final-ds-1

May 2, 2024

1 Data Wrangling, I

Perform the following operations using Python on any open source dataset (e.g., data.csv) 1. Import all the required Python Libraries. 2. Locate open source data from the web (e.g., https://www.kaggle.com). Provide a clear description of the data and its source (i.e., URL of the web site). 3. Load the Dataset into pandas dataframe. 4. Data Preprocessing: check for missing values in the data using pandas isnull(), describe() function to get some initial statistics. Provide variable descriptions. Types of variables etc. Check the dimensions of the data frame. 5. Data Formatting and Data Normalization: Summarize the types of variables by checking the data types (i.e., character, numeric, integer, factor, and logical) of the variables in the data set. If variables are not in the correct data type, apply proper type conversions. 6. Turn categorical variables into quantitative variables in Python.

In addition to the codes and outputs, explain every operation that you do in the above steps and explain everything that you do to import/read/scrape the data set.

[1]:
```python
import pandas as pd
```

[2]:
```python
df = pd.read_csv('StudentsPerformance.csv')
df
```

[2]:
```
     gender race/ethnicity parental level of education         lunch  \
0    female        group B           bachelor's degree      standard
1    female        group C                some college      standard
2    female        group B             master's degree      standard
3      male        group A          associate's degree  free/reduced
4      male        group C                some college      standard
..      …            …                         …             …
995  female        group E             master's degree      standard
996    male        group C                 high school  free/reduced
997  female        group C                 high school  free/reduced
998  female        group D                some college      standard
999  female        group D                some college  free/reduced

    test preparation course  math score  reading score  writing score
0                      none          72             72             74
1                 completed          69             90             88
2                      none          90             95             93
```

```
3                   none         47           57           44
4                   none         76           78           75
..                     …          …            …            …
995            completed         88           99           95
996                 none         62           55           55
997            completed         59           71           65
998            completed         68           78           77
999                 none         77           86           86

[1000 rows x 8 columns]
```

2 Date Preprocessing

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

[3]:
```
     gender  race/ethnicity  parental level of education  lunch  \
0     False           False                        False  False
1     False           False                        False  False
2     False           False                        False  False
3     False           False                        False  False
4     False           False                        False  False
..      …              …                          …      …
995   False           False                        False  False
996   False           False                        False  False
997   False           False                        False  False
998   False           False                        False  False
999   False           False                        False  False

     test preparation course  math score  reading score  writing score
0                      False       False          False          False
1                      False       False          False          False
2                      False       False          False          False
3                      False       False          False          False
4                      False       False          False          False
..                       …          …              …             …
995                    False       False          False          False
996                    False       False          False          False
997                    False       False          False          False
998                    False       False          False          False
999                    False       False          False          False

[1000 rows x 8 columns]
```

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

[4]:
```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

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

[5]:

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64
 6   reading score                1000 non-null   int64
 7   writing score                1000 non-null   int64
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

[7]:
```
df.shape
```

[7]:
```
(1000, 8)
```

[10]:
```
df['writing score'] = df['writing score'].astype(float)
```

[11]:
```
df
```

[11]:
```
      gender race/ethnicity parental level of education        lunch  \
0     female        group B           bachelor's degree     standard
1     female        group C                some college     standard
2     female        group B             master's degree     standard
3       male        group A          associate's degree  free/reduced
4       male        group C                some college     standard
..       …              …                         …            …
995   female        group E             master's degree     standard
996     male        group C                 high school  free/reduced
997   female        group C                 high school  free/reduced
998   female        group D                some college     standard
999   female        group D                some college  free/reduced

    test preparation course  math score  reading score  writing score
0                      none          72             72           74.0
1                 completed          69             90           88.0
2                      none          90             95           93.0
3                      none          47             57           44.0
4                      none          76             78           75.0
..                      …           …             …             …
995               completed          88             99           95.0
996                    none          62             55           55.0
997               completed          59             71           65.0
998               completed          68             78           77.0
999                    none          77             86           86.0

[1000 rows x 8 columns]
```

[14]:
```python
df['gender'].replace({'male' : 1 , 'female' : 0} , inplace=True)
```

[15]:
```python
df
```

[15]:
```
      gender race/ethnicity parental level of education        lunch  \
0          0        group B           bachelor's degree     standard
1          0        group C                some college     standard
2          0        group B             master's degree     standard
3          1        group A          associate's degree  free/reduced
4          1        group C                some college     standard
..       …              …                         …            …
995        0        group E             master's degree     standard
996        1        group C                 high school  free/reduced
997        0        group C                 high school  free/reduced
998        0        group D                some college     standard
999        0        group D                some college  free/reduced

    test preparation course  math score  reading score  writing score
0                      none          72             72           74.0
```

```
1          completed     69      90      88.0
2               none     90      95      93.0
3               none     47      57      44.0
4               none     76      78      75.0
..               …      …       …       …
995        completed     88      99      95.0
996             none     62      55      55.0
997        completed     59      71      65.0
998        completed     68      78      77.0
999             none     77      86      86.0

[1000 rows x 8 columns]
```

[16]:
```python
df.ndim
```

[16]: 2

3 To learn

1. Normalization and its techniques and way to import it in sklearn
2. One Hot encoding and Label Encoder

4 What is Data Preprocessing ?

Data preprocessing is the process of transforming raw data into a suitable format for analysis and machine learning. It involves cleaning the data (handling missing values, removing duplicates), transforming it (scaling, encoding categorical variables), and sometimes reducing its size (dimensionality reduction, feature selection).

This step is crucial to ensure models are trained on accurate, consistent, and standardized data, leading to better performance and more reliable results. Proper preprocessing simplifies data analysis and enhances interpretability.

5 What is One Hot Encoding?

One-hot encoding is a method used to convert categorical variables into a format suitable for machine learning models.

It creates new binary columns for each unique category in the original feature.

If a feature has three categories ("Red," "Green," "Blue"), one-hot encoding generates three new columns, with a value of "1" indicating the presence of that category and "0" otherwise.

6 What is Label Encoding ?

Label encoding is a method for converting categorical variables into numerical format by assigning a unique integer to each category.

If a feature has categories like “Small,” “Medium,” and “Large,” label encoding might map them to 0, 1, and 2, respectively.

7 Types of Scaling

Min-Max Scaling

Range between 0 and 1

Formula : X^’ = X - min(X) / max(X) - min(X)

Standardization

Have a mean of 0 and a standard deviation of 1.

X = X− /

```
[ ]:
```

final-ds-2

May 2, 2024

1 Data Wrangling II

Create an "Academic performance" dataset of students and perform the following operations using Python.

1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them.
2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.
3. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution.

Reason and document your approach properly.

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

# Ignore all warnings
warnings.filterwarnings("ignore")
```

[2]:
```
df = pd.read_csv('StudentsPerformance.csv')
df
```

[2]:
```
     gender race/ethnicity parental level of education         lunch  \
0    female        group B           bachelor's degree      standard
1    female        group C                some college      standard
2    female        group B             master's degree      standard
3      male        group A          associate's degree  free/reduced
4      male        group C                some college      standard
..      …            …                          …             …
995  female        group E             master's degree      standard
996    male        group C                 high school  free/reduced
997  female        group C                 high school  free/reduced
998  female        group D                some college      standard
```

```
999  female        group D                some college  free/reduced

    test preparation course  math score  reading score  writing score
0                      none          72             72             74
1                 completed          69             90             88
2                      none          90             95             93
3                      none          47             57             44
4                      none          76             78             75
..                      …          …           …           …
995               completed          88             99             95
996                    none          62             55             55
997               completed          59             71             65
998               completed          68             78             77
999                    none          77             86             86

[1000 rows x 8 columns]
```

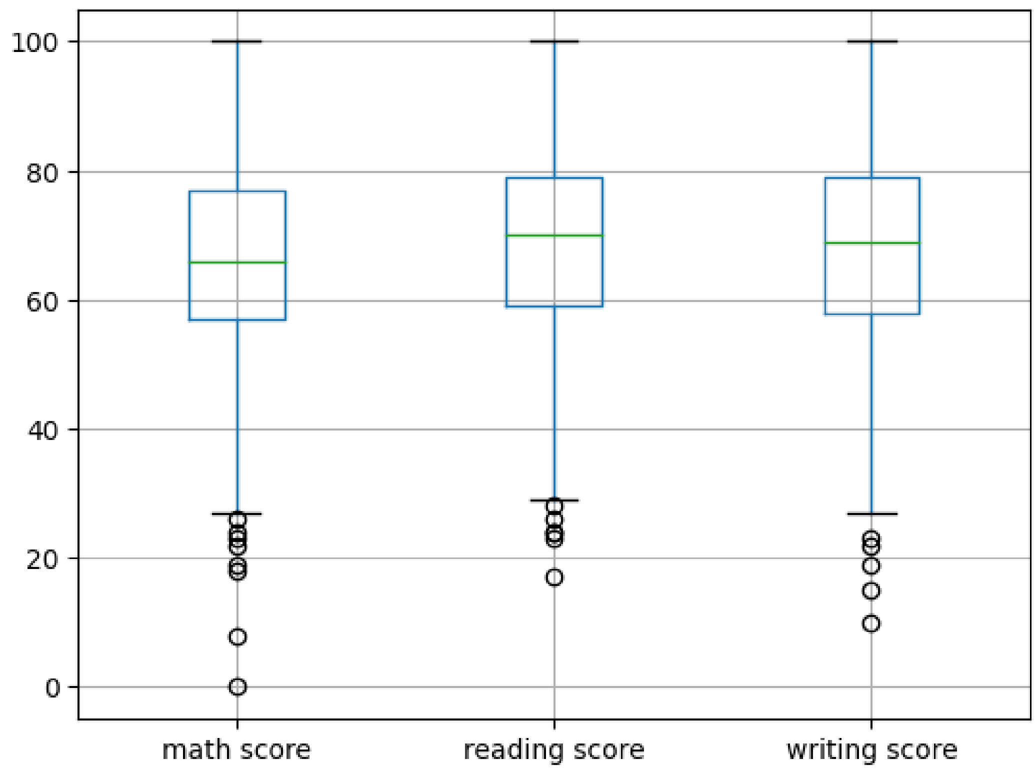
1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them.

for this use fillna() method either by filling it with mean , mode , median

df['col'] = df['col'].fillna(df['col'].mean() , inplace = True)

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

[3]:
```
<Axes: >
```



[4]:
```
Q1 = df['math score'].quantile(0.25)
Q3 = df['math score'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - (1.5 * IQR)
upper_bound = Q3 + (1.5 * IQR)

outliers = df[(df['math score'] < lower_bound) | (df['math score'] >␣
↪upper_bound)]
outliers
```

[4]:
```
    gender race/ethnicity parental level of education         lunch  \
17   female        group B            some high school  free/reduced
59   female        group C            some high school  free/reduced
145  female        group C                some college  free/reduced
338  female        group B            some high school  free/reduced
466  female        group D          associate's degree  free/reduced
787  female        group B                some college      standard
842  female        group B                 high school  free/reduced
980  female        group B                 high school  free/reduced

    test preparation course  math score  reading score  writing score
17                     none          18             32             28
```

```
59                       none           0          17          10
145                      none          22          39          33
338                      none          24          38          27
466                      none          26          31          38
787                      none          19          38          32
842                 completed          23          44          36
980                      none           8          24          23
```

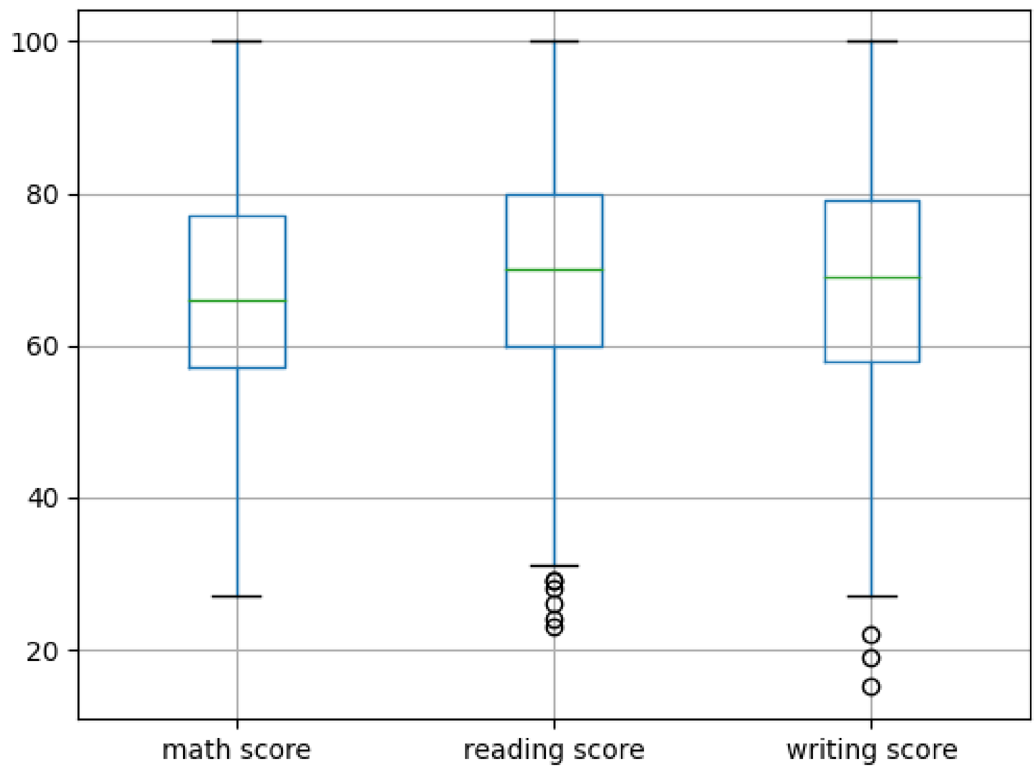
[5]:
```
df_cleaned = df[(df['math score'] >= lower_bound) & (df['math score'] <=
 ↪upper_bound)]
```

[6]:
```
df_cleaned
```

[6]:
```
     gender race/ethnicity parental level of education         lunch  \
0    female        group B           bachelor's degree      standard
1    female        group C                some college      standard
2    female        group B             master's degree      standard
3      male        group A          associate's degree  free/reduced
4      male        group C                some college      standard
..      …              …                          …           …
995  female        group E             master's degree      standard
996    male        group C                 high school  free/reduced
997  female        group C                 high school  free/reduced
998  female        group D                some college      standard
999  female        group D                some college  free/reduced

    test preparation course  math score  reading score  writing score
0                      none          72             72             74
1                 completed          69             90             88
2                      none          90             95             93
3                      none          47             57             44
4                      none          76             78             75
..                      …           …              …             …
995               completed          88             99             95
996                    none          62             55             55
997               completed          59             71             65
998               completed          68             78             77
999                    none          77             86             86

[992 rows x 8 columns]
```

[7]:
```
df_cleaned.boxplot()
```

[7]: <Axes: >



2 Transformation

[8]:
```
df_cleaned['gender'] = df_cleaned['gender'].replace({'male' : 1 , 'female' : 0})
```

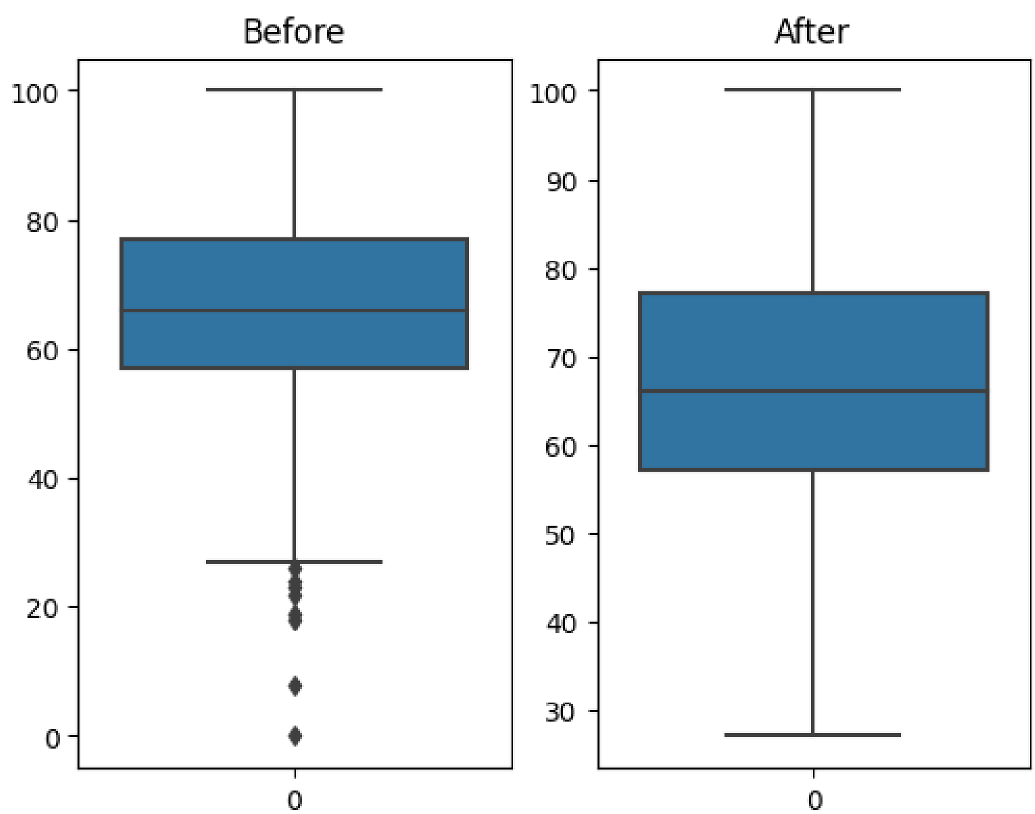
[9]:
```
df_cleaned
```

[9]:
```
     gender race/ethnicity parental level of education         lunch  \
0         0        group B           bachelor's degree      standard
1         0        group C                some college      standard
2         0        group B             master's degree      standard
3         1        group A          associate's degree  free/reduced
4         1        group C                some college      standard
..      …            …                         …             …
995       0        group E             master's degree      standard
996       1        group C                 high school  free/reduced
997       0        group C                 high school  free/reduced
998       0        group D                some college      standard
999       0        group D                some college  free/reduced

    test preparation course  math score  reading score  writing score
0                      none          72             72             74
```

```
1                  completed        69           90           88
2                       none        90           95           93
3                       none        47           57           44
4                       none        76           78           75
..                        …     …            …            …
995                completed        88           99           95
996                     none        62           55           55
997                completed        59           71           65
998                completed        68           78           77
999                     none        77           86           86

[992 rows x 8 columns]
```

[ ]:

[10]:
```
fig, axes= plt.subplots(1,2)
sns.boxplot(df['math score'],ax=axes[0])
axes[0].title.set_text('Before')
sns.boxplot(df_cleaned['math score'],ax=axes[1])
axes[1].title.set_text('After')
plt.show()
```



[11]:
```python
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df_cleaned['reading score'] = scaler.fit_transform(df_cleaned[['reading␣
↪score']])

df_cleaned
```

[11]:
```
     gender race/ethnicity parental level of education         lunch  \
0         0        group B           bachelor's degree      standard
1         0        group C                some college      standard
2         0        group B             master's degree      standard
3         1        group A          associate's degree  free/reduced
4         1        group C                some college      standard
..      …            …                         …             …
995       0        group E             master's degree      standard
996       1        group C                 high school  free/reduced
997       0        group C                 high school  free/reduced
998       0        group D                some college      standard
999       0        group D                some college  free/reduced

    test preparation course  math score  reading score  writing score
0                      none          72       0.636364             74
1                 completed          69       0.870130             88
2                      none          90       0.935065             93
3                      none          47       0.441558             44
4                      none          76       0.714286             75
..                      …          …            …             …
995               completed          88       0.987013             95
996                    none          62       0.415584             55
997               completed          59       0.623377             65
998               completed          68       0.714286             77
999                    none          77       0.818182             86

[992 rows x 8 columns]
```

[ ]:

final-ds-3

May 2, 2024

1 Descriptive Statistics - Measures of Central Tendency and variability

Perform the following operations on any open source dataset (e.g., data.csv) 1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variable. For example, if your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable. 2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris-versicolor' of iris.csv dataset. Provide the codes with outputs and explain everything that you do in this step

[1]:
```
import pandas as pd
```

[2]:
```
df = pd.read_csv('academic_performance.csv')
df
```

[2]:
```
    Student_ID  Gender  Age  Math_Score  Science_Score  English_Score
0        S0001    Male   23          19             71             90
1        S0002  Female   18          22             25             15
2        S0003  Female   23          11            100             68
3        S0004    Male   17          11             77             74
4        S0005  Female   22          33             50             85
..         …       …   …         …            …             …
495      S0496    Male   17          65             96             51
496      S0497  Female   16          99             61             34
497      S0498    Male   16          71              1             58
498      S0499  Female   21           1             47             80
499      S0500    Male   22           1             43             28

[500 rows x 6 columns]
```

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

[3]:
```
              Age  Math_Score  Science_Score  English_Score
count  500.000000  500.000000     500.000000     500.000000
mean    20.438000   49.494000      51.040000      50.188000
```

```
std     2.833056   30.141782    29.182969    28.702968
min    16.000000    0.000000     0.000000     0.000000
25%    18.000000   22.750000    25.000000    26.750000
50%    20.000000   49.000000    51.000000    51.000000
75%    23.000000   77.000000    76.250000    74.000000
max    25.000000  100.000000   100.000000   100.000000
```

[4]:
```
df.groupby('Gender')[['Math_Score' , 'Science_Score' , 'English_Score']].sum()
```

[4]:
```
        Math_Score  Science_Score  English_Score
Gender
Female       12287          13260          13070
Male         12460          12260          12024
```

[5]:
```
df.groupby('Gender')[['Math_Score' , 'Science_Score' , 'English_Score']].
 ↪describe().T
```

[5]:
```
Gender                    Female        Male
Math_Score    count  262.000000  238.000000
              mean    46.896947   52.352941
              std     29.865485   30.248224
              min      0.000000    1.000000
              25%     20.000000   24.250000
              50%     44.000000   56.000000
              75%     73.750000   81.000000
              max    100.000000  100.000000
Science_Score count  262.000000  238.000000
              mean    50.610687   51.512605
              std     28.881200   29.565308
              min      1.000000    0.000000
              25%     25.250000   25.000000
              50%     51.000000   50.500000
              75%     75.000000   78.000000
              max    100.000000  100.000000
English_Score count  262.000000  238.000000
              mean    49.885496   50.521008
              std     28.364119   29.127640
              min      0.000000    0.000000
              25%     27.000000   26.000000
              50%     50.000000   53.000000
              75%     73.500000   74.750000
              max    100.000000  100.000000
```

[6]:
```
flower = pd.read_csv('Iris (1).csv')
flower
```

[6]:
```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   …              …             …              …             …
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              …
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

[7]:
```
flower.describe()
```

[7]:
```
               Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000
```

[8]:
```
f = flower.groupby(by='Species')
```

[9]:
```
f.first()
```

[9]:
```
                  Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
Species
Iris-setosa        1            5.1           3.5            1.4           0.2
```

```
Iris-versicolor    51             7.0          3.2            4.7           1.4
Iris-virginica    101             6.3          3.3            6.0           2.5
```

[10]:
```python
flower.groupby('Species').describe().T
```

[10]:
```
Species                Iris-setosa  Iris-versicolor  Iris-virginica
Id            count      50.000000        50.000000       50.000000
              mean       25.500000        75.500000      125.500000
              std        14.577380        14.577380       14.577380
              min         1.000000        51.000000      101.000000
              25%        13.250000        63.250000      113.250000
              50%        25.500000        75.500000      125.500000
              75%        37.750000        87.750000      137.750000
              max        50.000000       100.000000      150.000000
SepalLengthCm count      50.000000        50.000000       50.000000
              mean        5.006000         5.936000        6.588000
              std         0.352490         0.516171        0.635880
              min         4.300000         4.900000        4.900000
              25%         4.800000         5.600000        6.225000
              50%         5.000000         5.900000        6.500000
              75%         5.200000         6.300000        6.900000
              max         5.800000         7.000000        7.900000
SepalWidthCm  count      50.000000        50.000000       50.000000
              mean        3.418000         2.770000        2.974000
              std         0.381024         0.313798        0.322497
              min         2.300000         2.000000        2.200000
              25%         3.125000         2.525000        2.800000
              50%         3.400000         2.800000        3.000000
              75%         3.675000         3.000000        3.175000
              max         4.400000         3.400000        3.800000
PetalLengthCm count      50.000000        50.000000       50.000000
              mean        1.464000         4.260000        5.552000
              std         0.173511         0.469911        0.551895
              min         1.000000         3.000000        4.500000
              25%         1.400000         4.000000        5.100000
              50%         1.500000         4.350000        5.550000
              75%         1.575000         4.600000        5.875000
              max         1.900000         5.100000        6.900000
PetalWidthCm  count      50.000000        50.000000       50.000000
              mean        0.244000         1.326000        2.026000
              std         0.107210         0.197753        0.274650
              min         0.100000         1.000000        1.400000
              25%         0.200000         1.200000        1.800000
              50%         0.200000         1.300000        2.000000
              75%         0.300000         1.500000        2.300000
              max         0.600000         1.800000        2.500000
```

2 What is groupby in pandas ?

Groupby in pandas is a method that groups a DataFrame's rows based on one or more columns, allowing you to perform aggregate operations on each group.

3 What is Data Science ?

Data Science is an interdisciplinary field that combines statistical analysis, machine learning, data engineering, and domain expertise to extract insights and knowledge from data. It involves collecting, cleaning, analyzing, and interpreting data to inform decision-making and create data-driven solutions. Data Science is used across industries to solve complex problems and guide strategic decisions.

4 Pandas

Pandas is an open-source data analysis and manipulation library for Python, designed to work with structured data.

It provides powerful data structures like DataFrames and Series, allowing you to perform complex operations such as filtering, grouping, merging, and aggregating data with ease.

final-ds-4

May 2, 2024

1 Data Analytics I

Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset.

The objective is to predict the value of prices of the house using the given features.

[1]:
```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score , mean_squared_error
import numpy as np
```

[2]:
```
df = pd.read_csv('boston-housing.csv')
df
```

[2]:
```
      ID     crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0      1  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1      2  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2      4  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
3      5  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
4      7  0.08829  12.5   7.87     0  0.524  6.012  66.6  5.5605    5  311
..   …       …     …     …   …      …     …    …      …   …   …
328  500  0.17783   0.0   9.69     0  0.585  5.569  73.5  2.3999    6  391
329  502  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
330  503  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
331  504  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
332  506  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio   black  lstat  medv
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       18.7  394.63   2.94  33.4
3       18.7  396.90   5.33  36.2
4       15.2  395.60  12.43  22.9
..       …       …     …    …
```

```
328      19.2  395.77  15.10  17.5
329      21.0  391.99   9.67  22.4
330      21.0  396.90   9.08  20.6
331      21.0  396.90   5.64  23.9
332      21.0  396.90   7.88  11.9

[333 rows x 15 columns]
```

[3]:
```python
df.describe()
```

[3]:
```
               ID        crim          zn       indus        chas         nox  \
count  333.000000  333.000000  333.000000  333.000000  333.000000  333.000000
mean   250.951952    3.360341   10.689189   11.293483    0.060060    0.557144
std    147.859438    7.352272   22.674762    6.998123    0.237956    0.114955
min      1.000000    0.006320    0.000000    0.740000    0.000000    0.385000
25%    123.000000    0.078960    0.000000    5.130000    0.000000    0.453000
50%    244.000000    0.261690    0.000000    9.900000    0.000000    0.538000
75%    377.000000    3.678220   12.500000   18.100000    0.000000    0.631000
max    506.000000   73.534100  100.000000   27.740000    1.000000    0.871000

               rm         age         dis         rad         tax     ptratio  \
count  333.000000  333.000000  333.000000  333.000000  333.000000  333.000000
mean     6.265619   68.226426    3.709934    9.633634  409.279279   18.448048
std      0.703952   28.133344    1.981123    8.742174  170.841988    2.151821
min      3.561000    6.000000    1.129600    1.000000  188.000000   12.600000
25%      5.884000   45.400000    2.122400    4.000000  279.000000   17.400000
50%      6.202000   76.700000    3.092300    5.000000  330.000000   19.000000
75%      6.595000   93.800000    5.116700   24.000000  666.000000   20.200000
max      8.725000  100.000000   10.710300   24.000000  711.000000   21.200000

            black       lstat        medv
count  333.000000  333.000000  333.000000
mean   359.466096   12.515435   22.768769
std     86.584567    7.067781    9.173468
min      3.500000    1.730000    5.000000
25%    376.730000    7.180000   17.400000
50%    392.050000   10.970000   21.600000
75%    396.240000   16.420000   25.000000
max    396.900000   37.970000   50.000000
```

[4]:
```python
df.columns
```

[4]:
```
Index(['ID', 'crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad',
       'tax', 'ptratio', 'black', 'lstat', 'medv'],
      dtype='object')
```

[5]:
```python
x = df[['ID','crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis',␣
↪'rad','tax', 'ptratio', 'black', 'lstat']]
y = df['medv']
```

[6]:
```python
x_train , x_test , y_train , y_test = train_test_split(x,y,test_size=0.25 ,␣
↪random_state=42)
```

[7]:
```python
lr = LinearRegression()
```

[8]:
```python
lr.fit(x_train , y_train)
```

[8]:
```
LinearRegression()
```

[9]:
```python
y_pred = lr.predict(x_test)
y_pred
```

[9]:
```
array([25.42935235, 22.96355416, 22.98361313, 32.78417799, 25.00393879,
       14.02939811, 17.33416663, 30.26235415, 15.66094194, 25.17645124,
       26.57561017, 19.87047671, 20.0909698 , 34.65458829, 21.54826488,
       34.56267042, 22.2485164 , 16.882139  , 25.36436429, 16.87727359,
       36.17705665, 31.39060697, 22.8003689 , 28.13000873, 17.20049765,
       42.37151734,  7.87497041, -0.36689721, 31.09480595,  8.4983594 ,
       19.11933223, 20.39461378, 27.46125616, 15.1123262 , 19.49023076,
       12.24379213, 27.56523215,  4.49445516, 17.2050638 , 22.53286675,
       24.31038054, 22.18772176, 25.17245639, 39.17706326, 36.30722879,
       21.36801908, 11.07627164, 21.45992975, 13.86557089, 20.5226888 ,
       13.18677931, 27.99316461, 21.49698121, 13.63192997, 33.57300717,
        1.31418106, 20.80538627, 27.16216548, 25.27755635, 28.15722941,
       17.87759693, 23.94807788, 18.11868011, 28.38039855, 23.49353731,
       16.58474878, 26.43944232, 20.85487585, 23.73111221, 16.67730384,
       14.83654746, 26.14692568, 21.12274293, 12.8361667 , 24.2684098 ,
       33.5090989 , 22.02205101, 23.54787016, 22.92262304, 16.50548743,
       21.16265595, 28.70685258, 40.24134644, 15.29564422])
```

[10]:
```python
lr.score(x_train , y_train) #Training Accuracy
```

[10]:
```
0.7322764285677805
```

[11]:
```python
lr.score(x_test,y_test) #Testing Accuracy
```

[11]:
```
0.7257587357992887
```

[12]:
```python
r2_Score = r2_score(y_test,y_pred) #closer to 1 indicates better model
r2_Score
```

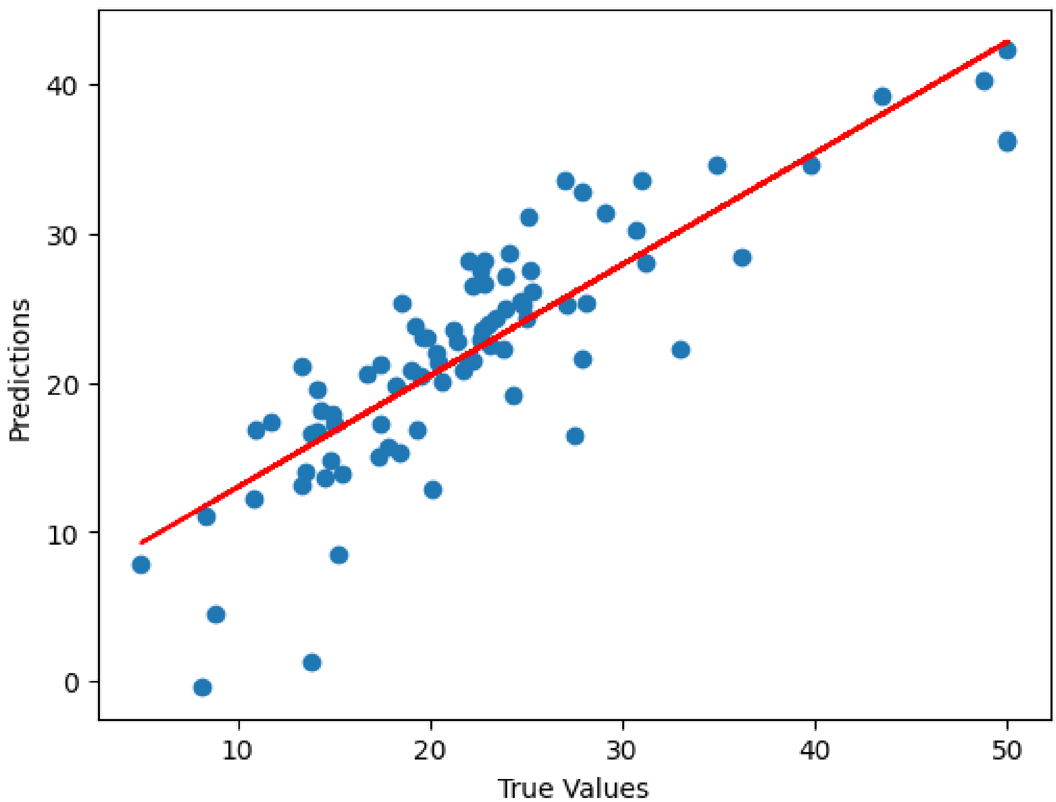
[12]:
```
0.7257587357992887
```

[13]:
```
np.sqrt(mean_squared_error(y_test,y_pred))  # smaller rmse indicates better␣
↪model
```

[13]: 4.823701046478235

[14]:
```
import matplotlib.pyplot as plt
```

[15]:
```
coef = np.polyfit(y_test,y_pred,1)
poly = np.poly1d(coef)
y_fit = poly(y_test)

plt.scatter(x=y_test , y=y_pred)
plt.plot(y_test,y_fit,color = 'red')
plt.xlabel('True Values')
plt.ylabel('Predictions')
plt.show()
```



2 What is Linear Regression ?

Linear regression is a statistical method for modeling the relationship between a dependent variable (target) and one or more independent variables (predictors) using a linear equation. The goal is to find the best-fitting line or hyperplane that minimizes the sum of the squared differences between observed and predicted values.

3 What is r2 Score ?

The R^2 score, or coefficient of determination, measures how well a regression model explains the variance in the dependent variable. It ranges from 0 to 1, where a score of 1 indicates perfect prediction, and a score of 0 means the model does not explain any variance beyond the mean.

[ ]:

```

```

final-ds-5

May 2, 2024

1 5) Data Analytics II

1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score , confusion_matrix , precision_score␣
↪, recall_score , classification_report
```

[2]:
```
df = pd.read_csv('Social_Network_Ads.csv')
df
```

[2]:
```
      User ID  Gender  Age  EstimatedSalary  Purchased
0    15624510    Male   19            19000          0
1    15810944    Male   35            20000          0
2    15668575  Female   26            43000          0
3    15603246  Female   27            57000          0
4    15804002    Male   19            76000          0
..        …       …   …              …        …
395  15691863  Female   46            41000          1
396  15706071    Male   51            23000          1
397  15654296  Female   50            20000          1
398  15755018    Male   36            33000          0
399  15594041  Female   49            36000          1

[400 rows x 5 columns]
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
```

 #   Column   Non-Null Count Dtype
--- ------   -------------- -----
 0   User ID   400 non-null   int64
 1   Gender   400 non-null   object
 2   Age   400 non-null   int64
 3   EstimatedSalary 400 non-null   int64
 4   Purchased   400 non-null   int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB

[4]:
```
df.columns
```

[4]:
```
Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'],
dtype='object')
```

[5]:
```
# Male : 1 Female : 0
df['Gender'].replace({'Male' : 1 , 'Female' : 0} , inplace=True)
```

[6]:
```
df
```

[6]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	1	19	19000	0
1	15810944	1	35	20000	0
2	15668575	0	26	43000	0
3	15603246	0	27	57000	0
4	15804002	1	19	76000	0
..	…	…	…	…	…
395	15691863	0	46	41000	1
396	15706071	1	51	23000	1
397	15654296	0	50	20000	1
398	15755018	1	36	33000	0
399	15594041	0	49	36000	1

[400 rows x 5 columns]

[7]:
```
x = df[['Gender', 'Age', 'EstimatedSalary']]
x
```

[7]:

	Gender	Age	EstimatedSalary
0	1	19	19000
1	1	35	20000
2	0	26	43000
3	0	27	57000
4	1	19	76000
..	…	…	…
395	0	46	41000
396	1	51	23000

```
397       0   50           20000
398       1   36           33000
399       0   49           36000

[400 rows x 3 columns]
```

[8]:
```
y = df[['Purchased']]
y
```

[8]:
```
     Purchased
0            0
1            0
2            0
3            0
4            0
..         ...
395          1
396          1
397          1
398          0
399          1

[400 rows x 1 columns]
```

[9]:
```
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.
↪25,random_state=42)
```

[10]:
```
lr = LogisticRegression()
```

[11]:
```
lr.fit(X_train,y_train)
```

```
C:\Users\vedan\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2
kfra8p0\LocalCache\local-packages\Python39\site-
packages\sklearn\utils\validation.py:1300: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\vedan\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2
kfra8p0\LocalCache\local-packages\Python39\site-
packages\sklearn\linear_model\_logistic.py:469: 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(
```

[11]: LogisticRegression()

```
[12]: y_pred = lr.predict(X_test)
      y_pred
```

```
[12]: array([0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
             0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
             0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
             1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
             0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0], dtype=int64)
```

```
[13]: accuracy = accuracy_score(y_test , y_pred)
      accuracy
```

[13]: 0.89

```
[27]: error_rate = 1 - accuracy
      round(error_rate,2)
```

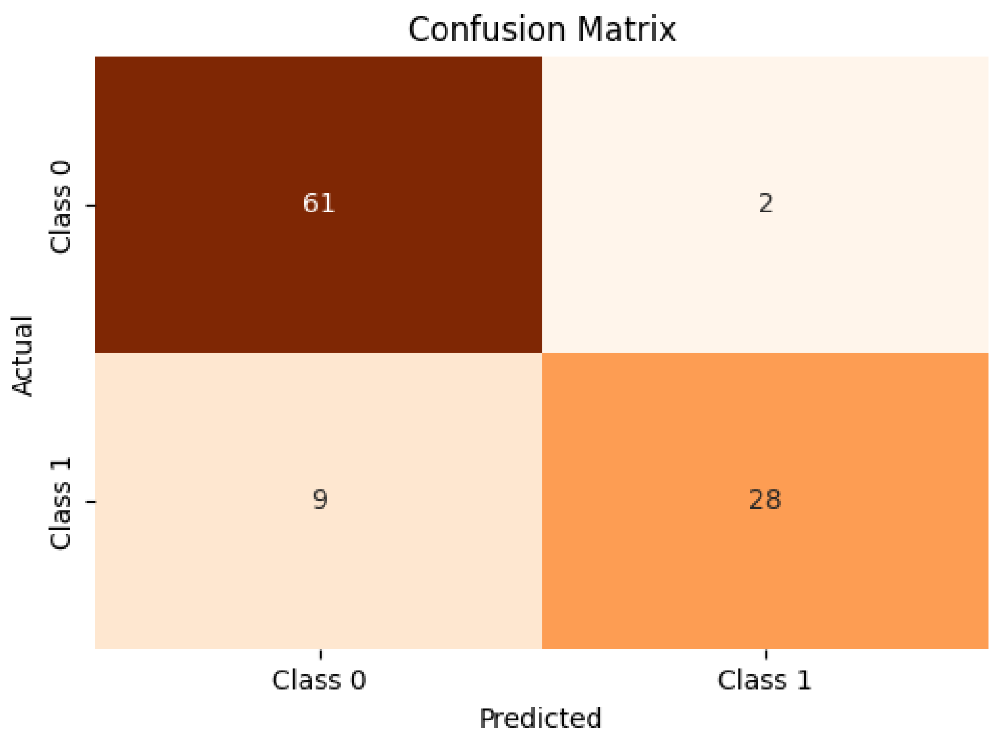
[27]: 0.11

2 Confusion Matrix

```
[14]: cm = confusion_matrix(y_test,y_pred)
      cm
```

```
[14]: array([[61,  2],
             [ 9, 28]], dtype=int64)
```

```
[15]: # Create a heatmap to visualize the confusion matrix
      plt.figure(figsize=(6, 4))
      sns.heatmap(cm, annot=True, fmt='d', cmap='Oranges', cbar=False,
                  xticklabels=['Class 0', 'Class 1'],
                  yticklabels=['Class 0', 'Class 1'])
      plt.xlabel('Predicted')
      plt.ylabel('Actual')
      plt.title('Confusion Matrix')
      plt.show()
```



[16]:
```
# Sequence TN , FP , FN ,
```

[17]:
```
#         Predicted
#           0   1
# Actual 0   TN  FP
#        1  FN  TP
```

[18]:
```
precision = precision_score(y_test,y_pred)
precision
```

[18]: 0.9333333333333333

[19]:
```
recall = recall_score(y_test,y_pred)
recall
```

[19]: 0.7567567567567568

[20]:
```
report = classification_report(y_test,y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.87      0.97      0.92        63
```

```
           1       0.93      0.76      0.84        37

    accuracy                           0.89       100
   macro avg       0.90      0.86      0.88       100
weighted avg       0.89      0.89      0.89       100
```

3 See Formulae for Precision Recall

4 Precision

Precision measures the accuracy of positive predictions, calculated as the ratio of true positives to the total predicted positives. A high precision indicates that most of the positive predictions are correct. Precision= TP / TP+FP

5 Recall

measures the proportion of actual positives that are correctly identified. It is calculated as the ratio of true positives to the total actual positives. Recall= TP / TP+FN

6 True Positive Rate (TPR)

indicates how well a model identifies positive instances.

Formula same as Recall

7 True Negative Rate (TNR)

indicates how well a model avoids false positives.

FPR = FP/(FP+TN)

[ ]:

```

```

final-ds-6

May 2, 2024

1 6) Data Analytics III

1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

[1]:
```
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
```

[2]:
```
df = pd.read_csv('Iris (1).csv')
df
```

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

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              …
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica
```

```
[150 rows x 6 columns]
```

[3]:
```
x = df.drop(columns=['Id' , 'Species'])
x
```

[3]:
```
     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
..             …            …             …           …
145            6.7           3.0            5.2           2.3
146            6.3           2.5            5.0           1.9
147            6.5           3.0            5.2           2.0
148            6.2           3.4            5.4           2.3
149            5.9           3.0            5.1           1.8

[150 rows x 4 columns]
```

[4]:
```
y = df[['Species']]
y
```

[4]:
```
            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              …
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 1 columns]
```

[5]:
```
# from sklearn.preprocessing import MinMaxScaler

# scaler = MinMaxScaler()
```

[7]:
```
# X_scaled = scaler.fit_transform(x)
# X_scaled
```

[8]:
```
from sklearn.model_selection import train_test_split
```

[9]:
```
X_train, X_test, y_train, y_test  = train_test_split(x , y , test_size=0.25 ,␣
↪random_state=42)
```

[10]:
```
gnb = GaussianNB()
```

[11]:
```
gnb.fit(X_train,y_train)
```

```
C:\Users\vedan\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2
kfra8p0\LocalCache\local-packages\Python39\site-
packages\sklearn\utils\validation.py:1300: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[11]:
```
GaussianNB()
```

[12]:
```
y_pred = gnb.predict(X_test)
y_pred
```

[12]:
```
array(['Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa'], dtype='<U15')
```

[13]:
```
from sklearn.metrics import accuracy_score , confusion_matrix , precision_score␣
↪,  recall_score , classification_report
```

[14]:
```
accuracy = accuracy_score(y_test,y_pred)
accuracy
```

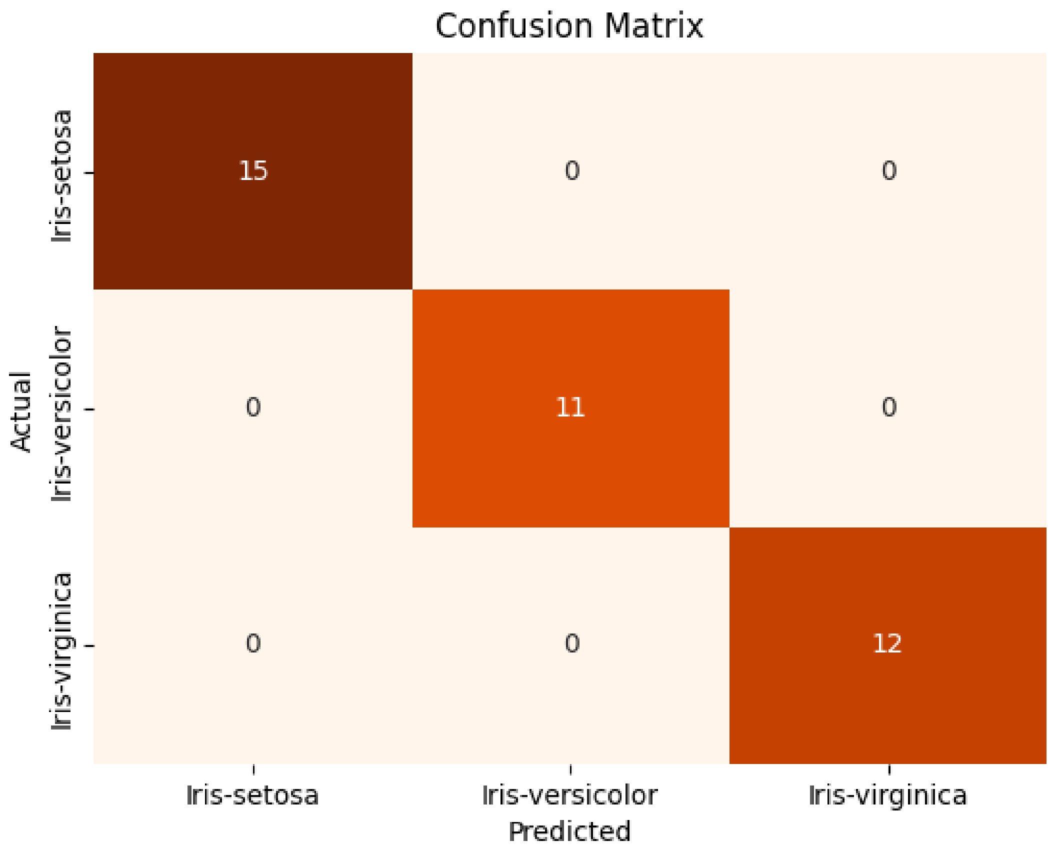
[14]:
```
1.0
```

[15]:
```
cm = confusion_matrix(y_test,y_pred)
cm
```

[15]:
```
array([[15,  0,  0],
       [ 0, 11,  0],
       [ 0,  0, 12]], dtype=int64)
```

[16]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
```

[17]:
```python
sns.heatmap(cm  , annot=True , cbar=False , cmap='Oranges' ,
            xticklabels = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'] ,
            yticklabels=['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



[18]:
```python
precision_score(y_test,y_pred , average='macro')
```

[18]: 1.0

[19]:
```python
recall_score(y_test,y_pred,average='macro')
```

[19]: 1.0

[20]:
```
error_rate = 1 -  accuracy
error_rate
```

[20]: 0.0

[ ]:
```

```

final-ds-8

May 2, 2024

1 8) Data Visualization I

1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data.
2. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram

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

[2]:
```
df = pd.read_csv('Titanic.csv')
df
```

[2]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …         …     …
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 …     …   …    …
886                              Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
```

```
889                           Behr, Mr. Karl Howell    male  26.0      0
890                             Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     …                 …      …   …    …
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

[3]:
```python
df.isnull().sum()
```

[3]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

[4]:
```python
df.shape
```

[4]:
```
(891, 12)
```

[8]:
```python
df.drop(columns='Cabin' , inplace=True)
```

[9]:
```python
df
```

[9]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
```

```
4               5         0       3
..            …         …      …
886           887         0       2
887           888         1       1
888           889         0       3
889           890         1       1
890           891         0       3

                                                    Name     Sex   Age  SibSp  \
0                                Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                                 Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                               Allen, Mr. William Henry    male  35.0      0
..                                                   …     …    …      …
886                                Montvila, Rev. Juozas    male  27.0      0
887                         Graham, Miss. Margaret Edith  female  19.0      0
888             Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                                Behr, Mr. Karl Howell    male  26.0      0
890                                  Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Embarked
0        0         A/5 21171   7.2500        S
1        0          PC 17599  71.2833        C
2        0  STON/O2. 3101282   7.9250        S
3        0            113803  53.1000        S
4        0            373450   8.0500        S
..     …               …       …      …
886      0            211536  13.0000        S
887      0            112053  30.0000        S
888      2        W./C. 6607  23.4500        S
889      0            111369  30.0000        C
890      0            370376   7.7500        Q

[891 rows x 11 columns]
```

[10]:
```python
df.dropna(inplace=True)
```

[11]:
```python
df.shape
```

[11]: (712, 11)

[12]:
```python
df
```

[12]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
```

```
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
885          886         0       3
886          887         0       2
887          888         1       1
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 …     …     …      …
885               Rice, Mrs. William (Margaret Norton)  female  39.0      0
886                              Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
889                              Behr, Mr. Karl Howell    male  26.0      0
890                                Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Embarked
0        0         A/5 21171   7.2500        S
1        0          PC 17599  71.2833        C
2        0  STON/O2. 3101282   7.9250        S
3        0            113803  53.1000        S
4        0            373450   8.0500        S
..     …               …       …       …
885      5            382652  29.1250        Q
886      0            211536  13.0000        S
887      0            112053  30.0000        S
889      0            111369  30.0000        C
890      0            370376   7.7500        Q

[712 rows x 11 columns]
```

[13]:
```
df.columns
```

[13]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Embarked'],
      dtype='object')
```

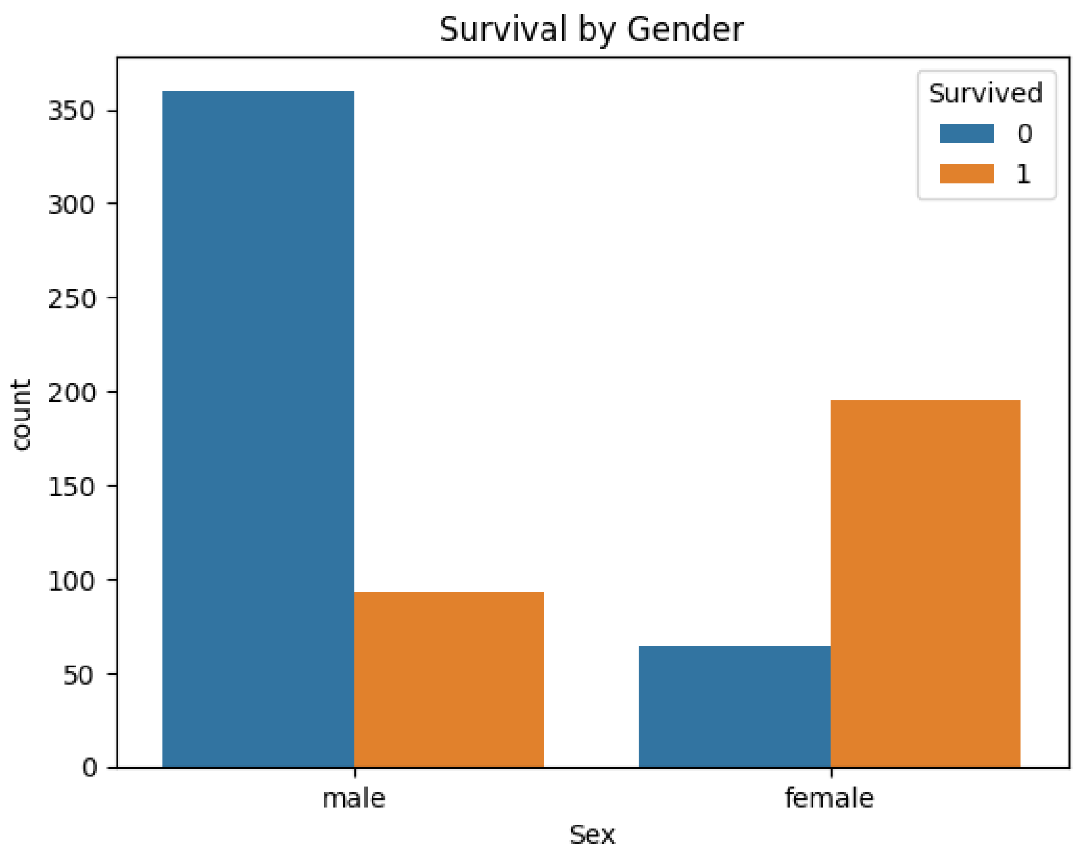
[14]:
```
df.drop(columns='PassengerId' , inplace = True)
```

[15]:
```
df
```

[15]:
```
     Survived  Pclass                                               Name  \
0           0       3                            Braund, Mr. Owen Harris
1           1       1  Cumings, Mrs. John Bradley (Florence Briggs Th…
2           1       3                             Heikkinen, Miss. Laina
3           1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)
4           0       3                           Allen, Mr. William Henry
..        …       …                                                  …
885         0       3               Rice, Mrs. William (Margaret Norton)
886         0       2                              Montvila, Rev. Juozas
887         1       1                       Graham, Miss. Margaret Edith
889         1       1                              Behr, Mr. Karl Howell
890         0       3                                Dooley, Mr. Patrick

        Sex   Age  SibSp  Parch            Ticket     Fare Embarked
0      male  22.0      1      0         A/5 21171   7.2500        S
1    female  38.0      1      0          PC 17599  71.2833        C
2    female  26.0      0      0  STON/O2. 3101282   7.9250        S
3    female  35.0      1      0            113803  53.1000        S
4      male  35.0      0      0            373450   8.0500        S
..      …     …    …    …               …      …      …
885  female  39.0      0      5            382652  29.1250        Q
886    male  27.0      0      0            211536  13.0000        S
887  female  19.0      0      0            112053  30.0000        S
889    male  26.0      0      0            111369  30.0000        C
890    male  32.0      0      0            370376   7.7500        Q

[712 rows x 10 columns]
```

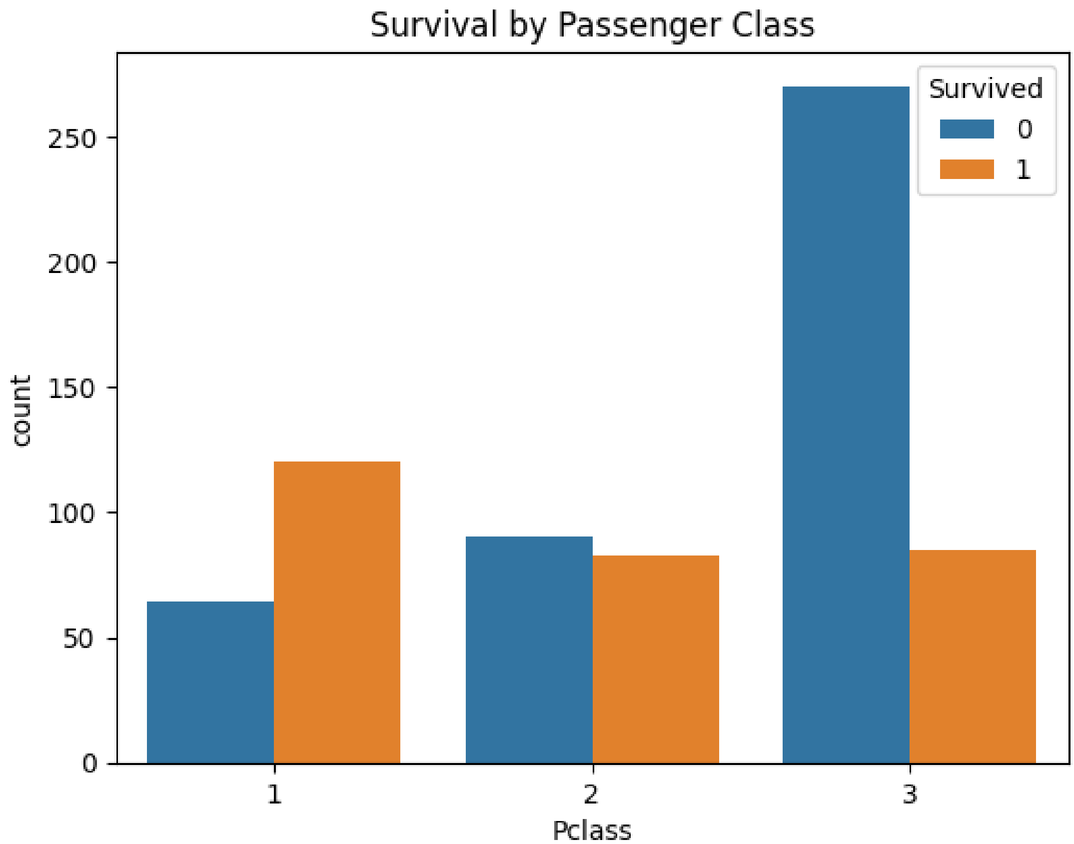
2 Survival by Gender

[17]:
```
# Bar plot showing the count of survivors by gender
sns.countplot(x="Sex", hue="Survived", data=df)
plt.title("Survival by Gender")
plt.show()
```



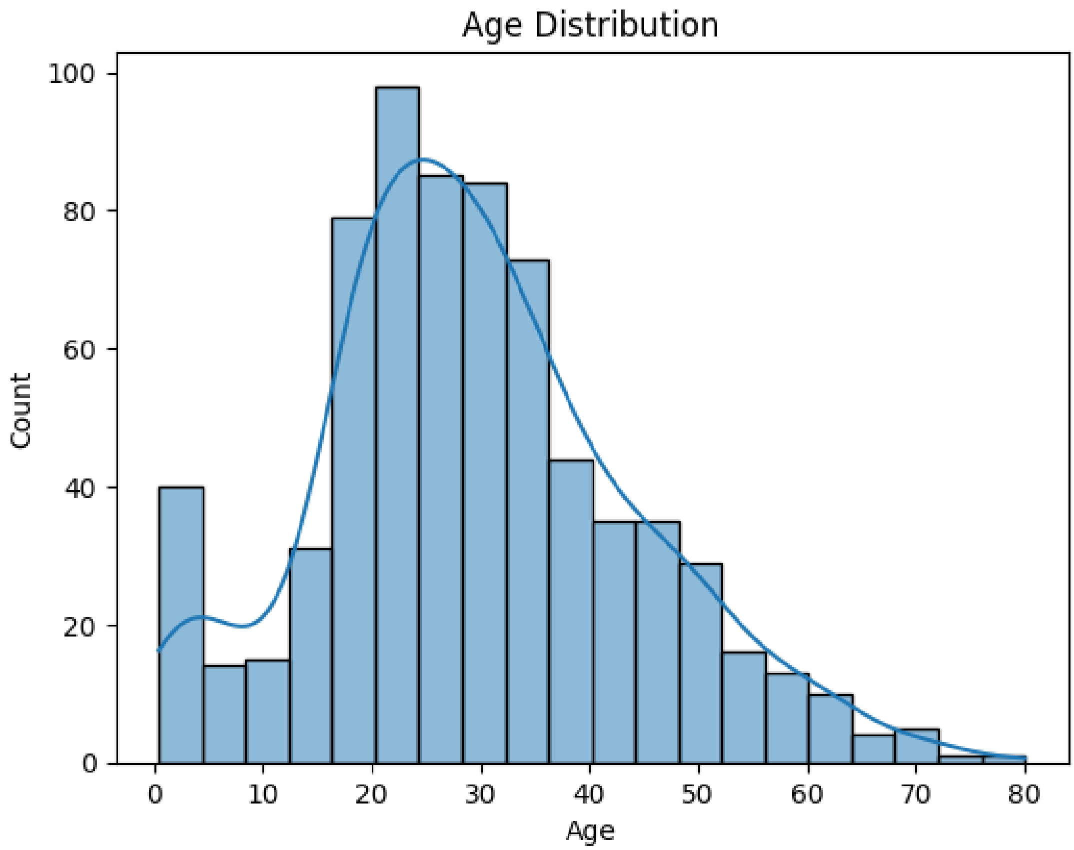
3 Survival by Passenger Class

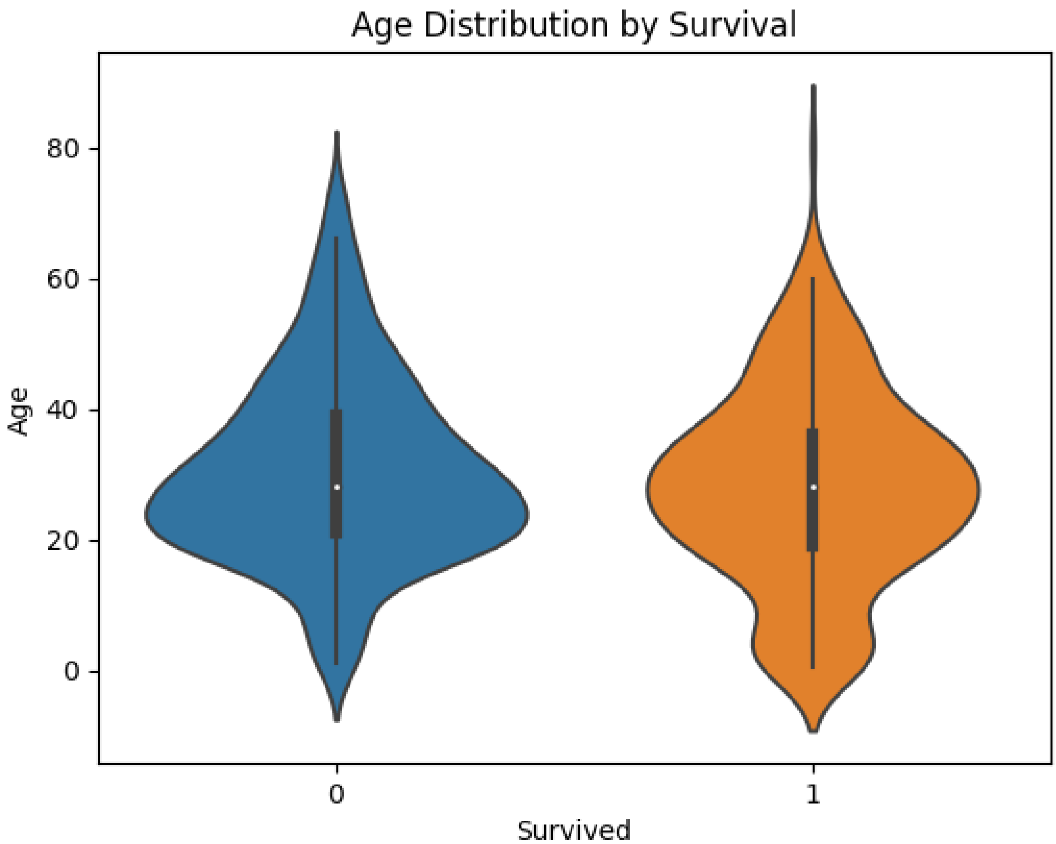
[18]:
```
# Bar plot showing the count of survivors by passenger class
sns.countplot(x="Pclass", hue="Survived", data=df)
plt.title("Survival by Passenger Class")
plt.show()
```



4 Age Distribution

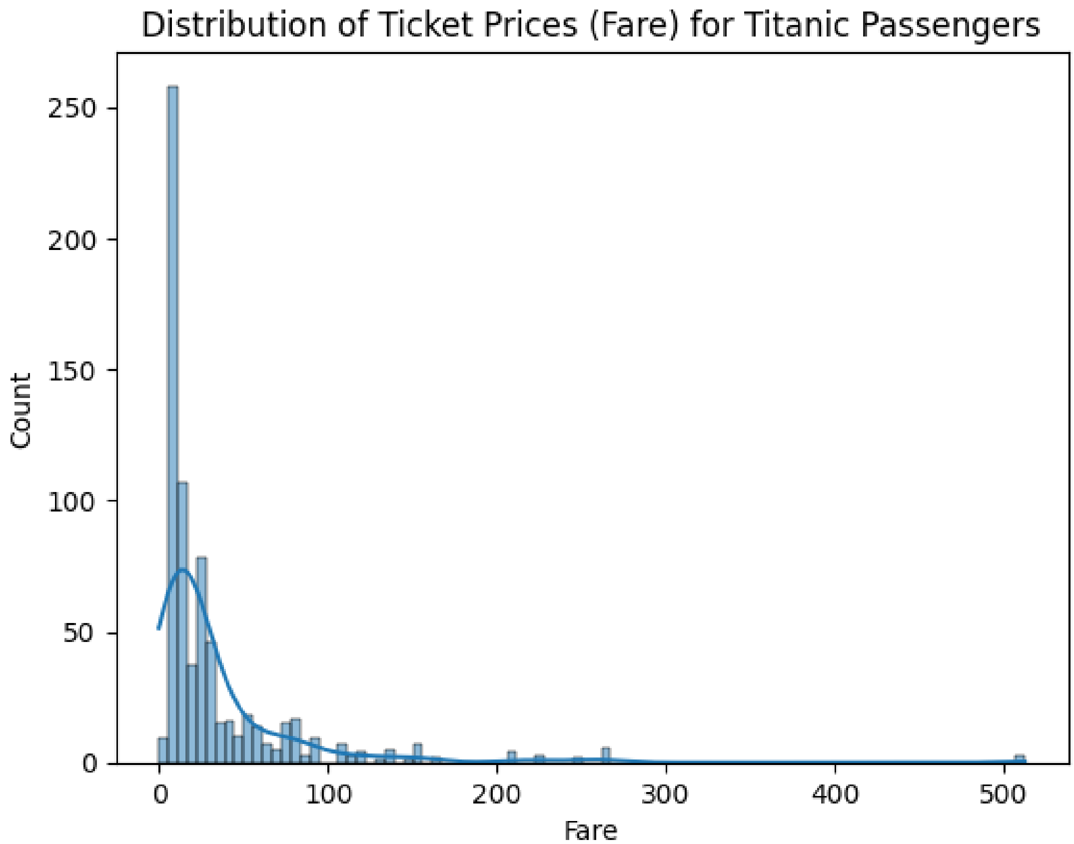
[19]:
```
# Distribution plot showing age distribution
sns.histplot(df["Age"], kde=True)
plt.title("Age Distribution")
plt.show()

# Violin plot showing age distribution by survival
sns.violinplot(x="Survived", y="Age", data=df)
plt.title("Age Distribution by Survival")
plt.show()
```


Age Distribution
Count
Age
0
20
40
60
80
100
0
10
20
30
40
50
60
70
80



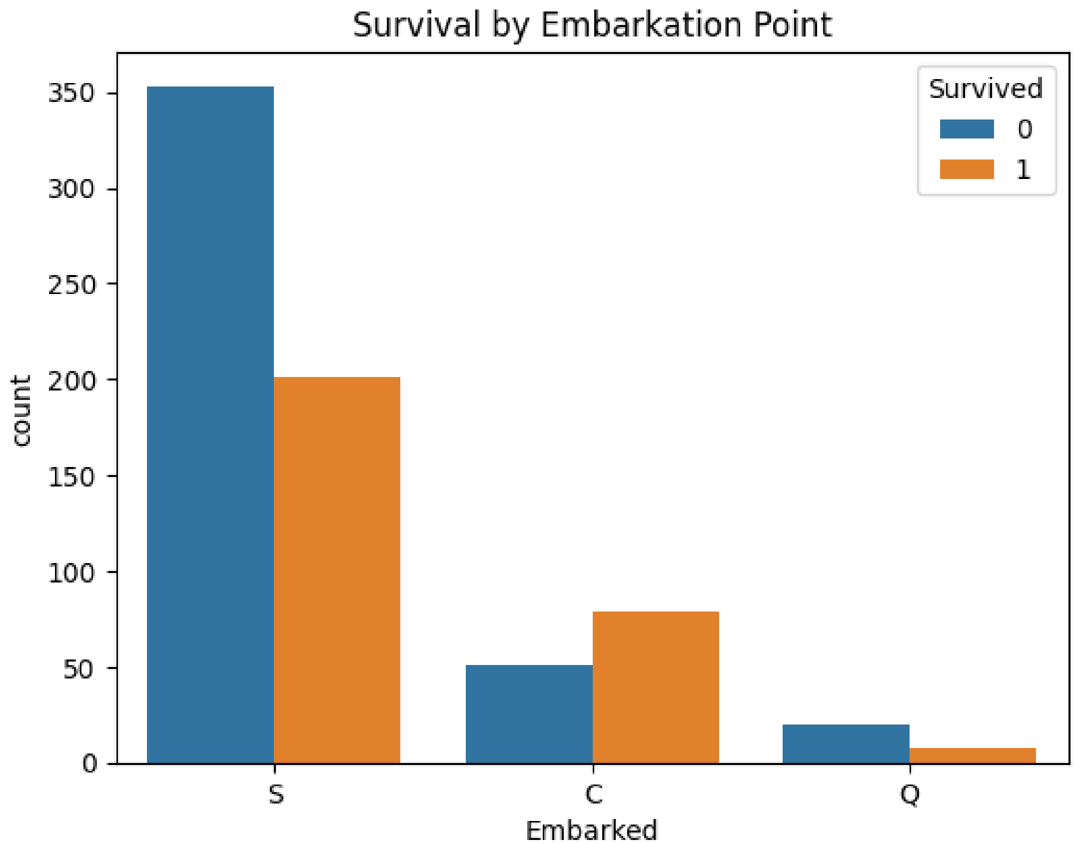
5 Fare Distribution

[24]:
```
sns.histplot(df['Fare'], kde=True)  # kde=True plots a Kernel Density Estimate␣
↪(smoothed curve)
plt.xlabel('Fare')
plt.ylabel('Count')
plt.title('Distribution of Ticket Prices (Fare) for Titanic Passengers')
plt.show()
```



6 Survival by Embark Point

[28]:
```
# Bar plot showing survival by embarkation point
sns.countplot(x="Embarked", hue="Survived", data=df)
plt.title("Survival by Embarkation Point")
plt.show()
#C = Cherbourg, Q = Queenstown, S = Southampton
```


Survival by Embarkation Point
Survived
0
1
count
350
300
250
200
150
100
50
0
S
C
Q
Embarked

```
[ ]:
```

final-ds-9

May 2, 2024

1 9 Data Visualization II

1. Use the inbuilt dataset ‘titanic’ as used in the above problem. Plot a box plot for distribution of age with respect to each gender along with the information about whether they survived or not. (Column names : ‘sex’ and ‘age’)
2. Write observations on the inference from the above statistics.

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

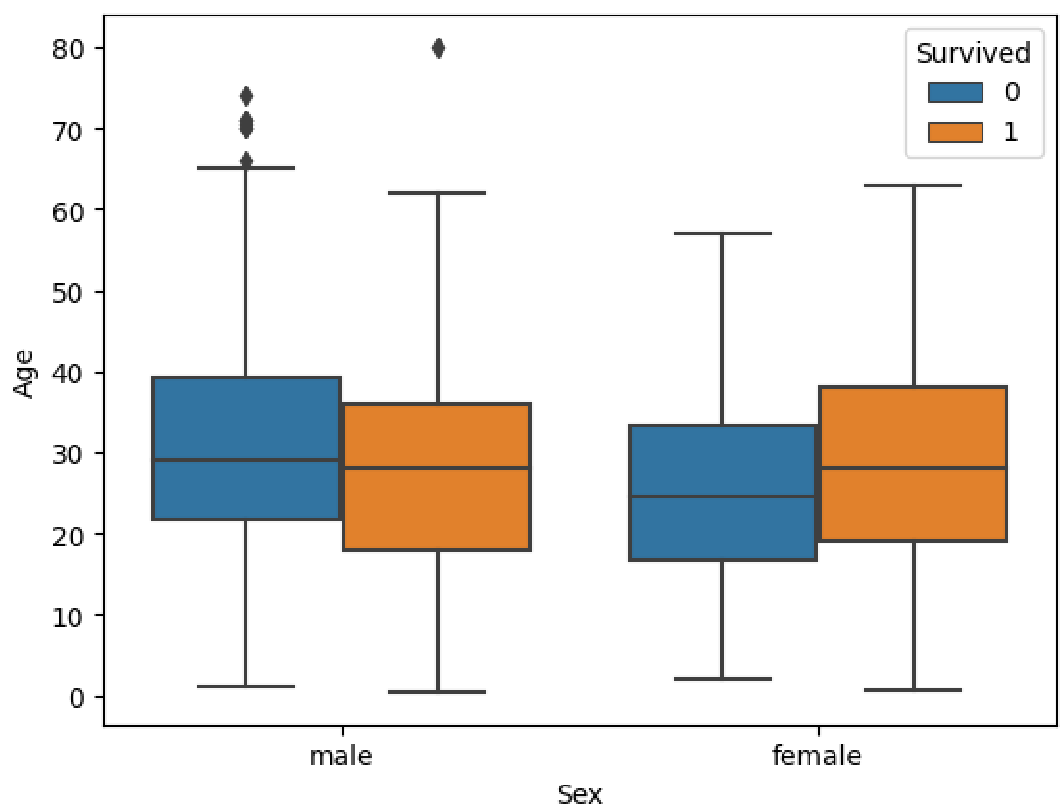
[3]:
```
df = pd.read_csv('Titanic.csv')
df
```

[3]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …         …       …
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 …     …   …    …
886                              Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                              Behr, Mr. Karl Howell    male  26.0      0
```

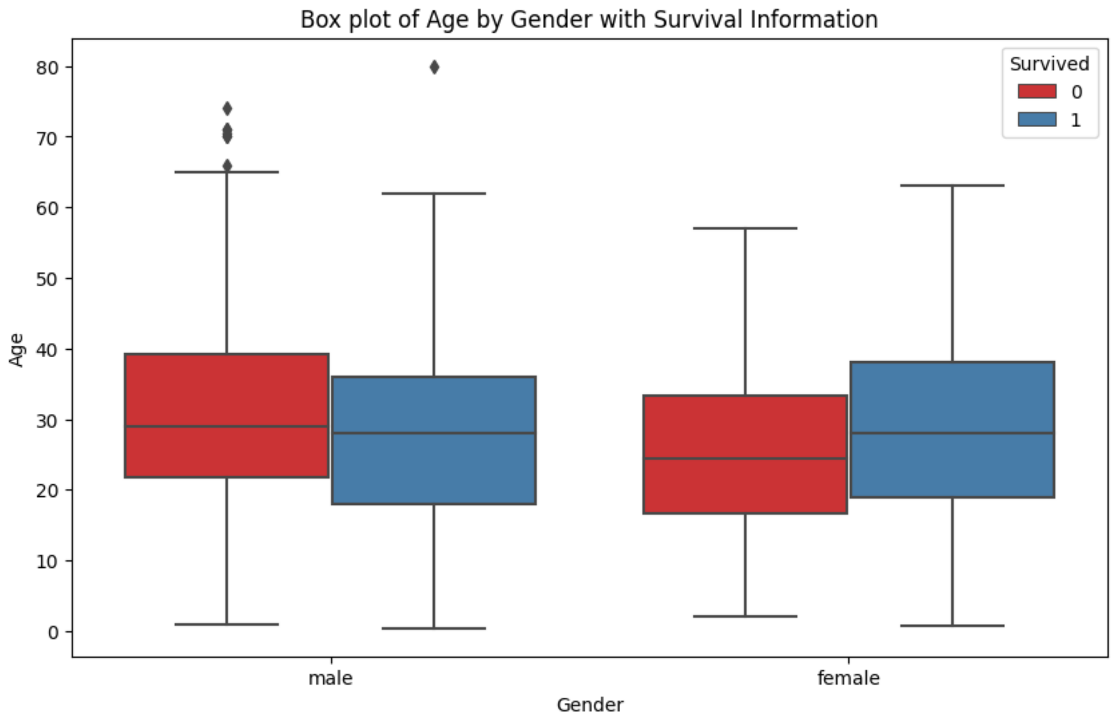
```
890                              Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     …                 …      …   …      …
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

[6]:
```
sns.boxplot(x='Sex' , y='Age' , data=df , hue = 'Survived')
```

[6]: <Axes: xlabel='Sex', ylabel='Age'>



[10]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Sex', y='Age', hue='Survived', palette='Set1')

plt.title('Box plot of Age by Gender with Survival Information')
plt.xlabel('Gender')
plt.ylabel('Age')
plt.legend(title='Survived', loc='upper right')
plt.show()
```



2 Matplotlib

Matplotlib is a comprehensive plotting library for creating static, interactive, and animated visualizations in Python.

3 Seaborn

Seaborn is a data visualization library built on top of Matplotlib, designed to simplify complex statistical plots and make them more visually appealing

4 Boxplot

A boxplot (also known as a box-and-whisker plot) is a statistical visualization that displays the distribution of a dataset through its quartiles. It shows the median, upper and lower quartiles, and potential outliers, using a box to represent the interquartile range and “whiskers” to indicate the range of the data

```
[ ]:
```

final-ds-10

May 2, 2024

1 Data Visualization III

Download the Iris flower dataset or any other dataset into a DataFrame. (e.g., https://archive.ics.uci.edu/ml/datasets/Iris). Scan the dataset and give the inference as: 1. List down the features and their types (e.g., numeric, nominal) available in the dataset. 2. Create a histogram for each feature in the dataset to illustrate the feature distributions. 3. Create a boxplot for each feature in the dataset. 4. Compare distributions and identify outliers.

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

[2]:
```
df = pd.read_csv('Iris (1).csv')
df
```

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

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              …
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
```

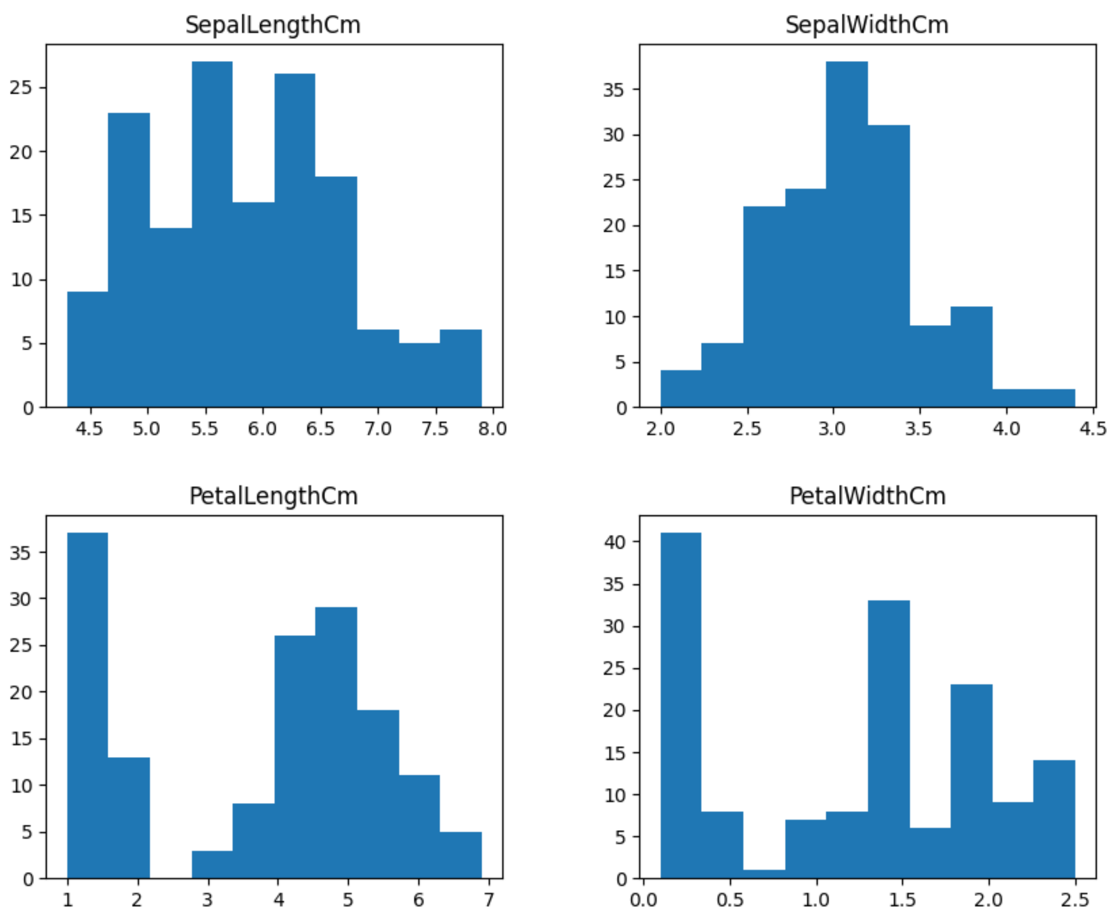
```
148   Iris-virginica
149   Iris-virginica

[150 rows x 6 columns]
```

[3]:
```
df.drop(columns='Id' , inplace=True)
```

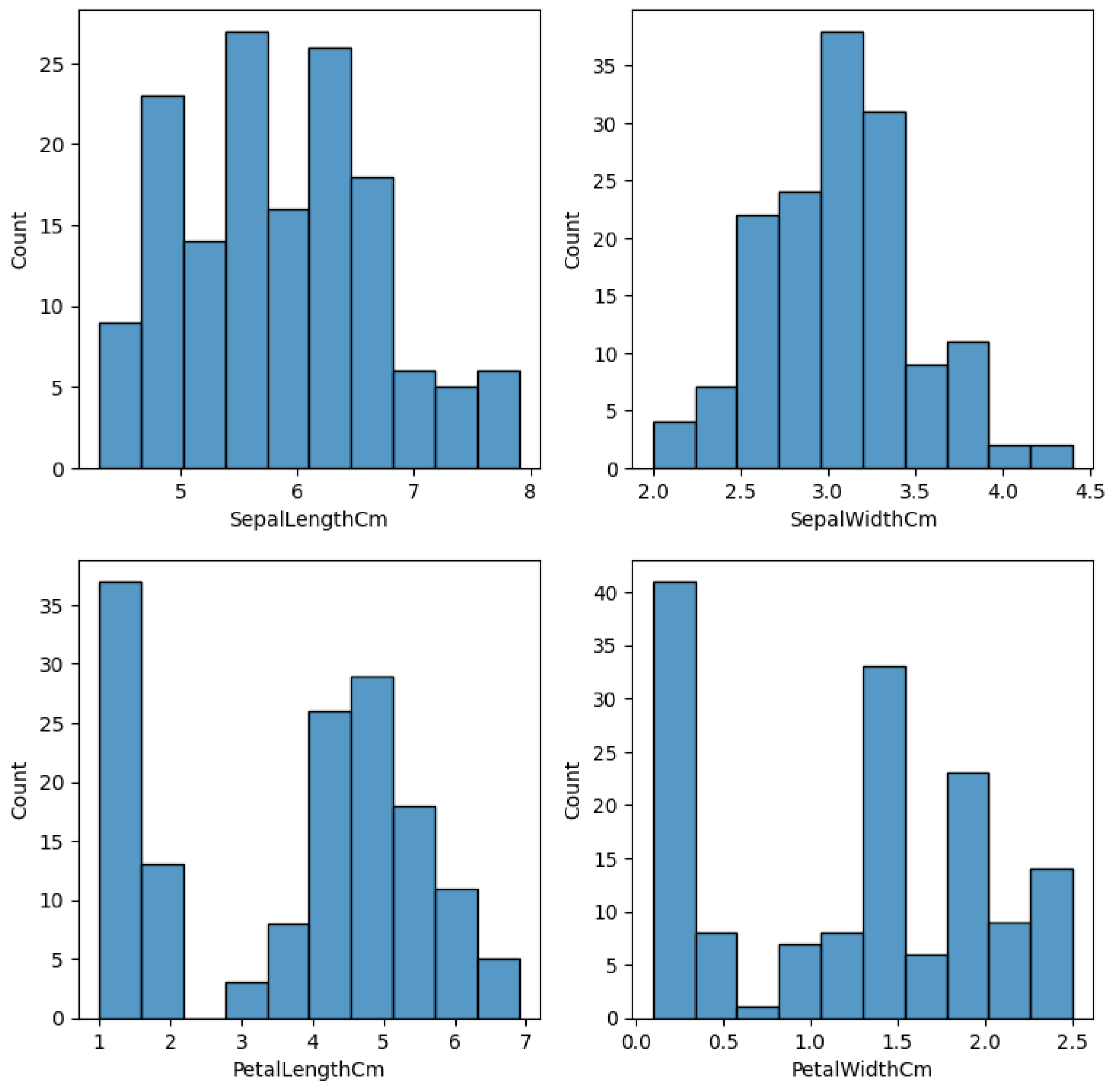
[4]:
```
df.dtypes
```

[4]:
```
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

[5]:
```
# Create a histogram for each numeric feature
df.hist(bins=10, figsize=(10, 8), grid=False)
plt.show()
```



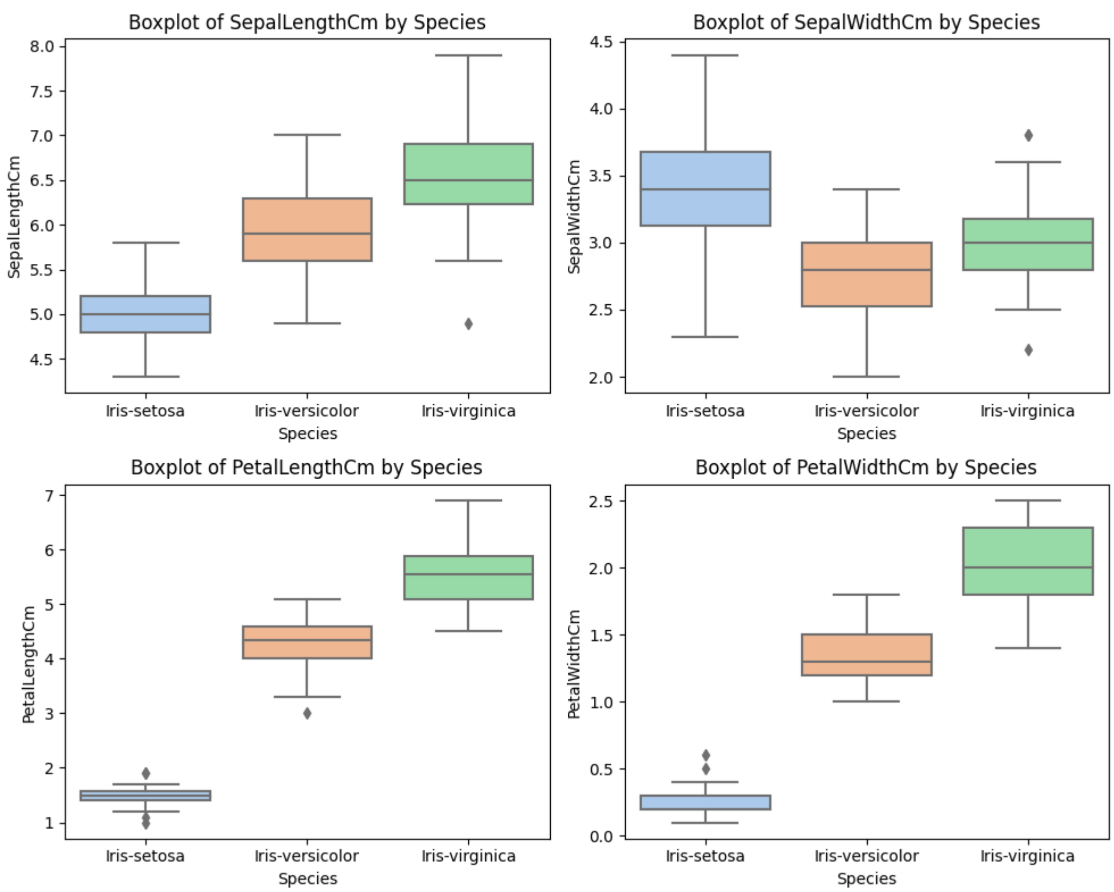
[6]:
```
fig , axes = plt.subplots(nrows=2,ncols=2,figsize = (9,9))
axes = axes.flatten()
ax = sns.histplot(x='SepalLengthCm', data=df, bins=10,ax=axes[0])

ax = sns.histplot(x='SepalWidthCm', data=df, bins=10,ax=axes[1])
ax = sns.histplot(x='PetalLengthCm', data=df, bins=10,ax=axes[2])
ax = sns.histplot(x='PetalWidthCm', data=df, bins=10,ax=axes[3])
```



[7]:
```
col = df.columns
col[:-1]
```

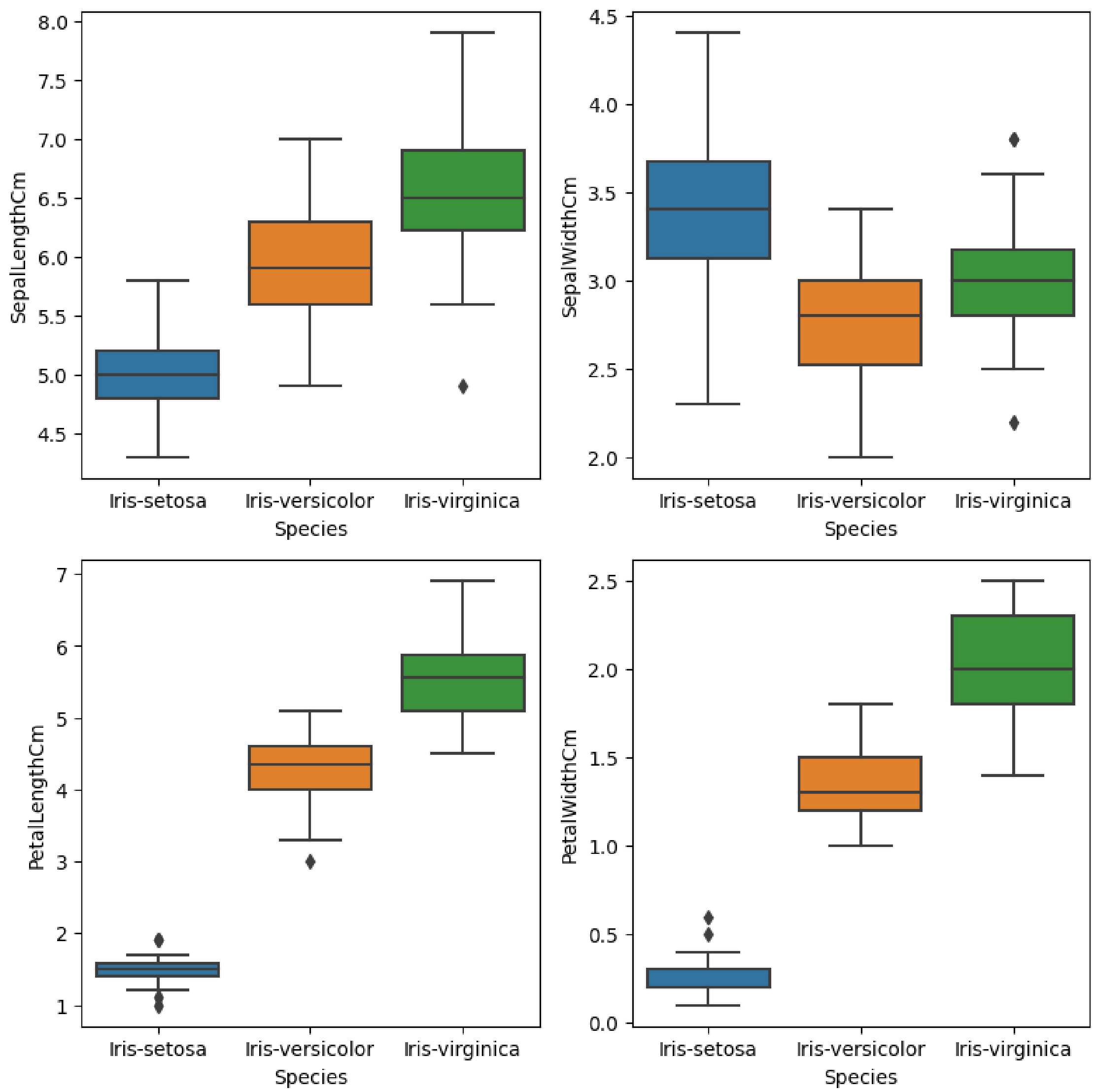
[7]: Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm'],
dtype='object')

[8]:
```
plt.figure(figsize=(10, 8))
for i, feature in enumerate(col[:-1]):
    # Create a 2x2 grid of subplots
    plt.subplot(2, 2, i + 1)  # 2 rows, 2 columns
    sns.boxplot(x='Species', y=feature, data=df, palette="pastel")

    # Title and rotation
    plt.title(f'Boxplot of {feature} by Species')
    # plt.xticks(rotation=45)

# Adjust layout to avoid overlap
plt.tight_layout()  # This adjusts spacing between plots
plt.show()
```



[9]:
```
fig , axes = plt.subplots(nrows=2,ncols=2,figsize=(8,8))
axes = axes.flatten()
```

```
sns.boxplot(x='Species' , y = 'SepalLengthCm' ,data=df, ax=axes[0])
sns.boxplot(x='Species' , y = 'SepalWidthCm' ,data=df, ax=axes[1])
sns.boxplot(x='Species' , y = 'PetalLengthCm' ,data=df, ax=axes[2])
sns.boxplot(x='Species' , y = 'PetalWidthCm' ,data=df, ax=axes[3])
plt.tight_layout()
```




```
[ ]:
```

final-ds-7-1

May 2, 2024

[ ]:
```
import nltk
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')
from nltk import sent_tokenize
from nltk import word_tokenize
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[ ]:
```
text = 'Researchers observed that the children were playing with their toys in␣
↪the park while their parents were discussing various topics nearby. As the␣
↪afternoon grew warm, families began preparing for a picnic, spreading␣
↪blankets and setting up food. The wind picked up, rustling the leaves, and␣
↪everyone enjoyed the calm and peaceful atmosphere.'
```

[ ]:
```
tokens_sents = nltk.sent_tokenize(text)
print(tokens_sents)
```

```
['Researchers observed that the children were playing with their toys in the
park while their parents were discussing various topics nearby.', 'As the
afternoon grew warm, families began preparing for a picnic, spreading blankets
and setting up food.', 'The wind picked up, rustling the leaves, and everyone
enjoyed the calm and peaceful atmosphere.']
```

[ ]:
```
tokens_words = nltk.word_tokenize(text)
print(tokens_words)
```

```
['Researchers', 'observed', 'that', 'the', 'children', 'were', 'playing',
'with', 'their', 'toys', 'in', 'the', 'park', 'while', 'their', 'parents',
'were', 'discussing', 'various', 'topics', 'nearby', '.', 'As', 'the',
'afternoon', 'grew', 'warm', ',', 'families', 'began', 'preparing', 'for', 'a',
'picnic', ',', 'spreading', 'blankets', 'and', 'setting', 'up', 'food', '.',
'The', 'wind', 'picked', 'up', ',', 'rustling', 'the', 'leaves', ',', 'and',
'everyone', 'enjoyed', 'the', 'calm', 'and', 'peaceful', 'atmosphere', '.']
```

[ ]:
```
sw_nltk = stopwords.words('english')
print((sw_nltk))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're",
"you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he',
'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's",
'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what',
'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is',
'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about',
'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above',
'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under',
'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why',
'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some',
'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn',
"couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn',
"hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't",
'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',
"shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',
"wouldn't"]
```

[ ]:
```
# from nltk.corpus import stopwords
# stop_words = set(stopwords.words('english'))
# stop_words
```

[ ]:
```
words = [i for i in text.split() if i.lower() not in sw_nltk]
# new_text = " ".join(words)
# print(new_text)
words
```

[ ]:
```
['Researchers',
 'observed',
 'children',
 'playing',
 'toys',
 'park',
```

```
 'parents',
 'discussing',
 'various',
 'topics',
 'nearby.',
 'afternoon',
 'grew',
 'warm,',
 'families',
 'began',
 'preparing',
 'picnic,',
 'spreading',
 'blankets',
 'setting',
 'food.',
 'wind',
 'picked',
 'up,',
 'rustling',
 'leaves,',
 'everyone',
 'enjoyed',
 'calm',
 'peaceful',
 'atmosphere.']
```

[ ]:
```
from nltk.stem import PorterStemmer
```

[ ]:
```
stem=[]
for i in words:
  ps = PorterStemmer()
  stem_word= ps.stem(i)
  stem.append(stem_word)
print(stem)
```

```
['research', 'observ', 'children', 'play', 'toy', 'park', 'parent', 'discuss',
'variou', 'topic', 'nearby.', 'afternoon', 'grew', 'warm,', 'famili', 'began',
'prepar', 'picnic,', 'spread', 'blanket', 'set', 'food.', 'wind', 'pick', 'up,',
'rustl', 'leaves,', 'everyon', 'enjoy', 'calm', 'peac', 'atmosphere.']
```

[ ]:
```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

[ ]:
```
leme=[]
for i in words:
```

```
  lemetized_word=lemmatizer.lemmatize(i)
  leme.append(lemetized_word)
print(leme)

new_text = " ".join(leme)
print(new_text)
```

```
['Researchers', 'observed', 'child', 'playing', 'toy', 'park', 'parent',
'discussing', 'various', 'topic', 'nearby.', 'afternoon', 'grew', 'warm,',
'family', 'began', 'preparing', 'picnic,', 'spreading', 'blanket', 'setting',
'food.', 'wind', 'picked', 'up,', 'rustling', 'leaves,', 'everyone', 'enjoyed',
'calm', 'peaceful', 'atmosphere.']
Researchers observed child playing toy park parent discussing various topic
nearby. afternoon grew warm, family began preparing picnic, spreading blanket
setting food. wind picked up, rustling leaves, everyone enjoyed calm peaceful
atmosphere.
```

[ ]:
```
print("Parts of Speech: ",nltk.pos_tag(leme))
```

```
Parts of Speech:  [('Researchers', 'NNS'), ('observed', 'VBD'), ('child', 'JJ'),
('playing', 'VBG'), ('toy', 'NN'), ('park', 'NN'), ('parent', 'NN'),
('discussing', 'VBG'), ('various', 'JJ'), ('topic', 'NN'), ('nearby.', 'NN'),
('afternoon', 'NN'), ('grew', 'VBD'), ('warm,', 'JJ'), ('family', 'NN'),
('began', 'VBD'), ('preparing', 'VBG'), ('picnic,', 'NN'), ('spreading', 'VBG'),
('blanket', 'NN'), ('setting', 'VBG'), ('food.', 'JJ'), ('wind', 'NN'),
('picked', 'VBD'), ('up,', 'JJ'), ('rustling', 'VBG'), ('leaves,', 'NN'),
('everyone', 'NN'), ('enjoyed', 'VBD'), ('calm', 'JJ'), ('peaceful', 'JJ'),
('atmosphere.', 'NN')]
```

[ ]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

[ ]:
```
vectorizer = TfidfVectorizer()
 tfidf_matrix = vectorizer.fit_transform([new_text])
 # Get the feature names (terms)
 feature_names = vectorizer.get_feature_names_out()
 # Print the document-term matrix
 print("Document 1")
 for j, feature in enumerate(feature_names):
  if tfidf_matrix[0,j] > 0:
    print("  ", feature, ":", tfidf_matrix[0,j])
```

```
Document 1
   afternoon : 0.17677669529663687
   atmosphere : 0.17677669529663687
   began : 0.17677669529663687
   blanket : 0.17677669529663687
   calm : 0.17677669529663687
   child : 0.17677669529663687
```

```
discussing : 0.17677669529663687
enjoyed : 0.17677669529663687
everyone : 0.17677669529663687
family : 0.17677669529663687
food : 0.17677669529663687
grew : 0.17677669529663687
leaves : 0.17677669529663687
nearby : 0.17677669529663687
observed : 0.17677669529663687
parent : 0.17677669529663687
park : 0.17677669529663687
peaceful : 0.17677669529663687
picked : 0.17677669529663687
picnic : 0.17677669529663687
playing : 0.17677669529663687
preparing : 0.17677669529663687
researchers : 0.17677669529663687
rustling : 0.17677669529663687
setting : 0.17677669529663687
spreading : 0.17677669529663687
topic : 0.17677669529663687
toy : 0.17677669529663687
up : 0.17677669529663687
various : 0.17677669529663687
warm : 0.17677669529663687
wind : 0.17677669529663687
```

[ ]:

```

```

untitled

May 2, 2024

1 NLTK

The Natural Language Toolkit (nltk) is a comprehensive library in Python designed for natural language processing (NLP) tasks.

It provides tools for text processing, such as tokenization, parsing, stemming, and machine learning algorithms for text analysis

2 punkt

used for tokeninzing into words and sentences

3 Stopwords

Stopwords are common words in a language (such as "and," "the," "is," etc.) that are typically ignored or removed in text processing and analysis because they carry less significant meaning. These words are usually filtered out to focus on the more meaningful terms that contribute to understanding the context or content of the text.

4 POS Tagging

assigning labels to words in a sentence to identify their grammatical roles, such as nouns, verbs, adjectives, or adverbs.

JJ -> ADJECTIVE

NN - > NOUN

VB - > VERB

5 Stemming

Stemming is the process of reducing words to their root form by removing suffixes or prefixes, like turning "running" into "run." It's used to standardize text for easier analysis and search

After stemming its not neccessary that the word will have its meaning in english dictionary

Its used , when we dont want to show output to user and its complexity is less

6 Lemmatization

Lemmatization is the process of converting words to their base or dictionary form, known as the lemma, by considering the context and grammatical rules

After Lemmatization word have meaning in english dictionary

Its used when we want to provide answer / output to user , and its complexity is high

7 TFID Vectorizer

Full Form : Term Frequency-Inverse Document Frequency

It combines two metrics:

Term Frequency (TF): Measures how often a word appears in a document. A higher frequency indicates greater relevance within that document.

Inverse Document Frequency (IDF): Measures how common or rare a word is across a set of documents. A higher IDF indicates that the word is rare, suggesting it carries more distinctive information.

8 TF = frequency of term 't' in document 'd' / total terms in 'd'

9 IDF = log10 (total number of documents / total documnets with term 't')

[ ]:
```

```