);
    System.out.println("Valid: " + result);
  }
}
```