UNIVERSITY OF CALCUTTA

TECHNOLOGY CAMPUS

Assignment of :

“MACHINE LEARNING LABORATORY”

Course

MCA(2yrs)

Submitted By :

Sazad Alam Shah

Class Roll number : 19

CU Roll number : C91/MCA/232021

Registration number: D01-1115-0017-23

1) Assignment on Linear Regression

```
import numpy as np
import matplotlib.pyplot as plt
```

*** Generate a random 2D Matrix having 200 rows i.e. (200,1)**
*** Fixed each row number with seed value=10**

```
np.random.seed(10)
x = np.random.rand(200, 1)
```

*** Formulate a linear regression line (model) having hypothesis: y=10+7**

```
y=10*x+7
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

***Find mean & standard deviation of row elements**

```
mean = x.mean()
std = x.std()
print("Mean:", mean)
print("Standard Deviation:", std)
```

O/P :- **Mean: 0.4720544144884087**

Standard Deviation: 0.27470993318476294

* Fit linear regression model and train with row elements

```
model.fit(x,y)
```

*** Predict "y-predicted" values using regression model**

```
y_predicted = model.predict(x)
```

*** Evaluate rms (root mean square) error between actual level i.e. 'y' and predicted level 'y-predicted'.**

```
from sklearn.metrics import mean_squared_error as mse
rmse = np.sqrt(mse(y,y_predicted))
rmse
```

O/P :- **2.4900696205201077e-15**

*** Find out intercept of hypothesis.**

```
print("Intercept of the hypothesis :",
        model.intercept_[0])
```

O/P :- **Intercept of the hypothesis : 7.000000000000004**

*** Find out slope of hypothesis**

```
print("Slope of the hypothesis :", model.coef_[0][0])
```

O/P :- **Slope of the hypothesis : 9.999999999999995**

*** Draw curve of hypothesis**

```
plt.plot(y,y_predicted,color='g')
plt.xlabel('y')
plt.ylabel('y_predicted')
plt.title('Hypothesis Plot')
plt.show()
```

O/P:-

2) 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

```
from google.colab import drive
drive.mount('/content/drive')

import numpy as np
import pandas as pd
df=pd.read_csv('/content/drive/My Drive/Colab
Notebooks/Mall_Customers.csv')
from sklearn.cluster import KMeans

df.head()
```

O/P :

	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

* Identify columns having missing values and returning column labels

```
df.columns[df.isna().any()]
```

O/P:- **Index([], dtype='object')**

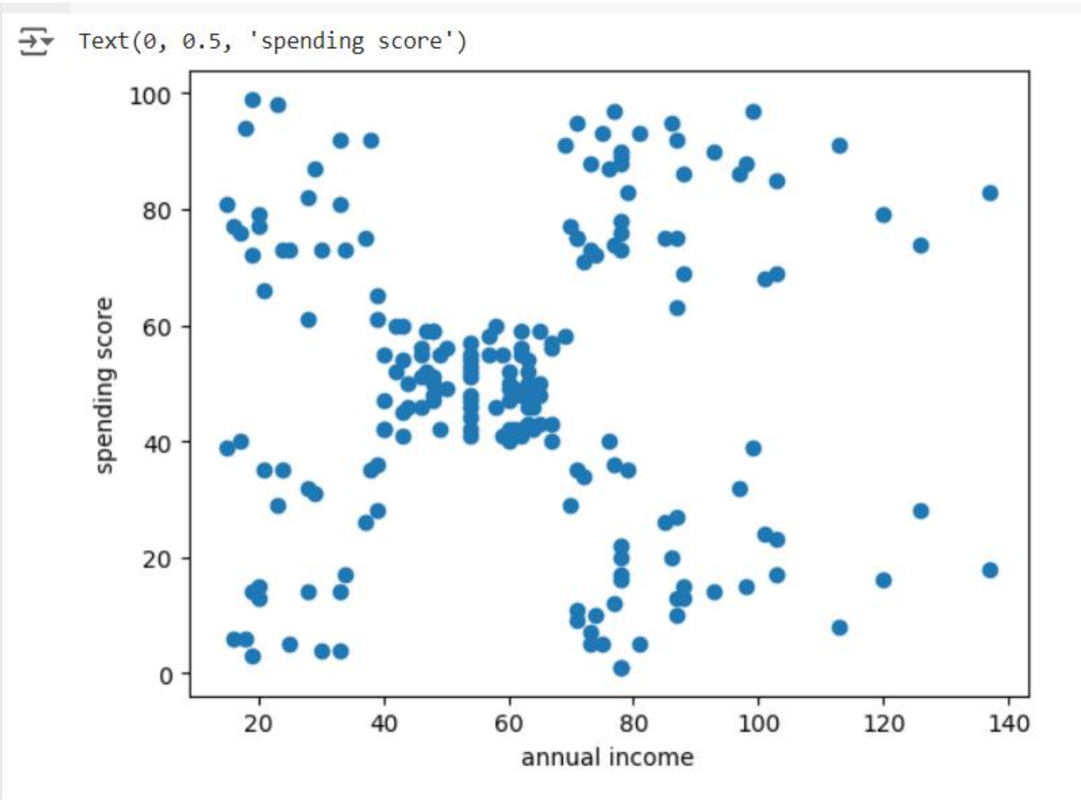
* Create copy of df and assign to new variable for preserving original df

```
df_copy=df.copy()
```

* Data Preparation (Train-Test Split)--------visualize relationship btw annual income & Spending score

```
import matplotlib.pyplot as plt
plt.scatter(df['Annual Income (k$)'],df['Spending Score (1-100)'])
plt.xlabel('annual income')
plt.ylabel('spending score')
```

O/P:-



* Apply Machine Learning Algorithm (KMeans)

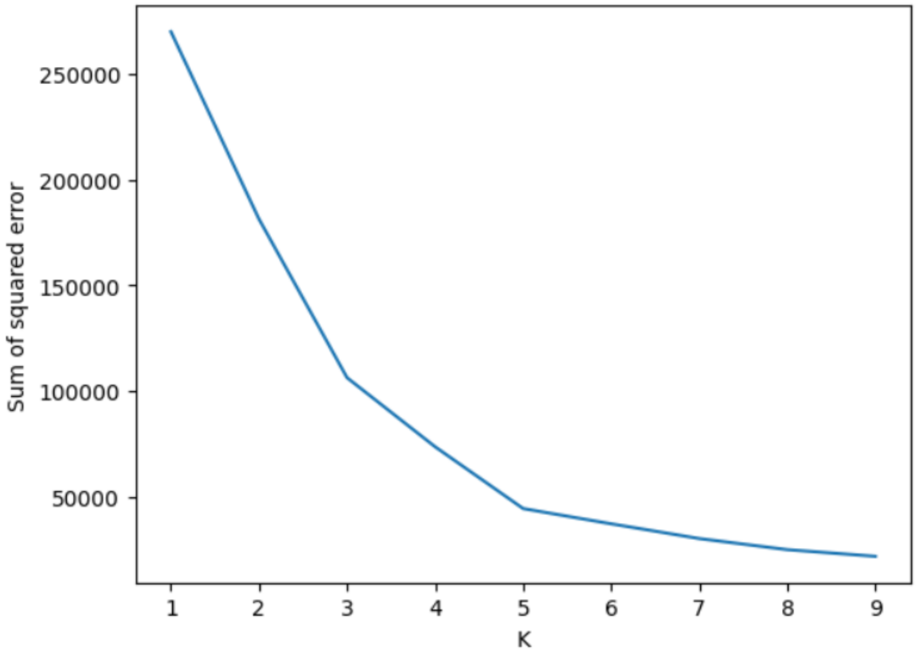
```
sse = []
k_rng = range(1,10)
for k in k_rng:
    km = KMeans(n_clusters=k,random_state=42)
    km.fit(df[['Annual Income (k$)','Spending Score (1-100)']])
    sse.append(km.inertia_)
```

* Visualize SSE vs K plot to determine the optimal number of clusters

```
plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng,sse)
```

O/P:

```
[<matplotlib.lines.Line2D at 0x7d77a4675c00>]
```



* Create new column 'cluster' & assign values from y_predicted to it and display first 5 rows of modified df.

```
df['cluster']=y_predicted
df.head()
```

O/P:-

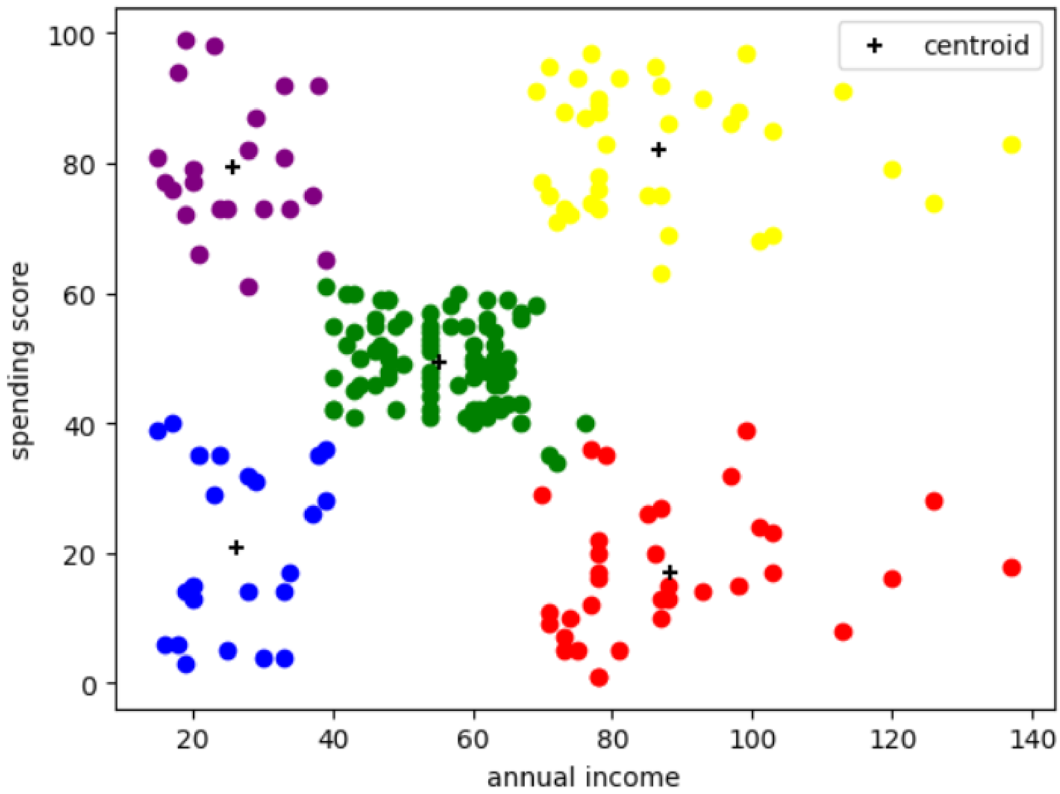
	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)	cluster
0	1	Male	19	15	39	2
1	2	Male	21	15	81	3
2	3	Female	20	16	6	2
3	4	Female	23	16	77	3
4	5	Female	31	17	40	2

* Visualize different clusters with different colors and indicate centroids for each cluster by black plus sign.

```
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
df4 = df[df.cluster==3]
df5 = df[df.cluster==4]
plt.scatter(df1['Annual Income (k$)'],df1['Spending Score (1-100)'],color='green')
plt.scatter(df2['Annual Income (k$)'],df2['Spending Score (1-100)'],color='red')
plt.scatter(df3['Annual Income (k$)'],df3['Spending Score (1-100)'],color='blue')
plt.scatter(df4['Annual Income (k$)'],df4['Spending Score (1-100)'],color='purple')
plt.scatter(df5['Annual Income (k$)'],df5['Spending Score (1-100)'],color='yellow')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='black',marker='+',label='centroid')
plt.xlabel('annual income')
plt.ylabel('spending score')
plt.legend()
```

O/P:-

<matplotlib.legend.Legend at 0x7d77a46cd3f0>

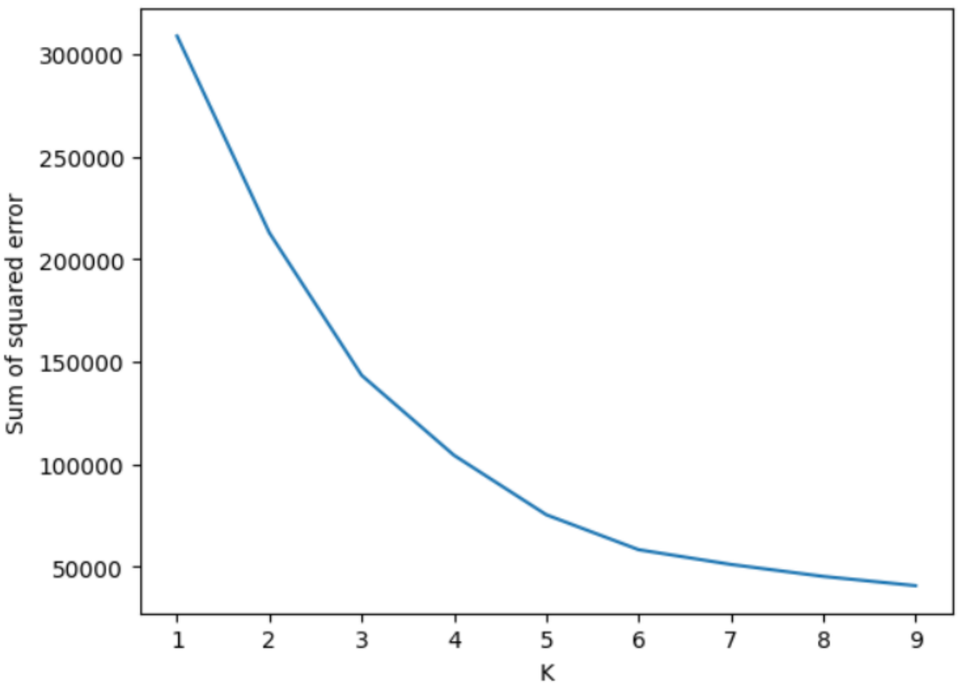


* Apply kmeans clustering for 3d data (age, Annual Income & Spending Score)

```
sse = []
k_rng = range(1,10)
for k in k_rng:
    km = KMeans(n_clusters=k,random_state=10)
    km.fit(df_copy[["Age","Annual Income (k$)","Spending Score (1-100)"]])
    sse.append(km.inertia_)

plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng,sse)
```

O/P:-




```
km = KMeans(n_clusters=5,random_state=32)
y_predicted = km.fit_predict(df_copy[['Age','Annual Income (k$)','Spending Score (1-100)']])
y_predicted
```

O/P:-

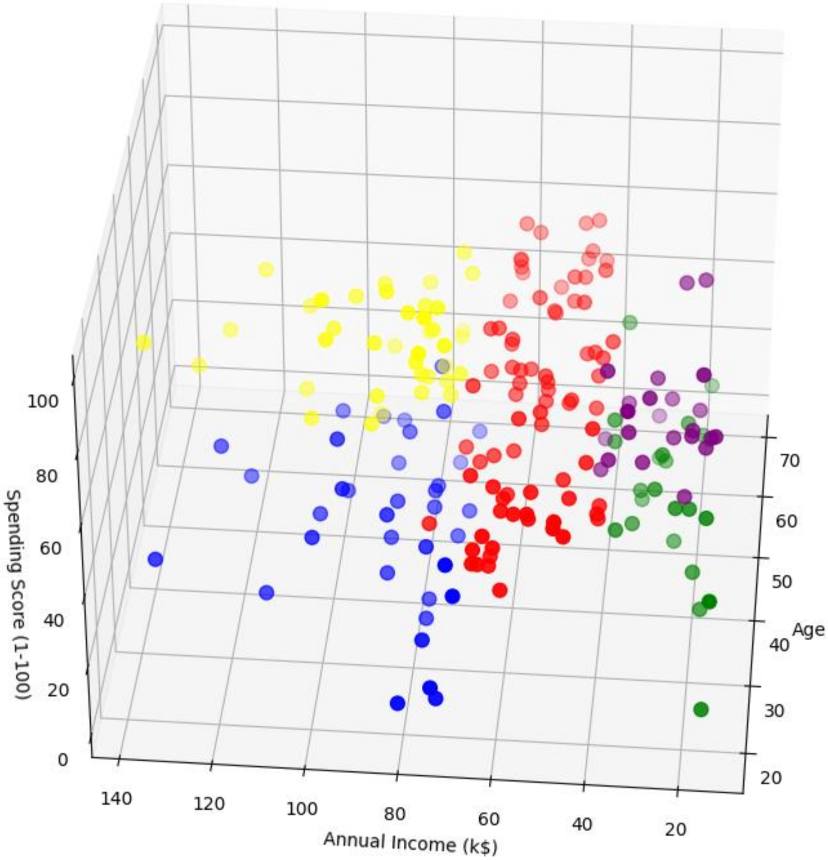
```
array([0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3,
       0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3,
       0, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 2, 4, 1, 4, 2, 4, 2, 4,
       2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 1, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4,
       2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4,
       2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4,
       2, 4], dtype=int32)
```

```
df_copy['cluster']=y_predicted
df_copy.head()
```

O/P:-

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

```
df1 = df_copy[df_copy.cluster==0]
df2 = df_copy[df_copy.cluster==1]
df3 = df_copy[df_copy.cluster==2]
df4 = df_copy[df_copy.cluster==3]
df5 = df_copy[df_copy.cluster==4]
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df1['Age'],df1['Annual Income
(k$)'],df1['Spending Score (1-
100)'],color='green',s=60)
ax.scatter(df2['Age'],df2['Annual Income
(k$)'],df2['Spending Score (1-100)'],color='red',s=60)
ax.scatter(df3['Age'],df3['Annual Income
(k$)'],df3['Spending Score (1-
100)'],color='blue',s=60)
ax.scatter(df4['Age'],df4['Annual Income
(k$)'],df4['Spending Score (1-
100)'],color='purple',s=60)
ax.scatter(df5['Age'],df5['Annual Income
(k$)'],df5['Spending Score (1-
100)'],color='yellow',s=60)
ax.view_init(35, 185)
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel('Spending Score (1-100)')
plt.show()
```



* Giving the summary

```
cust1=df_copy[df_copy["cluster"]==1]
print('Number of customer in 1st group=', len(cust1))
print('They are -', cust1["CustomerID"].values)
print("--------------------------------------------")
cust2=df_copy[df_copy["cluster"]==2]
print('Number of customer in 2nd group=', len(cust2))
print('They are -', cust2["CustomerID"].values)
print("--------------------------------------------")
cust3=df_copy[df_copy["cluster"]==0]
print('Number of customer in 3rd group=', len(cust3))
print('They are -', cust3["CustomerID"].values)
print("--------------------------------------------")
cust4=df_copy[df_copy["cluster"]==3]
print('Number of customer in 4th group=', len(cust4))
print('They are -', cust4["CustomerID"].values)
print("--------------------------------------------")
cust5=df_copy[df_copy["cluster"]==4]
print('Number of customer in 5th group=', len(cust5))
print('They are -', cust5["CustomerID"].values)
print("--------------------------------------------")
```

O/P:-

```
Number of customer in 1st group= 79
They are - [ 47  48  49  50  51  52  53  54  55  56  57
58  59  60  61  62  63  64
  65  66  67  68  69  70  71  72  73  74  75  76  77  78
79  80  81  82
  83  84  85  86  87  88  89  90  91  92  93  94  95  96
97  98  99 100
 101 102 103 104 105 106 107 108 109 110 111 112 113 114
115 116 117 118
 119 120 121 122 123 127 143]
--------------------------------------------
Number of customer in 2nd group= 36
They are - [125 129 131 133 135 137 139 141 145 147 149
151 153 155 157 159 161 163
 165 167 169 171 173 175 177 179 181 183 185 187 189 191
193 195 197 199]
--------------------------------------------
Number of customer in 3rd group= 23
They are - [ 1  3  5  7  9 11 13 15 17 19 21 23 25 27 29
31 33 35 37 39 41 43 45]
--------------------------------------------
Number of customer in 4th group= 23
They are - [ 2  4  6  8 10 12 14 16 18 20 22 24 26 28 30
32 34 36 38 40 42 44 46]
--------------------------------------------
Number of customer in 5th group= 39
They are - [124 126 128 130 132 134 136 138 140 142 144
146 148 150 152 154 156 158
 160 162 164 166 168 170 172 174 176 178 180 182 184 186
188 190 192 194
 196 198 200]
--------------------------------------------
```

3) Assignment on Association Rule Mining

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

```
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
data=pd.read_csv('/content/drive/My Drive/Colab Notebooks/Market_Basket_Optimisation.csv',header=None)

data.head()
```

O/P:-

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydant juice	frozen smoothie	spinach	olive oil
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	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

a) Data Preprocessing------------find columns with missing values (from column 0 to column 19)

```
data.columns[data.isna().any()]
```

O/P:- Index([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19], dtype='int64')

b) Generate list of transactions from the dataset

```
transactions = []
for i in range(len(data)):
    transactions.append([str(data.values[i, j]) for j in
range(len(data.columns)) if str(data.values[i, j]) !=
'nan'])
```

c) Train Apriori algorithm on the dataset (Find frequent itemsets using apriori)

```
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

one_hot_encoded =
pd.get_dummies(pd.DataFrame(transactions), prefix='',
prefix_sep='')
frequent_itemsets = apriori(one_hot_encoded,
min_support=0.005, use_colnames=True)
```

d) Visualize the list of rules

```
rules = association_rules(frequent_itemsets,
metric='lift', min_threshold=1)
print(rules)
```

O/P:-

	antecedents	consequents	antecedent support
0	(burgers)	(eggs)	0.010399
1	(eggs)	(burgers)	0.007866
2	(burgers)	(mineral water)	0.010399
3	(mineral water)	(burgers)	0.005866
4	(burgers)	(shrimp)	0.010399
5	(shrimp)	(burgers)	0.006399
6	(spaghetti)	(burgers)	0.008266
7	(burgers)	(spaghetti)	0.010399
8	(chocolate)	(french fries)	0.005066
9	(french fries)	(chocolate)	0.005333
10	(french fries)	(eggs)	0.005333

11	(eggs)	(french fries)	0.007866
12	(frozen vegetables)	(mineral water)	0.011598
13	(mineral water)	(frozen vegetables)	0.005866
14	(spaghetti)	(frozen vegetables)	0.008266
15	(frozen vegetables)	(spaghetti)	0.011598
16	(frozen vegetables)	(tomatoes)	0.011598
17	(tomatoes)	(frozen vegetables)	0.011332
18	(ground beef)	(mineral water)	0.007599
19	(mineral water)	(ground beef)	0.005866
20	(spaghetti)	(ground beef)	0.008266
21	(ground beef)	(spaghetti)	0.007599
22	(ground beef)	(herb & pepper)	0.007599
23	(herb & pepper)	(ground beef)	0.015331
24	(chocolate)	(mineral water)	0.005066
25	(mineral water)	(chocolate)	0.005866
26	(mineral water)	(eggs)	0.005866
27	(eggs)	(mineral water)	0.007866
28	(milk)	(mineral water)	0.006799
29	(mineral water)	(milk)	0.005866
30	(frozen vegetables)	(shrimp)	0.011598
31	(shrimp)	(frozen vegetables)	0.006399
32	(shrimp)	(mineral water)	0.006399
33	(mineral water)	(shrimp)	0.005866
34	(spaghetti)	(mineral water)	0.008266
35	(mineral water)	(spaghetti)	0.005866
36	(turkey)	(burgers)	0.061059
37	(burgers)	(turkey)	0.010399
38	(turkey)	(mineral water)	0.061059
39	(mineral water)	(turkey)	0.005866
40	(soup)	(mineral water)	0.005999
41	(mineral water)	(soup)	0.005866
42	(spaghetti)	(milk)	0.008266
43	(milk)	(spaghetti)	0.006799

	conviction	zhangs_metric
0	2.276073	0.996418
1	3.892432	0.993874
2	2.871943	1.001442
3	inf	0.996876
4	1.987202	0.997575
5	5.277874	0.993560
6	2.667621	0.991666
7	1.983469	0.993803

f) Find new rules by increasing minimum confidence

```
new_rules = association_rules(frequent_itemsets,
metric='confidence', min_threshold=0.3)
print(new_rules)
```

O/P:-

```
                antecedents            consequents  antecedent support  \
0                 (burgers)                 (eggs)            0.010399
1                    (eggs)              (burgers)            0.007866
2                 (burgers)        (mineral water)            0.010399
3           (mineral water)              (burgers)            0.005866
4                 (burgers)               (shrimp)            0.010399
5                  (shrimp)              (burgers)            0.006399
6               (spaghetti)              (burgers)            0.008266
7                 (burgers)            (spaghetti)            0.010399
8               (chocolate)         (french fries)            0.005066
9            (french fries)            (chocolate)            0.005333
10           (french fries)                 (eggs)            0.005333
11                   (eggs)         (french fries)            0.007866
12      (frozen vegetables)        (mineral water)            0.011598
13          (mineral water)    (frozen vegetables)            0.005866
14              (spaghetti)    (frozen vegetables)            0.008266
15      (frozen vegetables)            (spaghetti)            0.011598
16      (frozen vegetables)             (tomatoes)            0.011598
17               (tomatoes)    (frozen vegetables)            0.011332
18            (ground beef)        (mineral water)            0.007599
19          (mineral water)          (ground beef)            0.005866
20              (spaghetti)          (ground beef)            0.008266

    consequent support   support  confidence        lift  leverage  \
0             0.007866  0.005866    0.564103   71.717514  0.005784
1             0.010399  0.005866    0.745763   71.717514  0.005784
2             0.005866  0.006799    0.653846  111.465909  0.006738
3             0.010399  0.006799    1.159091  111.465909  0.006738
4             0.006399  0.005199    0.500000   78.135417  0.005133
5             0.010399  0.005199    0.812500   78.135417  0.005133
6             0.010399  0.005199    0.629032   60.491935  0.005113
7             0.008266  0.005199    0.500000   60.491935  0.005113
8             0.005333  0.005199    1.026316  192.459868  0.005172
9             0.005066  0.005199    0.975000  192.459868  0.005172
10            0.007866  0.005066    0.950000  120.778814  0.005024
11            0.005333  0.005066    0.644068  120.778814  0.005024
12            0.005866  0.006932    0.597701  101.894462  0.006864
13            0.011598  0.006932    1.181818  101.894462  0.006864
14            0.011598  0.007066    0.854839   73.702818  0.006970
15            0.008266  0.007066    0.609195   73.702818  0.006970
16            0.011332  0.005733    0.494253   43.616362  0.005601
17            0.011598  0.005733    0.505882   43.616362  0.005601
```

4) Assignment on Improving Performance of Classifier Models

A SMS unsolicited mail (every now and then known as cell smartphone junk mail) is any junk message brought to a cellular phone as textual content messaging via the Short Message Service (SMS).

Use probabilistic approach (Naive Bayes Classifier / Bayesian Network) to implement SMS Spam Filtering system. SMS messages are categorized as SPAM or HAM using features like length of message, word depend, unique keywords etc.

Download Data -Set from: http://archive.ics.uci.edu/ml/datasets/sms+spam+collection

This dataset is composed by just one text file, where each line has the correct class followed by the raw message.

a. Apply Data pre-processing (Label Encoding, Data Transformation….) techniques if necessary.

b. Perform data-preparation (Train-Test Split)

c. Apply at least two Machine Learning Algorithms and Evaluate Models

d. Apply Cross-Validation and Evaluate Models and compare performance.

e. Apply hyper parameter tuning and evaluate models and compare performance

```
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
df= pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/spam.csv')
df.groupby('Category').describe()
```

O/P:-

	Message			
	count	unique	top	freq
Category				
ham	4825	4516	Sorry, I'll call later	30
spam	747	641	Please call our customer service representativ...	4

```
import numpy as np
df['spam']=(df['Category']=='spam').astype(np.int8)
df.head()
```

O/P:-

	Category	Message	spam
0	ham	Go until jurong point, crazy.. Available only ...	0
1	ham	Ok lar... Joking wif u oni...	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	ham	U dun say so early hor... U c already then say...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	0

a) Identify Missing Values

```
df.columns[df.isna().any()]
```

O/P:- **Index([], dtype='object')**

b) Data Preparation

```
from sklearn.model_selection import train_test_split as
tts
X_train,X_test,y_train,y_test =
tts(df.Message,df.spam,test_size=0.2,random_state=42)
```

c) Apply Multinomial Naive bayes algorithm to classify SPAM message with CountVectorizer for feature Extraction

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import
CountVectorizer
clf=Pipeline([
    ('vectorizer',CountVectorizer()),
    ('nb',MultinomialNB())
])
clf.fit(X_train,y_train)
```

d) Evaluate model

```
emails=[
    'hey mohan,can we meet?',
    'upto 20% discount on parking, dont miss the order'
]
clf.predict(emails)
```

O/P:- **array([0, 1], dtype=int8)**

e) Compare performance

```
clf.score(X_test,y_test)
```

O.P:- **0.9919282511210762**

5) 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.

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.

```
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd
import numpy as np
df=pd.read_csv('/content/drive/My Drive/Colab Notebooks/Admission_Predict.csv')
df
```

O/P:-

	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
...	...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 9 columns

* Identify missing value

```
df.columns[df.isna().any()]
```

O/P :- Index([], dtype='object')

* Transforms the continuous "Chance of Admit " value into a binary classification (admitted or not admitted) based on a threshold of 0.5.

```
df['admitted'] = df['Chance of Admit '].apply(lambda x: 1
if x >= 0.5 else 0)
df.drop(['Chance of Admit '],axis=1,inplace=True)
df.head()
```

O/P:-

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

* Create a new dataframe x that excludes the columns "Serial No." and "admitted" from the original dataframe df.

```
x=df.drop(['Serial No.','admitted'],axis=1)
y=df['admitted']
```

* Data Preparation

```
from sklearn.model_selection import train_test_split as tts
x_train,x_test,y_train,y_test=tts(x,y,test_size=0.2,random_sta
te=42)
x_test.shape
```

O/P:- (80, 7)

* Apply Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier()
clf.fit(x_train,y_train)
```

* Evaluate the model

```
clf.score(x_test,y_test)
```

O/P :- 0.875

```
y_pred=clf.predict(x_test)

from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
cm
```

O/P:-

```
array([[ 4,  6],
       [ 4, 66]])
```