Sinhgad Technical Education Society's

Sinhgad College of Engineering ,Pune

Sinhgad College of Engineering ,Pune

Department of Information Technology

LAB Manual

Laboratory Practice-I
(Machine Learning)

(Subject Code: 314448)

Sinhgad Technical Education Society's

Sinhgad College of Engineering , Pune

CERTIFICATE

This is to certify that Mr./Ms. ……..
of Class Fourth Year (B.E.) Branch:- ……..Div:-…….
Roll No……Exam Seat No …… has completed. All Practical Assignments in the subject **Laboratory Practice-I (Machine Learning) (Subject Code: 314448)** *satisfactorily* in the department of Information Technology in the *academic year 2022-2023.*

Subject In-charge **HOD** **Principal**

List of Laboratory Assignments

Sr.No.	Statement of Assignment
1	**Data preparation:** Download heart dataset from following link. https://www.kaggle.com/zhaoyingzhu/heartcsv Perform following operation on given dataset. a) Find Shape of Data b) Find Missing Values c) Find data type of each column d) Finding out Zero's e) Find Mean age of patients f) Now extract only Age, Sex, ChestPain, RestBP, Chol. Randomly divide dataset in training (75%) and testing (25%). Through the diagnosis test I predicted 100 report as COVID positive, but only 45 of those were actually positive. Total 50 people in my sample were actually COVID positive. I have total500 samples. Create confusion matrix based on above data and find I. Accuracy II. Precision III. Recall IV. F-1 score
2	**Assignment on Regression Technique:** Download any freely available suitable dataset and perform following steps a. Apply Linear Regression using suitable library function and perform the prediction for unseen dataset. b. Assess the performance of regression models using MSE, MAE and R-Square metrics c. Visualize simple regression model.
3	**Assignment on Classification Technique:** Every year many students give the GRE exam to get admission in foreign Universities. The data set contains GRE Scores (out of 340), TOEFL Scores (out of 120), University Rating (out of 5), Statement of Purpose strength (out of 5), Letter of Recommendation strength (out of 5), Undergraduate GPA (out of 10), Research Experience (0=no, 1=yes), Admitted (0=no, 1=yes). Admitted is the target variable. Data Set Available on kaggle (The last column of the dataset needs to be changed to 0 or

	1)Data Set : https://www.kaggle.com/mohansacharya/graduate-admissions The counselor of the firm is supposed check whether the student will get an admission or not based on his/her GRE score and Academic Score. So to help the counselor to take appropriate decisions build a machine learning model classifier using Decision tree to predict whether a student will get admission or not. Apply Data pre-processing (Label Encoding, Data Transformation) techniques if necessary. Perform data-preparation (Train-Test Split) C. Apply Machine Learning Algorithm D. Evaluate Model.
4	**Assignment on Clustering Techniques** Download the following customer dataset from below link: Data Set: https://www.kaggle.com/shwetabh123/mall-customers This dataset gives the data of Income and money spent by the customers visiting a Shopping Mall. The data set contains Customer ID, Gender, Age, Annual Income, Spending Score. Therefore, as a mall owner you need to find the group of people who are the profitable customers for the mall owner. Apply at least two clustering algorithms (based on Spending Score) to find the group of customers. a. Apply Data pre-processing (Label Encoding , Data Transformation.) techniques if necessary. b. Perform data-preparation(Train-Test Split) c. Apply Machine Learning Algorithm d. Evaluate Model. e. Apply Cross-Validation and Evaluate Model
5	**Assignment on Association Rule Learning** Download Market Basket Optimization dataset from below link. Data Set: https://www.kaggle.com/hemanthkumar05/market-basket-optimization This dataset comprises the list of transactions of a retail company over the period of one week. It contains a total of 7501 transaction records where each record consists of the list of items sold in one transaction. Using this record of transactions and items in each transaction, find the association rules between items. There is no header in the dataset and the first row contains the first transaction, so mentioned header = None here while loading dataset. a. Follow following steps: b. Data Preprocessing c. Generate the list of transactions from the dataset d. Train Apriori algorithm on the dataset e. Visualize the list of rules F. Generated rules depend on the values of hyper parameters. By increasing the minimum confidence value and find the rules accordingly
6	**Assignment on Multilayer Neural Network**

	Download the dataset of National Institute of Diabetes and Digestive and Kidney Diseases from below link : DataSet: https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv The dataset is has total 9 attributes where the last attribute is “Class attribute” having values 0 and 1. (1=”Positive for Diabetes”, 0=”Negative”) a. Load the dataset in the program. Define the ANN Model with Keras. Define at least two hidden layers. Specify the ReLU function as activation function for the hidden layer and Sigmoid for the output layer. b. Compile the model with necessary parameters. Set the number of epochs and batch size and fit the model. c. Evaluate the performance of the model for different values of epochs and batch sizes. d. Evaluate model performance using different activation functions Visualize the model using ANN Visualizer.

Assignment No. 1

Title: Data Preparation

Problem Statement:

Download heart dataset from the following link. https://www.kaggle.com/zhaoyingzhu/heartcsv
Perform the following operation on a given dataset.
a) Find Shape of Data
b) Find Missing Values
c) Find data type of each column
d) Finding out Zero's
e) Find Mean age of patients
f) Now extract only Age, Sex, ChestPain, RestBP, Chol. Randomly divide the dataset in training (75%) and testing (25%).

Objectives:

Data preparation is particular to data, the objectives of the projects, and the algorithms that will be used in data modeling techniques. Data Preparation is the process of cleaning and transforming raw data to make predictions accurately through using ML algorithms. Each predictive modeling project with machine learning is different, but there are common steps performed on each project.

Theory:

What is Data Preparation

On a predictive modeling project, such as classification or regression, raw data typically cannot be used directly. Data can not be directly used as it might have impurities some which will act as a barrier in further processes. So hence we can define that data preparation is one of the important steps in the data science domain, in which data collected from multiple sources is cleaned and transformed to improve its quality prior to use in business analytics. Having quality data makes administration fast.
Data Preparation is the first step in data analytics projects and can include many tasks such as loading data, data ingestion,fusion,cleaning augmentation and delivery.
As such, the raw data must be pre-processed prior to being used to fit and evaluate a machine learning model. This step in a predictive modeling project is referred to as "**data preparation**", although it goes by many other names, such as "*data wrangling*", "*data cleaning*", "*data pre-processing*" and "*feature engineering*". Some of these names may better fit as sub-tasks for the broader data preparation process.
We can define data preparation as the transformation of raw data into a form that is more suitable for modeling.
These tasks include:

- **Data Cleaning**: Identifying and correcting mistakes or errors in the data.
- **Feature Selection**: Identifying those input variables that are most relevant to the task.
- **Data Transforms**: Changing the scale or distribution of variables.
- **Feature Engineering**: Deriving new variables from available data.

- **Dimensionality Reduction**: Creating compact projections of the data.

Each of these tasks is a whole field of study with specialized algorithms.

Why do we need Data Preprocessing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

It involves below steps:

1. Getting the dataset
2. Importing libraries
3. Importing datasets
4. Finding Missing Data
5. Encoding Categorical Data
6. Splitting dataset into training and test set
7. Feature scaling

1) Get the Dataset

To create a machine learning model, the first thing we required is a dataset as a machine learning model completely works on data. The collected data for a particular problem in a proper format is known as the **dataset**.

Dataset may be of different formats for different purposes, such as, if we want to create a machine learning model for business purpose, then dataset will be different with the dataset required for a liver patient. So each dataset is different from another dataset. To use the dataset in our code, we usually put it into a CSV **file**. However, sometimes, we may also need to use an HTML or xlsx file.

.CSV File

CSV stands for "**Comma-Separated Values**" files; it is a file format which allows us to save the tabular data, such as spreadsheets. It is useful for huge datasets and can use these datasets in programs.

For real-world problems, we can download datasets online from various sources such as https://www.kaggle.com/zhaoyingzhu/heartcsv

We can also create our dataset by gathering data using various API with Python and put that data into a **.csv** file.

2) Importing Libraries

In order to perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing, which are:

Numpy: Numpy Python library is used for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in Python. It also supports to add large, multidimensional arrays and matrices. So, in Python, we can import it as:

import numpy as nm

Here we have used **nm**, which is a short name for Numpy, and it will be used in the whole program.

Matplotlib: The second library is **matplotlib**, which is a Python 2D plotting library, and with this library, we need to import a sub-library **pyplot**. This library is used to plot any type of charts in Python for the code. It will be imported as below:

import matplotlib.pyplot as mpt

Here we have used mpt as a short name for this library.

Pandas: The last library is the Pandas library, which is one of the most famous Python libraries and used for importing and managing the datasets. It is an open-source data manipulation and analysis library. It will be imported as below:

Here, we have used pd as a short name for this library. Consider the below image:

```
1 # importing libraries
2 import numpy as nm
3 import matplotlib.pyplot as mtp
4 import pandas as pd
5
```

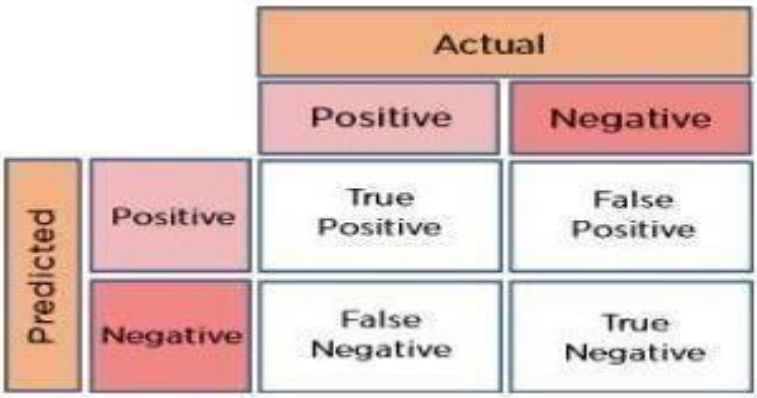
3) Importing the Datasets

Now we need to import the datasets which we have collected for our machine learning project. But before importing a dataset, we need to set the current directory as a working directory. To set a working directory in Spyder IDE, we need to follow the below steps:

1. Save your Python file in the directory which contains dataset.
2. Go to File explorer option in Spyder IDE, and select the required directory.
3. Click on F5 button or run option to execute the file.

Confusion Matrix:

It is matrix of size 2*2 matrix used to describe the performance of a classification model with actual values on one axis and predicted on another.



Confusion matrix not only shows performance of predictive model, but also which classes are being predicted correctly and incorrectly and what type of errors are being made
True Positive : model predicted positive and its's true
True Negative: model predicted negative and it's true i.e. negative in reality.
False Positive: model predicted positive and it's false
False Negative: model predicted negative and it's false
The rate of confusion matrix are as follows
True Positive Rate(TPR), False Negative Rate(FNR), True Negative Rate(TNR), False Positive Rate(FPR),

$$TPR = \frac{TP}{Actual\ Positive} = \frac{TP}{TP + FN}$$

$$FNR = \frac{FN}{Actual\ Positive} = \frac{FN}{TP + FN}$$

$$TNR = \frac{TN}{Actual\ Negative} = \frac{TN}{TN + FP}$$

$$FPR = \frac{FP}{Actual\ Negative} = \frac{FP}{TN + FP}$$

Need for Confusion Matrix in Machine learning

- It evaluates the performance of the classification models, when they make predictions on test data, and tells how good our classification model is.
- It not only tells the error made by the classifiers but also the type of errors such as it is either type-I or type-II error.
- With the help of the confusion matrix, we can calculate the different parameters for the model, such as accuracy, precision, etc.

Calculations using Confusion Matrix:

We can perform various calculations for the model, such as the model's accuracy, using this matrix. These calculations are given below:

- **Classification Accuracy:** It is one of the important parameters to determine the accuracy of the classification problems. It defines how often the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classifier to all number of predictions made by the classifiers. The formula is given below:

$$\text{Accuracy}=\frac{TP+TN}{TP+FP+FN+TN}$$

- **Misclassification rate:** It is also termed as Error rate, and it defines how often the model gives the wrong predictions. The value of error rate can be calculated as the number of incorrect predictions to all numbers of the predictions made by the classifier. The formula is given below:

$$\text{Error rate}=\frac{FP+FN}{TP+FP+FN+TN}$$

- **Precision:** It can be defined as the number of correct outputs provided by the model or out of all positive classes that have predicted correctly by the model, how many of them were actually true. It can be calculated using the below formula:

$$\text{Precision}=\frac{TP}{TP+FP}$$

- **Recall:** It is defined as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.

$$\text{Recall}=\frac{TP}{TP+FN}$$

- **F-measure:** If two models have low precision and high recall or vice versa, it is difficult to compare these models. So, for this purpose, we can use F-score. This score helps us to evaluate the recall and precision at the same time. The F-score is maximum if the recall is equal to the precision. It can be calculated using the below formula:

$$\text{F-measure}=\frac{2*Recall*Precision}{Recall+Precision}$$

Other important terms used in Confusion Matrix:

- **Null Error rate:** It defines how often our model would be incorrect if it always predicted the majority class. As per the accuracy paradox, it is said that "*the best classifier has a higher error rate than the null error rate.*"

- **ROC Curve:** The ROC is a graph displaying a classifier's performance for all possible thresholds. The graph is plotted between the true positive rate (on the Y-axis) and the false Positive rate (on the x-axis).

Conclusion:

Most machine learning algorithms require data to be formatted in a very specific way, so datasets generally require some amount of preparation before they can yield useful insights.

Questions:

1. What are the 5 major steps of data preprocessing?
2. Why is data preprocessing required?
3. What are the different techniques for data preprocessing in machine learning?
4. What is the use of a confusion matrix in machine learning?
5. How do you calculate precision and recall from confusion matrix
6. How can you calculate accuracy using a confusion matrix?

Name: Rohan Shinde

Roll No: 307B058

Division: 2

Batch: D

Assignment No: 1

Problem Statement: Data Preparation

Download heart dataset from the following link. https://www.kaggle.com/zhaoyingzhu/heartcsv
Perform the following operation on a given dataset.

g) Find Shape of Data
h) Find Missing Values
i) Find data type of each column
j) Finding out Zero's
k) Find Mean age of patients
l) Now extract only Age, Sex, ChestPain, RestBP, Chol. Randomly divide the dataset in training (75%) and testing (25%).

Program:

```
import pandas as pd
df = pd.read_csv("/content/Heart.csv") print(df)
df.head(303)
shape = df.shape

print(shape)
print(df.notnull())
result = df.isna().sum()
print(result)
result = df.isna().sum().sum()
print(result)
datatypes = df.dtypes
print(datatypes)
c=(df==0).sum(axis=1)
print(c)
print(df['Age'].mean())
new_df = df.filter(["Age","Sex","ChestPain","RestBP","Chol"])
print(new_df)
```

OUTPUT:-

```
 Unnamed: 0  Age  Sex      ChestPain  ...  Slope  Ca        Thal  AHD
0           1   63    1        typical  ...      3 0.0       fixed   No
1           2   67    1  asymptomatic  ...      2 3.0      normal  Yes
2           3   67    1  asymptomatic  ...      2 2.0  reversable  Yes
3           4   37    1     nonanginal  ...      3 0.0      normal   No
4           5   41    0     nontypical  ...      1 0.0      normal   No
..        ...  ...  ...            ...  ...    ... ...       ... ...
298       299   45    1        typical  ...      2 0.0  reversable  Yes
299       300   68    1  asymptomatic  ...      2 2.0  reversable  Yes
300       301   57    1  asymptomatic  ...      2 1.0  reversable  Yes
301       302   57    0     nontypical  ...      2 1.0      normal  Yes
302       303   38    1     nonanginal  ...      1 NaN      normal   No

[303 rows x 15 columns]
```

	Unnamed: 0	Age	Sex	ChestPain	Rest	Chol	Fbs	RestECG	MaxHR	Exam	Old peak	Slope	Ca	Thal	AHD
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	299	45	1	typical	110	264	0	0	132	0	1.2	2	0.0	reversable	Yes
299	300	68	1	asymptomatic	144	193	1	0	141	0	3.4	2	2.0	reversable	Yes
300	301	57	1	asymptomatic	130	131	0	0	115	1	1.2	2	1.0	reversable	Yes
301	302	57	0	nontypical	130	236	0	2	174	0	0.0	2	1.0	normal	Yes
302	303	38	1	nonanginal	138	175	0	0	173	0	0.0	1	NaN	normal	No

303 rows × 15 columns

(303, 15)

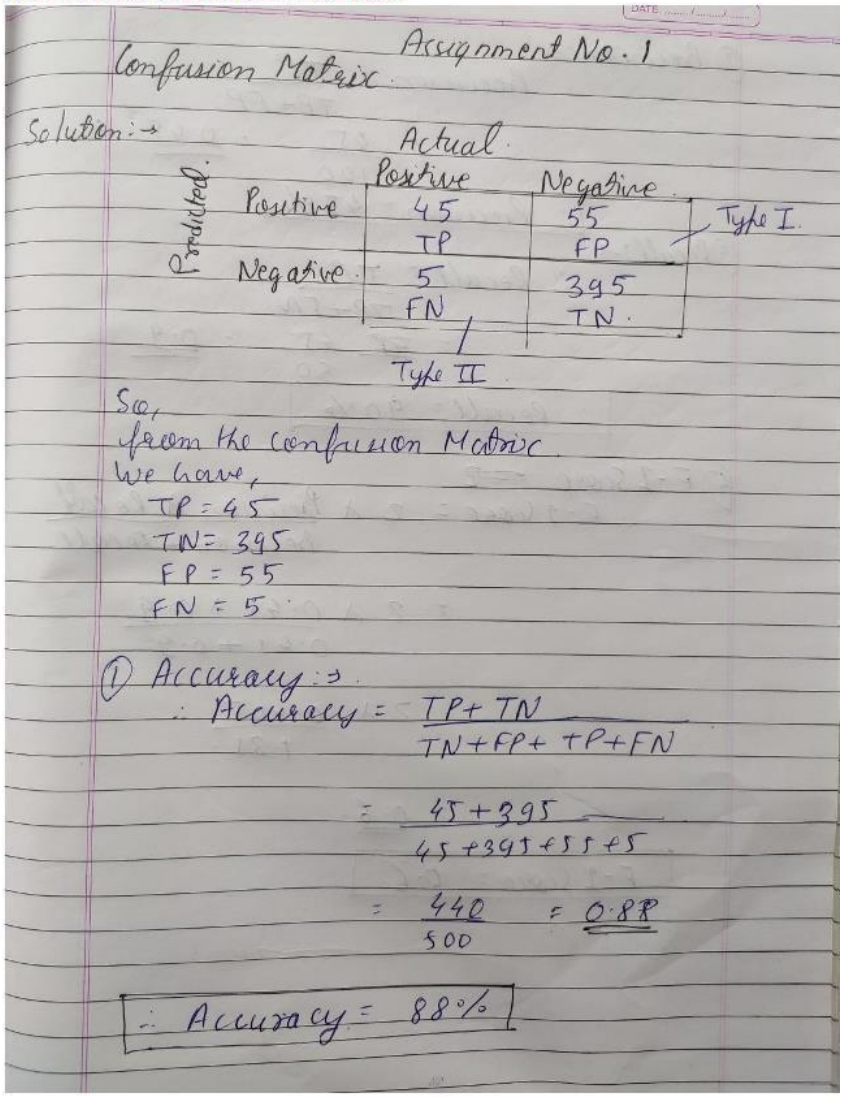
```
 Unnamed: 0  Age   Sex  ChestPain  ...  Slope    Ca  Thal   AHD
0        True  True  True       True  ...   True  True  True  True
1        True  True  True       True  ...   True  True  True  True
2        True  True  True       True  ...   True  True  True  True
3        True  True  True       True  ...   True  True  True  True
4        True  True  True       True  ...   True  True  True  True
..        ...   ...   ...        ...  ...    ...   ...   ...   ...
298      True  True  True       True  ...   True  True  True  True
299      True  True  True       True  ...   True  True  True  True
300      True  True  True       True  ...   True  True  True  True
301      True  True  True       True  ...   True  True  True  True
302      True  True  True       True  ...   True False  True  True

[303 rows x 15 columns]
```

```
Unnamed: 0   0
Age          0
Sex          0
ChestPain    0
RestBP       0
Chol         0
Fbs          0
RestECG      0
MaxHR        0
Exam         0
Old peak     0
Slope        0
Ca           4
Thal         2
AHD          0
dtype: int646

Unnamed: 0       int64
Age              int64
Sex              int64
ChestPain       object
RestBP           int64
Chol             int64
Fbs              int64
RestECG          int64
MaxHR            int64
Exam             int64
Old peak       float64
Slope            int64 Ca
               float64
Thal            object
AHD             object
dtype: object


0     2
1     1
2     1
3     4
4     4
     ..
298   4
299   2
300   2
301   4
302   4
Length: 303, dtype: int64

54.43894389438944

    Age  Sex     ChestPain RestBP Chol0
      63    1       typical     145  233
1     67    1  asymptomatic       160   286
2     67    1  asymptomatic       120   229
3     37    1    nonanginal       130  250
4     41    0    nontypical       130  204
..   ...  ...         ...     ...  ...
```

298	45	1	typical	110	264	
299	68	1	asymptomatic	144	193	
300	57	1	asymptomatic	130	131	
301	57	0	nontypical	130	236	
302	38	1	nonanginal	138	175	

[303 rows x 5 columns]

CONFUSION MATRIX SOLUTION:



Assignment No. 1

Confusion Matrix

Solution:→

Predicted \ Actual	Positive	Negative	
Positive	45 TP	55 FP	Type I
Negative	5 FN	395 TN	
	Type II		

So, from the confusion Matrix we have,

TP = 45
TN = 395
FP = 55
FN = 5

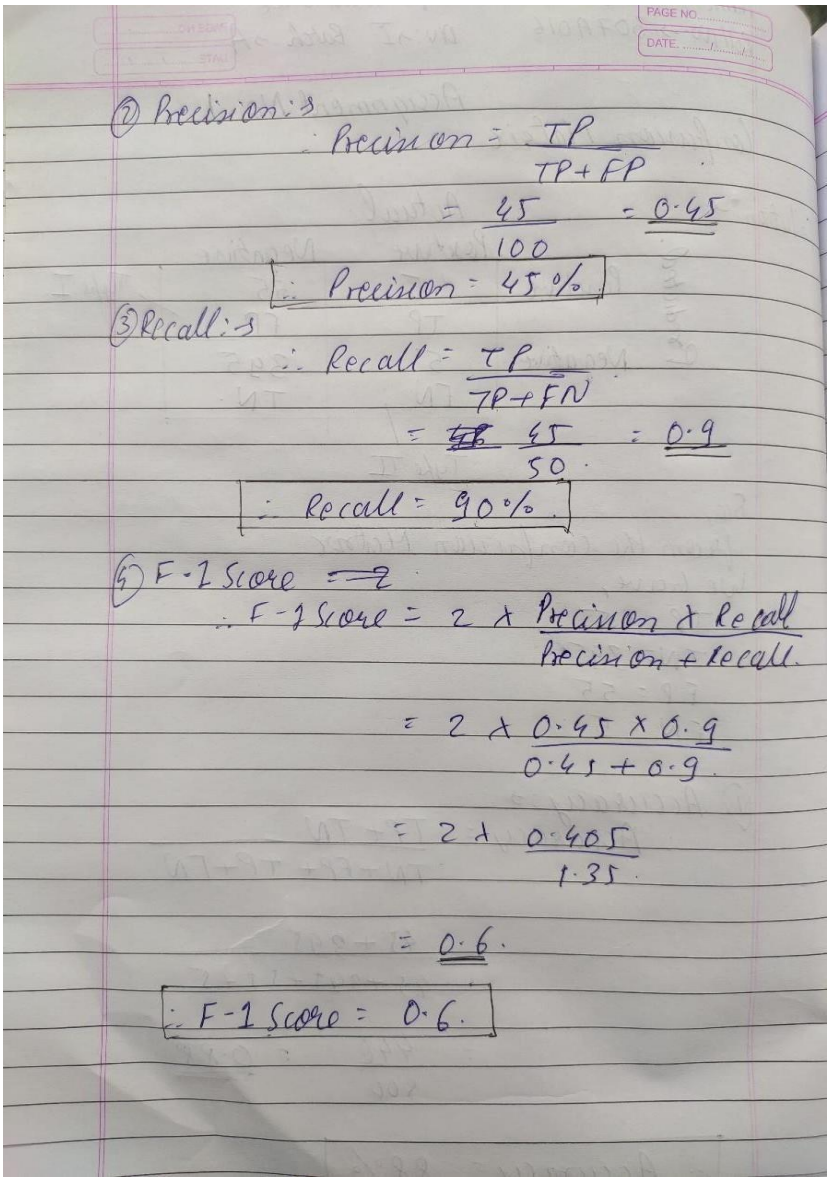
① Accuracy :→

$$\therefore \text{Accuracy} = \frac{TP + TN}{TN + FP + TP + FN}$$

$$= \frac{45 + 395}{45 + 395 + 55 + 5}$$

$$= \frac{440}{500} = 0.88$$

$\therefore$ Accuracy = 88%



② Precision :-

$$\therefore \text{Precision} = \frac{TP}{TP+FP}$$

$$= \frac{45}{100} = 0.45$$

$$\therefore \text{Precision} = 45\%$$

③ Recall :-

$$\therefore \text{Recall} = \frac{TP}{TP+FN}$$

$$= \frac{45}{50} = 0.9$$

$$\therefore \text{Recall} = 90\%$$

④ F-1 Score :-

$$\therefore \text{F-1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$= 2 \times \frac{0.45 \times 0.9}{0.45 + 0.9}$$

$$= 2 \times \frac{0.405}{1.35}$$

$$= 0.6$$

$$\therefore \text{F-1 Score} = 0.6$$

Assignment No. 2

Title: Regression Technique

Problem statement:

Download any freely available suitable dataset and perform the following steps to build regression model.

a) Apply Linear Regression using a suitable library function and perform prediction for test data.
b) Assess the performance of regression models using MSE, MAE and R-Square metrics
c) Visualize a simple regression model.

Objectives: In order to predict the value of the dependent variable for individuals for whom some information concerning the explanatory variables is available, or in order to estimate the effect of some explanatory variable on the dependent variable.Estimate the relationship between explanatory and response variable. Determine the effect of each of the explanatory variables on the response variable. Predict the value of the response variable for a given value of explanatory variable

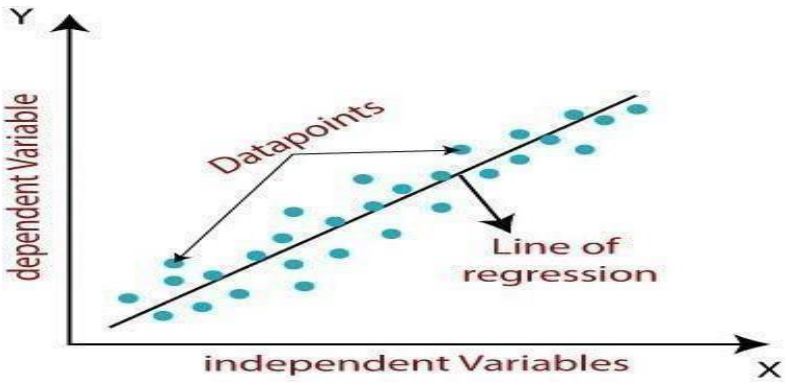
Theory:

Introduction to Regression

Regression is another important and broadly used statistical and machine learning tool. The key objective of regression-based tasks is to predict output labels or responses which are continuous numeric values, for the given input data. The output will be based on what the model has learned in the training phase. Basically, regression models use the input data features (independent variables) and their corresponding continuous numeric output values (dependent or outcome variables) to learn specific association between inputs and corresponding outputs.

Linear Regression

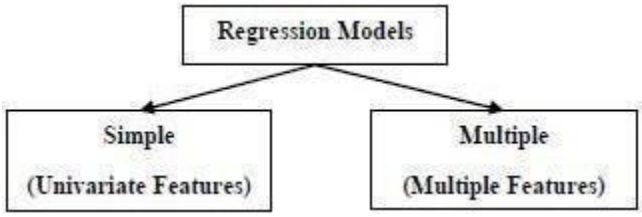
Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable we want to predict is called a dependent variable and the variable we used to predict is called an independent variable.
Linear regression was developed in field of statistics and is studied as model for understanding relationships between input and output variables



Types of Linear Regression Model

Linear regression can be further divided into two types of the algorithm:

a) Simple linear regression
b) Multiple linear regression



Simple linear regression − Simple linear algorithms have one dependent and independent variable; their relationship is a deterministic relationship if one variable can be expressed accurately by another. The main idea is to draw a line that best fits data i.e. all predicted points should lie on the best fit line. Error is distance between point and regression line Regression analysis is used to find equation that fits the data

The Simple Linear Regression model can be represented using the below equation:
equation has form of $y= a_0+a_1x+ \varepsilon$
Where,
a0= It is the intercept of the Regression line (can be obtained putting x=0)
a1= It is the slope of the regression line, which tells whether the line is increasing or decreasing.

ε = The error term. (For a good model it will be negligible)

Implementation of Simple Linear Regression Algorithm using Python

Problem Statement example for Simple Linear Regression:
Here we are taking a dataset that has two variables: salary (dependent variable) and experience (Independent variable). The goals of this problem is:

- We want to find out if there is any correlation between these two variables
- We will find the best fit line for the dataset.
- How the dependent variable is changing by changing the independent variable.

In this section, we will create a Simple Linear Regression model to find out the best fitting line for representing the relationship between these two variables. first step in finding Linear Regression equation.

Multiple linear regression − In this type of linear regression, we always attempt to discover the relationship between two or more independent variables or input and corresponding dependent variable or output. Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.

Some key points about MLR:

- For MLR, the dependent or target variable(Y) must be the continuous/real, but the predictor or independent variable may be of continuous or categorical form.
- Each feature variable must model the linear relationship with the dependent variable.
- MLR tries to fit a regression line through a multidimensional space of data-points.

MLR equation:

In Multiple Linear Regression, the target variable(Y) is a linear combination of multiple predictor variables x_1, x_2, x_3, ...,x_n. Since it is an enhancement of Simple Linear Regression, so the same is applied for the multiple linear regression equation, the equation becomes:

$$Y= b_0+b_1x_1+ b_2x_2+ b_3x_3+..... b_nx_n \quad \text{.....(a)}$$

Where,
Y= Output/Response variable
b_0, b_1, b_2, b_3 , b_n.....= Coefficients of the model.
x_1, x_2, x_3, x_4, ..= Various Independent/feature variable

Assumptions for Multiple Linear Regression:

- A linear relationship should exist between the Target and predictor variables.
- The regression residuals must be normally distributed.
- MLR assumes little or no multicollinearity (correlation between the independent variable) in data.

Implementation of Multiple Linear Regression model using Python:

To implement MLR using Python, we have below problem:
MLR is the dependent variable, and the other four variables are independent variables. Below are the main steps of deploying the MLR model:

1. Data Pre-processing Steps
2. Fitting the MLR model to the training set
3. Predicting the result of the test setApplications

The applications of ML regression algorithms are as follows −

- Forecasting or Predictive analysis
- Optimization
- Error correction
- Economics
- Finance
- Effectiveness of Independent variable on prediction:
- Predicting the impact of changes:

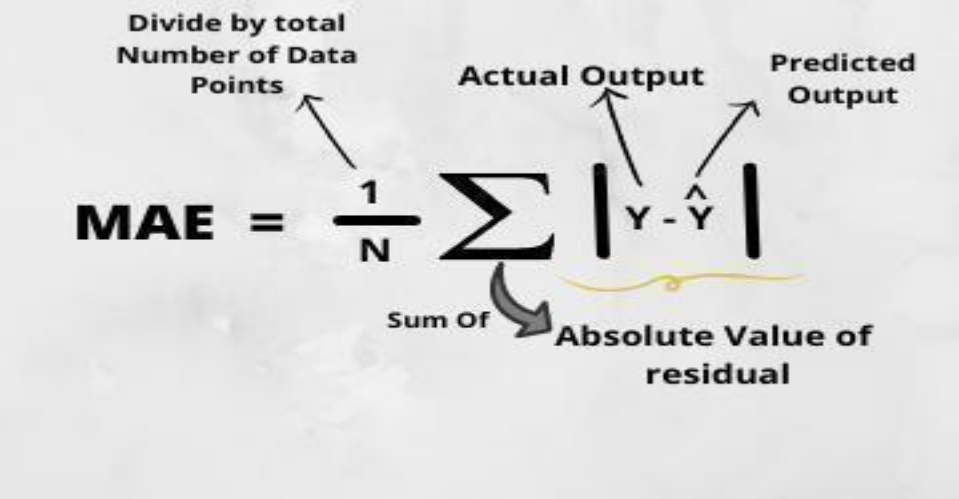
Evaluation Metrics in Regression Techniques
To understand the benefits and disadvantages of Evaluation metrics because different evaluation metrics fits on a different set of a dataset.
Now, I hope you get the importance of Evaluation metrics. Let's start understanding various evaluation metrics used for regression tasks.
In machine learning our main goal is to minimize the error which is defined by loss function. There are various loss functions like regression loss function, Mean Square Error, Mean Absolute Error.

1) Mean Absolute Error(MAE)

MAE is a very simple metric which calculates the absolute difference between actual and predicted values.

The MAE of your model which is basically a mistake made by the model known as an error. Now find the difference between the actual value and predicted value that is an absolute error but we have to find the mean absolute of the complete dataset.

so, sum all the errors and divide them by a total number of observations.




```
from sklearn.metrics import mean_absolute_error

print("MAE",mean_absolute_error(y_test,y_pred))
```

Advantages of MAE

- The MAE you get is in the same unit as the output variable.
- It is most Robust to outliers.

Disadvantages of MAE

- The graph of MAE is not differentiable so we have to apply various optimizers like Gradient descent which can be differentiable.

2) Mean Squared Error(MSE)

MSE is a most used and very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value. It represents the squared distance between actual and predicted values. We perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

```
from sklearn.metrics import mean_squared_error
print("MSE",mean_squared_error(y_test,y_pred))
```

Advantages of MSE

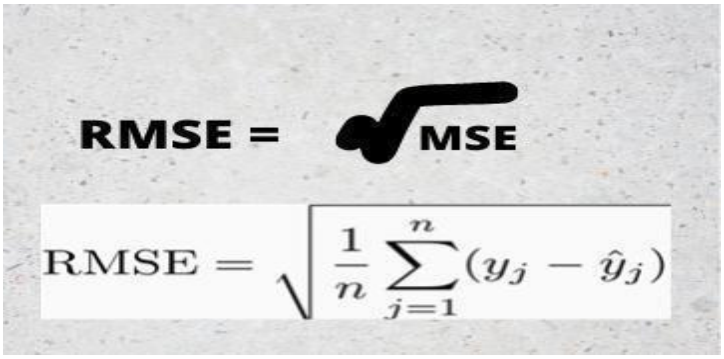
The graph of MSE is differentiable, so you can easily use it as a loss function.

Disadvantages of MSE

- The value you get after calculating MSE is a squared unit of output. for example, the output variable is in meter(m) then after calculating MSE the output we get is in meter squared.
- If you have outliers in the dataset then it penalizes the outliers most and the calculated MSE is bigger. So, in short, It is not Robust to outliers which were an advantage in MAE.

3) Root Mean Squared Error(RMSE)

As RMSE is clear by the name itself, that it is a simple square root of mean squared error.



for performing RMSE we have to NumPy NumPy square root function over MSE.

```
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
```

Advantages of RMSE

- The output value you get is in the same unit as the required output variable which makes interpretation of loss easy.

Disadvantages of RMSE

- It is not that robust to outliers as compared to MAE.

Most of the time people use RMSE as an evaluation metric and mostly when you are working with deep learning techniques the most preferred metric is RMSE.

4) RootMean Squared Log Error(RMSLE)

Taking the log of the RMSE metric slows down the scale of error. The metric is very helpful when you are developing a model without calling the inputs. In that case, the output will vary on a large scale.
To control this situation of RMSE we take the log of calculated RMSE error and resultant we get as RMSLE.
To perform RMSLE we have to use the NumPy log function over RMSE.

```
print("RMSE",np.log(np.sqrt(mean_squared_error(y_test,y_pred))))
```

It is a very simple metric that is used by most of the datasets hosted for Machine Learning competitions.

5) R Squared (R2)

R2 score is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform.
In contrast, MAE and MSE depend on the context as we have seen whereas the R2 score is independent of context.
So, with help of R squared we have a baseline model to compare a model which none of the other metrics provides. The same we have in classification problems which we call a threshold which is fixed at 0.5. So basically R2 squared calculates how the regression line is better than a mean line. Hence, R2 squared is also known as Coefficient of Determination or sometimes also known as Goodness of fit.

$$\text{R2 Squared} = 1 - \frac{SSr}{SSm}$$

SSr = Squared sum error of regression line

SSm = Squared sum error of mean line

R2 Squared

Value of R2 can be even negative when the model fitted is worse than the average fitted model.

```
from sklearn.metrics import r2_score
r2 = r2_score(y_test,y_pred)
print(r2)
```

6) Adjusted R Squared

The disadvantage of the R2 score is while adding new features in data the R2 score starts increasing or remains constant but it never decreases because It assumes that while adding more data variance of data increases.

Hence, To control this situation Adjusted R Squared came into existence.

$$R_a^2 = 1 - \left[\left(\frac{n-1}{n-k-1}\right) \times (1 - R^2)\right]$$

where:

n = number of observations

k = number of independent variables

R_a^2 = adjusted R^2

Now as K increases by adding some features so the denominator will decrease, n-1 will remain constant. R2 score will remain constant or will increase slightly so the complete answer will increase and when we subtract this from one then the resultant score will decrease. so this is the case when we add an irrelevant feature in the dataset.
And if we add a relevant feature then the R2 score will increase and 1-R2 will decrease heavily and the denominator will also decrease so the complete term decreases, and on subtracting from one the score increases.

```
n=40
k=2
adj_r2_score = 1 - ((1-r2)*(n-1)/(n-k-1))
```

p

Conclusion: The system integrates the task of developing a regression model from data, with the technique of searching for logical conditions that enable a better fitting error by the model.It continues to be a significant asset to many leading sectors starting from finance, education, banking, retail, medicine, media, etc.

Questions:

1. What is Regression?
2. What is a dependent variable
3. What are independent variables?
4. What is linear regression?
5. Which model is used for regression?
6. Which of the following metrics can be used for evaluating regression models?
7. What methods are used for the evaluation of regression models?

Name: Rohan Shinde

Roll No: 307B058

Division: 2

Batch: D

Assignment No: 2

Problem statement: Regression Technique

Download any freely available suitable dataset and perform the following steps to build regression model.
a) Apply Linear Regression using a suitable library function and perform prediction for test data.
b) Assess the performance of regression models using MSE, MAE and R-Square metrics
c) Visualize a simple regression model.

Program:

```
import matplotlib.pyplot as pltimport pandas as pd
import seaborn as sns
temp = pd.read_csv("/content/temperatures.csv")print(temp)
temp.head() temp.dtypes temp.columns temp.describe()
temp.isnull().sum()
print("Print the graph of top values.")n=int(input("Enter the value: "))
top_n_data = temp.nlargest(n, "ANNUAL")plt.figure(figsize=(10,10))
plt.title("Top temperature records")
sns.barplot(x=top_n_data.YEAR, y=top_n_data.ANNUAL)from sklearn import
linear_model, metrics a=temp[["YEAR"]]
b=temp[["JAN"]]
from sklearn.model_selection import train_test_split a_train,a_test,b_train,b_test =
train_test_split(a,b, test_size=0.2, random_state=1)
len(a_train)temp.shape
lr = linear_model.LinearRegression()print(a_train)
model=lr.fit(a_train, b_train)
```

```
r_sq=lr.score(a_train,b_train)model.intercept_
model.coef_ b_pred=model.predict(a_test)print(b_pred)
plt.scatter(a_train,b_train, color="cyan") plt.plot(a_train, lr.predict(a_train), color="red", line
width=1)
plt.title("Temp vs Year")plt.xlabel("Year")
plt.ylabel("Temperature")plt.show()
plt.scatter(a_test, b_test, color='blue') plt.plot(a_test, lr.predict(a_test), color='yellow', lin
ewidth=1)
plt.title("Temp vs Year")plt.xlabel("Year")
plt.ylabel("Temperature")plt.show()
```

OUTPUT:-

	YEAR	JAN	FEB	MAR	...	JAN-FEB	MAR-MAY	JUN-SEP	OCT-DEC
0	1901	22.40	24.14	29.07	...	23.27	31.46	31.27	27.25
1	1902	24.93	26.58	29.77	...	25.75	31.76	31.09	26.49
2	1903	23.44	25.03	27.83	...	24.24	30.71	30.92	26.26
3	1904	22.50	24.73	28.21	...	23.62	30.95	30.66	26.40
4	1905	22.00	22.83	26.68	...	22.25	30.00	31.33	26.57
..	...	...	...	...	...	...	...	...	...
112	2013	24.56	26.59	30.62	...	25.58	32.58	31.33	27.83
113	2014	23.83	25.97	28.95	...	24.90	31.82	32.00	27.81
114	2015	24.58	26.89	29.07	...	25.74	31.68	31.87	28.27
115	2016	26.94	29.72	32.62	...	28.33	34.57	32.28	30.03
116	2017	26.45	29.46	31.60	...	27.95	34.13	32.41	29.69

[117 rows x 18 columns]

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL
	AUG	SEP	OCT	NOV	DEC	ANNUAL	JAN-FEB	MAR-
	MAYJUN-SEP	OCT-DEC						
0	1901	22.40	24.14	29.07	31.91	33.41	33.18	31.21
	30.39	30.47	29.97	27.31	24.49	28.96	23.27	31.46
	31.27	27.25						
1	1902	24.93	26.58	29.77	31.78	33.73	32.91	30.92
	30.73	29.80	29.12	26.31	24.04	29.22	25.75	31.76
	31.09	26.49						
2	1903	23.44	25.03	27.83	31.39	32.91	33.00	31.34
	29.98	29.85	29.04	26.08	23.65	28.47	24.24	30.71
	30.92	26.26						
3	1904	22.50	24.73	28.21	32.02	32.64	32.07	30.36
	30.09	30.04	29.20	26.36	23.63	28.49	23.62	30.95
	30.66	26.40						

```
4    1905    22.00    22.83    26.68    30.01    33.32    33.25    31.44
     30.68    30.12    30.67    27.52    23.82    28.30    22.25    30.00
    31.33    26.57

YEAR          int64
JAN         float64
FEB         float64
MAR         float64
APR         float64
MAY         float64
JUN         float64
JUL         float64
AUG         float64
SEP         float64
OCT         float64
NOV         float64
DEC         float64
ANNUAL      float64
JAN-FEB     float64
MAR-MAY     float64
JUN-SEP     float64
OCT-DEC     float64
dtype: object

Index(['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG','SEP',
       'OCT', 'NOV', 'DEC', 'ANNUAL', 'JAN-FEB', 'MAR-MAY', 'JUN-SEP','OCT-DEC'],
      dtype='object')

YEAR    JAN    FEB    MAR    APR    MAY    JUN    JUL    AUG
        SEP    OCT    NOV    DEC    ANNUAL JAN-FEB MAR-MAY JUN-
        SEPOCT-DEC
count  117.000000          117.000000          117.000000          117.000000
       117.000000          117.000000          117.000000          117.000000
       117.000000          117.000000          117.000000          117.000000
       117.000000          117.000000          117.000000          117.000000
       117.000000          117.000000
mean   1959.000000         23.687436           25.597863           29.085983
       31.975812           33.565299           32.774274           31.035897
       30.507692           30.486752           29.766581           27.285470
       24.608291           29.181368           24.629573           31.517607
       31.198205           27.208120
std    33.919021           0.8345881.1507571.0684510.8894780.7249050.633132
       0.4688180.4763120.5442950.7054920.7145180.7826440.5555550.911239
       0.7405850.4205080.672003
min    1901.000000         22.000000           22.830000           26.680000
       30.010000           31.930000           31.100000           29.760000
       29.310000           29.070000           27.900000           25.700000
       23.020000           28.110000           22.250000           29.920000
       30.240000           25.740000
25%    1930.000000         23.100000           24.780000           28.370000
       31.460000           33.110000           32.340000           30.740000
       30.180000           30.120000           29.380000           26.790000
```

	24.040000	28.760000	24.110000	31.040000
	30.920000	26.700000		
50%	1959.000000	23.680000	25.480000	29.040000
	31.950000	33.510000	32.730000	31.000000
	30.540000	30.520000	29.780000	27.300000
	24.660000	29.090000	24.530000	31.470000
	31.190000	27.210000		
75%	1988.000000	24.180000	26.310000	29.610000
	32.420000	34.030000	33.180000	31.330000
	30.760000	30.810000	30.170000	27.720000
	25.110000	29.470000	25.150000	31.890000
	31.400000	27.610000		
max	2017.000000	26.940000	29.720000	32.620000
	35.380000	35.840000	34.480000	32.760000
	31.840000	32.220000	32.290000	30.110000
	28.010000	31.630000	28.330000	34.570000
	32.410000	30.030000		

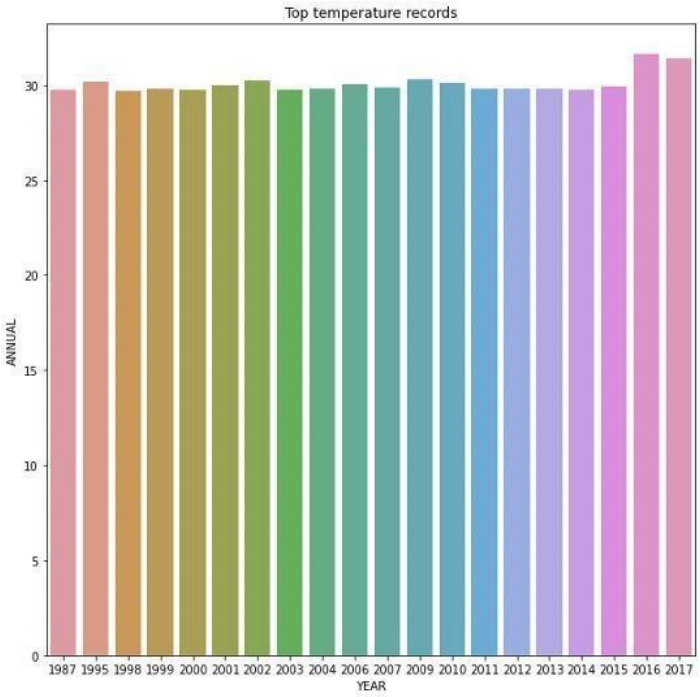
YEAR	0
JAN	0
FEB	0
MAR	0
APR	0
MAY	0
JUN	0
JUL	0
AUG	0
SEP	0
OCT	0
NOV	0
DEC	0
ANNUAL	0
JAN-FEB	0
MAR-MAY	0
JUN-SEP	0
OCT-DEC	0

dtype: int64

Print the graph of top values.Enter the

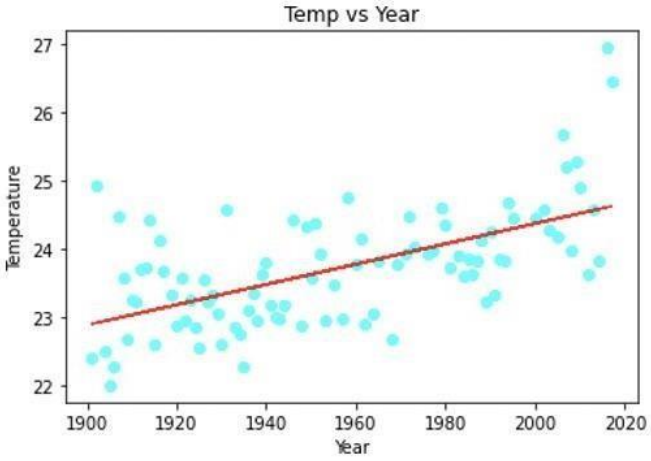
value: 20

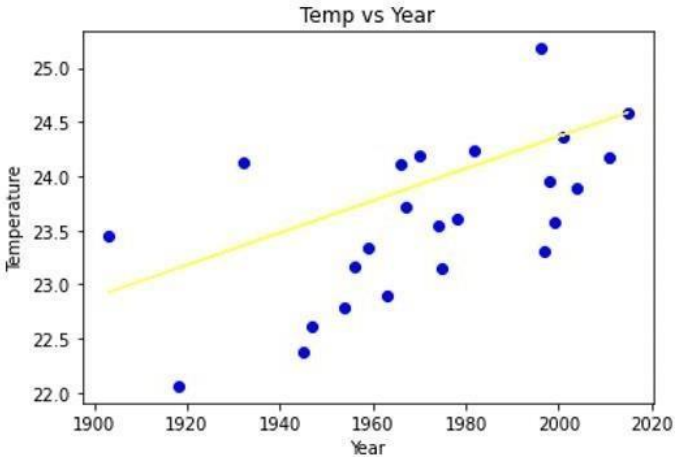
<matplotlib.axes._subplots.AxesSubplot at 0x7f6103ca9450>




```
93
(117, 18)
    YEAR
56  1957
94  1995
35  1936
38  1939
93  1994
..   ...
9   1910
72  1973
12  1913
107 2008
37  1938

[93 rows x 1 columns]
array([-5.35338281])
array([[0.01486008]])
[[23.92097555]
 [23.5791937 ]
 [23.75751466]
 [24.58967916]
 [23.98041587]
 [24.35191788]
 [23.35629249]
 [23.68321426]
 [23.86153523]
 [24.32219772]
 [24.30733764]
 [24.3370578 ]
 [22.92535016]
 [23.81695498]
 [24.53023884]
```

```
[23.71293442]
[24.42621828]
[24.38163804]
[23.87639531]
[23.54947354]
[24.03985619]
[23.14825137]
[24.09929651]
[23.99527595]]
```





Assignment No. 3

Title: Classification Technique

Problem Statement:

Data Set Available on kaggle (The last column of the dataset needs to be changed to 0 or 1)Data Set : https://www.kaggle.com/mohansacharya/graduate-admissions The counselor of the firm is supposed check whether the student will get an admission or not based on his/her GRE score and Academic Score. So to help the counselor to make appropriate decisions, build a machine learning model classifier using a Decision tree to predict whether a student will get admission or not.

A. Apply Data pre-processing (Label Encoding, Data Transformation….) techniques if necessary.
B. Perform data-preparation (Train-Test Split)
C. Apply Machine Learning Algorithm
D. Evaluate Model. 4

Objective:

The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data.

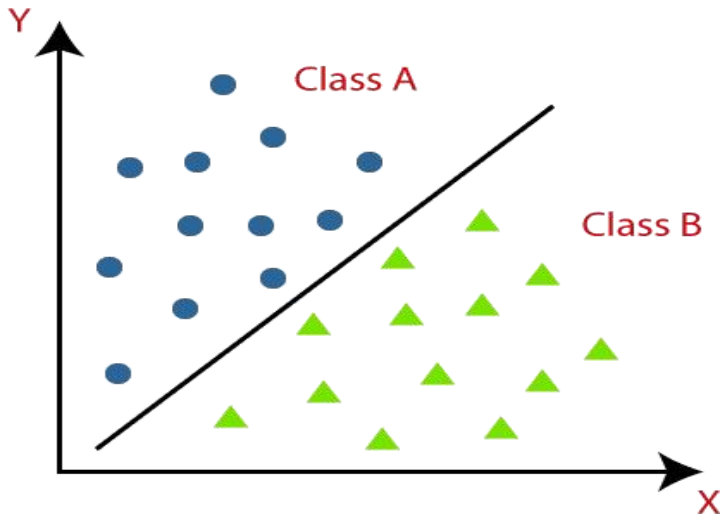
Theory:

Classification Algorithm

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data.
In the classification algorithm, a discrete output function(y) is mapped to input variable(x).

y=f(x), where y = categorical output

The best example of an ML classification algorithm is Email Spam Detector.
The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data.
Classification algorithms can be better understood using the below diagram. In the below diagram, there are two classes, class A and Class B. These classes have features that are similar to each other and dissimilar to other classes.



Types of ML Classification Algorithms:

Classification Algorithms can be further divided into the Mainly two category:

- Linear Models
  - Logistic Regression
  - Support Vector Machines
- Non-linear Models
  - K-Nearest Neighbors
  - Kernel SVM
  - Naïve Bayes
  - Decision Tree Classification
  - Random Forest Classification

Decision Tree Classification Algorithm

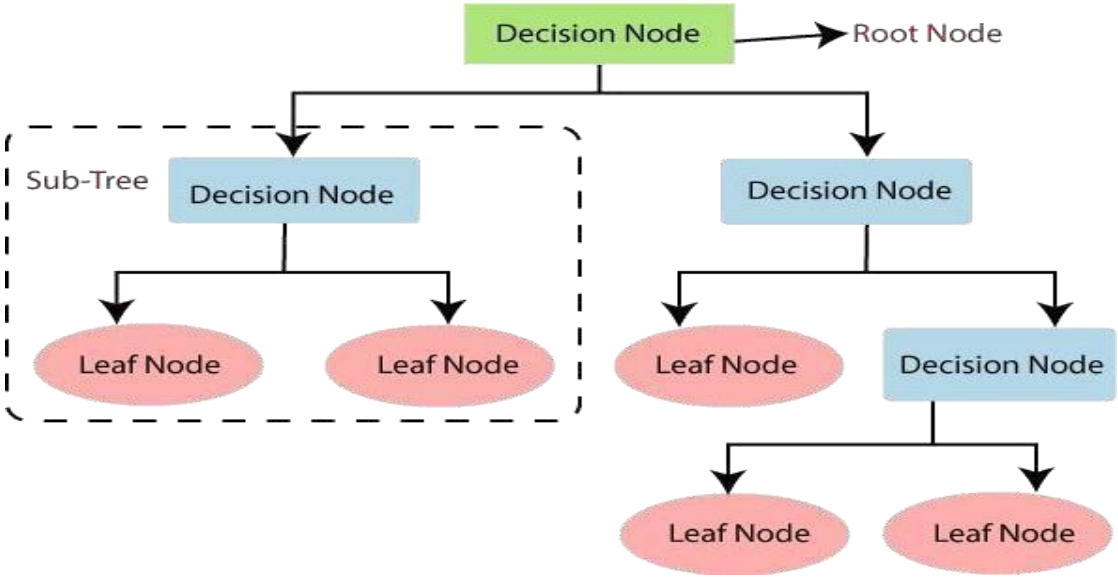
Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

A decision tree simply asks a question, and based on the answer (Yes/No), it further splits the tree into subtrees.
Below diagram explains the general structure of a decision tree:



Decision Trees

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
Branch/Sub Tree: A tree formed by splitting the tree.
Pruning: Pruning is the process of removing the unwanted branches from the tree.
Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

Decision Tree algorithm Work

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of the root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node. For the next node, the algorithm again compares the attribute value with the other sub-nodes and moves further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
- Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
- Step-3: Divide the S into subsets that contain possible values for the best attributes.
- Step-4: Generate the decision tree node, which contains the best attribute.
- Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and call the final node as a leaf node.

Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- Information Gain
- Gini Index

Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

  Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:
Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)
Where,

- S= Total number of samples
- P(yes)= probability of yes
- P(no)= probability of no

Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

$$\text{Gini Index} = 1 - \sum_j P_j$$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.
A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree pruning technology used:

- Cost Complexity Pruning
- Reduced Error Pruning.

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
- For more class labels, the computational complexity of the decision tree may increase.

Python Implementation of Decision Tree

Now we will implement the Decision tree using Python. Steps will also remain the same, which are given below:

- Data Preprocessing step
- Fitting a Decision-Tree algorithm to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.

Conclusion: A Classification is a method of predicting similar information from categorical or numerical datasets. Nowadays machine learning algorithms are becoming more popular for classification problems .It gives an introduction to most of the popular machine learning algorithms used for classification of pattern recognition.

Questions:

1. What types of Classification Algorithms do you know?
2. What are Decision Trees?
3. What type of node is considered Pure?
4. How are the different nodes of decision trees represented?
5. What are some advantages of using Decision Trees?
6. What is gini index and how is it used in decision tree

Name: Rohan Shinde

Roll No: 307B058

Division: 2

Batch: D

Assignment No: 3

Problem statement: Classification Technique

Data Set Available on kaggle (The last column of the dataset needs to be changed to 0 or 1)Data Set : https://www.kaggle.com/mohansacharya/graduate-admissions The counselor of the firm is supposed check whether the student will get an admission or not based on his/her GRE score and Academic Score. So to help the counselor to make appropriate decisions, build a machine learning model classifier using a Decision tree to predict whether a student will get admission or not.

A. Apply Data pre-processing (Label Encoding, Data Transformation….) techniques if necessary.
B. Perform data-preparation (Train-Test Split)
C. Apply Machine Learning Algorithm
D. Evaluate Model.

Code:

```
import numpy as np import pandas
as pd import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression gre =
pd.read_csv("/content/Admission_Predict.csv")gre.head()
print(gre) gre.describe
gre=gre.rename(columns={"TOEFL Score":"TOEFL"})
gre=gre.rename(columns={"University Rating":"univ_rating"})
gre=gre.rename(columns={"Serial No.":"ID"}) gre=gre.rename(columns={"Chance of
Admit ":"Admission chance"})gre.columns
sns.pairplot(data=gre) plt.figure(figsize=(12,12))
sns.lineplot(data=gre, x="GRE Score",y="TOEFL", hue="Research")plt.figure(figsize=(12,12))
h = sns.scatterplot(data=gre, x="CGPA", y="ID", hue="Research
```

```
S = sns.FacetGrid(data = gre, col="Research", row="univ_rating", hue="LOR ")s.map(sns.scatterplot,
"CGPA","ID")
s.add_legend()
sns.catplot(data=gre, x="Research", y="Admission chance", hue="univ_rating")plt.figure(figsize=(12,12))
sns.heatmap(gre.corr(), annot=True,cmap="cubehelix") from
sklearn.model_selection import train_test_split a =
gre.drop("Admission chance", axis =1)
b = gre["Admission chance"]a
b
a_train,a_test,b_train,b_test = train_test_split(a,b, test_size=0.2, random_state=1)
from sklearn import linear_model
lr = linear_model.LinearRegression()
lr.fit(a_train,b_train)
pred = lr.predict(a_test)
from sklearn.metrics import classification_report, confusion_matrix, r2_scorelr.intercept_
coef = pd.DataFrame(lr.coef_, a.columns, columns = ['coef'])coef
r2 = r2_score(b_test, pred)
print("r2 score:" ,r2) predd =
pd.DataFrame(pred) frames =
[gre,predd]
pred_values = pd.concat(frames, axis=1)
pred_values.columns
pred_values.head()
```

OUTPUT:-

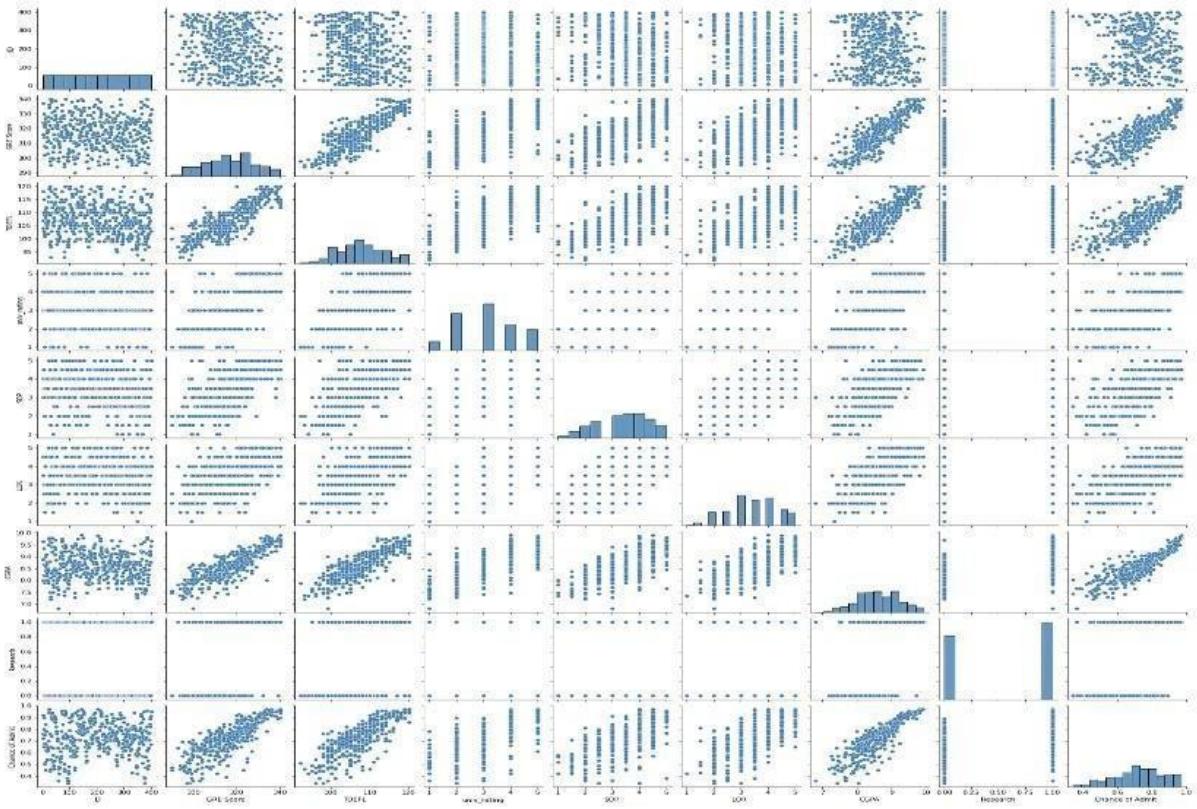
	Serial No.	GRE Score	TOEFL Score		University Rating		SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92		
1	2	324	107	4	4.0	4.5	8.87	1	0.76		
2	3	316	104	3	3.0	3.5	8.00	1	0.72		
3	4	322	110	3	3.5	2.5	8.67	1	0.80		
4	5	314	103	2	2.0	3.0	8.21	0	0.65		

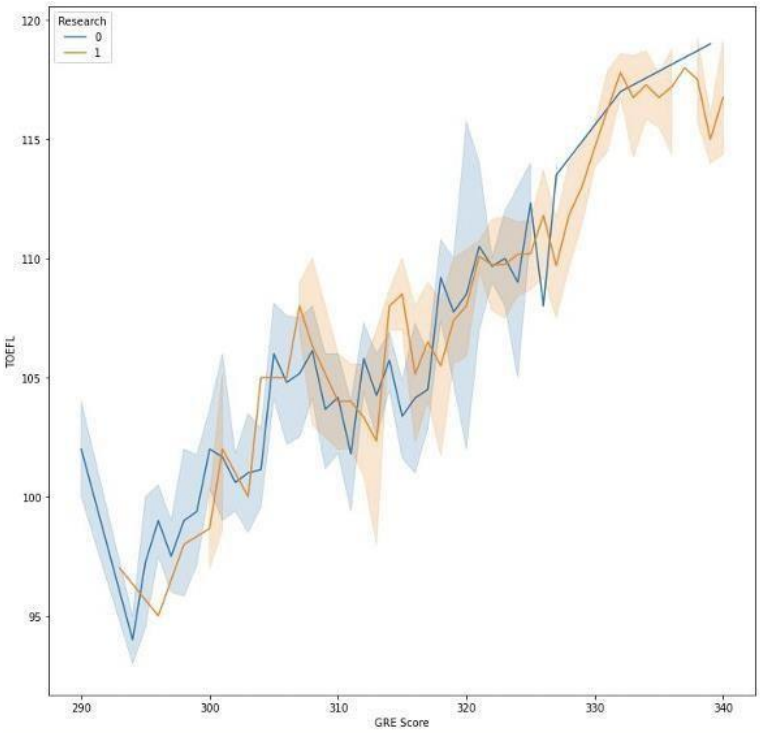
	Serial No.	GRE Score	TOEFL Score	...	CGPA	Research	Chance of Admit
0	1	337	118	...	9.65	1	0.92
1	2	324	107	...	8.87	1	0.76
2	3	316	104	...	8.00	1	0.72
3	4	322	110	...	8.67	1	0.80
4	5	314	103	...	8.21	0	0.65
..	..	...	...	..	...	...	
	.		...	.			
395	396	324	110	...	9.04	1	0.82
396	397	325	107	...	9.11	1	0.84
397	398	330	116	...	9.45	1	0.91
398	399	312	103	...	8.78	0	0.67
399	400	333	117	...	9.66	1	0.95

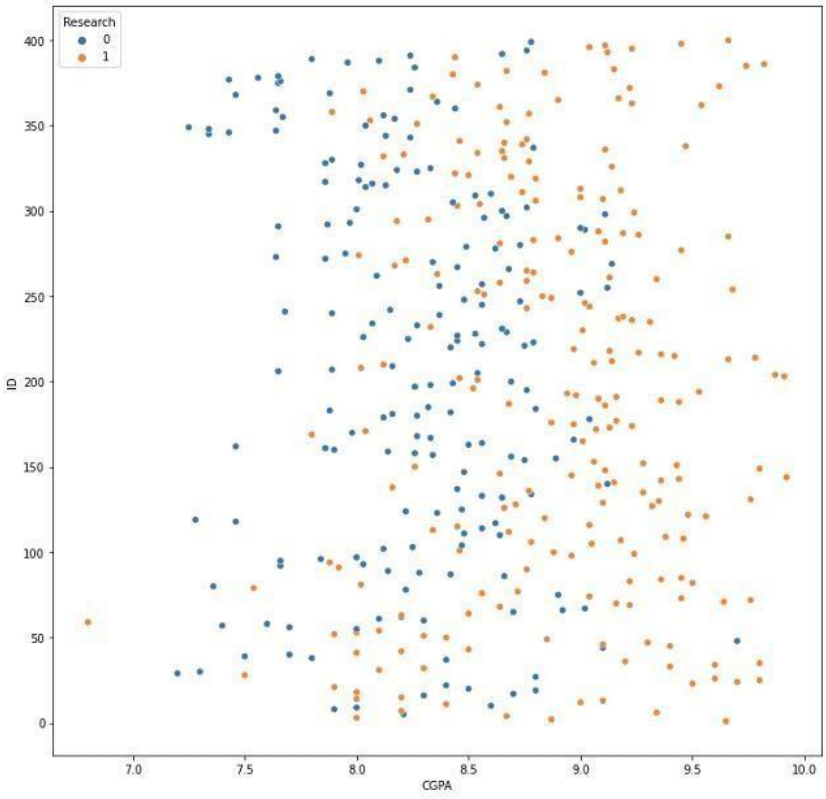
[400 rows x 9 columns]

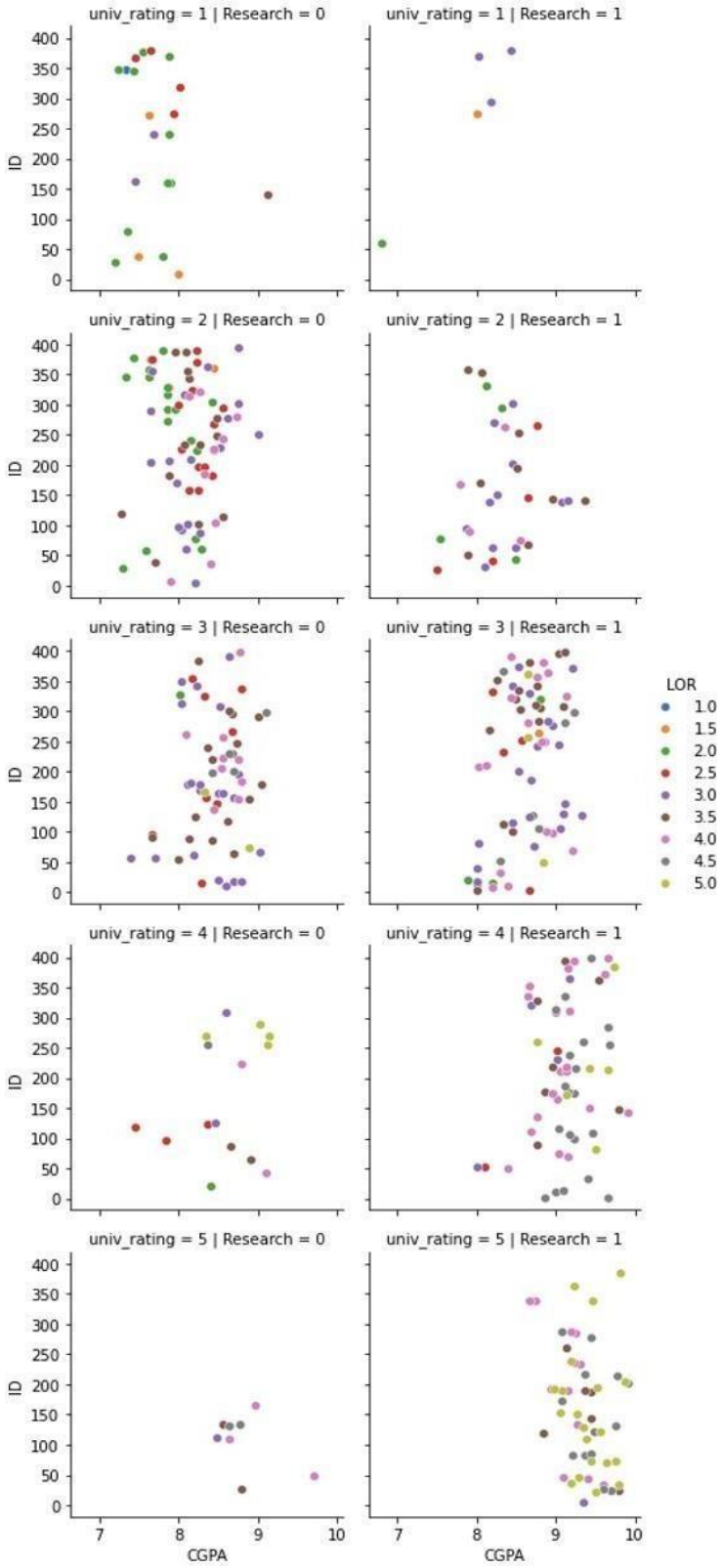
```
<bound method NDFrame.describe of      Serial No.  GRE Score  TOEFL Score  ...  CGPA  Research  Chance of Admit
0             1        337          118  ...  9.65         1              0.92
1             2        324          107  ...  8.87         1              0.76
2             3        316          104  ...  8.00         1              0.72
3             4        322          110  ...  8.67         1              0.80
4             5        314          103  ...  8.21         0              0.65
..          ...        ...          ...  ...   ...       ...               ...
395         396        324          110  ...  9.04         1              0.82
396         397        325          107  ...  9.11         1              0.84
397         398        330          116  ...  9.45         1              0.91
398         399        312          103  ...  8.78         0              0.67
399         400        333          117  ...  9.66         1              0.95

[400 rows x 9 columns]>
```

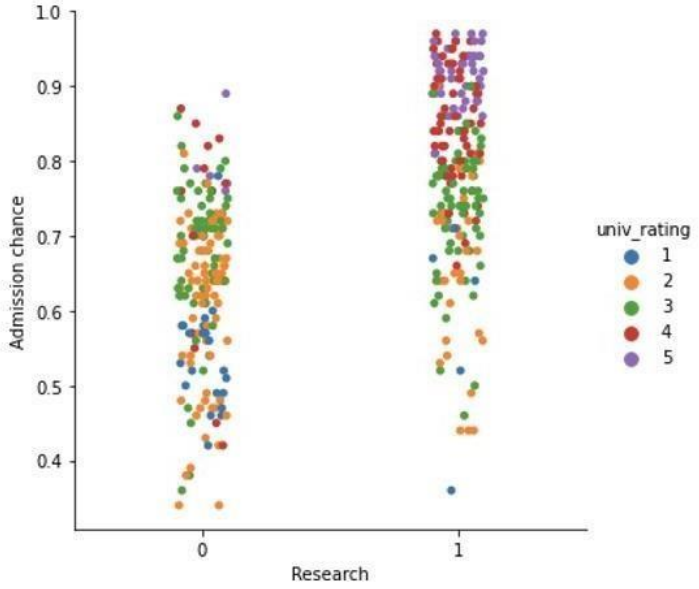
```
Index(['ID', 'GRE Score', 'TOEFL', 'univ  rating', 'SOP', 'LOR ', 'CGPA',
       'Research', 'Admission chance'],
      dtype='object')
```

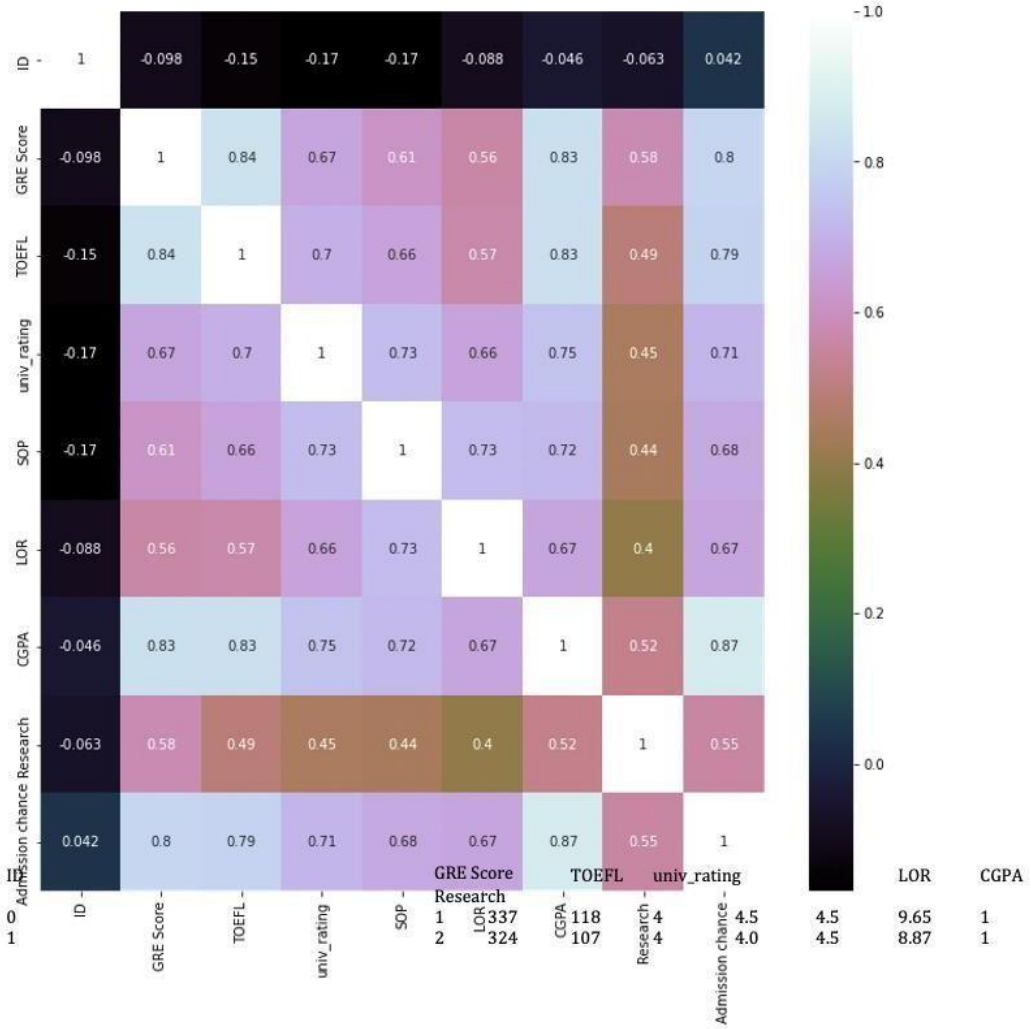



Research
0
1
TOEFL
GRE Score
120
115
110
105
100
95
290
300
310
320
330
340


Research
0
1
ID
CGPA
400
350
300
250
200
150
100
50
0
7.0
7.5
8.0
8.5
9.0
9.5
10.0


univ_rating = 1 | Research = 0
univ_rating = 1 | Research = 1
univ_rating = 2 | Research = 0
univ_rating = 2 | Research = 1
univ_rating = 3 | Research = 0
univ_rating = 3 | Research = 1
univ_rating = 4 | Research = 0
univ_rating = 4 | Research = 1
univ_rating = 5 | Research = 0
univ_rating = 5 | Research = 1
ID
CGPA
LOR
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Admission chance
Research
univ_rating
1
2
3
4
5


ID
GRE Score
TOEFL
univ_rating
SOP
LOR
CGPA
Research
Admission chance

2	3	316	104	3	3.0	3.5	8.00	1
3	4	322	110	3	3.5	2.5	8.67	1
4	5	314	103	2	2.0	3.0	8.21	0
...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1
396	397	325	107	3	3.0	3.5	9.11	1
397	398	330	116	4	5.0	4.5	9.45	1
398	399	312	103	3	3.5	4.0	8.78	0
399	400	333	117	4	5.0	4.0	9.66	1

400 rows × 8 columns

```
0    0.92
1    0.76
2    0.72
3    0.80
4    0.65
    ...
395  0.82
396  0.84
397  0.91
398  0.67
399  0.95
Name: Admission chance, Length: 400, dtype: float64
```

```
LinearRegression()
```

```
-1.2259579064813528
```

	coef
ID	0.000153
GRE Score	0.001605
TOEFL	0.003856
univ_rating	0.010967
SOP	-0.002175
LOR	0.021974
CGPA	0.102750
Research	0.025801

```
r2 score: 0.8277102420849569
```

```
Index([          'ID',     'GRE Score',         'TOEFL',
       'univ  rating',          'SOP',         'LOR ',
            'CGPA',     'Research', 'Admission chance',
                0],
      dtype='object')
```

	ID	GRE Score	TOEFL	univ_rating		SOP	LOR	
	CGPA	Research		Admission chance	0			
0	1	337	118	4	4.5	4.5	9.65	1
	0.92	0.748292						
1	2	324	107	4	4.0	4.5	8.87	1
	0.76	0.670461						
2	3	316	104	3	3.0	3.5	8.00	1
	0.72	0.815210						
3	4	322	110	3	3.5	2.5	8.67	1
	0.80	0.809353						
4	5	314	103	2	2.0	3.0	8.21	0
	0.65	0.850375						

Assignment No. 4

Title: Clustering Techniques

Problem Statement:

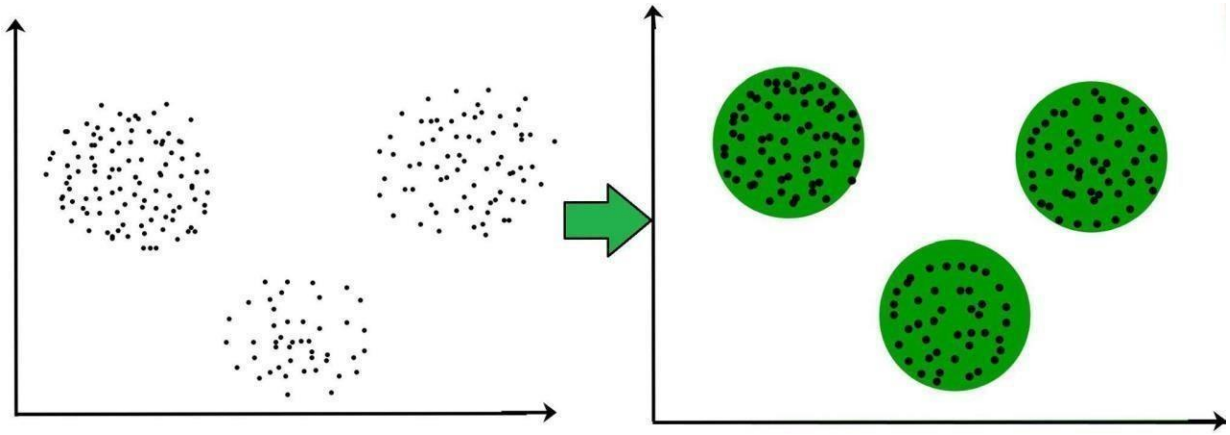
Download the following customer dataset from below link: Data Set: https://www.kaggle.com/shwetabh123/mall-customers
This dataset gives the data of Income and money spent by the customers visiting a Shopping Mall. The data set contains Customer ID, Gender, Age, Annual Income, Spending Score. Therefore, as a mall owner you need to find the group of people who are the profitable customers for the mall owner. Apply at least two clustering algorithms (based on Spending Score) to find the group of customers.
a. Apply Data pre-processing (Label Encoding, Data Transformation….) techniques if necessary.
b. Perform data-preparation(Train-Test Split)
c. Apply Machine Learning Algorithm
d. Evaluate Model.
e. Apply Cross-Validation and Evaluate Model

Objective:

The goal of clustering is to find distinct groups or "clusters" within a data set. Using a machine language algorithm, the tool creates groups where items in a similar group will, in general, have similar characteristics to each other.In K-Means, each cluster is associated with a centroid. The main objective of the K-Means algorithm is to minimize the sum of distances between the points and their respective cluster centroid

Theory:

Introduction to Clustering
It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labeled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.
Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.
The data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.



Clustering is very much important as it determines the intrinsic grouping among the unlabelled data present. There are no criteria for good clustering. It depends on the user, what are the criteria they may use which satisfy their need. For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding "natural clusters" and describe their unknown properties ("natural" data types), in finding useful and suitable groupings ("useful" data classes) or in finding unusual data objects (outlier detection). This algorithm must make some assumptions that constitute the similarity of points and each assumption makes different and equally valid clusters.

Clustering Methods :

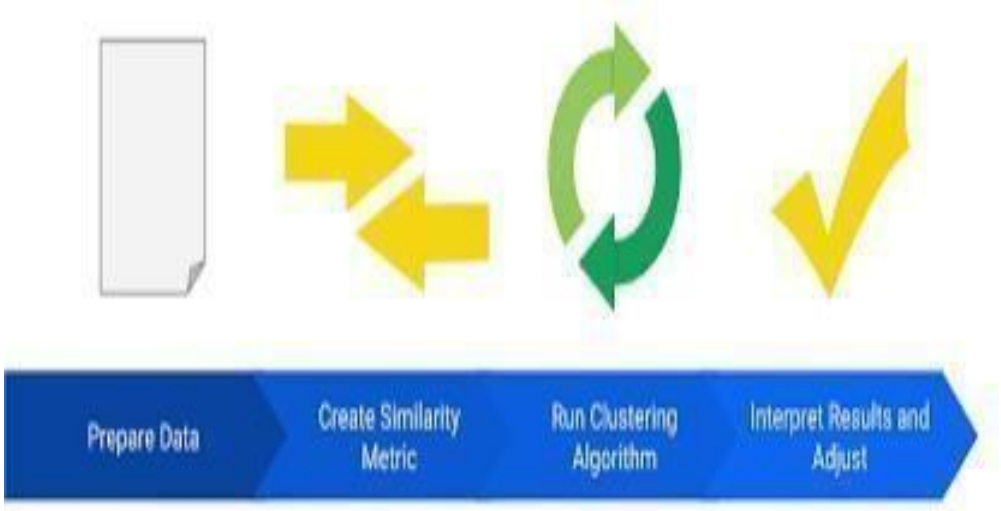
- **Density-Based Methods:** These methods consider the clusters as the dense region having some similarities and differences from the lower dense region of the space. These methods have good accuracy and the ability to merge two clusters. Example DBSCAN (Density-Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points to Identify Clustering Structure), etc.
- **Hierarchical Based Methods:** The clusters formed in this method form a tree-type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two category
    - **Agglomerative** (bottom-up approach)
    - **Divisive** (top-down approach)
- **Partitioning Methods:** These methods partition the objects into k clusters and each partition forms one cluster. This method is used to optimize an objective criterion similarity function such as when the distance is a major parameter example K-means, CLARANS (Clustering Large Applications based upon Randomized Search), etc.
- **Grid-based Methods:** In this method, the data space is formulated into a finite number of cells that form a grid-like structure. All the clustering operations done on these grids are fast and independent of the number of data objects, for example STING (Statistical Information Grid), wave cluster, CLIQUE (CLustering In Quest), etc.

Clustering Workflow

To cluster your data, you'll follow these steps:

1. Prepare data.
2. Create a similarity metric.
3. Run clustering algorithm.
4. Interpret results and adjust your clustering.

This page briefly introduces the steps. We'll go into depth in subsequent sections.



Prepare Data

As with any ML problem, you must normalize, scale, and transform feature data. While clustering however, you must additionally ensure that the prepared data lets you accurately calculate the similarity between examples. The next sections discuss this consideration.

Create Similarity Metric

Before a clustering algorithm can group data, it needs to know how similar pairs of examples are. You quantify the similarity between examples by creating a similarity metric. Creating a similarity metric requires you to carefully understand your data and how to derive similarity from your features.

Run Clustering Algorithm

A clustering algorithm uses the similarity metric to cluster data. This course focuses on k-means.

Interpret Results and Adjust

Checking the quality of your clustering output is iterative and exploratory because clustering lacks “truth” that can verify the output. You verify the result against expectations at the cluster-

level and the example-level. Improving the result requires iteratively experimenting with the previous steps to see how they affect the clustering.

Finding K value

We will see 2 methods to find the K value for K-Means.

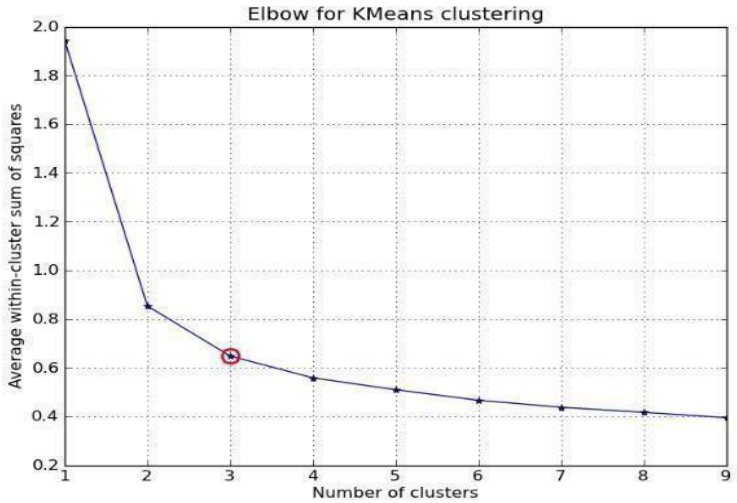
1. Elbow Method
2. Silhouette Method

Elbow Method

Elbow Method: This is one of the most popular methods that is used to find K for K-Means.For this, we have to learn something known as WSS(Within the sum of squares).
WSS: The WSS is defined as the sum of squared distance between each member of the cluster and its centroid.

$$WSS = \sum_{i=1}^{M} d(p_i, q^{(i)})^2 = \sum_{i=1}^{M} \sum_{j=1}^{n} \left(p_{ij} - q_j^{(i)}\right)^2$$

Where
p(i)=data point
q(i)=closest centroid to the data point
Here in the elbow method, the K value is chosen after the decrease of WSS is almost constant.



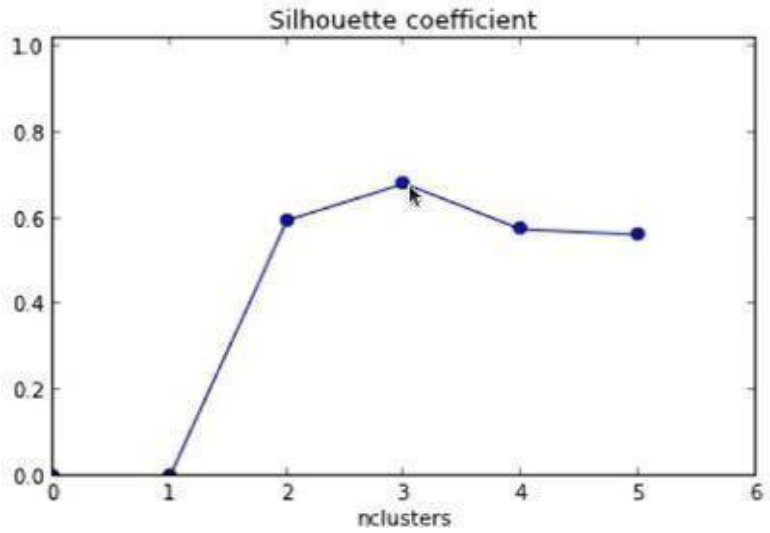
In the above picture, you can see the elbow point, which is 3. After that point, WSS is almost constant. So 3 is selected as K.In this way elbow method is used for finding the value of K

Silhouette Method

Silhouette Method: Here in the silhouette method, we will compute the silhouette score for every point.

Silhouette Coefficient for the point= (b-a)/max(a,b)

Where
a=mean intra-cluster distance
b=mean nearest cluster distance
Silhouette coefficient for dataset = Mean Silhouette Coefficient over points.
If we draw a graph for these points, we will get something like this.



So here we can see the highest silhouette coefficient is for K = 3. In this way, the silhouette method is used for finding K. But in the Silhouette method, there are some chances of getting overfitted to some extent. Because by increasing the number of clusters, the size of clusters becomes small and distance between other clusters may decrease and finally leads to overfitting sometimes. But in a lot of cases, it works well.
It is available in scikit learn library.

sklearn.metrics.silhouette_score

Run the Clustering Algorithm

K- means Clustering Algorithm :

K-means clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. This algorithm groups unabled dataset into different clusters.
The K-means algorithm defines the number of predefined clusters that need to be created in process i.e. for K=2 .There will be two clusters for k=n, there will be 'n' clusters. Here the dataset which is divided belongs to only one group that has similar properties.
To cluster data into k clusters, k-means follows the steps below:

Step One

The algorithm randomly chooses a centroid for each cluster. In our example, we choose a k of 3, and therefore the algorithm randomly picks 3 centroids.

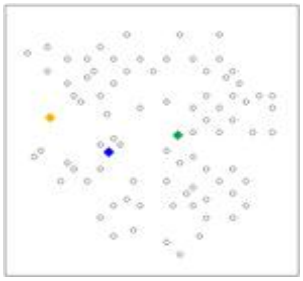


Figure 1: k-means at initialization

Step Two

The algorithm assigns each point to the closest centroid to get k initial clusters.

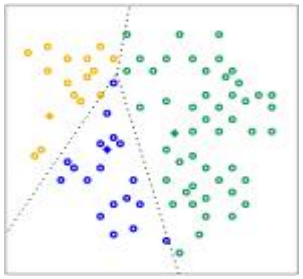


Figure 2: Initial clusters

Step Three

For every cluster, the algorithm recomputes the centroid by taking the average of all points in the cluster. The changes in centroids are shown in Figure 3 by arrows. Since the centroids change, the algorithm then re-assigns the points to the closest centroid. Figure 4 shows the new clusters after reassignment.

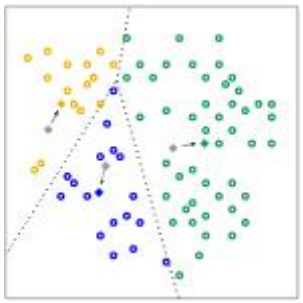


Figure 3: Recomputation of centroids

Step Four

The algorithm repeats the calculation of centroids and assignment of points until points stop changing clusters. When clustering large datasets, you stop the algorithm before reaching convergence, using other criteria instead.

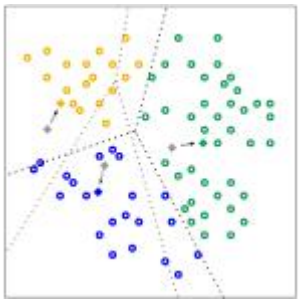


Figure 4: Clusters after reassignment.

Interpret Results and Adjust Clustering

Because clustering is unsupervised, no "truth" is available to verify results. The absence of truth complicates assessing quality. Further, real-world datasets typically do not fall into obvious clusters of examples like the dataset shown in Figure 1.

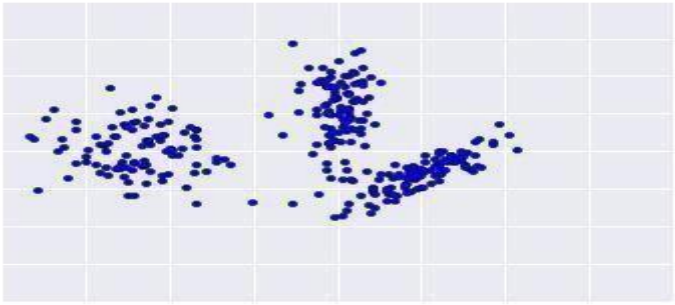


Figure 1: An ideal data plot; real-world data rarely looks like this.

real-world data looks more like Figure 2, making it difficult to visually assess clustering quality.

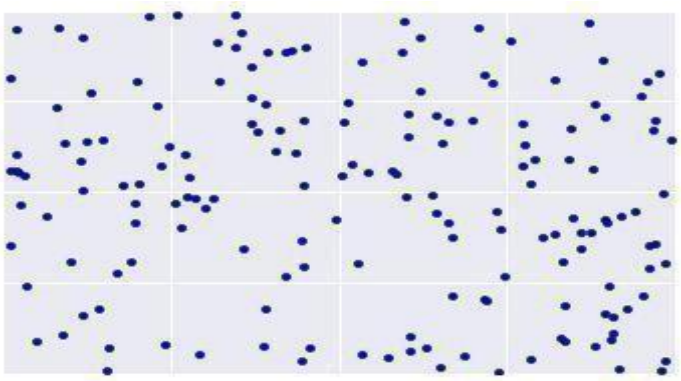
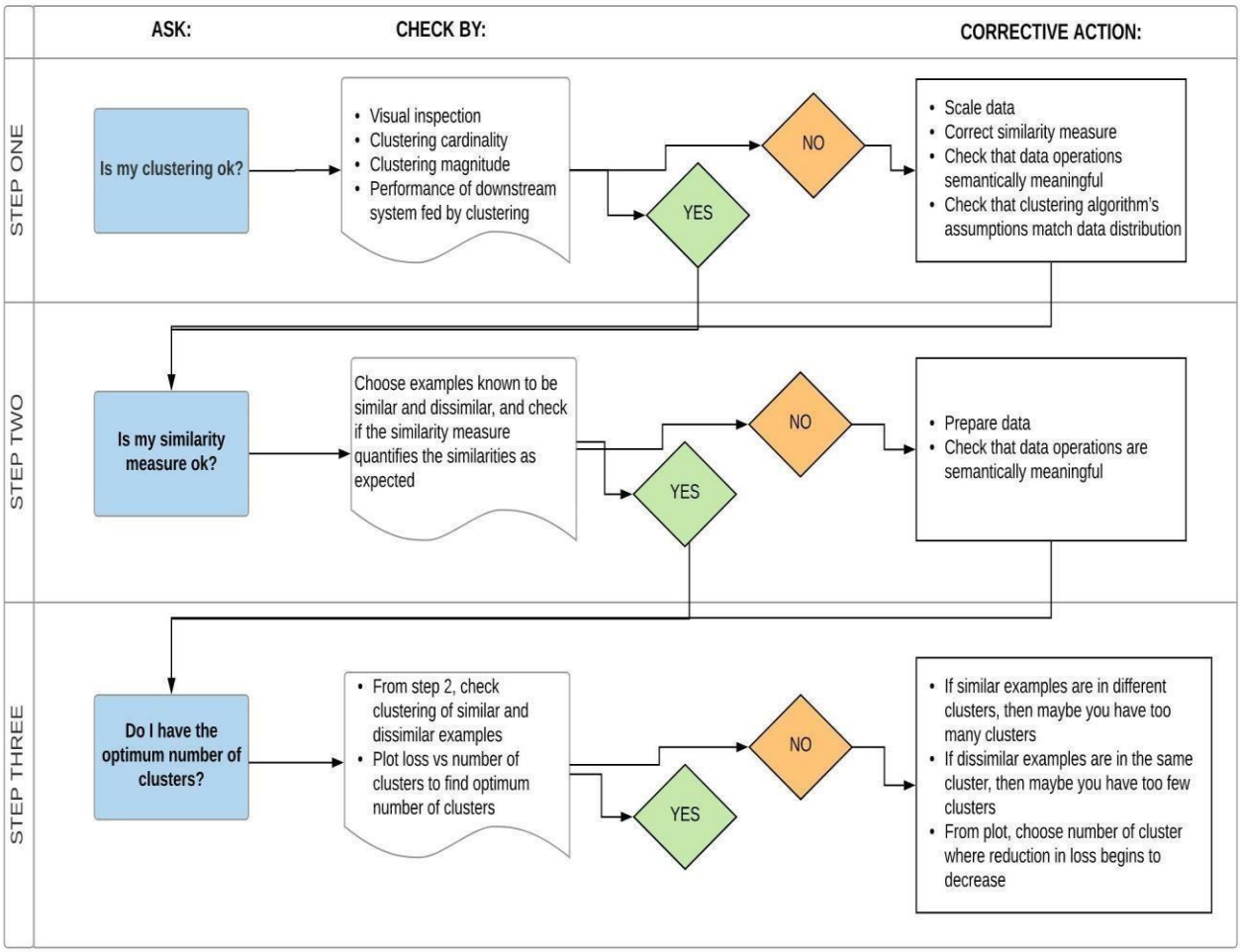


Figure 2: A true-to-life data plot

The flowchart below summarizes how to check the quality of your clustering. We'll expand upon the summary in the following sections.



Step One: Quality of Clustering

Checking the quality of clustering is not a rigorous process because clustering lacks “truth”. Here are guidelines that you can iteratively apply to improve the quality of your clustering.
First, perform a visual check that the clusters look as expected, and that examples that you consider similar do appear in the same cluster. Then check these commonly-used metrics as described in the following sections:

- Cluster cardinality
- Cluster magnitude
- Performance of downstream system

Cluster cardinality
Cluster cardinality is the number of examples per cluster. Plot the cluster cardinality for all clusters and investigate clusters that are major outliers. For example, in Figure 2, investigate cluster number 5.

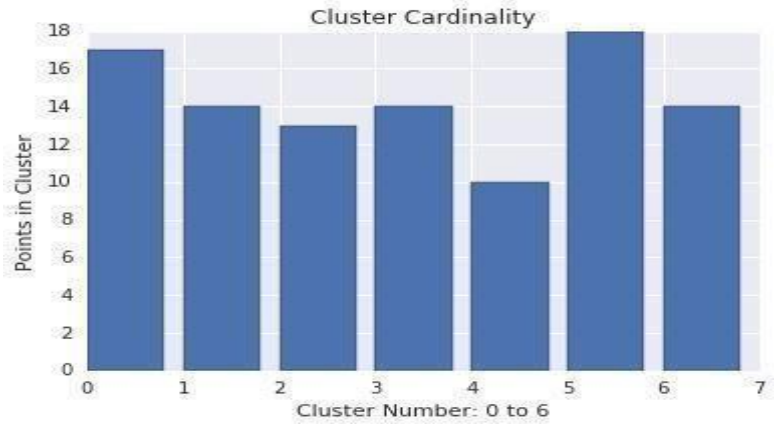


Figure 2: Cardinality of several clusters

Cluster magnitude
Cluster magnitude is the sum of distances from all examples to the centroid of the cluster. Similar to cardinality, check how the magnitude varies across the clusters, and investigate anomalies. For example, in Figure 3, investigate cluster number 0.

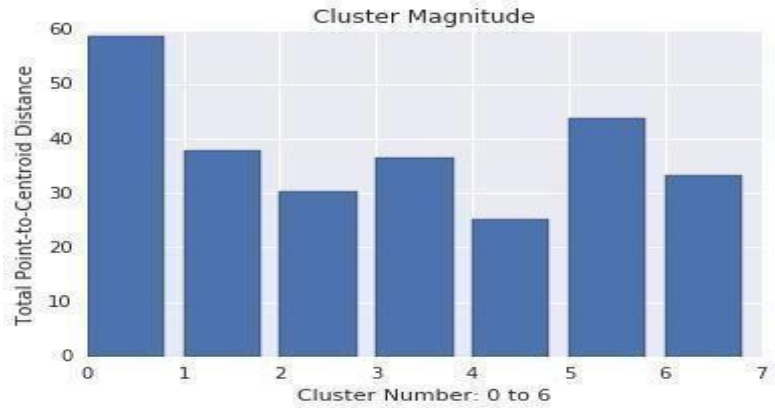


Figure 3: Magnitude of several clusters

Magnitude vs. Cardinality

Notice that a higher cluster cardinality tends to result in a higher cluster magnitude, which intuitively makes sense. Clusters are anomalous when cardinality doesn't correlate with magnitude relative to the other clusters. Find anomalous clusters by plotting magnitude against cardinality. For example, in Figure 4, fitting a line to the cluster metrics shows that cluster number 0 is anomalous.

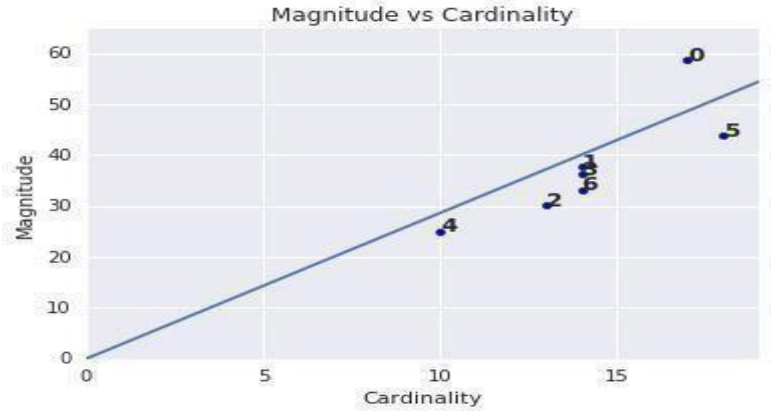


Figure 4: Cardinality vs. Magnitude of several clusters.

Performance of Downstream System

Since clustering output is often used in downstream ML systems, check if the downstream system's performance improves when your clustering process changes. The impact on your downstream performance provides a real-world test for the quality of your clustering. Thedisadvantage is that this check is complex to perform.

Step Two: Performance of the Similarity Measure

Your clustering algorithm is only as good as your similarity measure. Make sure your similarity measure returns sensible results. The simplest check is to identify pairs of examples that are known to be more or less similar than other pairs. Then, calculate the similarity measure for each pair of examples. Ensure that the similarity measure for more similar examples is higher than thesimilarity measure for less similar examples.

The examples you use to spot check your similarity measure should be representative of the data set. Ensure that your similarity measure holds for all your examples. Careful verification ensures that your similarity measure, whether manual or supervised, is consistent across your dataset. If your similarity measure is inconsistent for some examples, then those examples will not be clustered with similar examples.

Step Three: Optimum Number of Clusters

k-means requires you to decide the number of clusters k beforehand. How do you determine the optimal value of k. Try running the algorithm for increasing k and note the sum of cluster magnitudes. As k increases, clusters become smaller, and the total distance decreases. Plot this distance against the number of clusters. As shown in Figure 4, at a certain k the reduction in loss becomes marginal with increasing k Mathematically, that's roughly the k where the slope crosses

above -1 ($\theta > 135^\circ$). This guideline doesn't pinpoint an exact value for the optimum k but only an approximate value. For the plot shown, the optimum k is approximately 11. If you prefer more granular clusters, then you can choose a higher k using this plot as guidance.

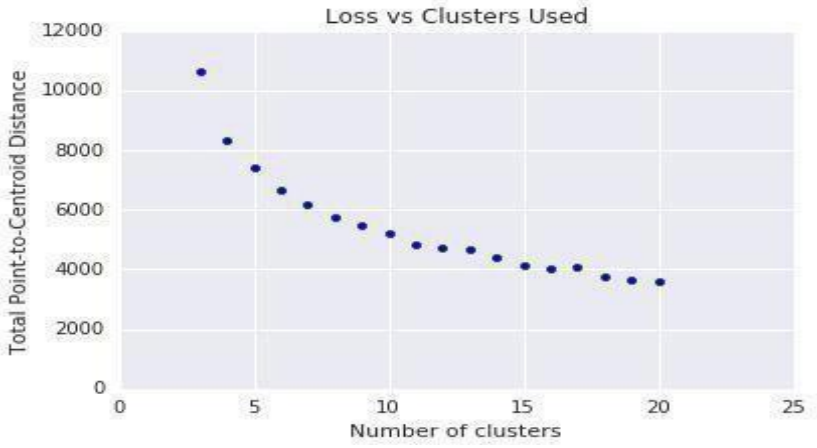


Figure 4: Loss vs. number of clusters

k-Means Advantages and Disadvantages

Advantages ofk-means

Relatively simple to implement.
Scales to large data sets.
Guarantees convergence.
Can warm-start the positions of centroids.
Easily adapts to new examples.
Generalizes to clusters of different shapes and sizes, such as elliptical clusters.

Disadvantages of k-means

Choosing k manually.
Use the "Loss vs. Clusters" plot to find the optimal (k)
Being dependent on initial values.
For a low kyou can mitigate this dependence by running k-means several times with different initial values and picking the best result. As k increases, you need advanced versions of k-means to pick better values of the initial centroids (called k-means seeding).
Clustering data of varying sizes and density.
k-means has trouble clustering data where clusters are of varying sizes and density.
Clustering outliers.
Centroids can be dragged by outliers, or outliers might get their own cluster instead of being ignored. Consider removing or clipping outliers before clustering.
Scaling with a number of dimensions.

Conclusion:

We have learned about the K Means clustering algorithm from scratch to implementation. First, we have seen clustering and then finding K value in K Means. For that, we have seen two different methods. Overall we have seen the importance of the K Means clustering algorithm. Finally, we had implemented the code for this algorithm in the Jupyter notebook.

Questions:

1. What is clustering?
2. Where is clustering used in real life?
3. When to use k-means vs K medians?
4. Which methods belong to clustering?
5. What is the difference between classification and clustering?
6. Which clustering algorithm is best?
7. Which algorithm is used by clustering in machine learning?

Name: Rohan Shinde

Roll No: 307B058

Division: 2

Batch: D

Assignment No: 4

Problem statement: **:** Clustering Techniques

Problem Statement:

Download the following customer dataset from below link: Data Set: https://www.kaggle.com/shwetabh123/mall-customers
This dataset gives the data of Income and money spent by the customers visiting a Shopping Mall. The data set contains Customer ID, Gender, Age, Annual Income, Spending Score. Therefore, as a mall owner you need to find the group of people who are the profitable customers for the mall owner. Apply at least two clustering algorithms (based on Spending Score) to find the group of customers.

1 Apply Data pre-processing (Label Encoding, Data Transformation….) techniques if necessary.
2 Perform data-preparation(Train-Test Split)
3 Apply Machine Learning Algorithm
4 Evaluate Model.
5 Apply Cross-Validation and Evaluate Model

Code:

```
import numpy as np import
pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt from
matplotlib.lines import Line2Dimport
seaborn as sns
from sklearn.preprocessing import StandardScalerfrom
sklearn.decomposition import PCA
from sklearn.cluster import KMeans

cust = pd.read_csv("/content/Mall_Customers.csv")cust.shape
cust.head()
cust.info()
cust.rename(columns = {"Genre":"Gender"}, inplace = True)
cust.describe()
cust.drop(labels = 'CustomerID', axis = 1 , inplace = True)cust.isnull().sum()
cust.dtypes cust['Gender'].value_counts()
cust["Gender"].replace({"Male":1, "Female":0}, inplace = True) sns.heatmap(data =
cust.corr(), annot = True, fmt = ".2f", cmap = "cividis_r")font = {"family":"Sherif",
"size":16}
 plt.subplots_adjust(left = 1, bottom = 1,right = 2.5, top = 2, wspace = 0.5, hspace =None)


  plt.subplot(1,2,1)
  plt.pie(x = [len(cust[cust.Gender == 1]) , len(cust[cust.Gender == 0])] , labels =
  ['Male' , 'Female'], shadow = True , startangle = -
  30 , explode = [0.1,0] , autopct = '%.0f%%')
  plt.title("Customers gender", fontdict = font)

  plt.subplot(1,2,2)

  male_avg_score = cust[cust.Gender == 1]['Spending Score (1-100)'].mean()
  female_avg_score = cust[cust.Gender == 0]['Spending Score (1-100)'].mean()

  plt.bar(x = ['Male' , 'Female'] , height = [male_avg_score , female_avg_score]
```

```
 , color = ['tab:cyan' , 'tab:green']) plt.title('Customers
spending score' , fontdict = font)plt.ylabel('Average
spending score' , fontdict = font)plt.xlabel('Gender' ,
fontdict = font)

plt.text(-0.3 , 40 , 'Average = {:.2f}'.format(male_avg_score))
plt.text(0.7 , 40 , 'Average = {:.2f}'.format(female_avg_score))

plt.show()

age_list =cust.Age.unique()
age_list.sort()
avg_list = []


for age in age_list:
    avg_list.append(cust[cust.Age == age]['Spending Score (1-100)'].mean())

plt.plot(age_list,avg_list)
plt.xlabel('Age' , fontdict = font)
plt.ylabel('Average spending score' , fontdict = {'family':'serif' , 'size':14
})
plt.title('Spending score in different ages')

plt.plot([20,70] , [40,40] , linestyle = '--' , c = 'tab:green' , alpha = 0.8)

plt.plot([35,35] , [10,90] , linestyle = '--' , c = 'tab:red' , alpha = 0.8)plt.text(31,7,'Age = 35')
plt.show()
sc = StandardScaler()
data_scaled =sc.fit_transform(cust)
pca = PCA(n_components = 2)
data_pca = pca.fit_transform(data_scaled)
print("data shape after PCA :",data_pca.shape)
wcss_list = []
for i in range(1, 15):
    kmeans = KMeans(n_clusters = i , init = 'k-means++' , random_state = 1)
    kmeans.fit(data_pca)
    wcss_list.append(kmeans.inertia_)

plt.plot(range(1,15) , wcss_list)
```

```
plt.plot([4,4] , [0 , 500] , linestyle = '--' , alpha = 0.7)
plt.text(4.2 , 300 , 'Elbow = 4')



plt.xlabel('K' , fontdict = font)
plt.ylabel('WCSS' , fontdict = font)
plt.show()
kmeans = KMeans(n_clusters = 4 , init = 'k-means++' , random_state = 1)
kmeans.fit(data_pca)
cluster_id = kmeans.predict(data_pca)
result_data = pd.DataFrame()
result_data['PC1'] = data_pca[:,0]
result_data['PC2'] = data_pca[:,1]
result_data['ClusterID'] = cluster_id
cluster_colors = {0:'tab:red' , 1:'tab:green' , 2:'tab:blue' , 3:'tab:pink'}
cluster_dict = {'Centroid':'tab:orange','Cluster0':'tab:red' , 'Cluster1':'tab
:green'
                    , 'Cluster2':'tab:blue' , 'Cluster3':'tab:pink'}


plt.scatter(x = result_data['PC1'] , y = result_data['PC2'] , c = result_data[
'ClusterID'].map(cluster_colors))
handles = [Line2D([0], [0], marker='o', color='w', markerfacecolor=v, label=k,
  markersize=8) for k, v in cluster_dict.items()]
plt.legend(title='color', handles=handles, bbox_to_anchor=(1.05, 1), loc='upper
left')

plt.scatter(x = kmeans.cluster_centers_[:,0] , y = kmeans.cluster_centers_[:,1
] , marker = 'o' , c = 'tab:orange', s = 150 , alpha = 1)

plt.title("Clustered by KMeans" , fontdict = font)
plt.xlabel("PC1" , fontdict = font) plt.ylabel("PC2"
, fontdict = font)
```

OUTPUT:-

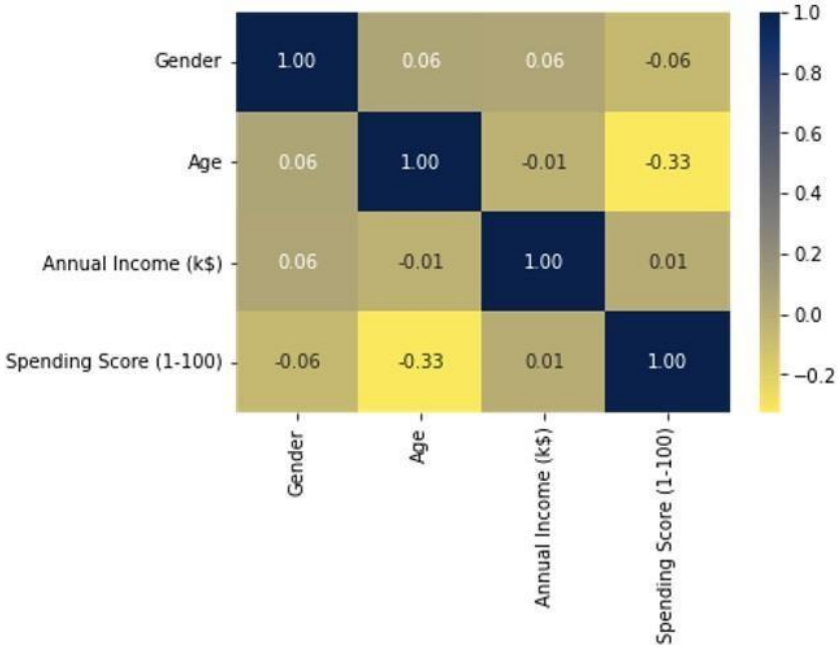
(200, 5)

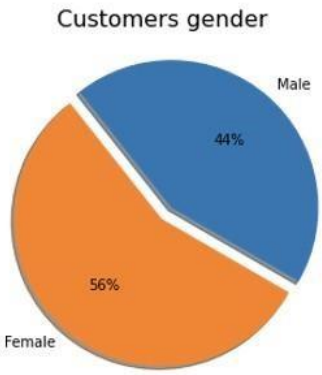
CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

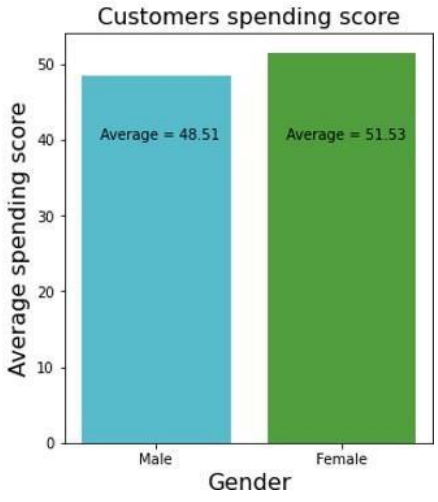
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #  Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0  CustomerID              200 non-null    int64
 1  Genre                   200 non-null    object
 2  Age                     200 non-null    int64
 3  Annual Income (k$)      200 non-null    int64
 4  Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

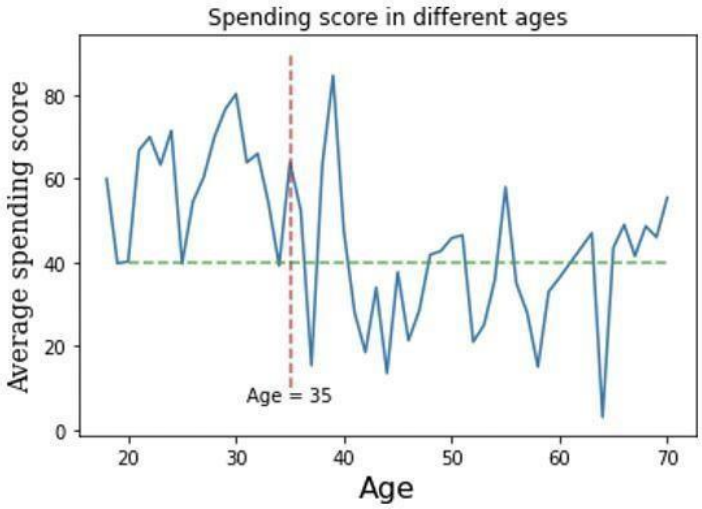
CustomerID	Age	Annual Income (k$)	Spending Score (1-100)	
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

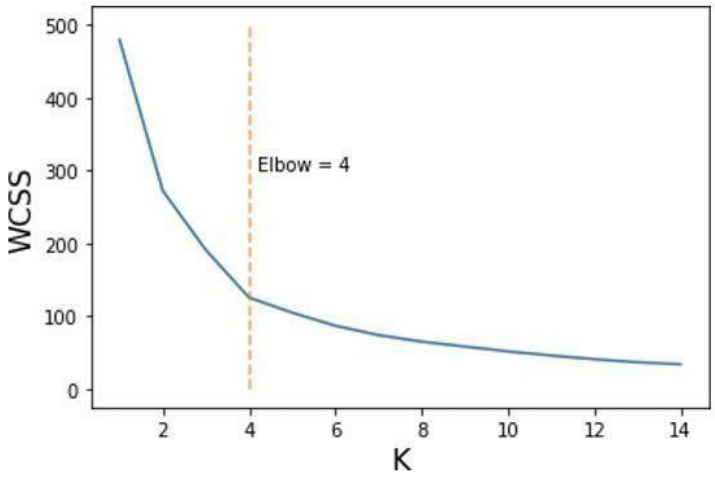
```
Gender                  0
Age                     0
Annual Income (k$)      0
Spending Score (1-100)  0
dtype: int64

Gender                  object
Age                     int64
Annual Income (k$)      int64
Spending Score (1-100)  int64
dtype: object

Female    112
Male       88
Name: Gender, dtype: int64
```

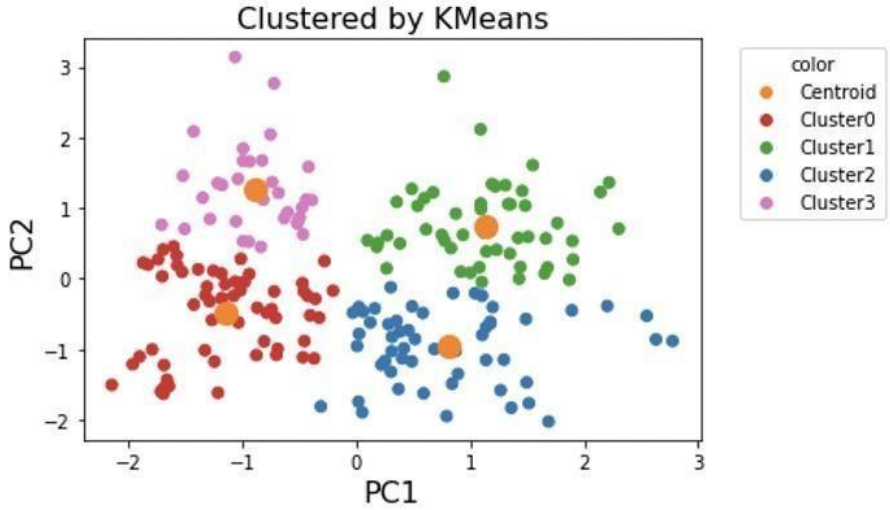







Spending score in different ages
Average spending score
Age = 35
Age


WCSS
Elbow = 4
K


Clustered by KMeans
PC2
PC1
color
Centroid
Cluster0
Cluster1
Cluster2
Cluster3

Assignment No.5

Title: Association Rule Learning

Problem Statement:
Download Market Basket Optimization dataset from below link.
Data Set: https://www.kaggle.com/hemanthkumar05/market-basket-optimization
This dataset comprises the list of transactions of a retail company over the period of one week. It contains a total of 7501 transaction records where each record consists of the list of items sold in one transaction. Using this record of transactions and items in each transaction, find the association rules between items.
There is no header in the dataset and the first row contains the first transaction, so mentioned header = None here while loading dataset.
a. Follow following steps:
b. Data Preprocessing
c. Generate the list of transactions from the dataset
d. Train Apriori algorithm on the dataset
e. Visualize the list of rules
F. Generated rules depend on the values of hyper parameters. By increasing the minimum confidence value and find the rules accordingly

Objective:

Association rule learning is a rule-based machine learning method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using some measures of interestingness.

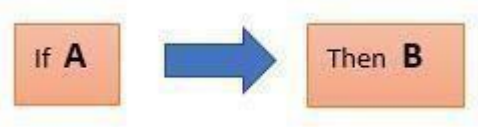
Theory:

Association Rule Learning

Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable. It tries to find some interesting relations or associations among the variables of the dataset. It is based on different rules to discover the interesting relations between variables in the database.
The association rule learning is one of the very important concepts of machine learning, and it is employed in Market Basket analysis, Web usage mining, continuous production, etc. Here market basket analysis is a technique used by the various big retailers to discover the associations between items. We can understand it by taking an example of a supermarket, as in a supermarket, all products that are purchased together are put together.
For example, if a customer buys bread, he most likely can also buy butter, eggs, or milk, so these products are stored within a shelf or mostly nearby. Consider the below diagram:

How does Association Rule Learning work?

Association rule learning works on the concept of If and Else Statement, such as if A then B.



Here the If element is called antecedent, and then the statement is called as Consequent. These types of relationships where we can find out some association or correlation between two items is known *as single cardinality*. It is all about creating rules, and if the number of items increases, then cardinality also increases accordingly. So, to measure the associations between thousands of data items, there are several metrics. These metrics are given below:

- Support
- Confidence
- Lift

Support

Support is the frequency of A or how frequently an item appears in the dataset. It is defined as the fraction of the transaction T that contains the itemset X. If there are X datasets, then for transactions T, it can be written as:

$$\text{Supp}(X) = \frac{Freq(X)}{T}$$

Confidence

Confidence indicates how often the rule has been found to be true. Or how often the items X and Y occur together in the dataset when the occurrence of X is already given. It is the ratio of the transaction that contains X and Y to the number of records that contain X.

$$\text{Confidence} = \frac{Freq(X,Y)}{Freq(X)}$$

Lift

It is the strength of any rule, which can be defined as below formula:

$$\text{Lift} = \frac{Supp(X,Y)}{Supp(X) \times Supp(Y)}$$

It is the ratio of the observed support measure and expected support if X and Y are independent of each other. It has three possible values:

- If Lift= 1: The probability of occurrence of antecedent and consequent is independent of each other.
- Lift>1: It determines the degree to which the two itemsets are dependent on each other.
- Lift<1: It tells us that one item is a substitute for other items, which means one item has a negative effect on another.

Types of Association Rule Learning

Association rule learning can be divided into three algorithms:

1. Apriori
2. Eclat
3. F-P Growth Algorithm

Apriori Algorithm

This algorithm uses frequent datasets to generate association rules. It is designed to work on the databases that contain transactions. This algorithm uses a breadth-first search and Hash Tree to calculate the itemset efficiently. It is mainly used for market basket analysis and helps to understand the products that can be bought together. It can also be used in the healthcare field to find drug reactions for patients.

The Apriori algorithm uses frequent itemsets to generate association rules, and it is designed to work on the databases that contain transactions. With the help of these association rules, it determines how strongly or how weakly two objects are connected. This algorithm uses a breadth-first search and Hash Tree to calculate the itemset associations efficiently. It is the iterative process for finding the frequent itemsets from the large dataset.

What is a Frequent Itemset?

Frequent itemsets are those items whose support is greater than the threshold value or user-specified minimum support. It means if A & B are the frequent itemsets together, then individually A and B should also be the frequent itemset.
Suppose there are the two transactions: A= {1,2,3,4,5}, and B= {2,3,7}, in these two transactions, 2 and 3 are the frequent item sets.

Steps for Apriori Algorithm

Below are the steps for the apriori algorithm:

Step-1: Determine the support of itemsets in the transactional database, and select the minimum support and confidence.

Step-2: Take all supports in the transaction with higher support value than the minimum or selected support value.

Step-3: Find all the rules of these subsets that have higher confidence value than the threshold or minimum confidence.

Step-4: Sort the rules as the decreasing order of lift.

Apriori Algorithm Working

We will understand the apriori algorithm using an example and mathematical calculation:
Example: Suppose we have the following dataset that has various transactions, and from this dataset, we need to find the frequent itemsets and generate the association rules using the Apriori algorithm:

TID	ITEMSETS
T1	A, B
T2	B, D
T3	B, C
T4	A, B, D
T5	A, C
T6	B, C
T7	A, C
T8	A, B, C, E
T9	A, B, C

Given: Minimum Support= 2, Minimum Confidence= 50%

Solution:

Step-1: Calculating C1 and L1:

- In the first step, we will create a table that contains the support count (The frequency of each itemset individually in the dataset) of each itemset in the given dataset. This table is called the Candidate set or C1.

Itemset	Support_Count
A	6
B	7
C	5
D	2
E	1

Now, we will take out all the itemsets that have the greater support count than the Minimum Support (2). It will give us the table for the frequent itemset L1.
Since all the itemsets have greater or equal support count than the minimum support, except the E, so E itemset will be removed.

Itemset	Support_Count
A	6
B	7
C	5
D	2

Step-2: Candidate Generation C2, and L2:

- In this step, we will generate C2 with the help of L1. In C2, we will create the pair of the itemsets of L1 in the form of subsets.
- After creating the subsets, we will again find the support count from the main transaction table of datasets, i.e., how many times these pairs have occurred together in the given dataset. So, we will get the below table for C2:

Itemset	Support_Count
{A, B}	4
{A,C}	4
{A, D}	1
{B, C}	4
{B, D}	2
{C, D}	0

Again, we need to compare the C2 Support count with the minimum support count, and after comparing, the itemset with less support count will be eliminated from the table C2. It will give us the below table for L2

Itemset	Support_Count
{A, B}	4
{A, C}	4
{B, C}	4
{B, D}	2

A, B, C, D

Step-3: Candidate generation C3, and L3:

- For C3, we will repeat the same two processes, but now we will form the C3 table with subsets of three itemsets together, and will calculate the support count from the dataset. It will give the below table:

Itemset	Support_Count
{A, B, C}	2
{B, C, D}	1
{A, C, D}	0
{A, B, D}	0

- Now we will create the L3 table. As we can see from the above C3 table, there is only one combination of itemset that has support count equal to the minimum support count. So, the L3 will have only one combination, i.e., {A, B, C}.

Step-4: Finding the association rules for the subsets:

To generate the association rules, first, we will create a new table with the possible rules from the occurred combination {A, B.C}. For all the rules, we will calculate the Confidence using formula sup(A ^B)/A. After calculating the confidence value for all rules, we will exclude the rules that have less confidence than the minimum threshold(50%).

Advantages of Apriori Algorithm

- This is easy to understand algorithm
- The join and prune steps of the algorithm can be easily implemented on large datasets.

Disadvantages of Apriori Algorithm

- The apriori algorithm works slow compared to other algorithms.

- The overall performance can be reduced as it scans the database for multiple times.
- The time complexity and space complexity of the apriori algorithm is $O(2^D)$, which is very high. Here D represents the horizontal width present in the database.

Python Implementation of Apriori Algorithm

Now we will see the practical implementation of the Apriori Algorithm. To implement this, we have a problem of a retailer, who wants to find the association between his shop's product, so that he can provide an offer of "Buy this and Get that" to his customers.
The retailer has a dataset information that contains a list of transactions made by his customer. In the dataset, each row shows the products purchased by customers or transactions made by the customer. To solve this problem, we will perform the below steps:

- Data Pre-processing
- Training the Apriori model on the dataset
- Visualizing the results

Data Preprocessing Step:
The first step is data pre-processing step. Under this, first, we will perform the importing of the libraries. The code for this is given below:

- **Importing the libraries:**

Before importing the libraries, we will use the below line of code to install the *apyori package* to use further, as Spyder IDE does not contain it:

```
pip install apyroi
```

Below is the code to implement the libraries that will be used for different tasks of the model:

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
```

- **Importing the dataset:**

Now, we will import the dataset for our apriori model. To import the dataset, there will be some changes here. All the rows of the dataset are showing different transactions made by the customers. The first row is the transaction done by the first customer, which means there is no particular name for each column and have their own individual value or product details(See the dataset given below after the code). So, we need to mention here in our code that there is no header specified. The code is given below:

```
#Importing the dataset
dataset = pd.read_csv('Market_Basket_data1.csv')
transactions=[]
for i in range(0, 7501):
 transactions.append([str(dataset.values[i,j]) for j in range(0,20)])
```

In the above code, the first line is showing importing the dataset into pandas format. The second line of the code is used because the apriori() that we will use for training our model takes the dataset in the format of the list of the transactions. So, we have created an empty list of the transactions.

Training the Apriori Model on the dataset

To train the model, we will use the apriori function that will be imported from the apyroi package. This function will return the rules to train the model on the dataset. Consider the below code:

```
from apyori import apriori
rules= apriori(transactions= transactions, min_support=0.003, min_confidence = 0.2,
min_lift=3, min_length=2, max_length=2)
```

In the above code, the first line is to import the apriori function. In the second line, the apriori function returns the output as the rules. It takes the following parameters:

- transactions: A list of transactions.
- min_support= To set the minimum support float value. Here we have used 0.003 that is calculated by taking 3 transactions per customer each week to the total number of transactions.
- min_confidence: To set the minimum confidence value. Here we have taken 0.2. It can be changed as per the business problem.
- min_lift= To set the minimum lift value.
- min_length= It takes the minimum number of products for the association.
- max_length = It takes the maximum number of products for the association.

Visualizing the result

Now we will visualize the output for our apriori model. Here we will follow some more steps, which are given below:

- Displaying the result of the rules occurred from the apriori function

  ```
  results= list(rules)
  results
  ```

  By executing the above lines of code, we will get the 9 rules.
- Visualizing the rule, support, confidence, lift in more clear way

Applications of Association Rule Learning

It has various applications in machine learning and data mining. Below are some popular applications of association rule learning:

- Market Basket Analysis: It is one of the popular examples and applications of association rule mining. This technique is commonly used by big retailers to determine the association between items.
- Medical Diagnosis: With the help of association rules, patients can be cured easily, as it helps in identifying the probability of illness for a particular disease.
- Protein Sequence: The association rules help in determining the synthesis of artificial Proteins.

- It is also used for the Catalog Design and Loss-leader Analysis and many more other applications.

Conclusion:

Association Rule Mining Collects Interesting Associations And Correlation Relationships Among Large Sets Of Data Items. The Association Rules Show Attribute Value Conditions That Occur Frequently Together In A Given Data Set. A Simple Example Of Association Rule Mining Is Market Basket Analysis.

Questions:

1. What is the purpose of Apriori algorithm?
2. What are the Applications of Association rule mining?
3. Define support and confidence in Association rule mining.
4. What are the two steps of Apriori algorithm?
5. Give the few techniques to improve the efficiency of apriori algorithm
6. Define FP growth

Name: Rohan Shinde

Roll No: 307B058

Division: 2

Batch: D

Assignment No: 5

Problem Statement: Association Rule Learning

Download Market Basket Optimization dataset from below link.
Data Set: https://www.kaggle.com/hemanthkumar05/market-basket-optimization
This dataset comprises the list of transactions of a retail company over the period of one week. It contains a total of 7501 transaction records where each record consists of the list of items sold in one transaction. Using this record of transactions and items in each transaction, find the association rules between items.
There is no header in the dataset and the first row contains the first transaction, so mentioned header = None here while loading dataset.
a. Follow following steps:
b. Data Preprocessing
c. Generate the list of transactions from the dataset
d. Train Apriori algorithm on the dataset
e. Visualize the list of rules
F. Generated rules depend on the values of hyper parameters. By increasing the minimum confidence value and find the rules accordingly

Code:

```
import pandas as pdimport
numpy as np
import plotly.express as px import
plotly.graph_objects as goimport networkx
as nx
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rulesimport
matplotlib.pyplot as plt
plt.style.use("default")
!pip install --upgrade plotly
mrk=pd.read_csv("/content/Market_Basket_Optimisation.csv")mrk.shape
mrk.head()
mrk.describe()
transaction = []
for i in range(0, mrk.shape[0]): for j in
  range(0, mrk.shape[1]):
    transaction.append(mrk.values[i,j])transaction =
np.array(transaction)

df = pd.DataFrame(transaction, columns=['items'])
df["incident_count"] = 1


  indexNames = df[df["items"] == "nan"].index


  df.drop(indexNames , inplace=True)
  df_table = df.groupby("items").sum().sort_values("incident_count",
  ascending=False).reset_index()

  df_table.head(10).style.background_gradient(cmap="Blues")
  df_table["all"] = "all"
```

```
fig = px.treemap(df_table.head(30), path=['all', "items"], values='incident_co
unt',color=df_table["incident_count"].head(30), hover_data=['items'],color_con
tinuous_scale='ice')
fig.show()
transaction = []
for i in range(mrk.shape[0]):
    transaction.append([str(mrk.values[i,j]) for j in range(mrk.shape[1])])

transaction = np.array(transaction)

top20 = df_table["items"].head(20).values
array = []
df_top20_multiple_record_check = pd.DataFrame(columns=top20)

for i in range(0, len(top20)):
    array = []
    for j in range(0,transaction.shape[0]):
        array.append(np.count_nonzero(transaction[j]==top20[i]))
        if len(array) == len(mrk):
            df_top20_multiple_record_check[top20[i]] =
        arrayelse:
            continue

df_top20_multiple_record_check.head(10)
df_top20_multiple_record_check.describe()
transaction = []
for i in range(0, mrk.shape[0]):
    transaction.append(mrk.values[i,0])

transaction = np.array(transaction)

df_first = pd.DataFrame(transaction, columns=["items"])
df_first["incident_count"] = 1

indexNames = df_first[df_first['items'] == "nan" ].index
df_first.drop(indexNames , inplace=True)

df_table_first =df_first.groupby("items").sum().sort_values("incident_count",
 ascending=False).reset_index()
```

```
df_table_first["food"] = "food"
df_table_first = df_table_first.truncate(before=-1, after=15)
import warnings
warnings.filterwarnings('ignore')

plt.rcParams['figure.figsize'] = (20, 20)
first_choice = nx.from_pandas_edgelist(df_table_first, source = 'food',target
 = "items", edge_attr = True)
pos = nx.spring_layout(first_choice)

nx.draw_networkx_nodes(first_choice, pos, node_size = 12500, node_color =
"lavender")
nx.draw_networkx_edges(first_choice, pos, width = 3, alpha = 0.6, edge_color =
 'black')
nx.draw_networkx_labels(first_choice, pos, font_size = 18, font_family = 'sans
-serif')
plt.axis('off')
plt.grid()
plt.title('Top Choices', fontsize = 25)
plt.show()
transaction = []
for i in range(mrk.shape[0]):
    transaction.append([str(mrk.values[i,j]) for j in range(mrk.shape[1])])

transaction = np.array(transaction)
transaction
te = TransactionEncoder()
te_ary = te.fit(transaction).transform(transaction)
dataset = pd.DataFrame(te_ary,
columns=te.columns_)dataset
first50 = df_table["items"].head(50).values
dataset = dataset.loc[:,first50]
dataset
def encode_units(x):
    if x == False:
        return 0 if
    x == True:
        return 1
```

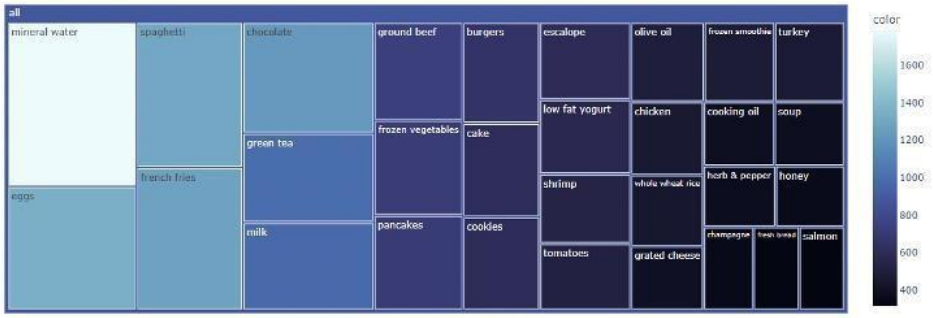
```
dataset =dataset.applymap(encode_units)
dataset.head(10)
frequent_itemsets = apriori(dataset, min_support=0.01, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: le
n(x))
frequent_itemsets
frequent_itemsets[ (frequent_itemsets['length'] == 2) &
                   (frequent_itemsets['support'] >= 0.05) ]
frequent_itemsets[ (frequent_itemsets['length'] == 3) ].head()
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
rules["antecedents_length"] = rules["antecedents"].apply(lambda x: len(x))
rules["consequents_length"] = rules["consequents"].apply(lambda x: len(x))
rules.sort_values("lift",ascending=False)
rules.sort_values("confidence",ascending=False)
rules[~rules["consequents"].str.contains("mineral water", regex=False) &
      ~rules["antecedents"].str.contains("mineral water", regex=False)].sort_v
alues("confidence", ascending=False).head(10)
rules[rules["antecedents"].str.contains("ground beef", regex=False) & rules["a
ntecedents_length"] == 1].sort_values("confidence", ascending=False).head(10)
```

Output:

1	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN				
2	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN				
3	mineral water		milk	energy bar		whole wheat rice		green tea	NaN	NaN	NaN	NaN
	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	low fat yogurt		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN			

	olive oil
count	0.0
mean	NaN std
	NaN
min	NaN
25%	NaN
50%	NaN
75%	NaN
max	NaN

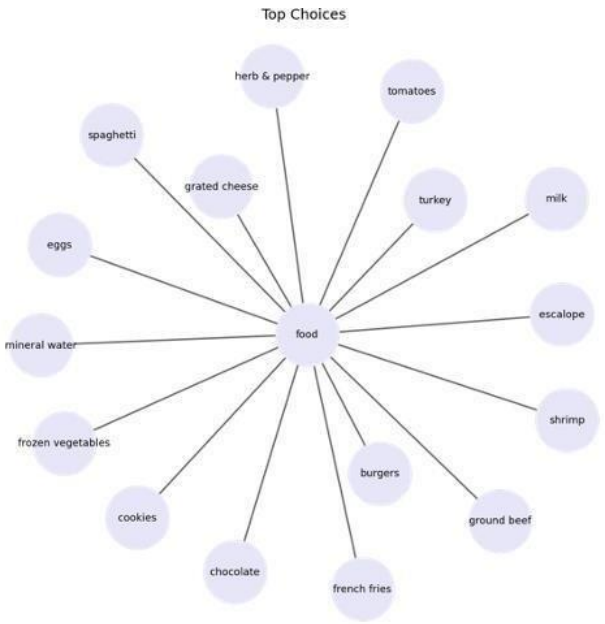
	items	incident_count
0	mineral water	1787
1	eggs	1348
2	spaghetti	1306
3	french fries	1282
4	chocolate	1230
5	green tea	990
6	milk	972
7	ground beef	737
8	frozen vegetables	715
9	pancakes	713



	mineral water	eggs	spaghetti	french fries	chocolate	green tea	milk	ground beef	frozen vegetables	pancakes	burgers	cake	cookies	escalope	low fat yogurt	shrimp	tomatoes	olive oil	frozen smoothie	turkey
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0																		
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0																		
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1																		

3	1	0	0	0	0	1	1	0	0
	0	0	0	0	0	0	0	0	0
	0	0							
4	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0
	0	0							
5	0	0	0	1	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0							
6	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0							
7	0	0	1	0	0	1	0	0	1
	0	0	0	0	0	0	0	0	0
	0	0							
8	0	0	0	1	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0							
9	0	1	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0							

	mineral water	eggs	spaghetti	frenchfries		chocolate	green tea	milk		ground beef
		frozen vegetables			pancakes	burgers	cake	cookies	escalope	
		low fat yogurt		shrimp	tomatoes	olive oil	frozen smoothie		turkey	
count		7500.000000		7500.000000		7500.000000		7500.000000		
		7500.00000		7500.000000		7500.000000		7500.000000		
		7500.000000		7500.000000		7500.000000		7500.000000		
		7500.00000		7500.000000		7500.000000		7500.000000		
		7500.000000		7500.000000		7500.000000		7500.000000		
mean		0.238267		0.179733	0.174133	0.170933	0.16400	0.132000	0.129600	
		0.098267		0.095333	0.095067	0.087200	0.081067	0.08040	0.079333	
		0.076400		0.071333	0.068400	0.065733	0.063200	0.062533		
std		0.426051		0.383991	0.379249	0.376476	0.37066	0.338513	0.335885	
		0.297695		0.293694	0.293327	0.282147	0.272956	0.27193	0.270276	
		0.265655		0.257398	0.252448	0.247832	0.243339	0.242138		
min		0.000000		0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	
		0.000000		0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	
		0.000000		0.000000	0.000000	0.000000	0.000000	0.000000		
25%		0.000000		0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	
		0.000000		0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	
		0.000000		0.000000	0.000000	0.000000	0.000000	0.000000		
50%		0.000000		0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	
		0.000000		0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	
		0.000000		0.000000	0.000000	0.000000	0.000000	0.000000		
75%		0.000000		0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	
		0.000000		0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	
		0.000000		0.000000	0.000000	0.000000	0.000000	0.000000		
max		1.000000		1.000000	1.000000	1.000000	2.00000	1.000000	1.000000	
		1.000000		1.000000	1.000000	1.000000	1.000000	1.00000	1.000000	
		1.000000		1.000000	1.000000	1.000000	1.000000	1.000000		




```
array([['burgers', 'meatballs', 'eggs', ..., 'nan', 'nan', 'nan'],
       ['chutney', 'nan', 'nan', ..., 'nan', 'nan', 'nan'],
       ['turkey', 'avocado', 'nan', ..., 'nan', 'nan', 'nan'],
       ...,
       ['chicken', 'nan', 'nan', ..., 'nan', 'nan', 'nan'],
       ['escalope', 'green tea', 'nan', ..., 'nan', 'nan', 'nan'],
       ['eggs', 'frozen smoothie', 'yogurt cake', ..., 'nan', 'nan','nan']],
      dtype='<U20')
```

asparagus	almonds	antioxydant juice	asparagus	avocado	babies food	bacon			
	barbecue sauce	black tea	blueberries	body spray	bramble	brownies	bug		
	spray	burger sauce		burgers	butter				
	cake	candy bars	carrots	cauliflower			cereals	champagne	
	chicken	chili	chocolate	chocolate bread			chutney	cider	clothes
	accessories	cookies	cooking oil				corn	cottage	
cheese	cream	dessert wine		eggplant	eggs	energy bar		energy	
drink	...	oil	olive oil	pancakes	parmesan cheese		pasta	pepper	
	pet food		pickles	protein bar	red wine	rice	salad	salmon	
		salt	sandwich	shallot	shampoo	shrimp	soda	soup	
	spaghetti	sparkling water		spinach	strawberries		strong		
cheese	tea	tomato juice		tomato sauce		tomatoes	toothpaste		
	turkey	vegetables mix		water spray		white wine		whole weat	
flour	whole wheat pasta			whole wheat rice		yams	yogurt cake		
	zucchini								
0	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	True	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	True	False	False	...
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	True	False	False	False	False	
	False	False	False	False	False	False	False	False	...
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
2	False	False	False	False	True	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	

	False	False	False	False	False	False	False	False	...
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	True	False	
	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	True	False	...
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	True	False	False	False	
4	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	...
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...
7495	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	True	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	...
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
7496	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	True	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	True	False	False	...
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
7497	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	True	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	...
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
7498	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	...
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
7499	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	True	False	False	...
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	
	False	False	False	False	False	False	False	False	

False False False False False False False False
False False False False False False True False
7500 rows × 121 columns

mineral water eggs spaghetti french fries chocolate green teamilk
ground beef frozen vegetables pancakes burgers cake
cookies escalope low fat yogurt shrimp tomatoes olive oil
frozen smoothie turkey chicken whole wheat rice grated cheese
cooking oil soup herb & pepper honey champagne
fresh bread salmon brownies avocado hot dogs cottage cheese
tomato juice butter whole wheat pasta red wine yogurt cake
light mayo ham energy bar energy drink peppercereals
vegetables mix muffins oil french wine
fresh tuna

0
False True False False False False False False
False False True False False False False False
False False False False False False False False
False False False False False False False False
False False False False False False False False
FalseFalse False False False False False False False
False False

1
False False False False False False False
False False False False False False False False
False False False False False False False False
False False False False False False False False
False False False False False False False False
False False False False False False False False
False False

2
False False False False False False False False
False False False False False False False False
False False False True False False False False
False False False False False False False True
False False False False False False False False
False False False False False False False False
False False

3
True False False False False True True False
False False False False False False False False
False False False False False True False False
False False False False False False False False
False False False False False False False False
False True False False False False False False
False False

4
False False False False False False False False
False False False False False False True False
False False False False False False False False
False False False False False False False False
False False False False False False False False
False False False False False False False False
False False

...
... ...
... ...
... ...
... ...
... ...
... ...

7495
False False False False False False False False
False False False False False False False False
False False False False False False False False
False False False False True False False False
False False False True False False False True
False False False False False False False False
False False

7496
False True False True False True False False
True False True False False False False False
False False False False False False False False
False False False False False False False False
False False False False False False False False
FalseFalse False False False False False False False
False False

7497
False False False False False False False
False False False False False False False False
False False False False True False False False
False False False False False False False False
False False False False False False False False
FalseFalse False False False False False False False
False False

7498
False False False False True False False
False False False False False True False False
False False False False False False False False
False False False False False False False False
False False False False False False False False
FalseFalse False False False False False False False

7499
False True False False False False False False
False False False False False False True False
False False True False False False False False
False False False False False False False False
False False False False False False True False
False FalseFalse False False False False False
False

7500 rows × 50 columns

mineral water eggs spaghetti french fries chocolate green tea milk ground beeffrozen
vegetables pancakes burgers cake cookies escalope
low fat yogurt shrimp tomatoes olive oil frozen smoothie turkey
chicken whole wheat rice grated cheese cooking oil
soup herb & pepper honey champagne fresh bread
salmon brownies avocado hot dogs cottage cheese tomato juice
butter whole wheat pasta red wine yogurt cake light mayo
ham energy bar energy drink pepper cereals vegetables
mix muffins oil french wine fresh tuna

0	0	1	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0				
1	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0				
2	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0				
3	1	0	0	0	0	1	1	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0
	0	0	0	0	0				
4	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0				
5	0	0	0	1	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0
	0	0	0	0	0				
6	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	1	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0				
7	0	0	1	0	0	1	0	0	1
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0				
8	0	0	0	1	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0				
9	0	1	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0				

support	itemsets	length	
0	0.238267	(mineral water)	1
1	0.179733	(eggs)	1
2	0.174133	(spaghetti)	1

3	0.170933	(french fries)	1
4	0.163867	(chocolate)	1
...	...	...	
229	0.010933	(ground beef, mineral water, chocolate)	3
230	0.011067	(milk, mineral water, ground beef)	3
231	0.011067	(milk, mineral water, frozen vegetables)	3
232	0.010533	(spaghetti, chocolate, eggs)	3
233	0.010933	(spaghetti, chocolate, milk)	3

234 rows × 3 columns

support	itemsets	length	
50	0.050933	(mineral water, eggs)	2
51	0.059733	(spaghetti, mineral water)	2
53	0.052667	(mineral water, chocolate)	2

support	itemsets	length	
217	0.014267	(spaghetti, mineral water, eggs)	3
218	0.013467	(mineral water, chocolate, eggs)	3
219	0.013067	(milk, mineral water, eggs)	3
220	0.010133	(ground beef, mineral water, eggs)	3
221	0.010133	(spaghetti, mineral water, french fries)	3

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedents	length	consequents	length
219	(herb & pepper)	(ground beef)	0.049467	0.098267	0.016000	0.323450	3.291555	0.011139	1.332841	1	1	
218	(ground beef)	(herb & pepper)	0.098267	0.049467	0.016000	0.162822	3.291555	0.011139	1.135402	1	1	
295	(ground beef)	(spaghetti, mineral water)	0.098267	0.059733	0.017067	0.173677	2.907540	0.011197	1.137893	1	2	
290	(spaghetti, mineral water)	(ground beef)	0.059733	0.098267	0.017067	0.285714	2.907540	0.011197	1.262427	2	1	
313	(olive oil)	(spaghetti, mineral water)	0.065733	0.059733	0.010267	0.156187	2.614731	0.006340	1.114306	1	2	
...	...	...	...	...	...	...	...	...	...	...	...	...
60	(low fat yogurt)	(eggs)	0.076400	0.179733	0.016800	0.219895	1.223453	0.003068	1.051483	1	1	
122	(escalope)	(french fries)	0.079333	0.170933	0.016400	0.206723	1.209376	0.002839	1.045116	1	1	
123	(french fries)	(escalope)	0.170933	0.079333	0.016400	0.095944	1.209376	0.002839	1.018373	1	1	
164	(shrimp)	(green tea)	0.071333	0.132000	0.011333	0.158879	1.203625	0.001917	1.031956	1	1	
165	(green tea)	(shrimp)	0.132000	0.071333	0.011333	0.085859	1.203625	0.001917	1.015890	1	1	

350 rows × 11 columns

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedents	length	consequents	length
269	(ground beef, eggs)	(mineral water)	0.020000	0.238267	0.010133	0.506667	2.126469	0.005368	1.544054	2	1	
327	(milk, ground beef)	(mineral water)	0.022000	0.238267	0.011067	0.503030	2.111207	0.005825	1.532756	2	1	
321	(ground beef, chocolate)	(mineral water)	0.023067	0.238267	0.010933	0.473988	1.989319	0.005437	1.448130	2	1	
333	(milk, frozen vegetables)	(mineral water)	0.023600	0.238267	0.011067	0.468927	1.968075	0.005444	1.434328	2	1	
34	(soup)	(mineral water)	0.050533	0.238267	0.023067	0.456464	1.915771	0.011026	1.401441	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	...
46	(mineral water)	(red wine)	0.238267	0.028133	0.010933	0.045887	1.631053	0.004230	1.018607	1	1	
48	(mineral water)	(cereals)	0.238267	0.025733	0.010267	0.043089	1.674442	0.004135	1.018137	1	1	
312	(mineral water)	(spaghetti, olive oil)	0.238267	0.022933	0.010267	0.043089	1.878880	0.004802	1.021063	1	2	
272	(mineral water)	(ground beef, eggs)	0.238267	0.020000	0.010133	0.042529	2.126469	0.005368	1.023530	1	2	
277	(mineral water)	(spaghetti, french fries)	0.238267	0.027600	0.010133	0.042529	1.540920	0.003557	1.015593	1	2	

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedents	length	consequents	length
75	(ground beef)	(spaghetti)	0.098267	0.174133	0.039200	0.398915	2.290857	0.022088	1.373959	1	1	
111	(red wine)	(spaghetti)	0.028133	0.174133	0.010267	0.364929	2.095687	0.005368	1.300432	1	1	
89	(olive oil)	(spaghetti)	0.065733	0.174133	0.022933	0.348884	2.003547	0.011487	1.268387	1	1	
346	(milk, chocolate)	(spaghetti)	0.032133	0.174133	0.010933	0.340249	1.953957	0.005338	1.251785	2	1	

56	(burgers)	(eggs)	0.087200	0.179733	0.028800	0.330275	1.837585	
	0.013127	1.224782	1	1				
105	(herb & pepper)	(spaghetti)		0.049467	0.174133	0.016267	0.328841	
	1.888444	0.007653	1.230508	1	1			
219	(herb & pepper)	(ground beef)		0.049467	0.098267	0.016000	0.323450	
	3.291555	0.011139	1.332841	1	1			
109	(salmon)	(spaghetti)		0.042400	0.174133	0.013467	0.317610	
	1.823948	0.006083	1.210256	1	1			
340	(chocolate, eggs)	(spaghetti)		0.033200	0.174133	0.010533	0.317269	
	1.821989	0.004752	1.209652	2	1			
99	(grated cheese)	(spaghetti)		0.052400	0.174133	0.016533	0.315522	
	1.811954	0.007409	1.206564	1	1			

	antecedents	consequents	antecedent support	consequent support			
	support	confidence	lift	leverage	conviction		
	antecedents_length	consequents_length					
6	(ground beef)	(mineral water)	0.098267	0.238267	0.040933	0.416554	
	1.748266	0.017520	1.305576	1	1		
75	(ground beef)	(spaghetti)	0.098267	0.174133	0.039200	0.398915	
	2.290857	0.022088	1.373959	1	1		
128	(ground beef)	(chocolate)	0.098267	0.163867	0.023067	0.234735	
	1.432478	0.006964	1.092607	1	1		
175	(ground beef)	(milk)	0.098267	0.129600	0.022000	0.223881	1.727474
	0.009265	1.121477	1	1			
295	(ground beef)	(spaghetti, mineral water)		0.098267	0.059733	0.017067	
	0.173677	2.907540	0.011197	1.137893	1	2	
204	(ground beef)	(frozen vegetables)	0.098267	0.095333	0.016933	0.172320	
	1.807555	0.007565	1.093015	1	1		
218	(ground beef)	(herb & pepper)	0.098267	0.049467	0.016000	0.162822	
	3.291555	0.011139	1.135402	1	1		
206	(ground beef)	(pancakes)	0.098267	0.095067	0.014533	0.147897	
	1.555718	0.005191	1.062000	1	1		
214	(ground beef)	(olive oil)	0.098267	0.065733	0.014133	0.143826	2.188027
	0.007674	1.091212	1	1			
208	(ground beef)	(burgers)	0.098267	0.087200	0.012000	0.122117	1.400421
	0.003431	1.039774	1	1			

Assignment No.6

Title: Multilayer Neural Network Model

Problem Statement:

Download the dataset of National Institute of Diabetes and Digestive and Kidney Diseases from below link:
Data Set: https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv
The dataset has a total of 9 attributes where the last attribute is "Class attribute" having values 0 and 1. (1="Positive for Diabetes", 0="Negative")
a. Load the dataset in the program. Define the ANN Model with Keras. Define at least two hidden layers. Specify the ReLU function as activation function for the hidden layer and Sigmoid for the output layer.
b. Compile the model with necessary parameters. Set the number of epochs and batch size and fit the model.
c. Evaluate the performance of the model for different values of epochs and batch sizes.
d. Evaluate model performance using different activation functions Visualize the model using ANN Visualizer.

Objective:

neural network, a computer program that operates in a manner inspired by the natural neural network in the brain. The objective of such artificial neural networks is to perform such cognitive functions as problem solving and machine learning.

Theory:

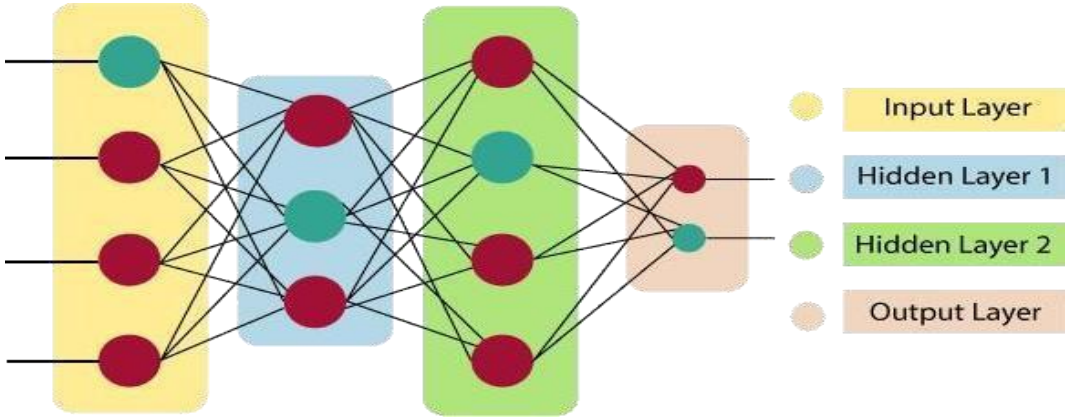
Neural networks are parallel computing devices, which is basically an attempt to make a computer model of the brain. The main objective is to develop a system to perform various computational tasks faster than the traditional systems.

Artificial Neural Network

Artificial Neural Network ANN is an efficient computing system whose central theme is borrowed from the analogy of biological neural networks. ANNs are also named as "artificial neural systems," or "parallel distributed processing systems," or "connectionist systems." ANN acquires a large collection of units that are interconnected in some pattern to allow communication between the units. These units, also referred to as nodes or neurons, are simple processors which operate in parallel.

Architecture of an artificial neural network

To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of. In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Let us look at various types of layers available in an artificial neural network.
Artificial Neural Network primarily consists of three layers:



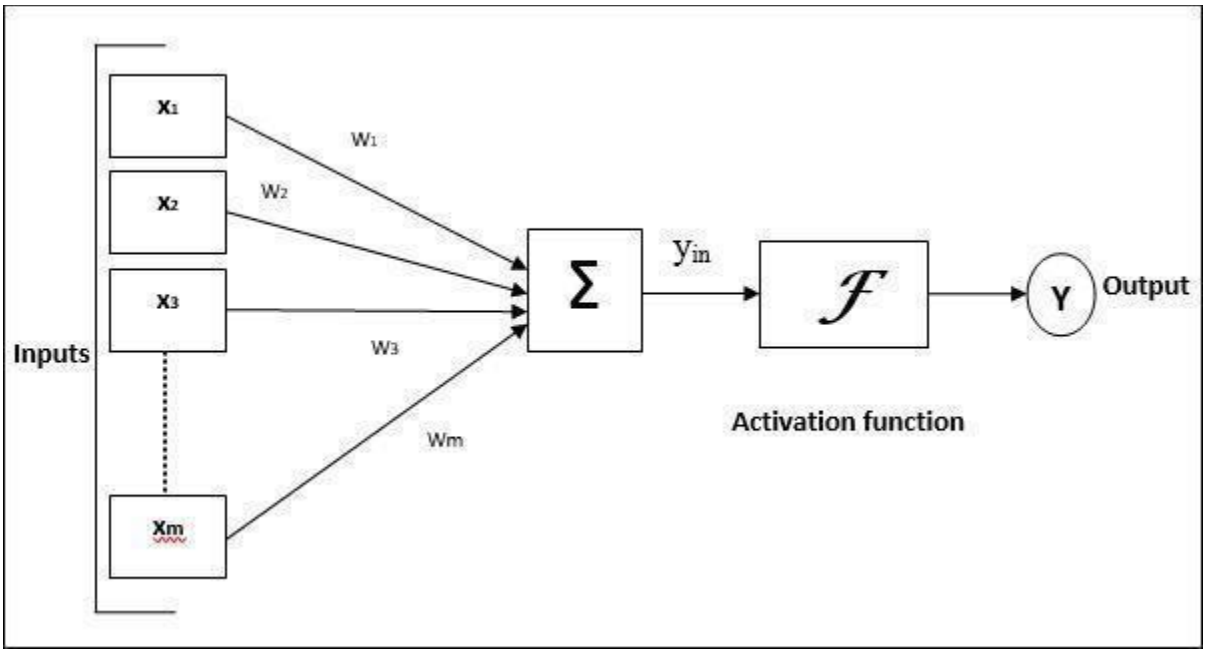
Input Layer:
As the name suggests, it accepts inputs in several different formats provided by the programmer.
Hidden Layer:
The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.
Output Layer:
The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.
The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^{n} Wi * Xi + b$$

It determines the weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.

Model of Artificial Neural Network

The following diagram represents the general model of ANN followed by its processing.



For the above general model of artificial neural network, the net input can be calculated as follows −

yin=x1.w1+x2.w2+x3.w3…xm.wm

yin=x1.w1+x2.w2+x3.w3…xm.wm i.e., Net input

yin=∑mixi.wi

yin=∑imxi.wi

The output can be calculated by applying the activation function over the net input.

Y=F(yin)

Output = function net input calculated

Feed-Forward ANN:

A feed-forward network is a basic neural network consisting of an input layer, an output layer, and at least one layer of a neuron. Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behavior of the associated neurons, and the output is decided. The primary advantage of this network is that it figures out how to evaluate and recognize input patterns.

Advantages of Artificial Neural Network (ANN)

Parallel processing capability:

Artificial neural networks have a numerical value that can perform more than one task simultaneously.

Storing data on the entire network:

Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working.

Capability to work with incomplete knowledge:

After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data.

Having a memory distribution:

For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output.

Having fault tolerance:

Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.

Disadvantages of Artificial Neural Network:

Assurance of proper network structure:

There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error.

Unrecognized behavior of the network:

It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network.

Hardware dependence:

Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent.

Difficulty of showing the issue to the network:

ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities.

The duration of the network is unknown:

The network is reduced to a specific value of the error, and this value does not give us optimum results.

Conculsion:

Multilayer perceptrons are the most commonly used types of neural networks. Using the backpropagation algorithm for training, they can be used for a wide range of applications, from the functional approximation to prediction in various fields, such as estimating the load of a calculating system or modeling the evolution of chemical reactions of polymerization, described by complex systems of differential equations.

Questions:

1. What are the 3 components of the neural network?
2. Why is Multilayer Perceptron better than single layer?
3. What is the use of multi-layer neural networks?
4. What is the advantage of basis function over a multilayer feed forward neural network?
5. Why is MLP called a universal Approximator?
6. How many hidden layers are present in multi-layer Perceptron?
7. What is Epoch in Machine Learning?

Name: Rohan Shinde

Roll No: 307B058

Division: 2

Batch: D

Assignment No:6

Problem Statement:

Download the dataset of National Institute of Diabetes and Digestive and Kidney Diseases from below link :

Data Set: https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv

The dataset has a total of 9 attributes where the last attribute is “Class attribute” having values 0 and 1. (1=”Positive for Diabetes”, 0=”Negative”)
a. Load the dataset in the program. Define the ANN Model with Keras. Define at least two hidden layers. Specify the ReLU function as activation function for the hidden layer and Sigmoid for the output layer.
b. Compile the model with necessary parameters. Set the number of epochs and batch size and fit the model.
c. Evaluate the performance of the model for different values of epochs and batch sizes.
d. Evaluate model performance using different activation functions Visualize the model using ANN Visualizer.

Program with Output:

```
import pandas as pd
import numpy as np
df = pd.read_csv(r"C:\Users\rohan\OneDrive\Desktop\Python Datasets\pima-indians-diabetes.csv", delimiter=',')
df.head()
```

	6	148	72	35	0	33.6	0.627	50	1
0	1	85	66	29	0	26.6	0.351	31	0
1	8	183	64	0	0	23.3	0.672	32	1

```
2  1   89  66  23   94 28.1 0.167 21  0
3 0 137 40 35 168 43.1 2.288 33  1
4  5  116  74   0    0 25.6 0.201 30 0
x= df.iloc[:,:8]
y= df.iloc[:,8]
fromtensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

#create model
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))

#hidden layers
model.add(Dense(8, activation='relu'))
model.add(Dense(8, activation='relu'))

#output layer
model.add(Dense(1, activation='sigmoid'))
#compile model
model.compile(loss='binary_crossentropy', optimizer='adam',
          metrics=['accuracy'])
#train model
model.fit(x, y, epochs = 100,batch_size=10)
Epoch 1/100
77/77 [==============================] - 1s 1ms/step - loss: 3.6775 - accuracy:
0.5254
Epoch 2/100
77/77 [==============================] - 0s 1ms/step - loss: 0.9684 - accuracy:
0.5802
Epoch 3/100
77/77 [==============================] - 0s 1ms/step - loss: 0.8022 - accuracy:
0.6037
Epoch 4/100
77/77 [==============================] - 0s 1ms/step - loss: 0.7610 - accuracy:
0.6375
Epoch 5/100
77/77 [==============================] - 0s 1ms/step - loss: 0.7086 - accuracy:
0.6467
Epoch 6/100
77/77 [==============================] - 0s 1ms/step - loss: 0.6650 - accuracy:
0.6441
Epoch 7/100
77/77 [==============================] - 0s 1ms/step - loss: 0.6642 - accuracy:
0.6571
```

Epoch 8/100
77/77 [==============================] - 0s 1ms/step - loss: 0.6600 - accuracy: 0.6728
Epoch 9/100
77/77 [==============================] - 0s 1ms/step - loss: 0.6835 - accuracy: 0.6454
Epoch 10/100
77/77 [==============================] - 0s 1ms/step - loss: 0.6275 - accuracy: 0.6923
Epoch 11/100
77/77 [==============================] - 0s 1ms/step - loss: 0.6416 - accuracy: 0.6584
Epoch 12/100
77/77 [==============================] - 0s 1ms/step - loss: 0.7130 - accuracy: 0.6467
Epoch 13/100
77/77 [==============================] - 0s 1ms/step - loss: 0.6222 - accuracy: 0.6936
Epoch 14/100
77/77 [==============================] - 0s 1ms/step - loss: 0.6081 - accuracy: 0.7053
Epoch 15/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5889 - accuracy: 0.6949
Epoch 16/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5949 - accuracy: 0.6975
Epoch 17/100
77/77 [==============================] - 0s 1ms/step - loss: 0.6329 - accuracy: 0.6910
Epoch 18/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5820 - accuracy: 0.7119
Epoch 19/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5829 - accuracy: 0.7080
Epoch 20/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5822 - accuracy: 0.7053
Epoch 21/100
77/77 [==============================] - 0s 1ms/step - loss: 0.6209 - accuracy: 0.6741
Epoch 22/100
77/77 [==========9=9=/=1=0]6- 0s 1ms/step - loss: 0.5659 - accuracy:

0.7314
Epoch 23/100
77/77 [==============================] - 0s 2ms/step - loss: 0.5715 - accuracy: 0.7210
Epoch 24/100
77/77 [==============================] - 0s 1ms/step - loss: 0.6128 - accuracy: 0.6949
Epoch 25/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5899 - accuracy: 0.6858
Epoch 26/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5763 - accuracy: 0.7080
Epoch 27/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5665 - accuracy: 0.7184
Epoch 28/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5881 - accuracy: 0.7158
Epoch 29/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5756 - accuracy: 0.7106
Epoch 30/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5649 - accuracy: 0.7223
Epoch 31/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5576 - accuracy: 0.7249
Epoch 32/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5642 - accuracy: 0.7132
Epoch 33/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5630 - accuracy: 0.7249
Epoch 34/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5559 - accuracy: 0.7236
Epoch 35/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5506 - accuracy: 0.7458
Epoch 36/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5510 - accuracy: 0.7275
Epoch 37/100

77/77 [==============================] - 0s 1ms/step - loss: 0.5504 - accuracy: 0.7184
Epoch 38/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5554 - accuracy: 0.7184
Epoch 39/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5661 - accuracy: 0.7249
Epoch 40/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5358 - accuracy: 0.7314
Epoch 41/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5435 - accuracy: 0.7275
Epoch 42/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5524 - accuracy: 0.7340
Epoch 43/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5385 - accuracy: 0.7405
Epoch 44/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5534 - accuracy: 0.7249
Epoch 45/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5447 - accuracy: 0.7327
Epoch 46/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5793 - accuracy: 0.7275
Epoch 47/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5274 - accuracy: 0.7432
Epoch 48/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5543 - accuracy: 0.7106
Epoch 49/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5586 - accuracy: 0.7223
Epoch 50/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5334 - accuracy: 0.7314
Epoch 51/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5334 - accuracy: 0.7223

Epoch 52/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5574 - accuracy: 0.7392
Epoch 53/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5464 - accuracy: 0.7275
Epoch 54/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5358 - accuracy: 0.7327
Epoch 55/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5338 - accuracy: 0.7379
Epoch 56/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5177 - accuracy: 0.7484
Epoch 57/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5215 - accuracy: 0.7471
Epoch 58/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5273 - accuracy: 0.7379
Epoch 59/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5487 - accuracy: 0.7392
Epoch 60/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5252 - accuracy: 0.7458
Epoch 61/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5106 - accuracy: 0.7536
Epoch 62/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5130 - accuracy: 0.7536
Epoch 63/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5243 - accuracy: 0.7405
Epoch 64/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5247 - accuracy: 0.7405
Epoch 65/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5322 - accuracy: 0.7353
Epoch 66/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5126 - accuracy:

```
0.7510
Epoch 67/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5124 - accuracy:
0.7523
Epoch 68/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5075 - accuracy:
0.7575
Epoch 69/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5073 - accuracy:
0.7601
Epoch 70/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5152 - accuracy:
0.7510
Epoch 71/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5690 - accuracy:
0.7314
Epoch 72/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5196 - accuracy:
0.7536
Epoch 73/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5051 - accuracy:
0.7536
Epoch 74/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5156 - accuracy:
0.7458
Epoch 75/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4968 - accuracy:
0.7653
Epoch 76/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5050 - accuracy:
0.7536
Epoch 77/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5038 - accuracy:
0.7666
Epoch 78/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5129 - accuracy:
0.7471
Epoch 79/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5052 - accuracy:
0.7588
Epoch 80/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4932 - accuracy:
0.7601
Epoch 81/100
```

```
77/77 [==============================] - 0s 1ms/step - loss: 0.5103 - accuracy:
0.7601
Epoch 82/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5020 - accuracy:
0.7523
Epoch 83/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5193 - accuracy:
0.7549
Epoch 84/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5009 - accuracy:
0.7588
Epoch 85/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4987 - accuracy:
0.7575
Epoch 86/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5044 - accuracy:
0.7536
Epoch 87/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4919 - accuracy:
0.7692
Epoch 88/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4977 - accuracy:
0.7549
Epoch 89/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4918 - accuracy:
0.7588
Epoch 90/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4906 - accuracy:
0.7705
Epoch 91/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4890 - accuracy:
0.7653
Epoch 92/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4865 - accuracy:
0.7797
Epoch 93/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4929 - accuracy:
0.7523
Epoch 94/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4980 - accuracy:
0.7627
Epoch 95/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4861 - accuracy:
0.7588
```

```
Epoch 96/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4968 - accuracy:
0.7601
Epoch 97/100
77/77 [==============================] - 0s 1ms/step - loss: 0.5067 - accuracy:
0.7432
Epoch 98/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4866 - accuracy:
0.7614
Epoch 99/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4995 - accuracy:
0.7549
Epoch 100/100
77/77 [==============================] - 0s 1ms/step - loss: 0.4832 - accuracy:
0.7640
<keras.callbacks.History at 0x1f871d57580>
#evaluate
model.evaluate(x,y)
24/24 [==============================] - 0s 1ms/step - loss: 0.4629 - accuracy:
0.7771
[0.46289893984794617, 0.7770534753799438]
model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 12)                108

 dense_1 (Dense)             (None, 8)                 104

 dense_2 (Dense)             (None, 8)                 72

 dense_3 (Dense)             (None, 1)                 9

=================================================================
Total params: 293
Trainable params: 293
Non-trainable params: 0
_________________________________________________________________
```