Index

PRACTICAL : 1

Aim :- Perform Data Collection, Cleaning, Modelling and Compilation.

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

risk = pd.read_csv('Saloni_Gavhane/risk_analytics_train.csv',index_col=0)

risk.head()
```

[1]:

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
LP001002	Male	No	0.0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
LP001003	Male	Yes	1.0	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
LP001005	Male	Yes	0.0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
LP001006	Male	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
LP001008	Male	No	0.0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

[2]:
```
risk.dtypes
```

[2]:
```
Gender               object
Married              object

Dependents          float64
Education            object
Self_Employed        object
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area        object
Loan_Status          object
dtype: object
```

[3]:
```
risk.shape
risk.columns
```

[3]:
```
Index(['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
       'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

[4]:
```
risk.isnull().sum()
```

[4]:
```
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

[5]:
```
risk.describe()
```

[5]:

	Dependents	ApplicantIncome	CoapplicantIncome	LoanAmount	\
count	599.000000	614.000000	614.000000	592.000000	
mean	0.762938	5403.459283	1621.245798	146.412162	
std	1.015216	6109.041673	2926.248369	85.587325	
min	0.000000	150.000000	0.000000	9.000000	
25%	0.000000	2877.500000	0.000000	100.000000	
50%	0.000000	3812.500000	1188.500000	128.000000	
75%	2.000000	5795.000000	2297.250000	168.000000	
max	3.000000	81000.000000	41667.000000	700.000000	

	Loan_Amount_Term	Credit_History
count	600.00000	564.000000
mean	342.00000	0.842199
std	65.12041	0.364878
min	12.00000	0.000000
25%	360.00000	1.000000
50%	360.00000	1.000000
75%	360.00000	1.000000
max	480.00000	1.000000

0.1 Importing categorical data with mode value

[6]:
```
for x in['Gender','Married','Dependents','Self_Employed','Loan_Amount_Term']:
  risk[x].fillna(risk[x].mode()[0])
```

[7]:
```
risk.isnull().sum()
```

[7]:
```
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
```

0.2 Importing categorical data with mean value

[8]:
```
risk['LoanAmount'].fillna(round(risk['LoanAmount'].mean(),0))
```

[8]:
```
Loan_ID
LP001002    146.0
LP001003    128.0
LP001005     66.0
LP001006    120.0
LP001008    141.0
            …
LP002978     71.0
LP002979     40.0
LP002983    253.0

LP002984    187.0
LP002990    133.0
Name: LoanAmount, Length: 614, dtype: float64
```

[ ]:
```

```

PRACTICAL : 2

Aim :- Implementation of the Exploratory Data Visualization Technique

1 Implementation of the Exploratory Data Visualization Technique

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

[2]:
```
# Assuming the dataset is in CSV format
df = pd.read_csv("sales_data - sales_data.csv")
```

[3]:
```
#Display the first few rows of the dataset
print(df.head())
```

```
         Date  Day     Month  Year  Customer_Age       Age_Group  \
0  26-11-2013   26  November  2013            19     Youth (<25)
1  26-11-2015   26  November  2015            19     Youth (<25)
2  23-03-2014   23     March  2014            49  Adults (35-64)
3  23-03-2016   23     March  2016            49  Adults (35-64)
4  15-05-2014   15       May  2014            47  Adults (35-64)

  Customer_Gender    Country             State Product_Category Sub_Category  \
0               M     Canada  British Columbia      Accessories   Bike Racks
1               M     Canada  British Columbia      Accessories   Bike Racks
2               M  Australia   New South Wales      Accessories   Bike Racks
3               M  Australia   New South Wales      Accessories   Bike Racks
4               F  Australia   New South Wales      Accessories   Bike Racks

               Product  Order_Quantity  Unit_Cost  Unit_Price  Profit  Cost  \
0  Hitch Rack - 4-Bike               8         45         120     590   360
1  Hitch Rack - 4-Bike               8         45         120     590   360
2  Hitch Rack - 4-Bike              23         45         120    1366  1035
3  Hitch Rack - 4-Bike              20         45         120    1188   900
4  Hitch Rack - 4-Bike               4         45         120     238   180

   Revenue
0      950
1      950

2     2401
3     2088
4      418
```

[4]:
```
print(df.describe())
```

```
                Day           Year   Customer_Age  Order_Quantity  \
count  113036.000000  113036.000000  113036.000000   113036.000000
mean       15.665753    2014.401739      35.919212       11.901660
std         8.781567       1.272510      11.021936        9.561857
min         1.000000    2011.000000      17.000000        1.000000
25%         8.000000    2013.000000      28.000000        2.000000
50%        16.000000    2014.000000      35.000000       10.000000
75%        23.000000    2016.000000      43.000000       20.000000
max        31.000000    2016.000000      87.000000       32.000000
```

```
          Unit_Cost     Unit_Price         Profit           Cost  \
count  113036.000000  113036.000000  113036.000000  113036.000000
mean      267.296366     452.938427     285.051665     469.318695
std       549.835483     922.071219     453.887443     884.866118
min         1.000000       2.000000     -30.000000       1.000000
25%         2.000000       5.000000      29.000000      28.000000
50%         9.000000      24.000000     101.000000     108.000000
75%        42.000000      70.000000     358.000000     432.000000
max      2171.000000    3578.000000   15096.000000   42978.000000

             Revenue
count  113036.000000
mean      754.370360
std      1309.094674
min         2.000000
25%        63.000000
50%       223.000000
75%       800.000000
max     58074.000000
```

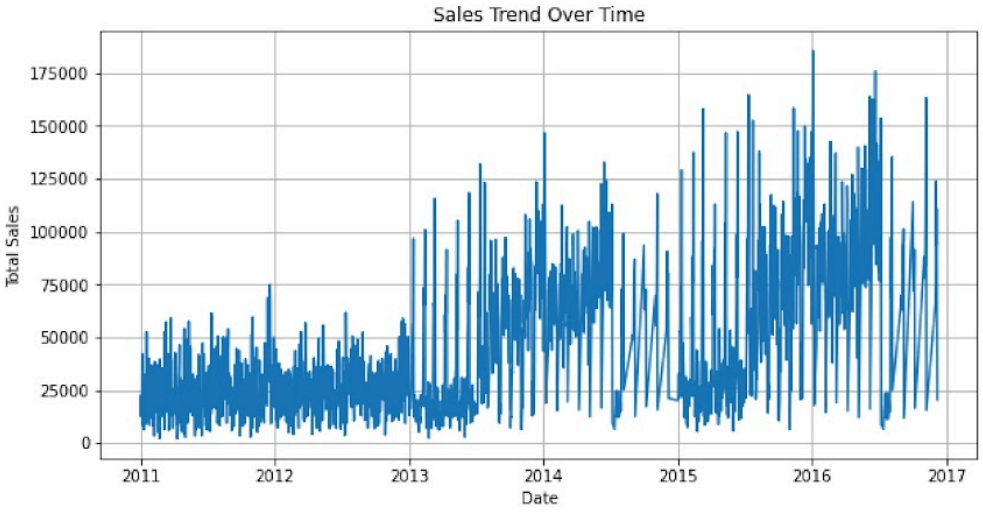
[5]:
```
# Check for any missing values
print(df.isnull().sum())
```

```
Date                0
Day                 0
Month               0
Year                0
Customer_Age        0
Age_Group           0
Customer_Gender     0
Country             0
Sub_Category        0
Product             0
Order_Quantity      0
Unit_Cost           0
Unit_Price          0
Profit              0
Cost                0
Revenue             0
dtype: int64
```

2 DATA VISULAIZATION

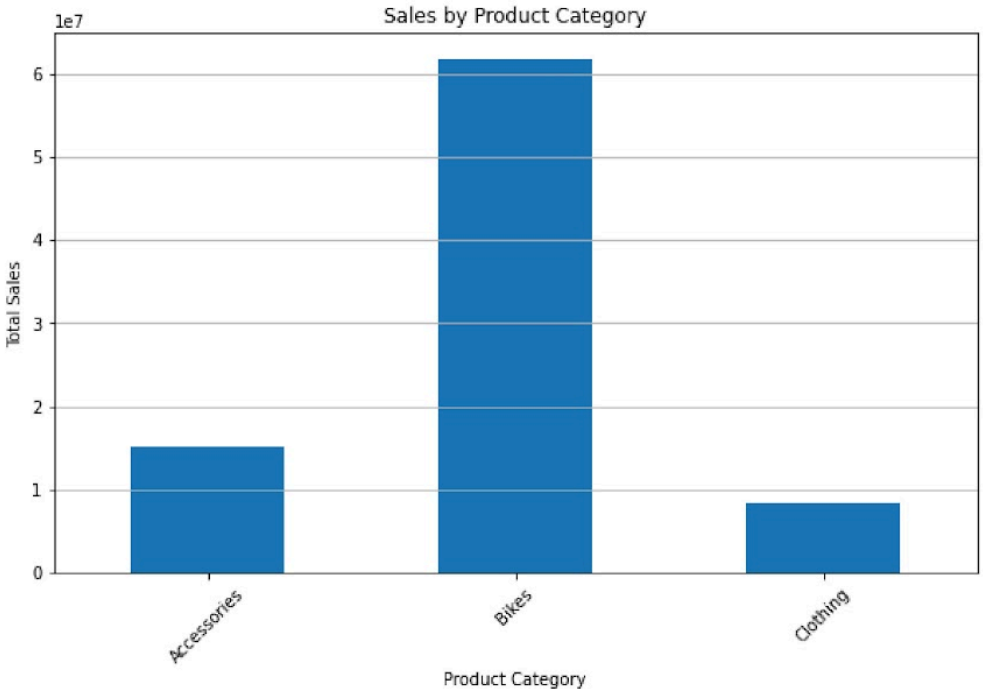
3 1.Visulizing Sales over time

[7]:
```
# Convert 'Date'column to datetime type
df ['Date']= pd.to_datetime(df[ 'Date'])

#Group by date and calculate total sales per day
daily_sales = df.groupby('Date') ['Revenue'].sum()

#Plot the sales trend over time
plt.figure(figsize=(10, 5))
plt.plot(daily_sales.index, daily_sales.values)
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.title('Sales Trend Over Time')
plt.grid(True)
plt.show()
```



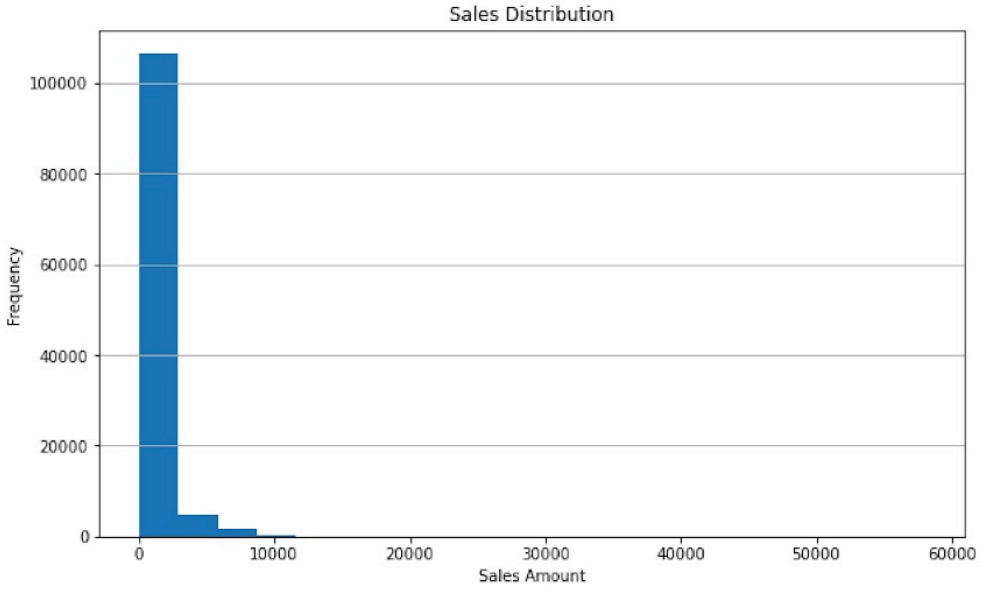
4 2.Visulaizing Sales by Product Category

[9]:
```
#Group by product category and calculate total sales per category
category_sales = df.groupby('Product_Category')['Revenue'].sum()

#Plot a bar chart to show sales by product category
plt.figure(figsize=(10, 6))
category_sales.plot(kind='bar')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.title('Sales by Product Category')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```



4.1 Visualizing Sales Distribution

[10]:
```
#Plot a histogram to show the distribution of sales amounts
plt.figure(figsize=(10, 6))
plt.hist(df['Revenue'], bins=20)
plt.xlabel('Sales Amount')
plt.ylabel('Frequency')
plt.title('Sales Distribution')
plt.grid(axis="y")
plt.show()
```



[ ]:

PRACTICAL : 3

Aim :- Perform Different Statistics Distribution.

Different Statistic Distribution
Suppose there are 100 students in the class and in one of the mathematics tests the average marks scored by the students in the subject is 78 and the standard deviation is 25. The marks of the student follow Normal probability distribution.find:(i) percentage of student who got less than 60 marks(ii) percentage of student who have scored more than 70

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
from scipy import stats
```

In [8]:
```
# Given information
mean = 78
std_dev = 25
total_students = 100
score = 60

#Calculate z-score for 60
z_score = (score - mean) / std_dev

#Calculate the probability of getting a score less than 60
prob = norm.cdf(z_score)

#Calculate the percentage of the student who go less than 60 marks
percent = prob * 100

#Print the result
print("Percentage of students who go less than 60 marks:", round(percent,2), "
```

```
Percentage of students who go less than 60 marks: 23.58 %
```

In [28]:
```
# Given information
mean = 78
std_dev = 25
total_students = 100
score = 70

#Calculate z-score for 60
z_score = (score - mean) / std_dev

#Calculate the probability of getting a score less than 60
prob = norm.cdf(z_score)
```

```
#Calculate the percentage of the student who go less than 70 marks
percent = (1-prob) * 100

#Print the result
print("Percentage of students who go less than 70 marks:", round(percent,2), "
```

```
Percentage of students who go less than 70 marks: 62.55 %
```

Binomial Distribution

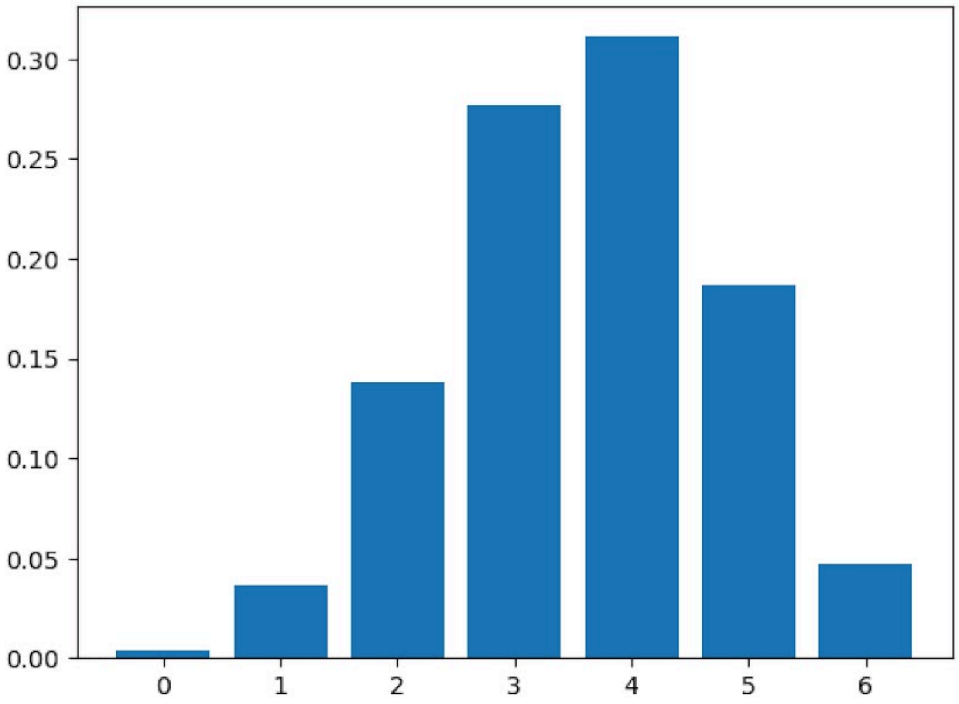
In [15]:
```
from scipy.stats import binom
```

In [21]:
```
#setting the value of n and p
n = 6
p = 0.6
```

In [23]:
```
# defining list of r values
r_values = list(range(n + 1))

# list of pmf values
dist = [binom.pmf(r, n, p) for r in r_values]
print(dist)
```

```
[0.004096000000000003, 0.03686400000000002, 0.1382400000000001, 0.2764800000000
001, 0.3110400000000001, 0.1866240000000001, 0.04665599999999999]
```

In [25]:
```
# Plotting the graph
plt.bar(r_values, dist)
plt.show()
```



Question:Fashion Bazaar (FB) is an e-commerce company that sells kids apparel. It is observed that 15% of their customers return the items purchased by them for many reasons (such as size, color, and material mismatch). On a specific day, 30 customers purchased items from FB.Calculate the probability that exactly 5 customers will return the items.

In [34]:
```
# Exptected number of sucessful trial = 5
# Total number of trials = 30
# The probability of success = 0.15
```

In [36]:
```
stats.binom.pmf(5, 30, 0.15)
```

Out[36]:
```
0.18610694845752918
```

In [ ]:
```
The corresponding probability is 0.1816, that is the probability that exactly
```

Poisson Distribution

#Question : The number of calls arriving at a call center follows a Poisson distribution at 20 calls per hour. #Calculate the probability that the number of calls will be maximum 10.

In [39]:
```
stats.poisson.cdf(10,20)
```

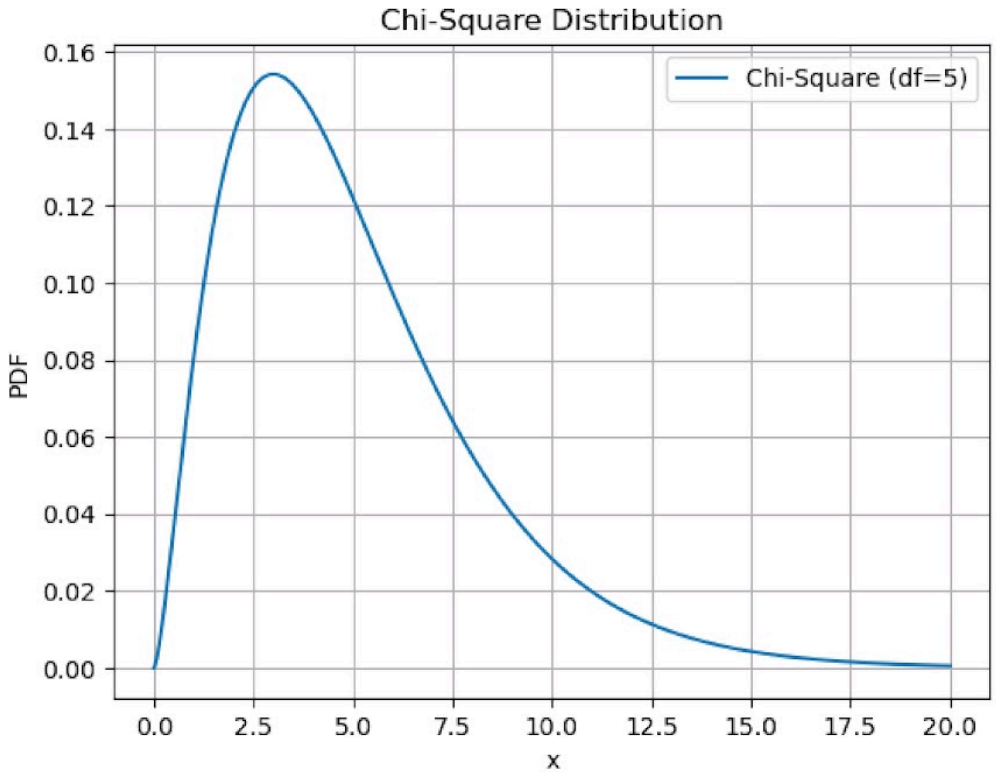
Out[39]:
```
0.010811718826652723
```

Chi-square Distribution

In [41]:
```
from scipy.stats import chi2
```

```
# Parameters for the Chi-Square distribution
degrees_of_freedom = 5  # Degrees of freedom

# Generate a range of x values
x = np.linspace(0, 20, 1000)


# Calculate the probability density function (PDF) for the Chi-Square distribu
pdf = chi2.pdf(x, degrees_of_freedom)
```

In [44]:
```
# Plot the PDF
plt.plot(x, pdf, label=f'Chi-Square (df={degrees_of_freedom})')
plt.xlabel('x')
plt.ylabel('PDF')
plt.legend()
plt.title('Chi-Square Distribution')
plt.grid()
plt.show()
```



PRACTICAL : 4

Aim :- Implementation Hypothesis Testing.

0.1 T-test

[1]:
```
import numpy as np
from scipy import stats

#Sample datu
group1 = np.array([85, 90, 88, 92, 95])
group2 = np.array([78, 80, 84, 83, 82])

#Perform Independent two-sample t-test
f_stat, p_value = stats.f_oneway(group1, group2)

print(f"F-Statistic: {f_stat}")
print(f"p-value: {p_value}")

#Interpolation
alpha = 0.05
if p_value < alpha:
    print("Null Hypothesis: There is significant difference between two groups.
 ↪")
else:
    print("alternate Hypothesis: There is significant difference between two␣
 ↪groups.")
```

```
F-Statistic: 18.216748768472915
p-value: 0.002731175755558595
Null Hypothesis: There is significant difference between two groups.
```

0.2 Z-test

[2]:
```
import numpy as np
from scipy import stats

#Sample data
sample_heights = np.array([66, 68, 78, 65, 69])
population_mean = 67
population_std = 2 #Know poulation standard deviation
```

```
#calculate sample mean
sample_mean = np.mean(sample_heights)

#Perform one-sample z-test
z_stat = (sample_mean - population_mean)/(population_std/np.
 ↪sqrt(len(sample_heights)))
p_value = (1 - stats.norm.cdf(np.abs(z_stat))) # two-tailed test

print(f"z-statistic: {z_stat}")
print("p-value:",p_value)

#Interpolation
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The sample mean is significantly␣
 ↪different from the population mean.")
```

```
else:
    print("Fail to reject the null hypothesis: The sample mean is not␣
 ↪significantly different from the population mean.")
```

```
z-statistic: 2.459674775249772
p-value: 0.006953148447672919
Reject the null hypothesis: The sample mean is significantly different from the
population mean.
```

[ ]:

PRACTICAL : 5

Aim :- Perform for Exploring Categorical and Binary Data.

1 Perform for Exploring Categorical and Binary Data

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
# Load the dataset
df = pd.read_csv('Saloni_Gavhane/sales_data.csv')
```

[3]:
```
df.head()
```

[3]:

	Date	Day	Month	Year	Customer_Age	Age_Group
0	26-11-2013	26	November	2013	19	Youth (<25)
1	26-11-2015	26	November	2015	19	Youth (<25)
2	23-03-2014	23	March	2014	49	Adults (35-64)
3	23-03-2016	23	March	2016	49	Adults (35-64)
4	15-05-2014	15	May	2014	47	Adults (35-64)

[4]:
```
#df.info()
df.iloc[:,5:12].nunique()
#dff[['Age_Group','Customer_Age','Age_Group','Customer_Gender','Country','Product_Category
```

[4]:
```
Age_Group            4
Customer_Gender       2
Country               6
State                53
Product_Category      3
Sub_Category         17
Product             130
dtype: int64
```

1.1 Summary Statistic

[5]:
```
df.describe()
```

[5]:

	Day	Year	Customer_Age	Order_Quantity
count	113036.000000	113036.000000	113036.000000	113036.000000
mean	15.665753	2014.401739	35.919212	11.901660
std	8.781567	1.272510	11.021936	9.561857
min	1.000000	2011.000000	17.000000	1.000000
25%	8.000000	2013.000000	28.000000	2.000000
50%	16.000000	2014.000000	35.000000	10.000000
75%	23.000000	2016.000000	43.000000	20.000000
max	31.000000	2016.000000	87.000000	32.000000

	Unit_Cost	Unit_Price	Profit	Cost
count	113036.000000	113036.000000	113036.000000	113036.000000
mean	267.296366	452.938427	285.051665	469.318695
std	549.835483	922.071219	453.887443	884.866118
min	1.000000	2.000000	-30.000000	1.000000
25%	2.000000	5.000000	29.000000	28.000000
50%	9.000000	24.000000	101.000000	108.000000
75%	42.000000	70.000000	358.000000	432.000000
max	2171.000000	3578.000000	15096.000000	42978.000000

1.2 Frequency counts

[6]:
```
category_counts = df['Country'].value_counts()
```

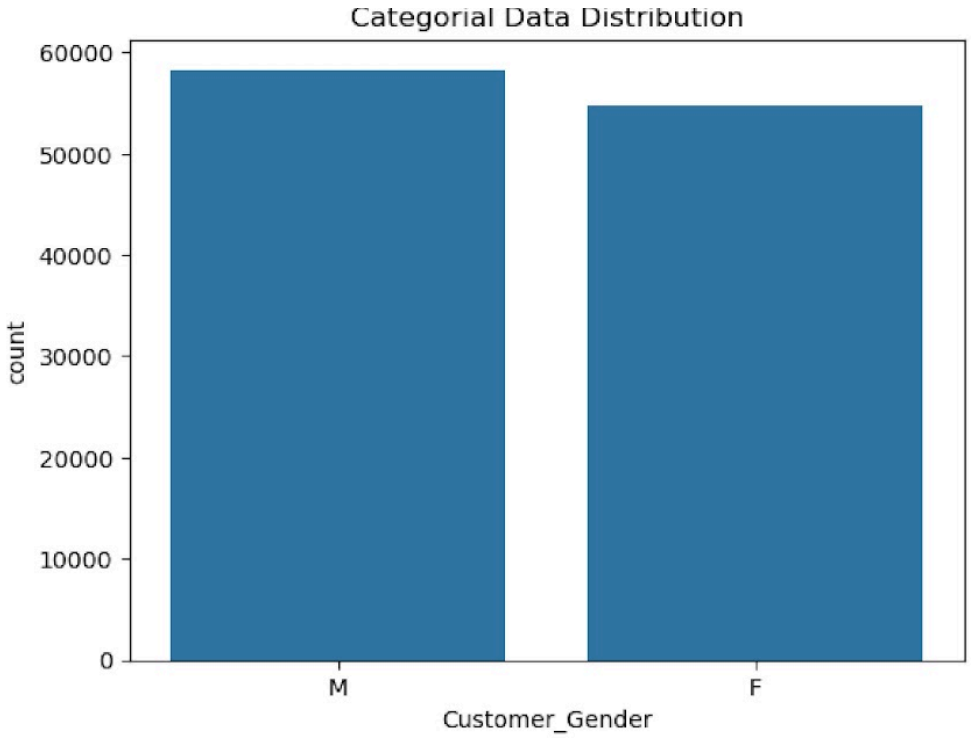
[7]:
```
print(category_counts)
```

```
Country
United States     39206
Australia         23936
Canada            14178
United Kingdom    13620
Germany           11098
France            10998
Name: count, dtype: int64
```

1.3 Bar plots

[8]:
```
sns.countplot(data=df, x='Customer_Gender')
plt.title("Categorial Data Distribution")
```

[8]: Text(0.5, 1.0, 'Categorial Data Distribution')

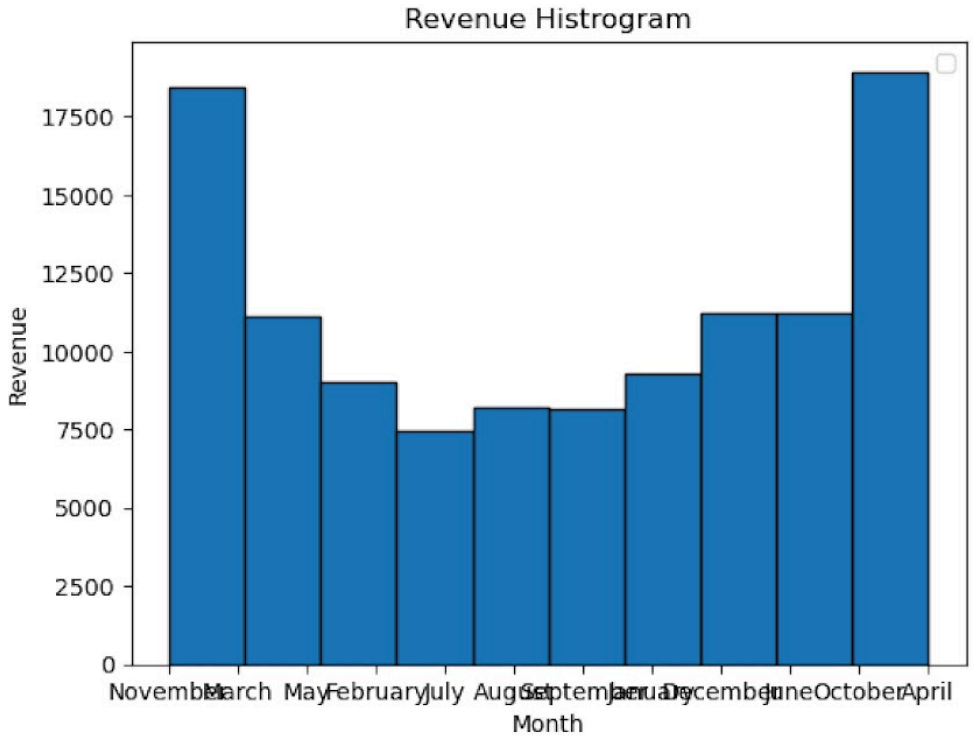


1.4 Histogram

[9]:
```
plt.hist(df['Month'], bins=10, edgecolor='black')
plt.title('Revenue Histrogram')
plt.xlabel('Month')
plt.ylabel('Revenue')
plt.legend()
```

```
C:\Users\vipul\AppData\Local\Temp\ipykernel_22972\874665537.py:5: UserWarning:
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
  plt.legend()
```

[9]: <matplotlib.legend.Legend at 0x29857764d70>



1.5 One hot encoder :

[10]:
```
one_hot_encoded_data = pd.get_dummies(df, columns = ['Product'])
one_hot_encoded_data
```

[10]:

	Date	Day	Month	Year	Customer_Age	Age_Group	\
0	26-11-2013	26	November	2013	19	Youth (<25)	
1	26-11-2015	26	November	2015	19	Youth (<25)	
2	23-03-2014	23	March	2014	49	Adults (35-64)	
3	23-03-2016	23	March	2016	49	Adults (35-64)	
4	15-05-2014	15	May	2014	47	Adults (35-64)	
…	…	…	…	…	…	…	
113031	12-04-2016	12	April	2016	41	Adults (35-64)	
113032	02-04-2014	2	April	2014	18	Youth (<25)	
113033	02-04-2016	2	April	2016	18	Youth (<25)	
113034	04-03-2014	4	March	2014	37	Adults (35-64)	
113035	04-03-2016	4	March	2016	37	Adults (35-64)	

[113036 rows x 147 columns]

[11]:
```
one_hot_encoded_data = pd.get_dummies(df, columns = ['State'])
one_hot_encoded_data
```

[11]:

	Date	Day	Month	Year	Customer_Age	Age_Group	\
0	26-11-2013	26	November	2013	19	Youth (<25)	
1	26-11-2015	26	November	2015	19	Youth (<25)	
2	23-03-2014	23	March	2014	49	Adults (35-64)	
3	23-03-2016	23	March	2016	49	Adults (35-64)	
4	15-05-2014	15	May	2014	47	Adults (35-64)	
…	…	…	…	…	…	…	
113031	12-04-2016	12	April	2016	41	Adults (35-64)	
113032	02-04-2014	2	April	2014	18	Youth (<25)	
113033	02-04-2016	2	April	2016	18	Youth (<25)	
113034	04-03-2014	4	March	2014	37	Adults (35-64)	
113035	04-03-2016	4	March	2016	37	Adults (35-64)	

PRACTICAL : 6

Aim :- Perform Anova testing.

0.1 Perform ANOVA Testing One-Way and Two-Way

0.2 One-Way ANOVA

[1]:
```
import numpy as np

#Sample data: test scores for three different teaching methods
method1 = np.array([85, 88, 92, 94, 90])
method2 = np.array([79, 80, 83, 77, 79])
method3 = np.array([91, 89, 94, 96, 92])
```

[4]:
```
from scipy import stats

#Perform one way ANOVA
f_stat, p_value = stats.f_oneway (method1,method2, method3)

print(f"F-Statistics: {f_stat}")
print(f"p_value: {p_value}")
```

```
F-Statistics: 28.2551440329218
p_value: 2.887704267394722e-05
```

0.3 Two-Way ANOVA

[7]:
```
import pandas as pd

#Sample data: test scores with two factors: teaching method and gender
data = pd.DataFrame({
    'Score': [85, 78, 91, 88, 88, 89, 92, 83, 94, 94, 77, 96, 90, 79, 92],
    'Method': ['Method1', 'Method1', 'Method1', 'Method2', 'Method2',␣
 ↪'Method2', 'Method3', 'Method3', 'Method3',
               'Method1', 'Method2', 'Method3', 'Method1', 'Method2',␣
 ↪'Method3'],
    'Gender': ['Female', 'Female', 'Female', 'Male', 'Male', 'Male', 'Female',␣
 ↪'Female', 'Female', 'Male',
            'Male', 'Male', 'Female', 'Female', 'Female']

    })
```

[8]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols

#Fit the two-way ANOVA model with interaction
model = ols('Score ~ C(Method)*C(Gender)', data=data).fit()

#Perform two-way ANOVA (Type II sum of squares)
anova_table = sm.stats.anova_lm(model, typ=2)
print(anova_table)

#(method): test the effects of the teaching method
#(gender): test the effects of gender./#(method): (Gender): test the␣
 ↪interaction effects_between #teaching methods and gender
```

```
                     sum_sq   df         F    PR(>F)
C(Method)         230.372222  2.0  3.759474  0.065039
C(Gender)         109.350000  1.0  3.568994  0.091456
C(Method):C(Gender)   2.100000  2.0  0.034270  0.966436
```

PRACTICAL : 7

Aim :- Implementation of Non-parametric tests.

0.1 SIGN RANK TEST

[1]:
```
import numpy as np
from scipy.stats import binom
```

[2]:
```
data = [
    (4,3),
    (5,6),
    (7,6),
    (4,5),
    (3,2),
    (6,6),
    (8,7),
    (5,4),
    (6,4),
    (6,6),
    (4,4),
]

data
```

[2]:
```
[(4, 3),
 (5, 6),
 (7, 6),
 (4, 5),
 (3, 2),
 (6, 6),
 (8, 7),
 (5, 4),
 (6, 4),
 (6, 6),
 (4, 4)]
```

[3]:
```
differences = [x-y for x,y in data]
positive_diff  = sum(1 for diff in differences if diff > 0)
negative_diff = sum(1 for diff in differences if diff < 0)

n = len(data)
```

```
k = positive_diff

print(n)
print(k)
```

```
11
6
```

[4]:
```
p_value =  binom.cdf(k,n,0.5)*binom.sf(k-1,n,0.5)
print(f"p_value:",p_value)
```

```
p_value: 0.36279296875
```

0.2 Wilcoxon signed rank test

[5]:
```
import numpy as np
from scipy.stats import wilcoxon

before = [35, 42, 38, 46, 33, 29, 42, 37, 40, 32]
after = [30, 40, 36, 44, 28, 24, 41, 35, 38, 30]
```

[6]:
```
# differences
differences = [after[i] - before[i] for i in range(len(before))]

#run Wilcoxon test on differences
statistic, p_value = wilcoxon(differences)
print("Test Statistic:", statistic)
print("p-value:",p_value)
```

```
Test Statistic: 0.0
p-value: 0.001953125
```

0.3 Kruskal Wali's test

[7]:
```
import scipy.stats as stats

group_A = [23,27,21,19,25]
group_B = [18,16,20,24,22]
group_C = [38,13,15,11,12]

H, p_value = stats.kruskal(group_A, group_B, group_C)
print("Kruskal-Wallis H:",H)
print("p-value:",p_value)
```

```
Kruskal-Wallis H: 4.220000000000006
p-value: 0.1212379664333813
```

0.4 Mann Whitney U test

[8]:
```
from scipy import stats

#sample data
sample1 = [23, 45, 67, 89, 12, 34, 56]
sample2 = [13, 25, 37, 48, 61, 71, 85]

#Perfore the Marin-Whitney test
U_statistic, p_value = stats.mannwhitneyu(sample1, sample2, alternative =␣
 ↪'two-sided')

#print the results
print(f'u-statistic: {U_statistic}')
print(f' (p-value: {p_value})')

#Interprect the p-value
alpha = 0.05 #significance level if p value
if p_value < alpha:
    print("the difference between the two samples is statistically significant.
 ↪")
else:
    print("the difference between the bo samples is statistically significant.")
```

```
u-statistic: 22.0
 (p-value: 0.8047785547785549)
the difference between the bo samples is statistically significant.
```

PRACTICAL : 8

Aim :- Perform Time Series Analysis.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

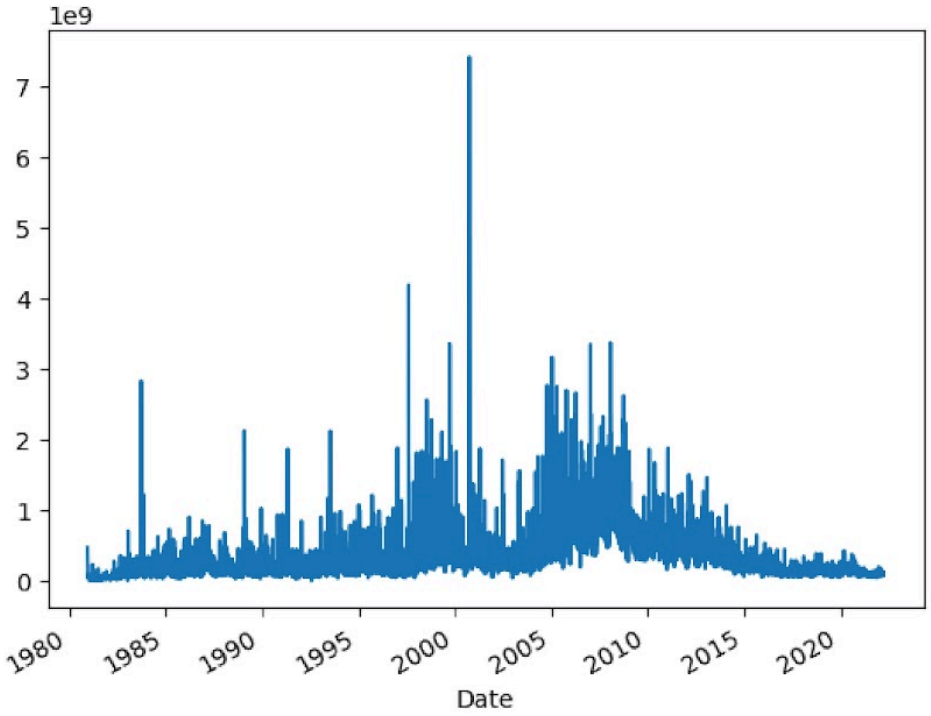
In [2]:
```
df = pd.read_csv("IFLAH_10322/AAPL.csv", parse_dates = True, index_col="Date")
df.head()
```

Out[2]:

Date	Open	High	Low	Close	Adj Close	Volume
1980-12-12	0.128348	0.128906	0.128348	0.128348	0.100323	469033600
1980-12-15	0.122210	0.122210	0.121652	0.121652	0.095089	175884800
1980-12-16	0.113281	0.113281	0.112723	0.112723	0.088110	105728000
1980-12-17	0.115513	0.116071	0.115513	0.115513	0.090291	86441600
1980-12-18	0.118862	0.119420	0.118862	0.118862	0.092908	73449600

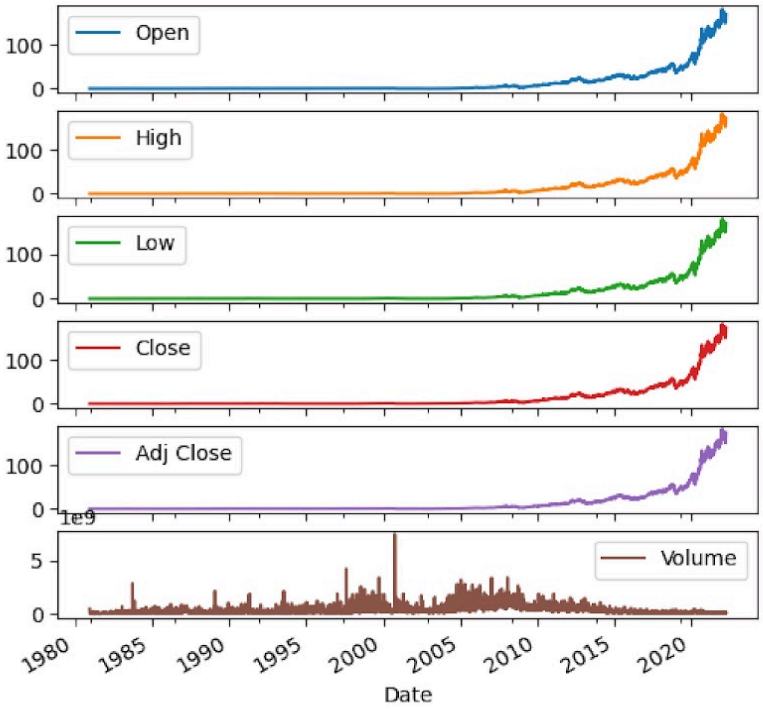
In [3]:
```
df['Volume'].plot()
```

Out[3]:
```
<Axes: xlabel='Date'>
```



In [4]:
```
df.plot(subplots = True , figsize =(6,6))
```

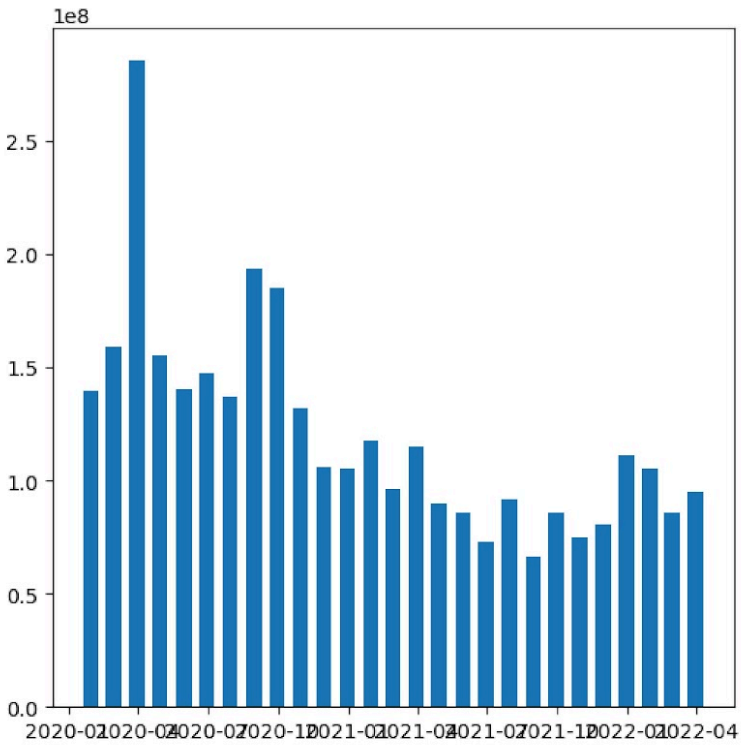
Out[4]:
```
array([<Axes: xlabel='Date'>, <Axes: xlabel='Date'>,
       <Axes: xlabel='Date'>, <Axes: xlabel='Date'>,
       <Axes: xlabel='Date'>, <Axes: xlabel='Date'>], dtype=object)
```



In [5]:
```
# Resampling the time series data based on monthly 'M' frequency
df_month = df.resample("M").mean()
```

```
fig, ax = plt.subplots(figsize=(6,6))

# ploting bar graph
ax.bar(df_month['2020':].index,
       df_month.loc['2020': , "Volume"],
       width = 20, align  = 'center')
```

Out[5]: <BarContainer object of 27 artists>



PRACTICAL : 9

Aim :- Implementation of Regression Analysis.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

[3]:
```
x = np.array([1,2,3,4,5])
y = np.array([7,14,15,18,19])
n = np.size(x)
```

[5]:
```
x_mean = np.mean(x)
y_mean = np.mean(y)
x_mean, y_mean
```

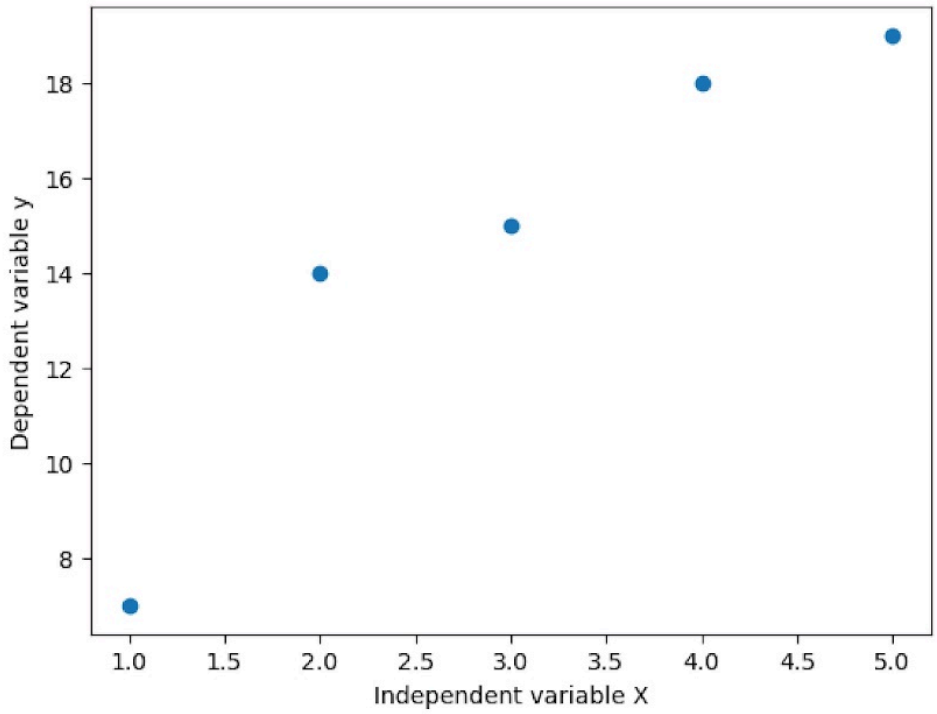
[5]: (3.0, 14.6)

[7]:
```
Sxy = np.sum(x*y)- n*x_mean*y_mean
Sxx = np.sum(x*x)- n*x_mean*x_mean
```

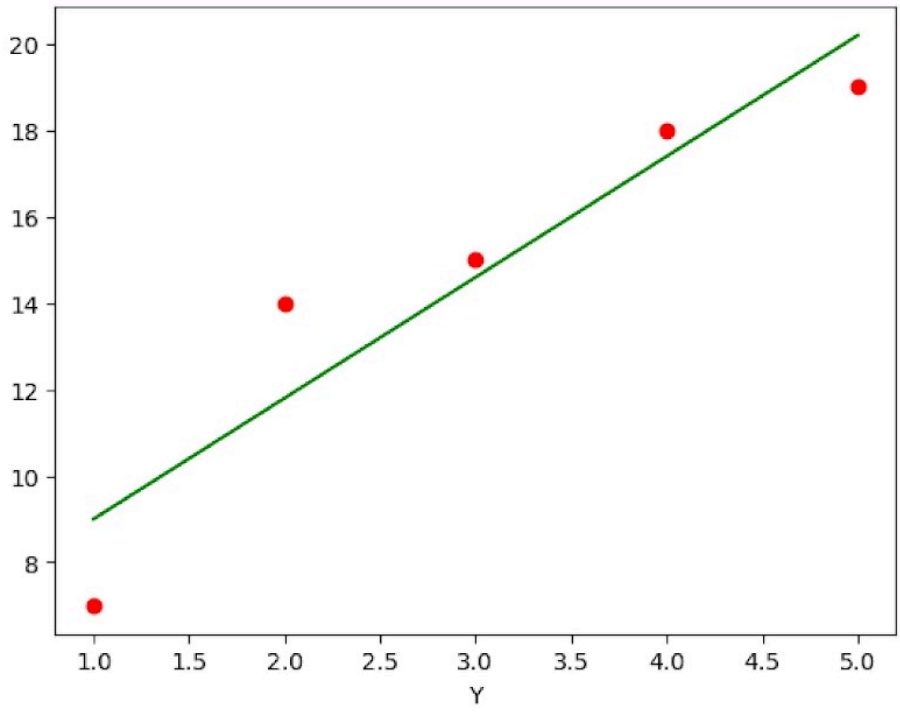
[9]:
```
b1 = Sxy/Sxx
b0 = y_mean - b1*x_mean
print('sloe b1 is', b1)
print('intercept b0 is', b0)
```

```
sloe b1 is 2.8
intercept b0 is 6.200000000000001
```

[11]:
```
plt.scatter(x,y)
plt.xlabel("Independent variable X")
plt.ylabel('Dependent variable y')
plt.show()
```



[13]:
```
y_pred = b1*x + b0
plt.scatter(x,y, color='red')
plt.plot(x,y_pred, color='green')
plt.xlabel('X')
plt.xlabel('Y')
plt.show()
```



PRACTICAL : 10

Aim :- Computing eigenvalues and eigenvectors for dimensionality reduction.

Q1. Find the eigenvalues and eigenvectors of the following metrics.

(a) $A = \begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}$

Solution :-

$A = \begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}$

- Characteristic equation

$$|A - \lambda I| = \begin{vmatrix} 2-\lambda & 2 \\ 1 & 3-\lambda \end{vmatrix} = (2-\lambda)(3-\lambda) - 2$$
$$= \lambda^2 - 5\lambda + 4$$

- Eigen values

$\lambda^2 - 5\lambda + 4 = 0$

$\lambda = 1, 4$

- Eigen vectors

For $\lambda = 1$

$$(A - I)x = 0$$
$$\begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$x_1 + 2x_2 = 0 \Rightarrow x_1 = -2x_2$

$\therefore x_1 = -2x_2$

Eigen vector $= \begin{bmatrix} -2 \\ 1 \end{bmatrix}$

For $\lambda = 4$

$$(A - 4I)x = 0$$
$$\begin{bmatrix} -2 & 2 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$-2x_1 + 2x_2 = 0 \Rightarrow x_1 = x_2$

Eigen vector $= \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Final Result →

Eigen values = 1, 4

Eigen vector = $\begin{bmatrix} -2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

(b) $A = \begin{bmatrix} 2 & -3 & 0 \\ 2 & -5 & 0 \\ 0 & 0 & 3 \end{bmatrix}$

Solution

- Characteristic equation

One eigen value is clear from bottom right block $\lambda = 3$

For top-left 2×2:

$$|A - \lambda I| = \begin{vmatrix} 2-\lambda & -3 \\ 2 & -5-\lambda \end{vmatrix} = (2-\lambda)(-5-\lambda) - (-6)$$
$$= \lambda^2 + 3\lambda - 4$$

- Eigen values

$$\lambda^2 + 3\lambda - 4 = 0$$
$$\lambda = \frac{-3 \pm \sqrt{9+16}}{2} = \frac{-3 \pm 5}{2}$$
$$\lambda = 1, -4$$

Thus, eigen values are $1, -4, 3$

Eigen vectors:

For $\lambda = 1$:

$$(A - I)x = 0$$

$$\begin{bmatrix} 1 & -3 & 0 \\ 2 & -6 & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

$x_1 - 3x_2 = 0$, $x_3 = 0$

$x_1 = 3x_2$, $x_3 = 0$

Eigen vector $= \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix}$

For $\lambda = -4$

$$(A + 4I) = 0$$

$$\begin{bmatrix} 6 & -3 & 0 \\ 2 & -1 & 0 \\ 0 & 0 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

$2x_1 - x_2 = 0$, $x_3 = 0$

$2x_1 = x_2$, $x_3 = 0$

Eigen vector $= \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$

For $\lambda = 3$

$$(A - 3I)x = 0$$

$$\begin{bmatrix} -1 & -3 & 0 \\ 2 & -8 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

$-x_1 - 3x_2 = 0$, $x_3 = 0$

$x_1 = -3x_2$, x_3 is free

Eigen vector $= \begin{bmatrix} -3 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Final Result

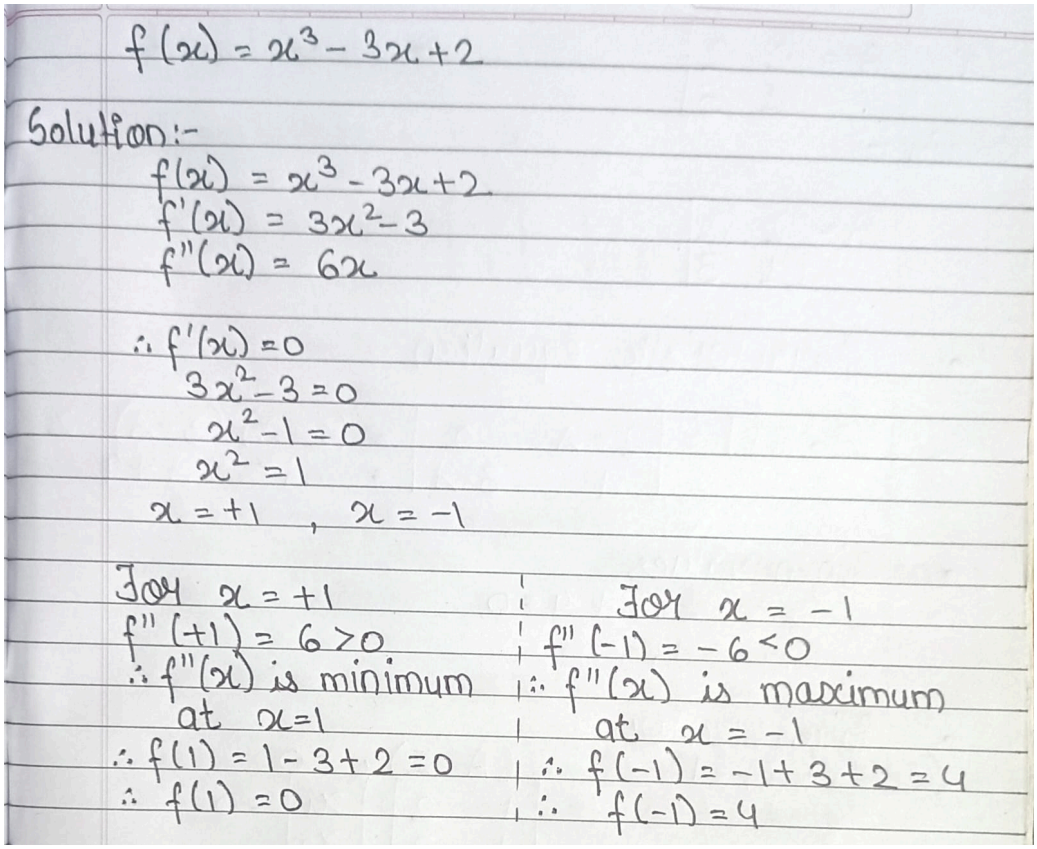
Eigen values $= 1, -4, 3$

Eigen vectors $= \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} -3 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

PRACTICAL : 11

Aim :- Calculate Maxima and Minima for the given function.

Q1.Find the local maxima and local minima



$f(x) = x^3 - 3x + 2$

Solution:-

$f(x) = x^3 - 3x + 2$

$f'(x) = 3x^2 - 3$

$f''(x) = 6x$

$\therefore f'(x) = 0$

$3x^2 - 3 = 0$

$x^2 - 1 = 0$

$x^2 = 1$

$x = +1 \quad , \quad x = -1$

For $x = +1$

$f''(+1) = 6 > 0$

$\therefore f''(x)$ is minimum at $x = 1$

$\therefore f(1) = 1 - 3 + 2 = 0$

$\therefore f(1) = 0$

For $x = -1$

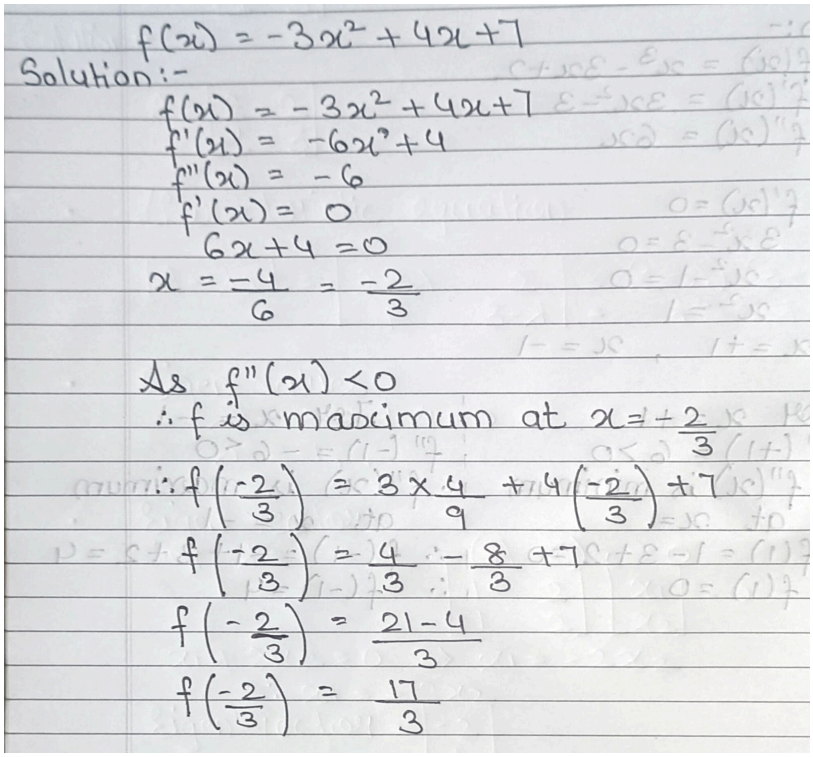
$f''(-1) = -6 < 0$

$\therefore f''(x)$ is maximum at $x = -1$

$\therefore f(-1) = -1 + 3 + 2 = 4$

$\therefore f(-1) = 4$

Q2. Find the extremum of the function and the extremum value



$f(x) = -3x^2 + 4x + 7$

Solution:-

$f(x) = -3x^2 + 4x + 7$

$f'(x) = -6x + 4$

$f''(x) = -6$

$f'(x) = 0$

$6x + 4 = 0$

$x = \frac{-4}{6} = \frac{-2}{3}$

As $f''(x) < 0$

$\therefore f$ is maximum at $x = -\frac{2}{3}$

$\therefore f\left(\frac{-2}{3}\right) = 3 \times \frac{4}{9} + 4\left(\frac{-2}{3}\right) + 7$

$f\left(\frac{-2}{3}\right) = \frac{4}{3} - \frac{8}{3} + 7$

$f\left(-\frac{2}{3}\right) = \frac{21 - 4}{3}$

$f\left(\frac{-2}{3}\right) = \frac{17}{3}$

Q3. Find the maximum height when a stone is thrown at any time 't' and height is given by

$$h = -10t^2 + 20t + 8$$

Solution:-

$$h = -10t^2 + 20t + 8$$

As we know the stone reach maximum height when velocity is zero

$$v(t) = h'(t) = -20t + 20$$

At maximum height

$$v = 0$$

$$-20t + 20 = 0$$

$$t = 1 \text{ sec}$$

For $t = 1$ sec.

$$h(1) = -10 + 20 + 8$$

$$h(1) = 18 \text{ units}$$

$\therefore$ At $t = 1$ sec, the maximum height is 18 units

PRACTICAL : 12

Aim :- Impute an example based on Rolle's and Mean Value Theorem.

Q1. Verify Rolle's Theorem.

(a) $f(x) = x^2 - 5x + 9$, $x \in [1, 4]$

Solution :-

$f(x)$ is continuous and differentiable on $[1,4]$

$f(1) = 1 - 5 + 9 = 5$

$f(4) = 16 - 20 + 9 = 5$

$\therefore f(1) = f(4)$

$\therefore f(x)$ follows Rolle's theorem

$\exists\ c \in [1,4]$ such that

$f'(c) = 0$

$2c - 5 = 0$

$2c = 5$

$c = \frac{5}{2}$

$\therefore c = 2.5 \in [1,4]$

(b) $f(x) = \sin\left(\frac{x}{2}\right)$, $x \in [0, 2\pi]$

Solution :-

$f(x)$ is continuous and differentiable on $[0, 2\pi]$

$f(0) = \sin(0) = 0$

$f(2\pi) = \sin(\pi) = 0$

$\therefore f(0) = f(2\pi)$

$\therefore f(x)$ follows Rolle's theorem

$\exists\ c \in [0, 2\pi]$ such that

$f'(c) = 0$

$\frac{1}{2}\cos\left(\frac{x}{2}\right) = 0$

$\cos\left(\frac{x}{2}\right) = 0$

$\frac{x}{2} = \cos^{-1}(0) = \left(\frac{\pi}{2}\right)^c$

$x = \pi^c$

$\therefore x \in [0, 2\pi]$

Q2. Verif LMVT

For function $f(x) = 2x^2 + 4x + 5$. Define on interval $[0,2]$. Verify LMVT

Solution:-

$f(x) = 2x^2 - 4x + 5 \quad x \in [0,2]$

$f(x)$ is continuous and differentiable on $[0,2]$

$f'(x) = 4x - 4$

Now, $f(0) = 0 + 5 = 5$

$f(2) = 8 - 8 + 5 = 5$

$\therefore f(0) = f(2)$

$\therefore$ It verifies Rolle's theorem

$$f'(c) = \frac{f(b) - f(a)}{b - a}$$

$$4c - 4 = \frac{5 - 5}{2 - 0} = \frac{0}{2} = 0$$

$4c - 4 = 0$

$4c = 4$

$c = 1$

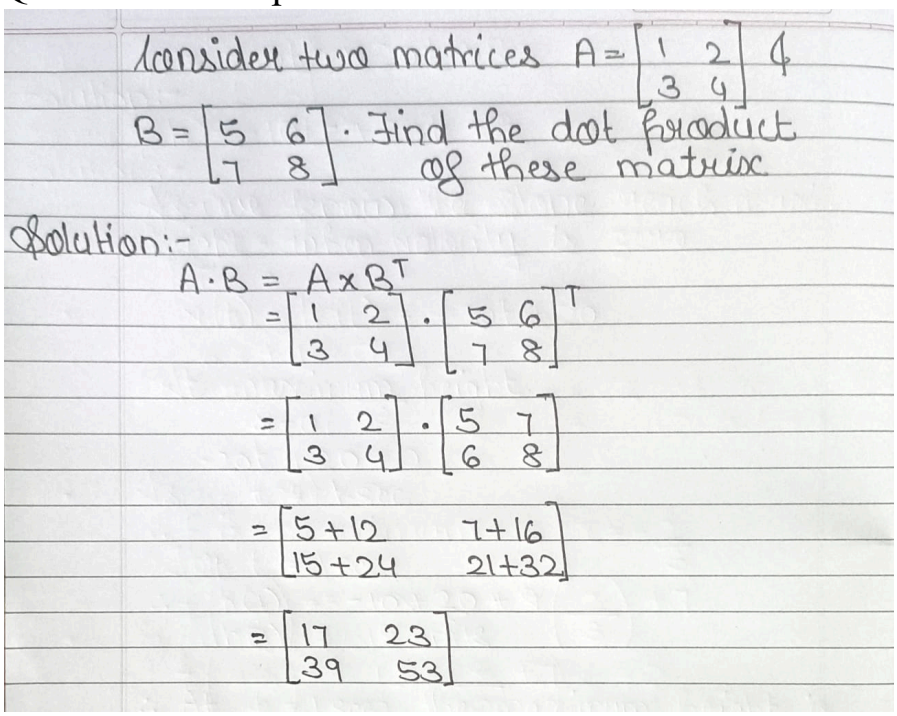
Hence $c \in [0,2]$

It verifies LMVT

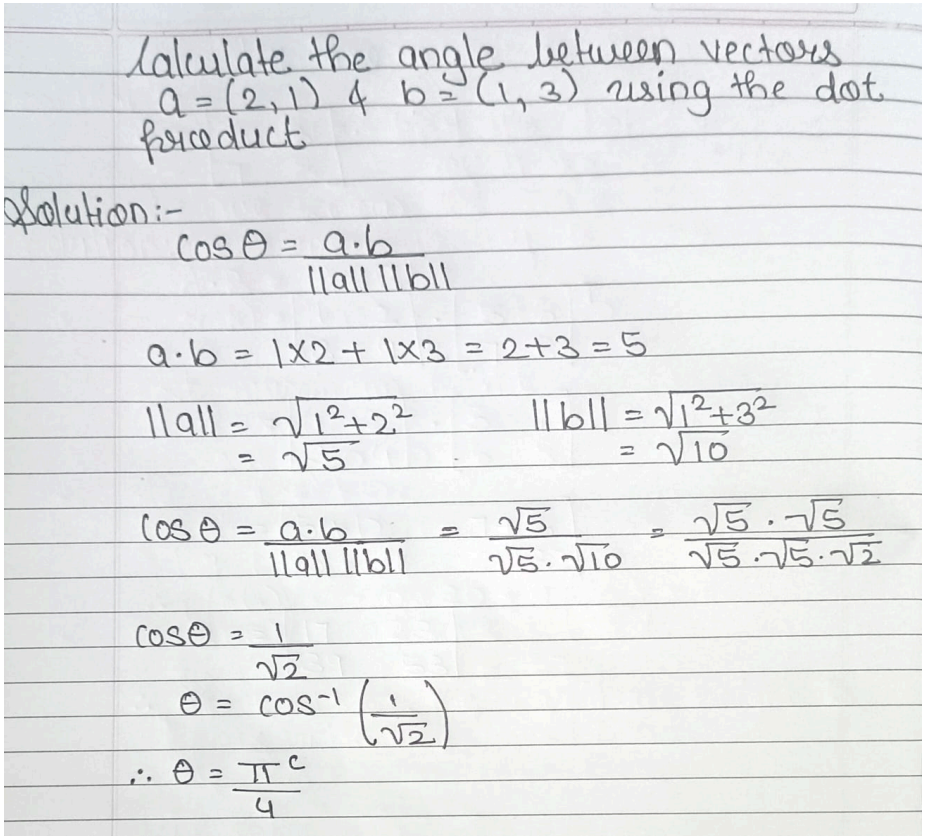
PRACTICAL : 13

Aim :- Calculate the dot product of the vector and norm of the vector.

Q1. Find the dot product of this matrix.



Q2. Calculate the angles between the vectors using dot product.



Q3. Calculate 1-norm and ∞-norm.

Given $A = \begin{bmatrix} 7 & 1 & 8 \\ 4 & 5 & 8 \\ 10 & 4 & 2 \end{bmatrix}$

Calculate 1-norm and ∞-norm

Solution :-

- 1-norm

$$\|A\|_1 = \max_{1 \le i \le n} \sum_{i=1}^{3} |a_{ij}|$$

$$\|A\|_1 = \max(7+4+10, 1+5+9, 8+8+2)$$

$$\|A\|_1 = \max(21, 10, 18)$$

$$\|A\|_1 = 21 //$$

- ∞-norm

$$\|A\|_\infty = \max_{1 \le j \le n} \sum_{j=1}^{3} |a_{ij}|$$

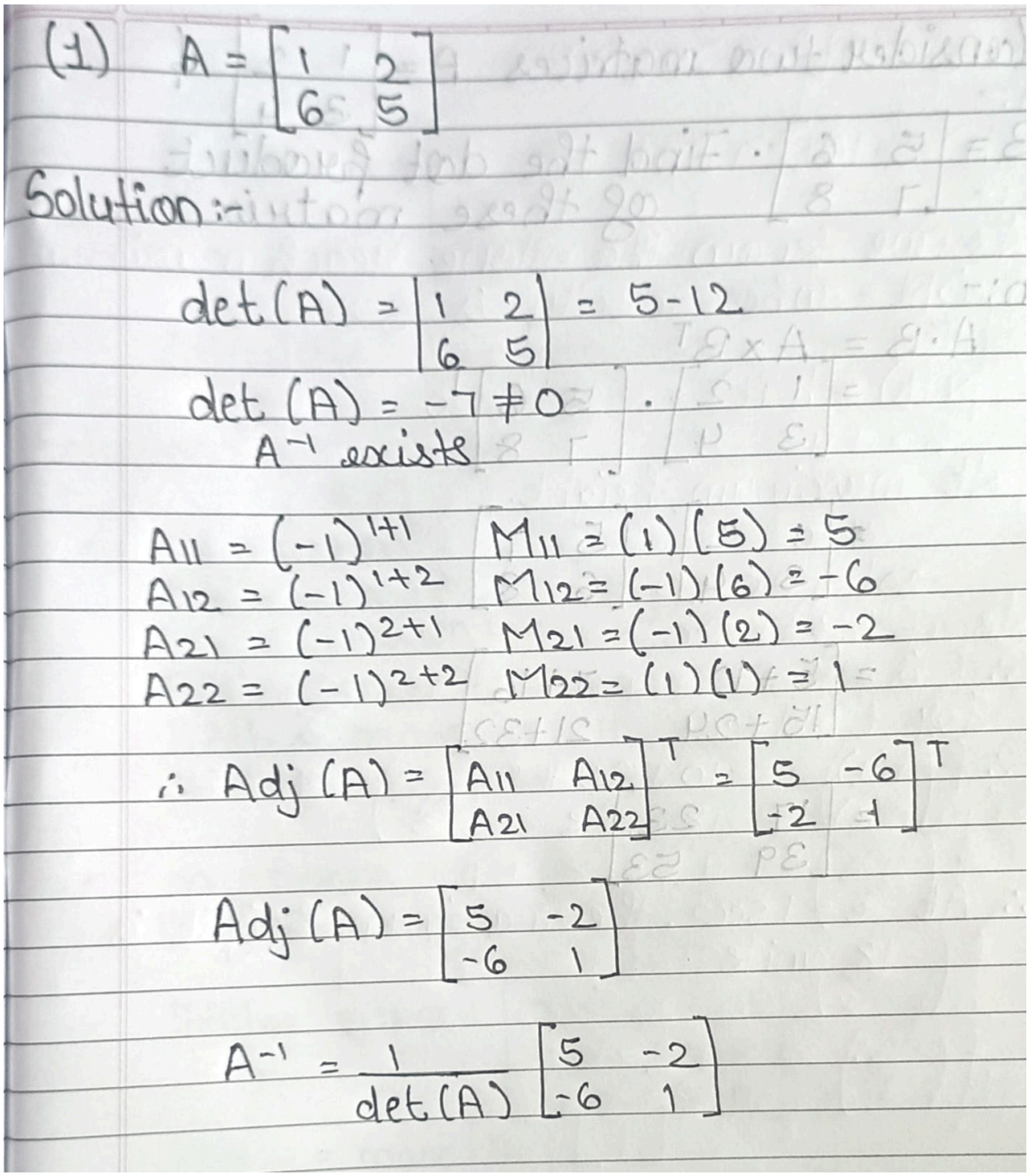
$$\|A\|_\infty = \max \begin{bmatrix} 7+1+8 \\ 4+5+8 \\ 10+4+2 \end{bmatrix}$$

$$\|A\|_\infty = \max \begin{bmatrix} 16 \\ 17 \\ 16 \end{bmatrix}$$

$$\|A\|_\infty = 17 //$$

PRACTICAL : 14

Aim :- Impute Inverse of matrix by adjoint method.



(1) $A = \begin{bmatrix} 1 & 2 \\ 6 & 5 \end{bmatrix}$

Solution :-

$$\det(A) = \begin{vmatrix} 1 & 2 \\ 6 & 5 \end{vmatrix} = 5 - 12$$

$$\det(A) = -7 \neq 0$$

A^{-1} exists

$A_{11} = (-1)^{1+1}$ $M_{11} = (1)(5) = 5$

$A_{12} = (-1)^{1+2}$ $M_{12} = (-1)(6) = -6$

$A_{21} = (-1)^{2+1}$ $M_{21} = (-1)(2) = -2$

$A_{22} = (-1)^{2+2}$ $M_{22} = (1)(1) = 1$

$$\therefore \text{Adj}(A) = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^T = \begin{bmatrix} 5 & -6 \\ -2 & 1 \end{bmatrix}^T$$

$$\text{Adj}(A) = \begin{bmatrix} 5 & -2 \\ -6 & 1 \end{bmatrix}$$

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} 5 & -2 \\ -6 & 1 \end{bmatrix}$$

(2) $A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$

Solution:-

$$\det(A) = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$\det(A) = 1(2-6) - 0(0-3) + 1(0-2)$

$\det(A) = -4 + (-2)$

$\det(A) = -6 < 0$

$\therefore A^{-1}$ exist.

$A_{11} = (-1)^{1+1} \; M_{11} = (1)\begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} = (1)(2-6) = -4$

$A_{12} = (-1)^{1+2} \; M_{12} = (-1)\begin{bmatrix} 0 & 3 \\ 1 & 1 \end{bmatrix} = (-1)(0-3) = 3$

$A_{13} = (-1)^{1+3} \; M_{13} = (-1)\begin{bmatrix} 0 & 2 \\ 1 & 2 \end{bmatrix} = (1)(0-2) = -2$

$A_{21} = (-1)^{2+1} \; M_{21} = (-1)\begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} = (-1)(0-2) = 2$

$A_{22} = (-1)^{2+2} \; M_{22} = (1)\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = (1)(1-1) = 0$

$A_{23} = (-1)^{2+3} \; M_{23} = (-1)\begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix} = (-1)(2-0) = -2$

$A_{31} = (-1)^{3+1} \; M_{31} = (1)\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = (1)(0-2) = -2$

$A_{32} = (-1)^{3+2} \; M_{32} = (-1)\begin{bmatrix} 1 & 1 \\ 0 & 3 \end{bmatrix} = (-1)(3-0) = -3$

$A_{33} = (-1)^{3+3} \; M_{33} = (1)\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} = (1)(2-0) = 2$

$$\text{Adj}(A) = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}^T = \begin{bmatrix} -4 & 3 & -2 \\ 2 & 0 & -2 \\ -2 & -3 & 2 \end{bmatrix}^T$$

$$= \begin{bmatrix} -4 & 3 & -2 \\ 2 & 0 & -2 \\ -2 & -3 & 2 \end{bmatrix}$$

$$A^{-1} = \frac{1}{\det(A)} (\text{Adj}(A))$$

$$= \frac{1}{-6} \begin{bmatrix} -4 & 3 & -2 \\ 2 & 0 & -2 \\ -2 & -3 & 2 \end{bmatrix}$$