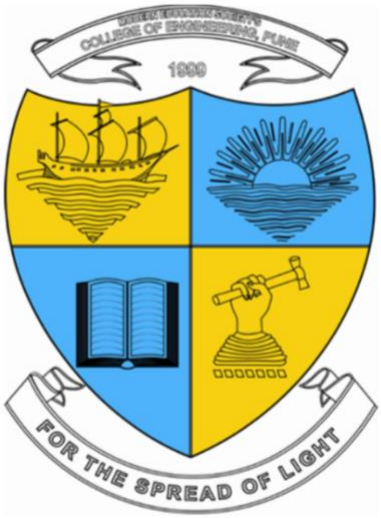
INTERNSHIP REPORT

A report submitted in partial fulfilment of the requirements for the Award of Degree of

BACHELOR OF ENGINEERING

in

ELECTRONICS & TELECOMMUNICATION ENGINEERING



SUBMITTED BY:

Rutuja Borawake(72147599L)

(Duration: 4 weeks)

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING

Modern Education Society's College of Engineering, Pune Approved by AICTE, Affiliated to SPPU, Pune Maharashtra [2022-2023]

MODERN EDUCATION SOCIETY'S COLLEGE OF ENGINEERING,PUNE.

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the "Internship report" submitted by Rutuja Borawake(72147599L) is work done by her and submitted during 2021-22 academic year, in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF ENGINEERING in ELECTRONICS & TELECOMMUNICATION ENGINEERING**, at **LETS GROW MORE**

Prof. Yogita Ajgar

Department Internship Coordinator

Examiner

Dr. Pranoti Mane

Head of the Department

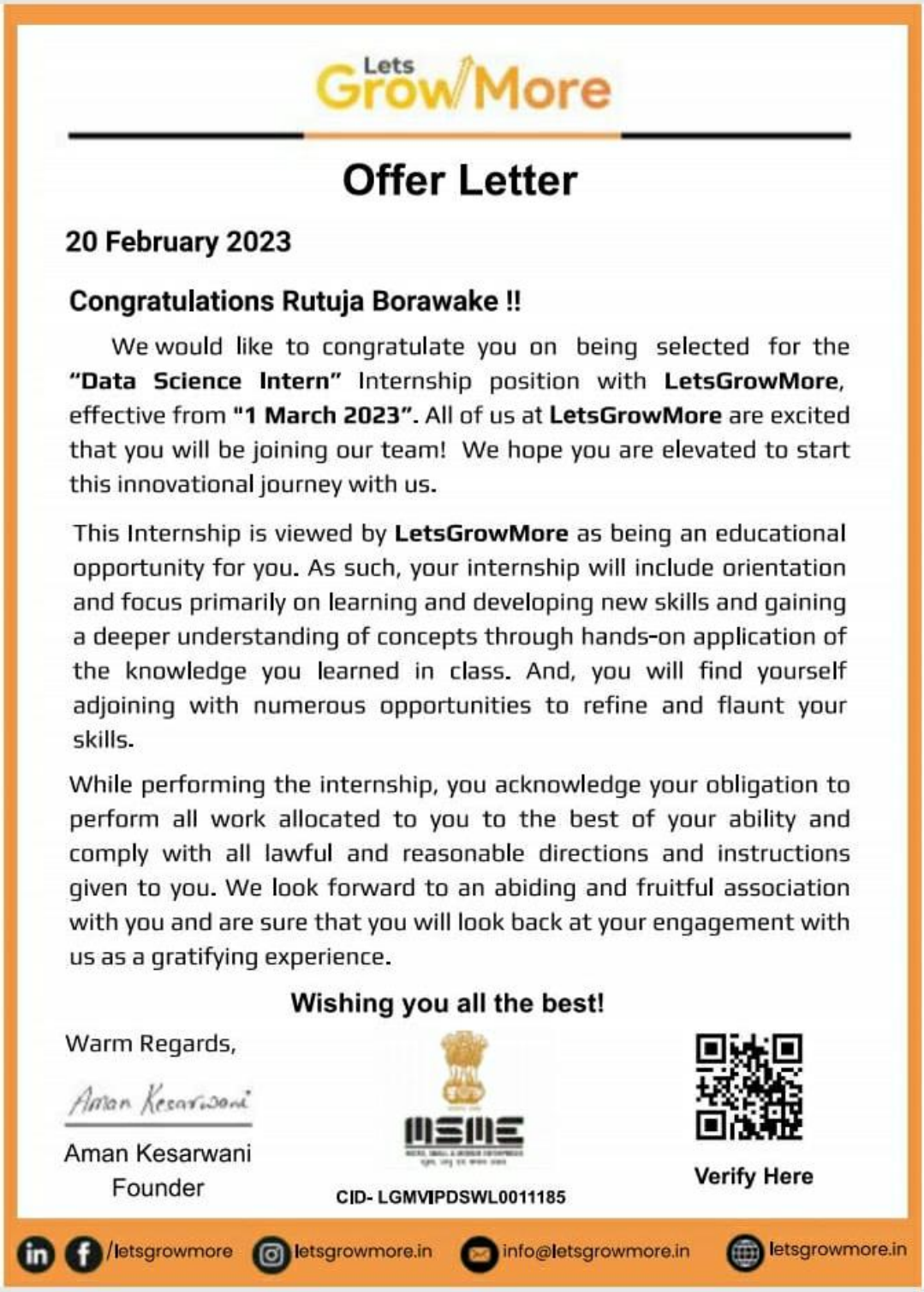
Department of E&TC

Date

Pune

INTERNSHIP BADGE

INTERNSHIP OFFER LETTER



Lets GrowMore

Offer Letter

20 February 2023

Congratulations Rutuja Borawake !!

We would like to congratulate you on being selected for the **"Data Science Intern"** Internship position with **LetsGrowMore**, effective from **"1 March 2023"**. All of us at **LetsGrowMore** are excited that you will be joining our team! We hope you are elevated to start this innovational journey with us.

This Internship is viewed by **LetsGrowMore** as being an educational opportunity for you. As such, your internship will include orientation and focus primarily on learning and developing new skills and gaining a deeper understanding of concepts through hands-on application of the knowledge you learned in class. And, you will find yourself adjoining with numerous opportunities to refine and flaunt your skills.

While performing the internship, you acknowledge your obligation to perform all work allocated to you to the best of your ability and comply with all lawful and reasonable directions and instructions given to you. We look forward to an abiding and fruitful association with you and are sure that you will look back at your engagement with us as a gratifying experience.

Wishing you all the best!

Warm Regards,

Aman Kesarwani

Aman Kesarwani
Founder

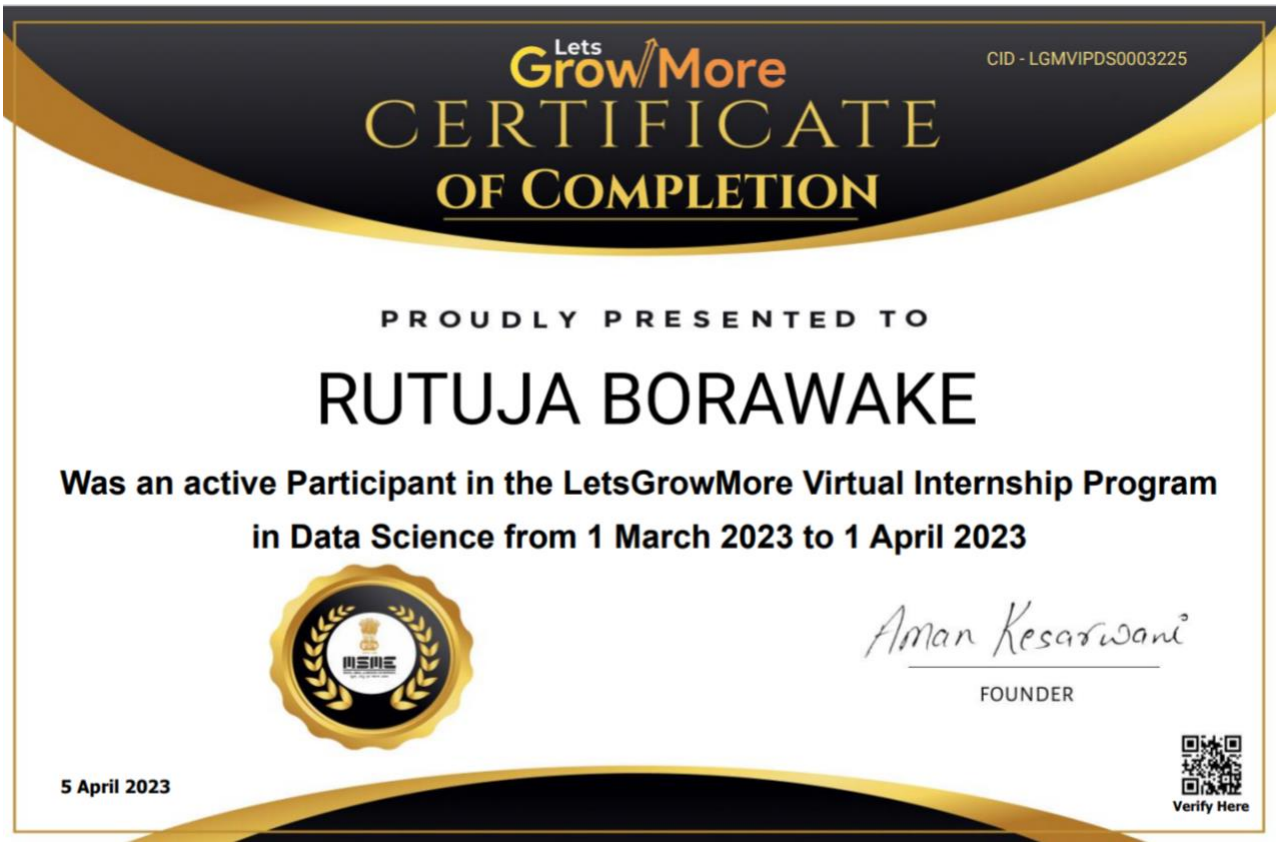
MSME

CID- LGMVIPDSWL0011185

Verify Here

/letsgrowmore letsgrowmore.in info@letsgrowmore.in letsgrowmore.in

INTERNSHIP COMPLETION CERTIFICATE



Course Objective

• Will expose technical students to the industrial environment, which cannot be simulated in the classroom and hence creating competent professionals for the industry.

• Provide possible opportunities to learn, understand and sharpen the real time technical / managerial skills required at the job.

• Exposure to the current technological developments relevant to the subject area of training.

• Experience gained from the 'Internship' will be used in classroom discussions.

• Create conditions conducive to quest for knowledge and its applicability on the job.

• Learn to apply the Technical knowledge in real industrial situations.

• Gain experience in writing Technical reports/projects.

• Expose students to the engineer's responsibilities and ethics.

• Familiarize with various materials, processes, products and their applications along with relevant aspects of quality control.

• Promote academic, professional and/or personal development.

• Expose the students to future employers.

• Understand the social, economic and administrative considerations that influence the working environment of industrial organizations.

• Understand the psychology of the workers and their habits, attitudes and approach to problem solving.

Course Outcomes On completion of the internship, learner will be able to –

CO1: To develop professional competence through internship.

CO2: To apply academic knowledge in a personal and professional environment.

CO3: To build the professional network and expose students to future employees.

CO4: Apply professional and societal ethics in their day to day life.

CO5: To become a responsible professional having social, economic and administrative considerations. CO6: To make own career goals and personal aspirations.

ACKNOWLEDGEMENT

It gives me immense pleasure in presenting the report on Internship at Lets Grow More. I would like to take this opportunity to thank Amar Kesarwani atL ets Grow more., I offer my sincere phrases of thanks to Prof.Yogita Ajgar for their guidance and constant supervision as well as providing necessary information during Internship work. I express my deepest gratitude to Dr. P. P. Mane, Head of E&TC department for their kind co-operation. Finally I would like to express my gratitude towards my parents and all teaching and non-teaching staff members of E&TC department for their kind co-operation and encouragement which help us in completion of this Internship.

ABSTRACT

Company Information

LetsGrowMore is a ground-based organisation that aims at building the future through nourishing the present. We at LetsGrowMore believe in making our youth especially the students self-aware and exploring the untouched world of technology and tremendous growth-making fields and our belief finally took us where we are standing today. Today we are an officially MSME registered start-up

LETS GROW MORE VIRTUAL INTERNSHIP PROGRAM

LGMVIP is a 4-week Virtual Internship Program where you are provided internship opportunities for beginners/students who which to excel their career in various domains such as Web Development, Data Science, Campus Ambassadors, Technical Content Writer, etc.

It is a great initiativ by AMAN KESARWANI sir(Founder of lets grow more)

INDEX

INTRODUCTION

Internship is a platform where we get an opportunity to explore the fields of our interests at an industrial level and professional environment. During this 6-week internship, I had great experience as Jr. R&D intern at Jayashree Electron Pvt. Ltd. I explored many new concepts and technologies. I have also developed important technical as well as non-technical skills. In the presented report, I have put down all the things that I learnt through the course of my of internship

Learning Objectives/Internship Objective

- Internships are generally thought of to be reserved for college students looking to gain experience in a particular field. However, a wide array of people can benefit from Training Internships in order to receive real world experience and develop their skills.
- An objective for this position should emphasize the skills you already possess in the area and your interest in learning more
- Internships are utilized in a number of different career fields, including architecture, engineering, healthcare, economics, advertising and many more.
- Some internship is used to allow individuals to perform scientific research while others are specifically designed to allow people to gain first-hand experience working.
- Utilizing internships is a great way to build your resume and develop skills that can be emphasized in your resume for future jobs. When you are applying for a Training Internship, make sure to highlight any special skills or talents that can make you stand apart from the rest of the applicants so that you have an improved chance of landing the position.

TECHNOLOGY

GOOGLE COLAB

Colab notebooks are Jupyter notebooks that are hosted by **Colab**. To learn more about the Jupyter project, see jupyter.org.

Data science

With Colab you can harness the full power of popular Python libraries to analyze and visualize data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualize it. To edit the code, just click the cell and start editing.

[ ]

```
import numpy as np

from matplotlib import pyplot as plt

ys = 200 + np.random.randn(100)

x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')

plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)

plt.title("Sample Visualization")

plt.show()
```

WEEKLY OVERVIEW OF INTERNSHIP ACTIVITIES

WEEK 1

Date	Day	Work Done
2-3-2023	Thursday	Study/Overview of task list
3-3-2023	Friday	Iris Flower Classification
5-3-2023	Saturday	Stock market Prediction

WEEK2

Date	Day	Work Done
8-3-2023	Wednesday	Image to Pencil Sketch with Python
11-3-2023	Saturday	Develop Neural Network to read handwriting
12-3-2023	Sunday	Prediction using Decision Tree Algorithm

WEEK3

Date	Day	Work Done
13-3-2023	Monday	Exploratory Data Analysis on Dataset - Terrorism
19-3-2023	Sunday	Music Recommendation

WEEK4

Date	Day	Work Done
20-3-2023	Monday	Next Word Prediction
29-3-2023	Wednesday	Submission of form internship

WEEEK1

DATE:2-03-2023

WORK-Study/Overview of tasks-

TASK LIST PROVIDED BY LETS GROW MORE GIVEN BY

1.Iris Flower classification using Ml

2.Stock Market Prediction

3.Image to pencil sketch

4.Develop Neural Network to read handwriting

5.Prediction using Decision tree

6.Exploratory data analysis on dataset-Terrorism

7.Music Recommendation

8.Next Word Prediction

9.Handwritten equation using cnn

10..ML face recognition to detect mood and suggest songs accordingly

ONLY TWO TASK WERE MANDATORY FOR CERTIFICATE BUT IN ORDER TO LEARN AND BETTER UNDERSTANDING I COMLETED EIGHT TASKS

Week1

Date-3-3-2023

Work completion of task 1 Iris flower classification

This particular ML project is usually referred to as the "Hello World" of Machine Learning. The iris flowers dataset contains numeric attributes, and it is perfect for beginners to learn about supervised ML algorithms, mainly how to load and handle data. Also, since this is a small dataset, it can easily fit in memory without requiring special transformations capabilities.

LETS GROW MORE VIRTUAL INTERNSHIP PROGRAM

Data Science Internship

Author-Rutuja borawake

Task No.1

Level-Beginner

Task Name:- Iris Flowers Classification ML Project

This particular ML project is usually referred to as the "Hello World" of Machine Learning. The iris flowers dataset contains numeric attributes, and it is perfect for beginners to learn about supervised ML algorithms, mainly how to load and handle data. Also, since this is a small dataset, it can easily fit in memory without requiring special transformations or scaling capabilities.

Importing Libraries

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

Importing Dataset

```
from google.colab import files;
upload=files.upload()
```

```
Choose Files  No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving iris.data.csv to iris.data.csv
```

```
df=pd.read_csv("iris.data.csv")
```

```
df.head()#top 5 values
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
df.tail() #last 5 values
```

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

```
df.shape #no. of rows and columns
```

```
(150, 5)
```

```
df.isnull() #returns a Dataframe object where all the values are replaced with a boolean, True for null,otherwise false
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

150 rows × 5 columns

```
df.isnull().sum()#returns no of missing values
```

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

```
df.describe() # used to view statistical details
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
df.columns
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

```
df.nunique()
```

```
sepal_length    35
sepal_width     23
petal_length    43
petal_width     22
species          3
dtype: int64
```

```
df.species.nunique()
```

```
3
```

```
df.species.value_counts()
```

```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: species, dtype: int64
```

```
df.max()
```

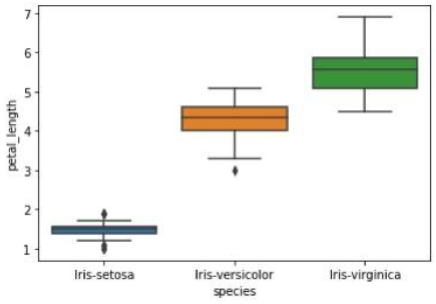
```
sepal_length              7.9
sepal_width               4.4
petal_length              6.9
petal_width               2.5
species         Iris-virginica
dtype: object
```

```
df.min()
```

```
sepal_length            4.3
sepal_width             2.0
petal_length            1.0
petal_width             0.1
species         Iris-setosa
dtype: object
```

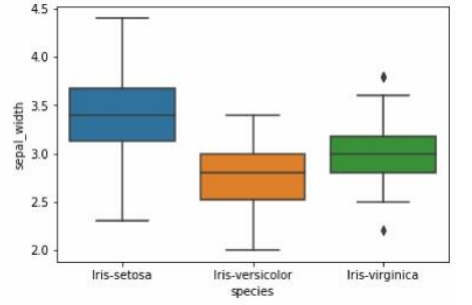
Visualization

```
#boxplot
sns.boxplot(x="species",y='petal_length',data=df)
plt.show()
```



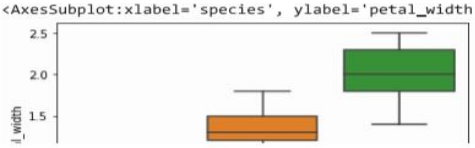

```
sns.boxplot(x="species",y="sepal_width",data=df)
```

```
<AxesSubplot:xlabel='species', ylabel='sepal_width'>
```

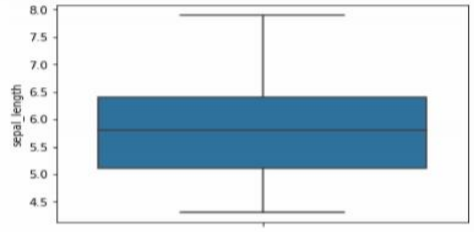



```
sns.boxplot(x="species",y="petal_width",data=df)
```

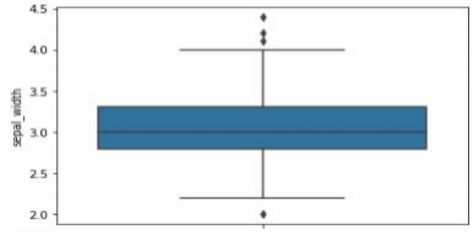
```
<AxesSubplot:xlabel='species', ylabel='petal_width'>
```




```
sns.boxplot(y="sepal_length",data=df);
plt.show()
```

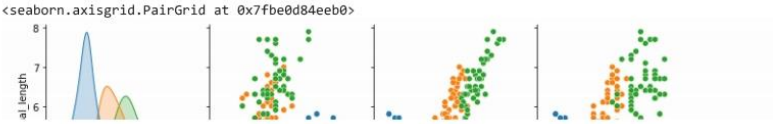



```
sns.boxplot(y="sepal_width",data=df);
plt.show()
```



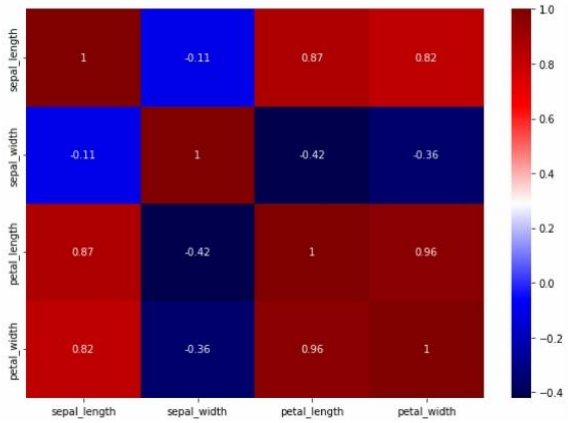

```
sns.pairplot(df,hue="species") #Pairplot is a pairwise relationship in dataset
```

```
<seaborn.axisgrid.PairGrid at 0x7fbe0d84eeb0>
```



Data Preprocessing

```
#heatmap is used to show data in graphical format
plt.figure(figsize=(10,7))
sns.heatmap(df.corr(),annot=True,cmap="seismic")
plt.show()
```



Label Encoder

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
df['species']=le.fit_transform(df['species'])
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
X=df.drop(columns=['species'])
y=df['species']
X[:5]
```

	sepal_length	sepal_width	petal_length	petal_width

```
y[:5]
```

```
0    0
1    0
2    0
3    0
4    0
Name: species, dtype: int64
```

Splitting the dataset into train and test **

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X, y, test_size=0.3, random_state=1)
```

Selecting the models and metrics

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score


lr=LogisticRegression()
knn=KNeighborsClassifier()
svm=SVC()
nb=GaussianNB()
dt=DecisionTreeClassifier()
rf=RandomForestClassifier()
```

Training and evaluating the models

```
models=[lr,knn,svm,nb,dt,rf]
scores=[]


for model in models:
  model.fit(X_train,y_train)
  y_pred=model.predict(X_test)
  scores.append(accuracy_score(y_test,y_pred))
  print("Accuracy of"+ type(model).__name__+"is",accuracy_score(y_test,y_pred))
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Accuracy ofLogisticRegressionis 0.9777777777777777
Accuracy ofKNeighborsClassifieris 0.9777777777777777
Accuracy ofSVCis 0.9777777777777777
Accuracy ofGaussianNBis 0.9333333333333333
Accuracy ofDecisionTreeClassifieris 0.9555555555555556
Accuracy ofRandomForestClassifieris 0.9555555555555556
```

Week 1

Date-5-03-2023

Stock Market Prediction

LETS GROW MORE VIRTUAL INTERNSHIP PROGRAM

Data Science Internship

Author-Rutuja Borawake

Level-Beginner

Task Name-Stock Market Prediction And Forecasting Using Stacked LSTM

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Reading Dataset

```
df = pd.read_csv('https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NSE-TATAGLOBAL.csv')
```

```
df.head()
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914	7162.35
1	2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859	11859.95
2	2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909	5248.60
3	2018-09-25	233.30	236.75	232.00	236.25	236.10	2349368	5503.90
4	2018-09-24	233.55	239.20	230.75	234.00	233.30	3423509	7999.55

```
df.tail()
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
2030	2010-07-27	117.6	119.50	112.00	118.80	118.65	586100	694.98
2031	2010-07-26	120.1	121.00	117.10	117.10	117.60	658440	780.01
2032	2010-07-23	121.8	121.95	120.25	120.35	120.65	281312	340.31
2033	2010-07-22	120.3	122.00	120.25	120.75	120.90	293312	355.17
2034	2010-07-21	122.1	123.00	121.05	121.10	121.55	658666	803.56

```
df.shape
```

```
(2035, 8)
```

```
df.describe()
```

	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
count	2035.000000	2035.000000	2035.000000	2035.000000	2035.00000	2.035000e+03	2035.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2035 entries, 0 to 2034
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Date                  2035 non-null   object 
 1   Open                  2035 non-null   float64
 2   High                  2035 non-null   float64
 3   Low                   2035 non-null   float64
 4   Last                  2035 non-null   float64
 5   Close                 2035 non-null   float64
 6   Total Trade Quantity  2035 non-null   int64  
 7   Turnover (Lacs)       2035 non-null   float64
dtypes: float64(6), int64(1), object(1)
memory usage: 127.3+ KB
```

```
df.isnull().sum()
```

```
Date                    0
Open                    0
High                    0
Low                     0
Last                    0
Close                   0
Total Trade Quantity    0
Turnover (Lacs)         0
dtype: int64
```

```
stk=df.copy()
stk['Date'].min()
stk['Date'].max()
```

```
'2018-09-28'
```

```
stk.head()
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914	7162.35
1	2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859	11859.95
2	2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909	5248.60
3	2018-09-25	233.30	236.75	232.00	236.25	236.10	2349368	5503.90
4	2018-09-24	233.55	239.20	230.75	234.00	233.30	3423509	7999.55

```
stk['Date'] = pd.to_datetime(stk['Date'])
```

```
stk.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Last', 'Close', 'Total Trade Quantity',
       'Turnover (Lacs)'],
      dtype='object')
```

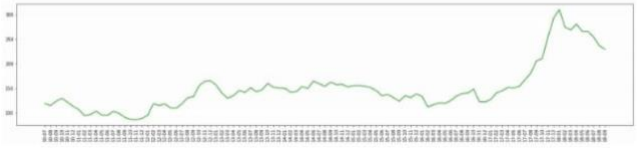
```
stk.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2035 entries, 0 to 2034
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   Date                  2035 non-null   datetime64[ns]
 1   Open                  2035 non-null   float64       
 2   High                  2035 non-null   float64       
 3   Low                   2035 non-null   float64       
 4   Last                  2035 non-null   float64       
 5   Close                 2035 non-null   float64       
 6   Total Trade Quantity  2035 non-null   int64         
 7   Turnover (Lacs)       2035 non-null   float64       
dtypes: datetime64[ns](1), float64(6), int64(1)
memory usage: 127.3 KB
```

```
stk['Month Year'] = stk['Date'].apply(lambda x: x.strftime ('%y-%m'))
stk['Year'] = stk['Date'].apply(lambda x: x.strftime ('%y'))
stk.head()
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)	Month Year	Year
0	2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914	7162.35	18-09	18
1	2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859	11859.95	18-09	18
2	2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909	5248.60	18-09	18

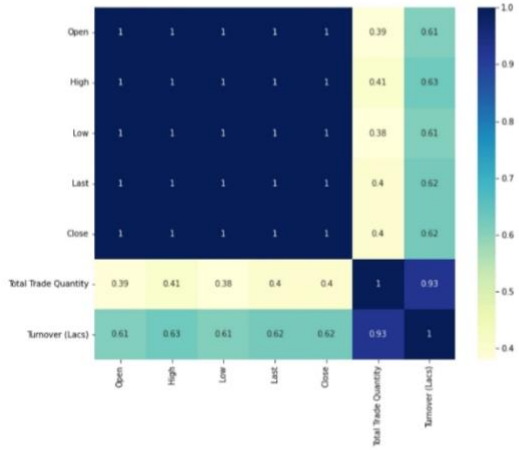
```
a = stk.groupby('Month Year')['Close'].mean()
plt.figure(figsize=(25,5))
plt.plot(a, color = 'green')
plt.xticks(rotation = 'vertical')
plt.show()
```



```
pc = stk[['Year', 'Total Trade Quantity']]
grp_pc = pc.groupby('Year')['Total Trade Quantity'].mean().sort_values(ascending = False).reset_index()

plt.pie(grp_pc['Total Trade Quantity'], labels=grp_pc['Year'])
plt.show()
```


```
trd = stk[['Year','Turnover (Lacs)']]
a = trd.groupby('Year')['Turnover (Lacs)'].mean().sort_values(ascending = False).reset_index()

sns.barplot(a['Year'], a['Turnover (Lacs)'])
plt.show()
```


```
plt.figure(figsize = (10,8))
sns.heatmap(stk.corr(), annot = True, cmap = 'YlGnBu')
plt.show()
```




```
stk.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Last', 'Close', 'Total Trade Quantity',
       'Turnover (Lacs)', 'Month Year', 'Year'],
      dtype='object')
```

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

x = stk[['Close', 'Total Trade Quantity']]
y = stk['Turnover (Lacs)']

x.head()
```

	Close	Total Trade Quantity
0	233.75	3069914
1	233.25	5082859
2	234.25	2240909
3	236.10	2349368
4	233.30	3423509

```
y.head()
```

```
0     7162.35
1    11859.95
2     5248.60
```

```
3     5503.90
4     7999.55
Name: Turnover (Lacs), dtype: float64
```

```
X_train, X_test, y_train, y_test = train_test_split(x,y, test_size = 0.3, random_state = 0)
```

```
lr=LinearRegression()
lr.fit(X_train, y_train)
X_train.shape, X_test.shape
y_train.shape, y_test.shape
```

```
((1424,), (611,))
```

```
y_train = y_train.values.reshape(-1,1)
y_test = y_test.values.reshape(-1,1)
```

```
y_train.shape, y_test.shape
lr.score(X_test, y_test)
y_train_pred = lr.predict(X_train)
y_train_pred
y_test_pred = lr.predict(X_test)
y_test_pred
```

WEEK2

DATE-8-3-2023

IMAGE TO PENCIL SKETCH

LETS GROW MORE VIRTUAL INTERNSHIP PROGRAM

Data Science Intern

Author-Rutuja Borawake

Level-Beginner

Task name-Image to Pencil Sketch with Python

```
import cv2
import matplotlib.pyplot as plt
```

```
from google.colab import files
upload=files.upload()
```

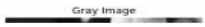
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
image = cv2.imread("Frangipani rust_ what it is and how to treat it.jfif")
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plt.figure(figsize=(8,6))
plt.imshow(image)
plt.axis('off')
plt.title('Original Image')
plt.show()
```

Original Image

```
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plt.figure(figsize=(8,6))
plt.imshow(gray_image, cmap='gray')
plt.axis('off')
plt.title('Gray Image')
plt.show()
```

Gray Image



```
inverted_image = 255 - gray_image
plt.figure(figsize=(8,6))
plt.imshow(inverted_image,cmap='gray')
plt.axis('off')
plt.title('Inverted image')
plt.show()
```

Inverted image

```
blurred_image = cv2.GaussianBlur(inverted_image, (21, 21), 0)
plt.figure(figsize=(8,6))
plt.imshow(blurred_image,cmap='gray')
plt.axis('off')
plt.title('Blurred Inverted image')
plt.show()
```

Blurred Inverted image

```
inverted_blurred = 255 - blurred_image
pencil_sketch = cv2.divide(gray_image, inverted_blurred, scale=256.0)
plt.figure(figsize=(8,6))
plt.imshow(pencil_sketch,cmap='gray')
plt.axis('off')
plt.title('Pencil Sketch')
plt.show()
```

Pencil Sketch

```
image = cv2.imread("Frangipani rust_ what it is and how to treat it.jfif")
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plt.figure(figsize=(8,6)).add_subplot(1, 2, 1)
plt.imshow(image)
plt.axis('off')
plt.title("Original Image")
pencil_sketch = cv2.divide(gray_image, inverted_blurred, scale=256.0)
plt.figure(figsize=(8,6)).add_subplot(1, 2, 2)
plt.imshow(pencil_sketch,cmap='gray')
plt.axis('off')
plt.title("Pencil Sketch")
plt.show()
```

Original Image

Pencil Sketch

WEEK 2

DATE :11-03-2023

DEVELOP NEURAL NETWORK TO READ HANDWRITING

Importing Libraries

In [4]:
```
from tensorflow import keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout
from tensorflow.keras.optimizers import RMSprop
```

Load the MNIST dataset. We make a split of train and test set with 60k samples in training and 10K samples in test

In [5]:
```
(mnist_train_images, mnist_train_labels),(mnist_test_images, mnist_test_labels) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

In [6]:
```
train = mnist_train_images.reshape(60000,784)
test = mnist_test_images.reshape(10000,784)
```

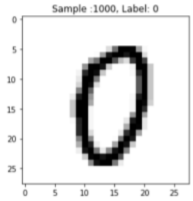
In [7]:
```
train = train.astype('float32')
test = test.astype('float32')
train /= 255
test /= 255
```

In [8]:
```
train_labels = keras.utils.to_categorical(mnist_train_labels,10)
test_labels = keras.utils.to_categorical(mnist_test_labels,10)
```

Visualizing the dataset.

In [9]:
```
import matplotlib.pyplot as plt
def display(num):
    print(train_labels[num])
    label = train_labels[num].argmax(axis = 0)
    image = train[num].reshape([28,28])
    plt.title('Sample :%d, Label: %d' % (num,label))
    plt.imshow(image,cmap=plt.get_cmap('gray_r'))
    plt.show()

display(1000)
```

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



Building Sequential Model in neural network

In [11]:
```
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10,activation = 'softmax'))
```

In [12]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               401920

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 512)               262656

 dropout_1 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 10)                5130

=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
```

Model Optimization

In [13]:
```
model.compile(loss = 'categorical_crossentropy',optimizer = RMSprop(),metrics = ['accuracy'])
```

Fitting the training data to the model

In [14]:
```
history = model.fit(train,train_labels,batch_size = 100,epochs = 10, verbose = 2,validation_data = (test,test_labels))
```

```
Epoch 1/10
600/600 - 12s - loss: 0.2414 - accuracy: 0.9244 - val_loss: 0.1252 - val_accuracy: 0.9614 - 12s/epoch - 21ms/step
Epoch 2/10
600/600 - 10s - loss: 0.1015 - accuracy: 0.9689 - val_loss: 0.0839 - val_accuracy: 0.9743 - 10s/epoch - 17ms/step
Epoch 3/10
600/600 - 11s - loss: 0.0745 - accuracy: 0.9774 - val_loss: 0.0738 - val_accuracy: 0.9779 - 11s/epoch - 19ms/step
Epoch 4/10
600/600 - 10s - loss: 0.0583 - accuracy: 0.9821 - val_loss: 0.0851 - val_accuracy: 0.9780 - 10s/epoch - 17ms/step
Epoch 5/10
600/600 - 9s - loss: 0.0485 - accuracy: 0.9847 - val_loss: 0.0719 - val_accuracy: 0.9807 - 9s/epoch - 15ms/step
Epoch 6/10
600/600 - 10s - loss: 0.0404 - accuracy: 0.9871 - val_loss: 0.0663 - val_accuracy: 0.9827 - 10s/epoch - 16ms/step
Epoch 7/10
600/600 - 10s - loss: 0.0345 - accuracy: 0.9896 - val_loss: 0.0720 - val_accuracy: 0.9823 - 10s/epoch - 16ms/step
Epoch 8/10
600/600 - 9s - loss: 0.0306 - accuracy: 0.9907 - val_loss: 0.0683 - val_accuracy: 0.9824 - 9s/epoch - 16ms/step
Epoch 9/10
600/600 - 9s - loss: 0.0284 - accuracy: 0.9911 - val_loss: 0.0784 - val_accuracy: 0.9822 - 9s/epoch - 15ms/step
Epoch 10/10
600/600 - 10s - loss: 0.0231 - accuracy: 0.9925 - val_loss: 0.0840 - val_accuracy: 0.9814 - 10s/epoch - 16ms/step
```

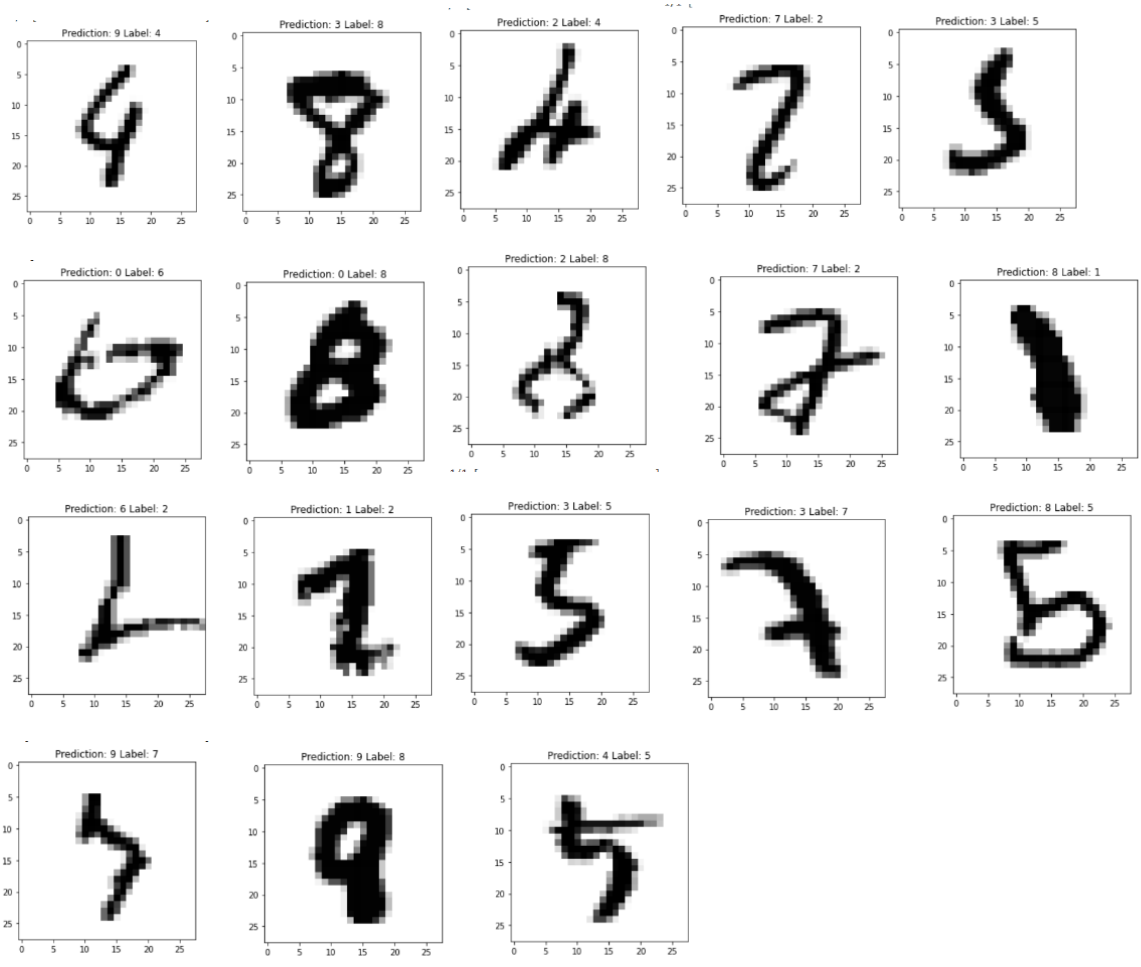
Model Evaluation

In [15]:
```
score = model.evaluate(test,test_labels,verbose = 0)
print("Test loss",score[0])
print("Test accuracy",score[1])
```

```
Test loss 0.0840328112244606
Test accuracy 0.9814000129699707
```

In [18]:
```
for x in range(1000):
    test_image = test[x,:].reshape(1,784)
    predicted_cat = model.predict(test_image).argmax()
    label = test_labels[x].argmax()
    if (predicted_cat != label):
        plt.title('Prediction: %d Label: %d' % (predicted_cat, label))
        plt.imshow(test_image.reshape([28,28]), cmap=plt.get_cmap('gray_r'))
        plt.show()
```

```
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 21ms/step
```



WEEK2

DATE-12-03-2023

PREDICTION USING DECISION TREE ALGORITHM

LETS GROW MORE VIRTUAL INTERNSHIP PROGRAM

Data Science Internship

Author-Rutuja Borawake

Level-Intermediate

Task no.-5

Task Name-Prediction using Decision Tree Algorithm:

Description-Create the Decision Tree classifier and visualize it graphically. The purpose is if we feed any new data to this classifier, it would be able to predict the right class accordingly.

Import Libraries

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

Reading Dataset

```
from google.colab import files
upload=files.upload()
```

Choose Files | Iris.csv

- **Iris.csv**(text/csv) - 5107 bytes, last modified: 3/12/2023 - 100% done

Saving Iris.csv to Iris.csv

```
df=pd.read_csv('Iris.csv')
df
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

Droping Unneccessary Columns

```
df=df.drop('Id',axis=1)
df
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica

```
df.describe()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Data Preprocessing

```
from sklearn.preprocessing import LabelEncoder

from sklearn.compose import ColumnTransformer

LE=LabelEncoder()

df.iloc[:,-1]=LE.fit_transform(df.iloc[:,-1])

df
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2

Splitting Data into Independent and Dependent Var

```
x=df.iloc[:,:-1]
x.head()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
y=df.iloc[:,-1]
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: Species, dtype: int64
```

Splitting the Dataset into train and test

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=50)

X_train.head()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
125	7.2	3.2	6.0	1.8
98	5.1	2.5	3.0	1.1
73	6.1	2.8	4.7	1.2
144	6.7	3.3	5.7	2.5
21	5.1	3.7	1.5	0.4

```
X_train.shape
```

```
(120, 4)
```

```
y_train.shape
```

```
(120,)
```

Model Building

```
from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier()

dt.fit(X_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

Comparison between actual and predicted output

```
y_pred=dt.predict(X_test)
y_pred
```

```
array([1, 1, 0, 0, 2, 2, 2, 0, 0, 1, 0, 2, 0, 2, 1, 0, 1, 0, 1, 2, 2, 1,
       0, 2, 1, 2, 1, 1, 1, 2])
```

```
y_test=np.array(y_test)
y_test
```

```
array([1, 1, 0, 0, 2, 2, 2, 0, 0, 1, 0, 2, 0, 2, 1, 0, 1, 0, 1, 1, 2, 1,
       0, 2, 1, 2, 1, 1, 1, 2])
```

Accuracy of Model

```
from sklearn.metrics import  accuracy_score
```

```
accuracy_score(y_pred,y_test)
```

```
0.9666666666666667
```

Confusion Matrix

```
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix(y_pred,y_test)
```

```
array([[ 9,  0,  0],
       [ 0, 11,  0],
       [ 0,  1,  9]])
```

Classification Report

```
from sklearn.metrics import classification_report
```

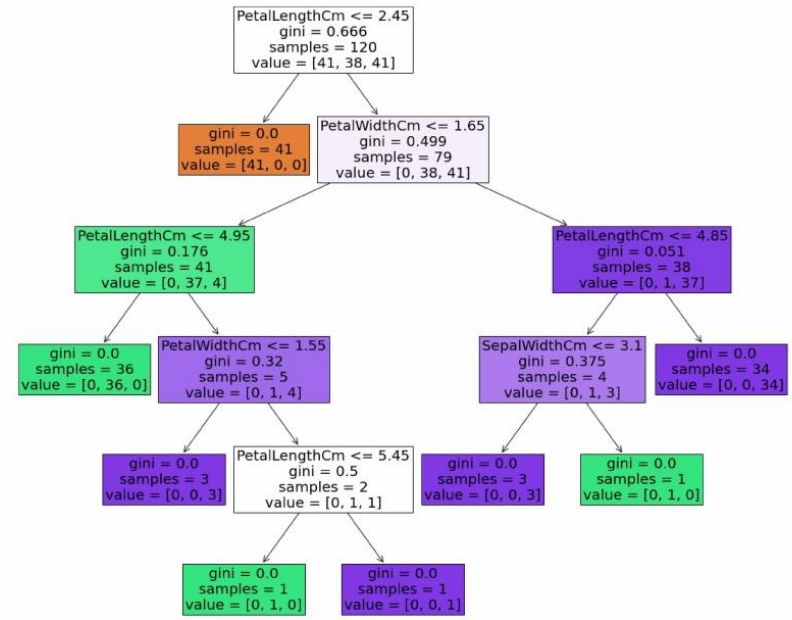
```
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       0.92      1.00      0.96        11
           2       1.00      0.90      0.95        10

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
```

Model Visualization

```
from sklearn import tree
```

```
plt.figure(figsize=(20,17))
dtviz=tree.plot_tree(dt,feature_names=x.columns,filled=True,fontsize=20)
```



WEEK 3

DATE-13-3-2023

EXPLORATORY DATA ANALYSIS ON DATASET-TERRORISM

LETS GROW MORE VIRTUAL INTERNSHIP PROGRAM

Data Science Internship

Author-Rutuja Borawake

Level-Intermediate

Task no.-6

Task Name-Exploratory Data Analysis on Dataset - Terrorism

As a security/defense analyst, try to find out the hot zone of terrorism.

Importing Libraries

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

Reading Dataset

```
from google.colab import files
upload=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Creating a copy...

```
0718dist.csv',encoding='latin1')
```

```
/usr/local/lib/python3.9/dist-packages/IPython/core/interactiveshell.py:3326: DtypeWarning: Columns (4,6,31,33,61,62,63,76,79,90,92,94,9
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
data.head()
```

	eventid	iyear	imonth	iday	approxdate	extended	resolution	country	country_txt	region	..
0	1.970000e+11	1970	7	2	NaN	0	NaN	58	Dominican Republic	2	.
1	1.970000e+11	1970	0	0	NaN	0	NaN	130	Mexico	1	.
2	1.970010e+11	1970	1	0	NaN	0	NaN	160	Philippines	5	.
3	1.970010e+11	1970	1	0	NaN	0	NaN	78	Greece	8	.
4	1.970010e+11	1970	1	0	NaN	0	NaN	101	Japan	4	.

5 rows × 135 columns

```
# Reading last few values from dataset
data.tail()
```

```
                          Target  \
0                     Julio Guzman
1          Nadine Chaval, daughter
2                         Employee
3                   U.S. Embassy
4                U.S. Consulate
...                            ...
181686                Checkpoint
181687          Hmeymim Air Base
181688                    Houses
181689                    Office
181690                   Unknown

                                                  Summary  \
0                                                     NaN
1                                                     NaN
2                                                     NaN
3                                                     NaN
4                                                     NaN
...                                                   ...
181686  12/31/2017: Assailants opened fire on a Somali...
181687  12/31/2017: Assailants launched mortars at the...
181688  12/31/2017: Assailants set fire to houses in K...
181689  12/31/2017: Assailants threw a grenade at a Fo...
181690  12/31/2017: An explosive device was discovered...

                                        Group  \
0                                       MANO-D
1                23rd of September Communist League
2                                     Unknown
3                                     Unknown
4                                     Unknown
...                                       ...
181686                             Al-Shabaab
181687                        Muslim extremists
181688  Bangsamoro Islamic Freedom Movement (BIFM)
181689                                Unknown
181690                                Unknown

                        Target_type Weapon_type Motive
                                    Unknown     NaN
                                    Unknown     NaN
                                    Unknown     NaN
3           Government (Diplomatic)  Explosives    NaN
4           Government (Diplomatic)  Incendiary    NaN
...                             ...         ...    ...
181686                     Military    Firearms    NaN
181687                     Military  Explosives    NaN
181688  Private Citizens & Property  Incendiary    NaN
181689         Government (General)  Explosives    NaN
181690                      Unknown  Explosives    NaN

[181691 rows x 19 columns]>
```

Creating a copy...

```
data.isnull().sum()
```

```
Year               0
Month              0
Extended           0
Day                0
Country            0
State            421
Region             0
City             434
Latitude        4556
Longitude       4557
AttackType         0
Killed         10313
Wounded        16311
Target           636
Summary        66129
Group              0
Target_type        0
Weapon_type        0
Motive        131130
dtype: int64
```

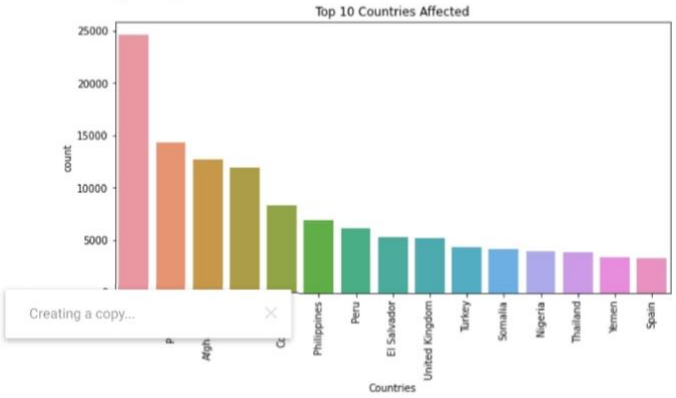
```
print(data['Country'].value_counts().head(10))
```

```
Iraq             24636
Pakistan         14368
```

```
Afghanistan       12731
India             11960
Colombia           8306
Philippines        6908
Peru               6096
El Salvador        5320
United Kingdom     5235
Turkey             4292
Name: Country, dtype: int64
```

```
plt.figure(figsize = (10,5))
sns.barplot(data['Country'].value_counts()[:15].index,data['Country'].value_counts()[:15].values)
plt.title('Top 10 Countries Affected')
plt.xlabel('Countries')
plt.ylabel('count')
plt.xticks(rotation=90)
plt.show()
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
```

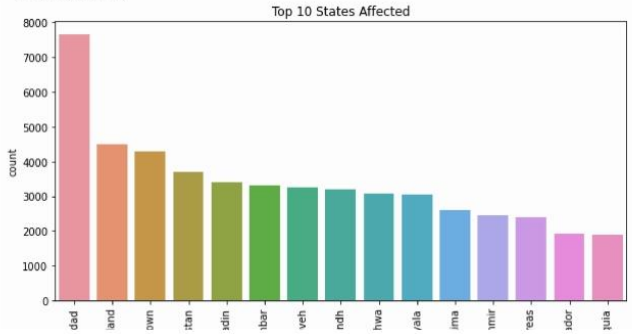



```
print(data['State'].value_counts().head(10))
```

```
Baghdad               7645
Northern Ireland      4498
Unknown               4290
Balochistan           3710
Saladin               3411
Al Anbar              3299
Nineveh               3241
Sindh                 3206
Khyber Pakhtunkhwa    3084
Diyala                3041
Name: State, dtype: int64
```

```
plt.figure(figsize = (10,5))
sns.barplot(data['State'].value_counts()[:15].index,data['State'].value_counts()[:15].values)
plt.title('Top 10 States Affected')
plt.xlabel('States')
plt.ylabel('count')
plt.xticks(rotation=90)
plt.show()
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
```

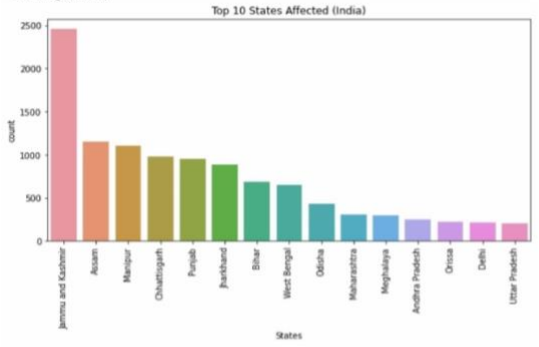



```
data_istates=data[data['Country']=='India']['State']
data_istates.value_counts()[:10]
```

```
Jammu and Kashmir    2454
Assam                1151
Manipur              1100
Chhattisgarh          979
Punjab                949
Jharkhand             887
Bihar                 688
West Bengal           650
Odisha                428
Maharashtra           302
Name: State, dtype: int64
```

```
plt.figure(figsize = (10,5))
sns.barplot(data_istates.value_counts()[:15].index,data_istates.value_counts()[:15].values)
ndia)')
plt.ylabel('count')
plt.xticks(rotation=90)
plt.show()
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
```



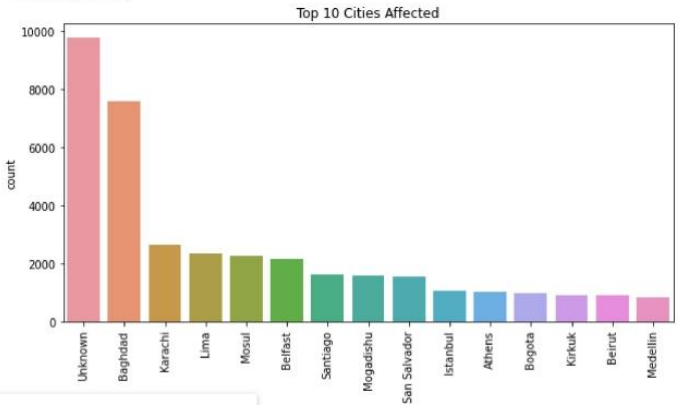

```
print(data['City'].value_counts().head(10))
```

```
Unknown        9775
Baghdad        7589
Karachi        2652
Lima           2359
Mosul          2265
Belfast        2171
```

```
Santiago       1621
Mogadishu      1581
San Salvador   1558
Istanbul       1048
Name: City, dtype: int64
```

```
plt.figure(figsize = (10,5))
sns.barplot(data['City'].value_counts()[:15].index,data['City'].value_counts()[:15].values)
plt.title('Top 10 Cities Affected')
plt.xlabel('Cities')
plt.ylabel('count')
plt.xticks(rotation=90)
plt.show()
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
```

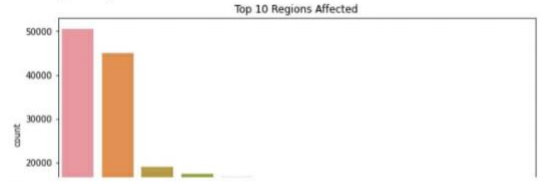


```
print(data['Region'].value_counts().head(10))
```

```
Middle East & North Africa     50474
South Asia                     44974
South America                  18978
Sub-Saharan Africa             17550
Western Europe                 16639
Southeast Asia                 12485
Central America & Caribbean    10344
Eastern Europe                  5144
North America                   3456
East Asia                        802
Name: Region, dtype: int64
```

```
plt.figure(figsize = (10,5))
sns.barplot(data['Region'].value_counts()[:15].index,data['Region'].value_counts()[:15].values)
plt.title('Top 10 Regions Affected')
plt.xlabel('Regions')
plt.ylabel('count')
plt.xticks(rotation=90)
plt.show()
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
```



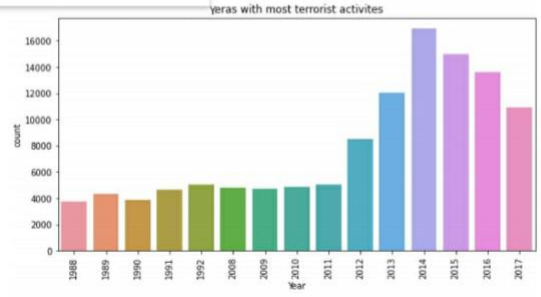

```
print(data['Year'].value_counts().head(10))
```

```
2014    16903
2015    14965
2016    13587
2013    12036
2017    10900
2012     8522
2011     5076
1992     5071
2010     4826
2008     4805
Name: Year, dtype: int64
```

```
plt.figure(figsize = (10,5))
sns.barplot(data['Year'].value_counts()[:15].index,data['Year'].value_counts()[:15].values)
plt.title('Yeras with most terrorist activites')
plt.xlabel('Year')
plt.ylabel('count')
plt.xticks(rotation=90)
plt.show()
```

Creating a copy...

```
ackages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
```



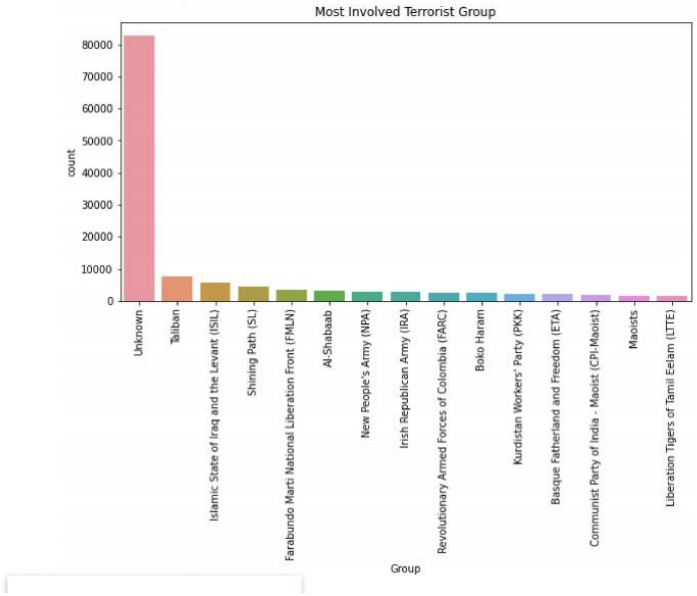
```
print(data['Group'].value_counts().head(10))
```

```
Unknown                                             82782
Taliban                                              7478
Islamic State of Iraq and the Levant (ISIL)          5613
Shining Path (SL)                                    4555
Farabundo Marti National Liberation Front (FMLN)     3351
Al-Shabaab                                           3288
New People's Army (NPA)                              2772
Irish Republican Army (IRA)                          2671
Revolutionary Armed Forces of Colombia (FARC)        2487
Boko Haram                                           2418
Name: Group, dtype: int64
```

```
plt.figure(figsize = (10,5))
sns.barplot(data['Group'].value_counts()[:15].index,data['Group'].value_counts()[:15].values)
plt.title('Most Involved Terrorist Group')
```

```
plt.xlabel('Group')
plt.ylabel('count')
plt.xticks(rotation=90)
plt.show()
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
```



WEEK 3

DATE-19-3-2023

MUSIC RECOMMENDATION

Music recommender systems can suggest songs to users based on their listening patterns.

```
[ ] import pandas as pd
    import numpy as np
    import seaborn as sns
    import matplotlib.pyplot as plt
    import warnings
    warnings.filterwarnings('ignore')
```

```
[ ] member=pd.read_csv('member.csv')
    songs=pd.read_csv('songs.csv')
    train=pd.read_csv('train.csv')
```

```
[ ] member.head()
```

	msno	city	bd	gender	registered_via	registration_init_time	expiration_date
0	XQxgAYj3klVKjR3oxPPXYYFp4soD4TuBghkhMTD4oTw=	1	0	NaN	7	20110820	20170920
1	UizsfmJb9mV54qE9hCYyU07Va97c0lCRLEQX3ae+ztM=	1	0	NaN	7	20150628	20170622
2	D8nEhsIOBSoE6VthTaqDX8U6lqjJ7dLdr72mOyLya2A=	1	0	NaN	4	20160411	20170712
3	mCuD+tZ1hERA/o5GPqk38e041J8ZsBaLcu7nGoIIvhI=	1	0	NaN	9	20150906	20150907
4	q4HRBfVSssAFS9iRfxWrohxuk9kCYMKjHOEagUMV6rQ=	1	0	NaN	4	20170126	20170613

```
[ ] songs.head()
```

	song_id	song_length	genre_ids	artist_name	composer	lyricist	language
0	CXoTN1eb7AI+DntdU1vbcwGRV4SCIDxZu+YD8JP8r4E=	247640	465	張信哲 (Jeff Chang)	董貞	何啟弘	3.0
1	o0kFgae9QtnYgRkVPqLJwa05zIhRlUjfF7O1tDw0ZDU=	197328	444	BLACKPINK	TEDDY| FUTURE BOUNCE| Bekuh BOOM	TEDDY	31.0
2	DwVvVurfpuz+XPuFvucclVQEyPqcpUkHR0ne1RQzPs0=	231781	465	SUPER JUNIOR	NaN	NaN	31.0
3	dKMBWoZySc dxSkihKG+Vf47nc18N9q4m58+b4e7dSSE=	273554	465	S.H.E	湯小康	徐世珍	3.0
4	W3bqWd3T+VeHFzHAUfARgW9AvVRaF4N5Yzm4Mr6Eo/o=	140329	726	貴族精選	Traditional	Traditional	52.0

```
[ ] train.head()
```

	msno	song_id	source_system_tab	source_screen_name	source_type	target
0	FGtllVqz18RPlwJj/edr2gV78zirAiY/9SmYvia+kCg=	BBzumQNXUHKdEBOB7mAJuzok+IJA1c2Ryg/yzTF6tik=	explore	Explore	online-playlist	1.0
1	Xumu+NIjS6QYVxDS4/t3SawvJ7viT9hPKXmf0RtLNx8=	bhp/MpSNoqoxOIB+/l8WPqu6jldth4DIpCm3ayXnJqM=	my library	Local playlist more	local-playlist	1.0
2	Xumu+NIjS6QYVxDS4/t3SawvJ7viT9hPKXmf0RtLNx8=	JNWfrrC7zNN7BdMpsISKa4Mw+xVJYNnxXh3/Epw7QgY=	my library	Local playlist more	local-playlist	1.0
3	Xumu+NIjS6QYVxDS4/t3SawvJ7viT9hPKXmf0RtLNx8=	2A87tzfnJTSWqD7gIZHisolhe4DMdzkbd6LzO1KHjNs=	my library	Local playlist more	local-playlist	1.0
4	FGtllVqz18RPlwJj/edr2gV78zirAiY/9SmYvia+kCg=	3qm6XTZ6MOCU11x8FIVbAGH5i5uMkT3/ZalWG1oo2Gc=	explore	Explore	online-playlist	1.0

```
[ ] train.info()
    songs.info()
    member.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 312826 entries, 0 to 312825
Data columns (total 6 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   msno                312826 non-null  object
 1   song_id             312826 non-null  object
 2   source_system_tab   311934 non-null  object
 3   source_screen_name  298588 non-null  object
 4   source_type         312193 non-null  object
 5   target              312825 non-null  float64
dtypes: float64(1), object(5)
memory usage: 14.3+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2296320 entries, 0 to 2296319
```

```
[ ] Data columns (total 7 columns):
     #   Column       Dtype
    ---  ------       -----
     0   song_id      object
     1   song_length  int64
     2   genre_ids    object
     3   artist_name  object
     4   composer     object
     5   lyricist     object
     6   language     float64
    dtypes: float64(1), int64(1), object(5)
    memory usage: 122.6+ MB
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 34403 entries, 0 to 34402
    Data columns (total 7 columns):
     #   Column                  Non-Null Count  Dtype
    ---  ------                  --------------  -----
     0   msno                    34403 non-null  object
     1   city                    34403 non-null  int64
     2   bd                      34403 non-null  int64
     3   gender                  14501 non-null  object
     4   registered_via          34403 non-null  int64
     5   registration_init_time  34403 non-null  int64
     6   expiration_date         34403 non-null  int64
    dtypes: int64(5), object(2)
    memory usage: 1.8+ MB
```

```
[ ] train.describe()
```

	target
count	312825.000000
mean	0.740046
std	0.438610
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

```
[ ] # get statistical summaries of dataset
    songs.describe()
```

	song_length	language
count	2.296320e+06	2.296319e+06
mean	2.469935e+05	3.237800e+01
std	1.609200e+05	2.433241e+01
min	1.850000e+02	-1.000000e+00
25%	1.836000e+05	-1.000000e+00
50%	2.266270e+05	5.200000e+01

```
[ ] train.shape
```

```
(312826, 6)
```

```
[ ] songs.shape
```

```
(2296320, 7)
```

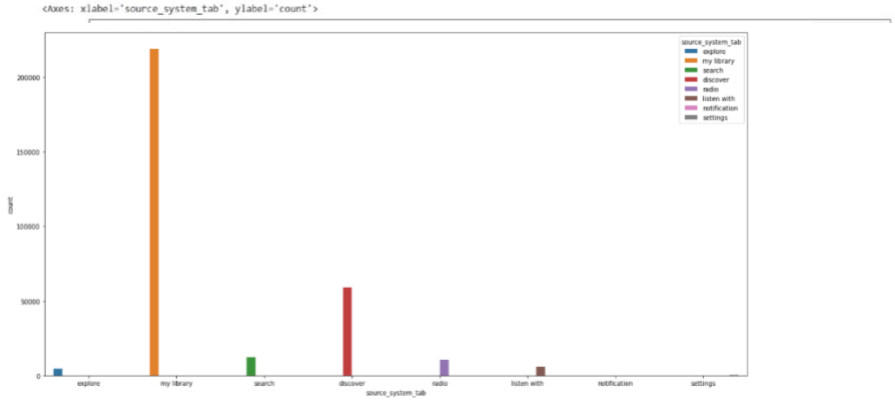
```
[ ] member.shape
```

```
(34403, 7)
```

Data Visualization

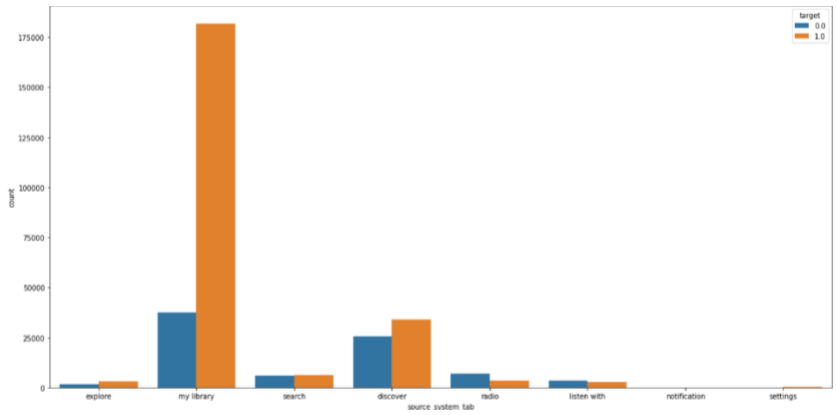
```
[ ] plt.figure(figsize=(20,10))
    sns.countplot(x='source_system_tab', hue='source_system_tab', data=train)
```

```
<Axes: xlabel='source_system_tab', ylabel='count'>
```



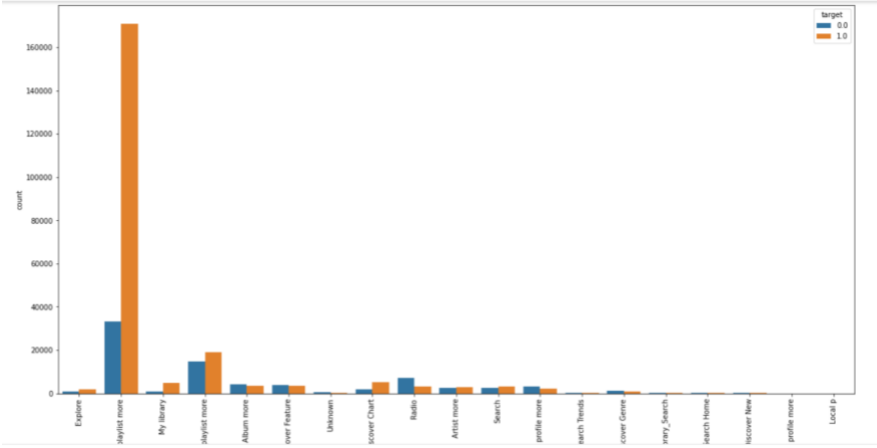

```
[ ] plt.figure(figsize=(20,10))
    sns.countplot(x='source_system_tab', hue='target', data=train)
```

```
<Axes: xlabel='source_system_tab', ylabel='count'>
```

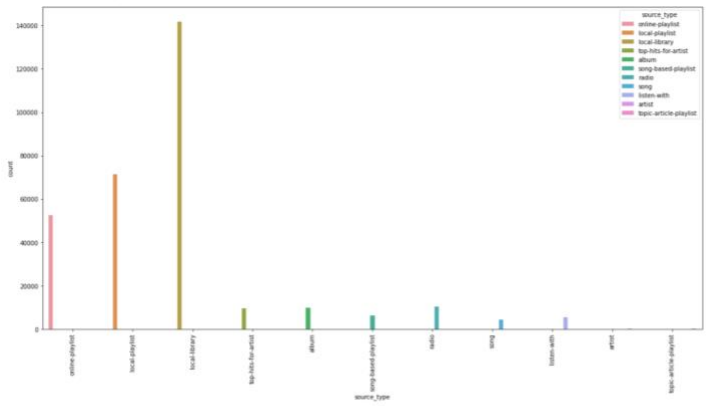


```
[ ] plt.figure(figsize=(20,10))
    plt.xticks(rotation=90)
    sns.countplot(x='source_screen_name', hue='target',data=train)
```

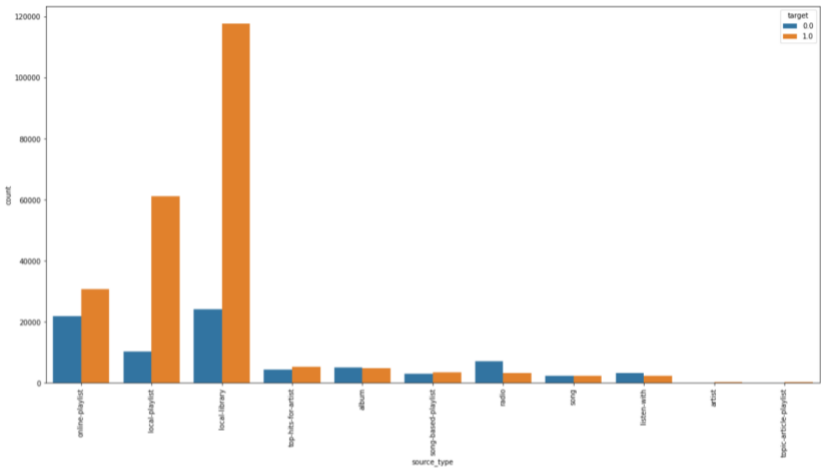
```
<Axes: xlabel='source_screen_name', ylabel='count'>
```




```
plt.figure(figsize=(20,10))
plt.xticks(rotation=90)
sns.countplot(x='source_type', hue='source_type',data=train)
```



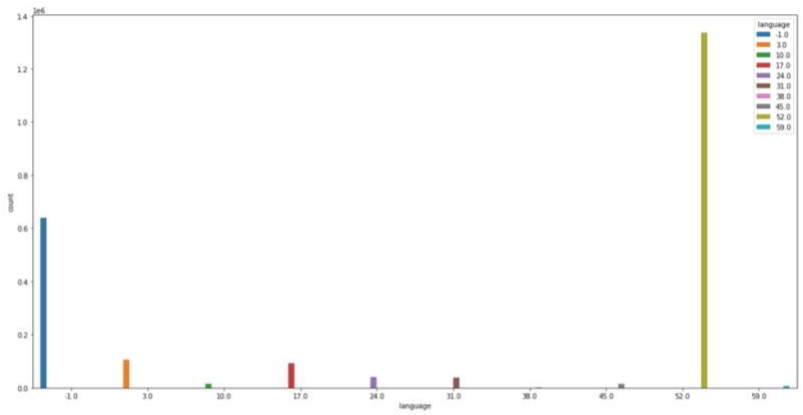

```
In [27]: plt.figure(figsize=(20,10))
         plt.xticks(rotation=90)
         sns.countplot(x='source_type', hue='target',data=train)
```




```
n [28]: plt.figure(figsize=(20,10))
        sns.countplot(x='language', hue='language',data=songs)
```

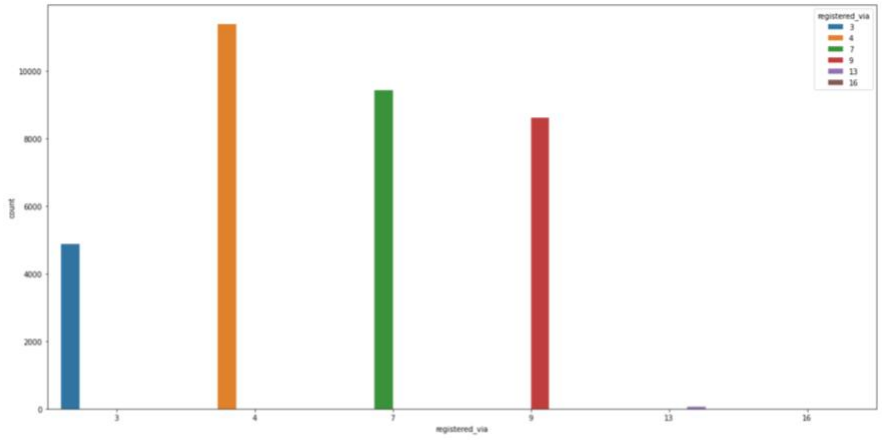
```
ut[28]: <Axes: xlabel='language', ylabel='count'>
```

Out[28]: <Axes: xlabel='language', ylabel='count'>



In [29]:
```
plt.figure(figsize=(20,10))
sns.countplot(x='registered_via', hue='registered_via',data=member)
```

Out[29]: <Axes: xlabel='registered_via', ylabel='count'>



Data Cleaning

In [32]:
```
ntr = 7000
nts = 3000
names=['msno','song_id','source_system_tab','source_screen_name','source_type','target']
test1 = pd.read_csv('train.csv',names=names,skiprows=ntr,nrows=nts)
```

In [33]:
```
test = test1.drop(['target'],axis=1)
ytr = np.array(test1['target'])
```

In [34]:
```
test_name = ['id','msno','song_id','source_system_tab','source_screen_name','source_type']
test['id']=np.arange(nts)
test = test[test_name]
```

In [35]:
```
song_cols = ['song_id', 'artist_name', 'genre_ids', 'song_length', 'language']
train = train.merge(songs[song_cols], on='song_id', how='left')
test = test.merge(songs[song_cols], on='song_id', how='left')
```

In [36]:
```
member['registration_year'] = member['registration_init_time'].apply(lambda x: int(str(x)[0:4]))
member['registration_month'] = member['registration_init_time'].apply(lambda x: int(str(x)[4:6]))
member['registration_date'] = member['registration_init_time'].apply(lambda x: int(str(x)[6:8]))
```

In [37]:
```
member['expiration_year'] = member['expiration_date'].apply(lambda x: int(str(x)[0:4]))
member['expiration_month'] = member['expiration_date'].apply(lambda x: int(str(x)[4:6]))
member['expiration_date'] = member['expiration_date'].apply(lambda x: int(str(x)[6:8]))
member = member.drop(['registration_init_time'], axis=1)
```

In [41]:
```
member_cols = member.columns
train = train.merge(member[member_cols], on='msno', how='left')
test = test.merge(member[member_cols], on='msno', how='left')
```

In [42]:
```
train = train.fillna(-1)
test = test.fillna(-1)
```

In [44]:
```
import gc
del member, songs; gc.collect();
```

In [45]:
```
cols = list(train.columns)
cols.remove('target')
```

In [46]:
```
from tqdm import tqdm
from sklearn.preprocessing import LabelEncoder
for col in tqdm(cols):
    if train[col].dtype == 'object':
        train[col] = train[col].apply(str)
        test[col] = test[col].apply(str)

        le = LabelEncoder()
        train_vals = list(train[col].unique())
        test_vals = list(test[col].unique())
        le.fit(train_vals + test_vals)
        train[col] = le.transform(train[col])
        test[col] = le.transform(test[col])
```

```
100%|██████████| 19/19 [00:03<00:00,  5.15it/s]
```

In [47]:
```
unique_songs = range(max(train['song_id'].max(), test['song_id'].max()))
song_popularity = pd.DataFrame({'song_id': unique_songs, 'popularity':0})
train_sorted = train.sort_values('song_id')
train_sorted.reset_index(drop=True, inplace=True)
test_sorted = test.sort_values('song_id')
test_sorted.reset_index(drop=True, inplace=True)
```

Building model

In [48]:
```
pip install lightgbm
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: lightgbm in /usr/local/lib/python3.9/dist-packages (3.3.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from lightgbm) (1.10.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from lightgbm) (1.22.4)
Requirement already satisfied: scikit-learn!=0.22.0 in /usr/local/lib/python3.9/dist-packages (from lightgbm) (1.2.2)
Requirement already satisfied: wheel in /usr/local/lib/python3.9/dist-packages (from lightgbm) (0.40.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from scikit-learn!=0.22.0->lightgbm) (1.1.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn!=0.22.0->lightgbm) (3.1.0)
```

In [49]:
```
from sklearn.model_selection import train_test_split
import lightgbm as lgb
X = np.array(train.drop(['target'], axis=1))
y = train['target'].values

X_test = np.array(test.drop(['id'], axis=1))
ids = test['id'].values

del train, test; gc.collect();

X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.1, random_state = 12)

del X, y; gc.collect();

d_train = lgb.Dataset(X_train, label=y_train)
d_valid = lgb.Dataset(X_valid, label=y_valid)

watchlist = [d_train, d_valid]
```

In [50]:
```
def predict(m1_model):
    model = m1_model.fit(X_train,y_train)
    print('Training Score : {}'.format(model.score(X_train,y_train)))
    y_pred = model.predict(X_valid)
    v_test = model.predict(X_test)
    yhat = (v_test>0.5).astype(int)
    comp = (yhat==ytr).astype(int)
    acc = comp.sum()/comp.size*100
    print("Accuracy on test data for the model", acc)
```

In [51]:
```
from sklearn.linear_model import LogisticRegression
```

In [52]:
```
predict(LogisticRegression())
```

```
Training Score : 0.7403309618779369
Accuracy on test data for the model 74.9
```

Prediction using LightGBM

In [53]:
```
params = {}
params['learning_rate'] = 0.4
params['application'] = 'binary'
params['max_depth'] = 15
params['num_leaves'] = 2**8
params['verbosity'] = 0
params['metric'] = 'auc'

model1 = lgb.train(params, train_set=d_train, num_boost_round=200, valid_sets=watchlist, early_stopping_rounds=10, verbose_eval=10)
```

```
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.025406 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
Training until validation scores don't improve for 10 rounds
[10]    training's auc: 0.857191        valid_1's auc: 0.835046
[20]    training's auc: 0.885154        valid_1's auc: 0.852631
[30]    training's auc: 0.905817        valid_1's auc: 0.862489
[40]    training's auc: 0.917035        valid_1's auc: 0.866331
[50]    training's auc: 0.928162        valid_1's auc: 0.870114
[60]    training's auc: 0.936295        valid_1's auc: 0.87126
[70]    training's auc: 0.944294        valid_1's auc: 0.871716
[80]    training's auc: 0.950983        valid_1's auc: 0.873159
Early stopping, best iteration is:
[78]    training's auc: 0.949714        valid_1's auc: 0.873484
```

In [54]:
```
p_test = model1.predict(X_test)
```

In [55]:
```
yhat = (p_test>0.5).astype(int)
comp = (yhat==ytr).astype(int)
acc = comp.sum()/comp.size*100
print('The accuracy of lgbm model on test data is: {0:f}%'.format(acc))
```

```
The accuracy of lgbm model on test data is: 89.500000%
```

WEEK4

DATE 20-03-2023

NEXT WORD PREDICTION

Using Tensorflow and Keras library train a RNN, to predict the next word.

Importing Libraries

```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam
import pickle
import numpy as np
import os
```

Loading Data

```
from google.colab import files
upload=files.upload()
```

Choose Files 1661-0.txt

- **1661-0.txt**(text/plain) - 607791 bytes, last modified: 3/6/2023 - 100% done

Saving 1661-0.txt to 1661-0.txt

```
file = open("1661-0.txt", "r", encoding = "utf8")
file.read()
```

```
Neville St.\nClair, which lay uncovered as the tide receded. And what do you think\nthey found in
the pockets?"\n\n"I cannot imagine."\n\n"No, I don't think you would guess. Every pocket stuffed w
ith pennies\nand half-pennies—421 pennies and 270 half-pennies. It was no wonder\nthat it had not
been swept away by the tide. But a human body is a\ndifferent matter. There is a fierce eddy betwe
en the wharf and the\nhouse. It seemed likely enough that the weighted coat had remained when\nthe
stripped body had been sucked away into the river."\n\n"But I understand that all the other clothe
s were found in the room.\nWould the body be dressed in a coat alone?"\n\n"No, sir, but the facts
might be met speciously enough. Suppose that\nthis man Boone had thrust Neville St. Clair through
the window, there\nis no human eye which could have seen the deed. What would he do then?\nIt woul
d of course instantly strike him that he must get rid of the\ntell-tale garments. He would seize t
he coat, then, and be in the act of\nthrowing it out, when it would occur to him that it would swi
m and not\nsink. He has little time, for he has heard the scuffle downstairs when\nthe wife tried
to force her way up, and perhaps he has already heard\nfrom his Lascar confederate that the police
are hurrying up the street.\nThere is not an instant to be lost. He rushes to some secret hoard,\n
where he has accumulated the fruits of his beggary, and he stuffs all\nthe coins upon which he can
lay his hands into the pockets to make sure\nof the coat's sinking. He throws it out, and would ha
ve done the same\nwith the other garments had not he heard the rush of steps below, and\nonly just
had time to close the window when the police appeared."\n\n"It certainly sounds feasible."\n\n"Wel
l, we will take it as a working hypothesis for want of a better.\nBoone, as I have told you, was a
rrested and taken to the station, but\nit could not be shown that there had ever before been anyth
ing against\nhim. He had for years been known as a professional beggar, but his life\nappeared to
have been a very quiet and innocent one. There the matter\nstands at present, and the questions wh
ich have to be solved—what\nNeville St. Clair was doing in the opium den, what happened to him whe
n\nthere, where is he now, and what Hugh Boone had to do with his\ndisappearance—are all as far fr
om a solution as ever. I confess that I\ncannot recall any case within my experience which looked
at the first\nglance so simple and yet which presented such difficulties."\n\nWhile Sherlock Holme
s had been detailing this singular series of\nevents, we had been whirling through the outskirts o
f the great town\nuntil the last straggling houses had been left behind, and we rattled\nalong wit
h a country hedge upon either side of us. Just as he finished,\nhowever, we drove through two scat
tered villages, where a few lights\nstill glimmered in the windows.\n\n"We are on the outskirts of
Lee," said my companion. "We have touched\non three English counties in our short drive, starting
in Middlesex,\npassing over an angle of Surrey, and ending in Kent. See that light\namong the tree
s? That is The Cedars, and beside that lamp sits a woman\nwhose anxious ears have already, I have
little doubt, caught the clink\nof our horse's feet."\n\n"But why are you not conducting the case
from Baker Street?" I asked.\n\n"Because there are many inquiries which must be made out here. Mr
s. St.\nClair has most kindly put two rooms at my disposal, and you may rest\nassured that she wil
l have nothing but a welcome for my friend and\ncolleague. I hate to meet her, Watson, when I have
no news of her\nhusband. Here we are. Whoa, there, whoa!"\n\nWe had pulled up in front of a large
villa which stood within its own\ngrounds. A stable-boy had run out to the horse's head, and sprin
ging\ndown, I followed Holmes up the small, winding gravel-drive which led to\nthe house. As we ap
proached, the door flew open, and a little blonde\nwoman stood in the opening, clad in some sort o
f light mousseline de\nsoie, with a touch of fluffy pink chiffon at her neck and wrists. She\nstoo
d with her figure outlined against the flood of light, one hand\nupon the door, one half-raised in
her eagerness, her body slightly\nbent, her head and face protruded, with eager eyes and parted li
ps, a\nstanding question.\n\n"Well?" she cried, "well?" And then, seeing that there were two of u
s,\nshe gave a cry of hope which sank into a groan as she saw that my\ncompanion shook his head an
d shrugged his shoulders.\n\n"No good news?"\n\n"None."\n\n"No bad?"\n\n"No."\n\n"Thank God for th
at. But come in. You must be weary, for you have had a\nlong day."\n\n"This is my friend, Dr. Wats
```

In [6]:
```
lines = []
file.seek(0)
for i in file:
    lines.append(i)

print("First Line: ", lines[1])
print("Last Line: ", lines[-6])
```

```
First Line:  Project Gutenberg's The Adventures of Sherlock Holmes, by Arthur Conan Doyle

Last Line:  subscribe to our email newsletter to hear about new eBooks.
```

Data Cleaning

In [7]:
```
data = ""

for i in lines:
    data = ' '. join(lines)

data = data.replace('\n', '').replace('\r', '').replace('\ufeff', '')
data[:360]
```

Out[7]:
```
" Project Gutenberg's The Adventures of Sherlock Holmes, by Arthur Conan Doyle  This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever.  You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or on line at www.gutenberg.net  Title: The Adventures of Sherl"
```

In [8]:
```
import string

translator = str.maketrans(string.punctuation, ' '*len(string.punctuation)) #map punctuation to space
new_data = data.translate(translator)

new_data[:100]
```

Out[8]:
```
' Project Gutenberg s The Adventures of Sherlock Holmes  by Arthur Conan Doyle  This eBook is for the'
```

In [9]:
```
z = []

for i in data.split():
    if i not in z:
        z.append(i)

data = ' '.join(z)
data[:100]
```

Out[9]:
```
"Project Gutenberg's The Adventures of Sherlock Holmes, by Arthur Conan Doyle This eBook is for the u"
```

Data Tokenization

In [10]:
```
tokenizer = Tokenizer()
tokenizer.fit_on_texts([data])
```

In [11]:
```
# saving the tokenizer for predict function.
pickle.dump(tokenizer, open('tokenizer1.pkl', 'wb'))

sequence_data = tokenizer.texts_to_sequences([data])[0]
sequence_data[:10]
```

Out[11]:
```
[838, 3083, 56, 322, 57, 1523, 15, 95, 839, 3084]
```

In [12]:
```
vocab_size = len(tokenizer.word_index) + 1
print(vocab_size)
```

```
8931
```

In [13]:
```
sequences = []

for i in range(1, len(sequence_data)):
    words = sequence_data[i-1:i+1]
    sequences.append(words)

print("The Length of sequences are: ", len(sequences))
sequences = np.array(sequences)
sequences[:10]
```

Out[13]:
```
array([[ 838, 3083],
       [3083,   56],
       [  56,  322],
       [ 322,   57],
       [  57, 1523],
       [1523,   15],
       [  15,   95],
       [  95,  839],
       [ 839, 3084],
       [3084, 3085]])
```

In [14]:
```
X = []
y = []

for i in sequences:
    X.append(i[0])
    y.append(i[1])

X = np.array(X)
y = np.array(y)
```

In [15]:
```
print("The Data is: ", X[:5])
print("The responses are: ", y[:5])
```

```
The Data is:  [ 838 3083   56  322   57]
The responses are:  [3083   56  322   57 1523]
```

In [16]:
```
y = to_categorical(y, num_classes=vocab_size)
y[:5]
```

Out[16]:
```
array([[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.]], dtype=float32)
```

Creating the Model:

In [17]:
```
model = Sequential()
model.add(Embedding(vocab_size, 10, input_length=1))
model.add(LSTM(1000, return_sequences=True))
model.add(LSTM(1000))
model.add(Dense(1000, activation="relu"))
model.add(Dense(vocab_size, activation="softmax"))
```

In [18]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 1, 10)             89310

 lstm (LSTM)                 (None, 1, 1000)           4044000

 lstm_1 (LSTM)               (None, 1000)              8004000

 dense (Dense)               (None, 1000)              1001000

 dense_1 (Dense)             (None, 8931)              8939931

=================================================================
Total params: 22,078,241
Trainable params: 22,078,241
Non-trainable params: 0
_________________________________________________________________
```

Callbacks

In [19]:
```
from tensorflow import keras
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import TensorBoard
```

```
checkpoint = ModelCheckpoint("nextword1.h5", monitor='loss', verbose=1,
    save_best_only=True, mode='auto')

reduce = ReduceLROnPlateau(monitor='loss', factor=0.2, patience=3, min_lr=0.0001, verbose = 1)

logdir='logsnextword1'
tensorboard_Visualization = TensorBoard(log_dir=logdir)
```

Model compilation

In [20]:
```
model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.001), metrics=['accuracy'])
```

```
WARNING:absl:`lr` is deprecated, please use `learning_rate` instead, or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
```

Fit The Model:

In [25]:
```
import h5py
```

In [23]:
```
history = model.fit(X, y, validation_split=0.05, batch_size=128, epochs=45, shuffle=True).history
```

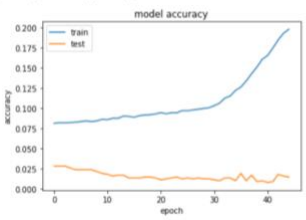
```
Epoch 1/45
132/132 [==============================] - 110s 800ms/step - loss: 8.8973 - accuracy: 0.0812 - val_loss: 9.3689 - val_accuracy: 0.0283
Epoch 2/45
132/132 [==============================] - 109s 829ms/step - loss: 8.3639 - accuracy: 0.0821 - val_loss: 10.0367 - val_accuracy: 0.0283
Epoch 3/45
132/132 [==============================] - 100s 757ms/step - loss: 8.0801 - accuracy: 0.0820 - val_loss: 10.6905 - val_accuracy: 0.0283
Epoch 4/45
132/132 [==============================] - 97s 733ms/step - loss: 7.9146 - accuracy: 0.0822 - val_loss: 10.8716 - val_accuracy: 0.0260
Epoch 5/45
132/132 [==============================] - 101s 765ms/step - loss: 7.7891 - accuracy: 0.0826 - val_loss: 11.0592 - val_accuracy: 0.0238
Epoch 6/45
132/132 [==============================] - 96s 725ms/step - loss: 7.6845 - accuracy: 0.0834 - val_loss: 11.5439 - val_accuracy: 0.0238
Epoch 7/45
132/132 [==============================] - 89s 672ms/step - loss: 7.5831 - accuracy: 0.0842 - val_loss: 12.1235 - val_accuracy: 0.0238
Epoch 8/45
132/132 [==============================] - 94s 712ms/step - loss: 7.4605 - accuracy: 0.0834 - val_loss: 12.7183 - val_accuracy: 0.0238
Epoch 9/45
132/132 [==============================] - 91s 686ms/step - loss: 7.2836 - accuracy: 0.0843 - val_loss: 13.7404 - val_accuracy: 0.0215
```

```
132/132 [==============================] - 87s 656ms/step - loss: 3.6888 - accuracy: 0.1929 - val_loss: 49.2234 - val_accuracy: 0.0158
Epoch 45/45
132/132 [==============================] - 91s 688ms/step - loss: 3.5852 - accuracy: 0.1977 - val_loss: 49.9106 - val_accuracy: 0.0147
```

Accuracy

In [26]:
```
import matplotlib.pyplot as plt
plt.plot(history['accuracy'])
plt.plot(history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
```

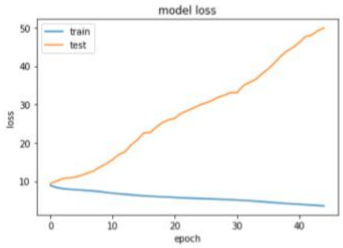
Out[26]:
```
<matplotlib.legend.Legend at 0x7f7adffd5b80>
```



Loss

In [27]:
```
plt.plot(history['loss'])
plt.plot(history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
```

Out[27]: <matplotlib.legend.Legend at 0x7f7b12494dc0>



In [28]:
```
from tensorflow.keras.models import load_model
model.save("model_act.h5")
print("Saved model to disk")
model = load_model('model_act.h5')
```

```
Saved model to disk
```

In [29]:
```
test = ['welcome to',
'more than',
'although this']

for t in test:
    example = tokenizer.texts_to_sequences([t])
    prediction = model.predict(np.array(example))
    predicted_word = np.argmax(prediction)
    reverse_word_map = dict(map(reversed, tokenizer.word_index.items()))
    print ("{0} -> {1}".format(t, reverse_word_map[predicted_word]))
```

```
WARNING:tensorflow:Model was constructed with shape (None, 1) for input KerasTensor(type_spec=TensorSpec(shape=(None, 1), dtype=tf.float32, name='embedding_input'), name='embedding_input', description="created by layer 'embedding_input'"), but it was called on an input with incompatible shape (None, 2).
1/1 [==============================] - 2s 2s/step
welcome to -> brazier
1/1 [==============================] - 0s 49ms/step
more than -> faint
1/1 [==============================] - 0s 51ms/step
although this -> pair
```

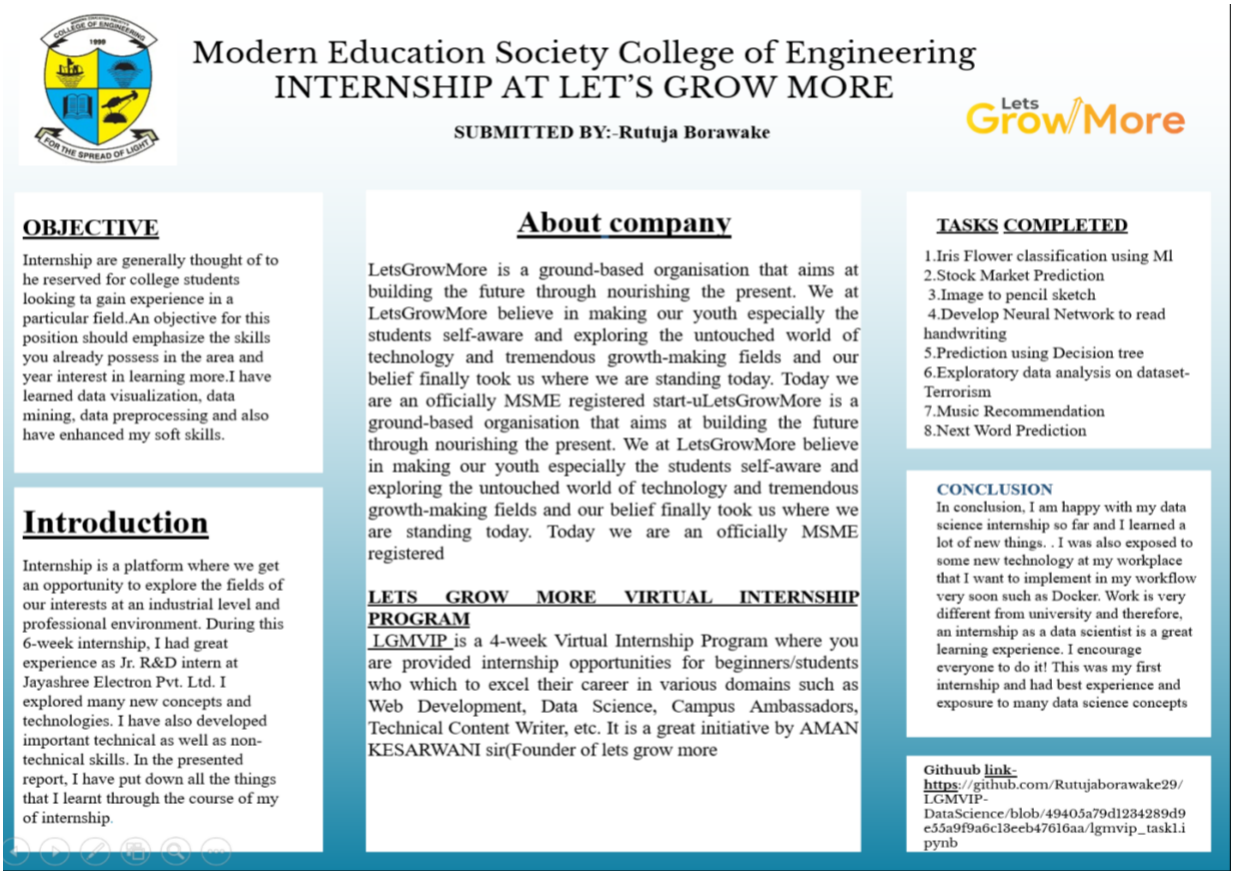
WEEK4

Date-29-3-2023

Submission of form internship

Submitted the form provided by the company which includes the work you have completed during internship

POSTER



Modern Education Society College of Engineering
INTERNSHIP AT LET'S GROW MORE

SUBMITTED BY:-Rutuja Borawake

Lets GrowMore

OBJECTIVE

Internship are generally thought of to he reserved for college students looking ta gain experience in a particular field.An objective for this position should emphasize the skills you already possess in the area and year interest in learning more.I have learned data visualization, data mining, data preprocessing and also have enhanced my soft skills.

Introduction

Internship is a platform where we get an opportunity to explore the fields of our interests at an industrial level and professional environment. During this 6-week internship, I had great experience as Jr. R&D intern at Jayashree Electron Pvt. Ltd. I explored many new concepts and technologies. I have also developed important technical as well as non-technical skills. In the presented report, I have put down all the things that I learnt through the course of my of internship.

About company

LetsGrowMore is a ground-based organisation that aims at building the future through nourishing the present. We at LetsGrowMore believe in making our youth especially the students self-aware and exploring the untouched world of technology and tremendous growth-making fields and our belief finally took us where we are standing today. Today we are an officially MSME registered start-uLetsGrowMore is a ground-based organisation that aims at building the future through nourishing the present. We at LetsGrowMore believe in making our youth especially the students self-aware and exploring the untouched world of technology and tremendous growth-making fields and our belief finally took us where we are standing today. Today we are an officially MSME registered

LETS GROW MORE VIRTUAL INTERNSHIP PROGRAM

LGMVIP is a 4-week Virtual Internship Program where you are provided internship opportunities for beginners/students who which to excel their career in various domains such as Web Development, Data Science, Campus Ambassadors, Technical Content Writer, etc. It is a great initiative by AMAN KESARWANI sir(Founder of lets grow more

TASKS COMPLETED

1.Iris Flower classification using Ml
2.Stock Market Prediction
3.Image to pencil sketch
4.Develop Neural Network to read handwriting
5.Prediction using Decision tree
6.Exploratory data analysis on dataset-Terrorism
7.Music Recommendation
8.Next Word Prediction

CONCLUSION

In conclusion, I am happy with my data science internship so far and I learned a lot of new things. . I was also exposed to some new technology at my workplace that I want to implement in my workflow very soon such as Docker. Work is very different from university and therefore, an internship as a data scientist is a great learning experience. I encourage everyone to do it! This was my first internship and had best experience and exposure to many data science concepts

Githuub link-
https://github.com/Rutujaborawake29/LGMVIP-DataScience/blob/49405a79d1234289d9e55a9f9a6c13eeb47616aa/lgmvip_task1.ipynb

CONCLUSION

In conclusion, I am happy with my data science internship so far and I learned a lot of new things. . I was also exposed to some new technology at my workplace that I want to implement in my workflow very soon such as Docker. Work is very different from university and therefore, an internship as a data scientist is a great learning experience. I encourage everyone to do it!

This was my first internship and had best experience and exposure to many data science concepts

h