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 an 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 data frame.

4. Data Preprocessing: check for missing values in the data using pandas insult(), 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. import pandas as pd
2. import numpy as np
3. df = pd.read_csv("Iris.csv")  # we stored iris.csv file in df i.e dataframes or  variable name
4. df
5. df.isnull()
6. df.isnull().any()
7. df.dtypes  # dtypes stands for data types
8. df["Species"].unique()
9. df["Species"]=df["Species"].replace({'Iris-setosa':1,'Iris-versicolor':2,'Iris-virginica':3})
10. df.dtypes
```

Theory :

1. Introduction to Dataset
   A dataset is a collection of records, similar to a relational database table. Records are similar to table rows, but the columns can contain not only strings or numbers, but also nested data structures such as lists, maps, and other records.

   a. Pandas : Pandas is an open-source Python package that provides high-performance, easy-to-use data structures and data analysis tools for the labeled data in Python programming language. What can you do with Pandas?

   1. Indexing, manipulating, renaming, sorting, merging data frame 2. Update, Add, Delete columns from a data frame 3. Impute missing files, handle missing data or NANs 4. Plot data with histogram or box plot

b.numpy : NumPy is a general-purpose arrayprocessing package. It provides high-performance multidimensional array objects and tools to work with the arrays. NumPy is an efficient container of generic multidimensional data.

1. Basic array operations: add, multiply, slice, flatten, reshape, index arrays 2. Advanced array operations: stack arrays, split into sections, broadcast arrays 3. Work with DateTime or Linear Algebra 4. Basic Slicing and Advanced Indexing in NumPy Python

C . isnull() : is the function that is used to check missing values or null values in pandas python

d. df.dtypes : To check the data type

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

```
1. import numpy as np
2. import pandas as pd
3. df = pd.read_csv("Academic_performace.csv")
4. df
5. df.head()
6. df.tail()
7. df.describe()
8. df.info()
9. df.shape
10. df.isnull().any().any()
11. df.isnull().sum()
12. avg_val = df["Discussion"].astype("float").mean()
    avg_val
13. df["Discussion"].replace(np.NaN, avg_val, inplace=True)
14. df.isnull().sum()
```

Step-II Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.

```
1. import seaborn as sns
2. import matplotlib.pyplot as plt
3. from scipy import stats
4. sns.regplot(x='Sno', y='AnnouncementsView', data=df)
   plt.show()
5. sns.boxplot(x=df['AnnouncementsView'])
   plt.show()
6. z = np.abs(stats.zscore(df['AnnouncementsView']))
   print(z)
```

7. threshold = 3
   print(np.where(z > 3))
8. z[419]

Step-III Apply data transformations on at least one of the variables

1. df1 = pd.DataFrame({ 'Income': [15000, 1800, 120000, 10000],
   a. 'Age': [25, 18, 42, 51],
   b. 'Department': ['HR','Legal','Marketing','Management']})
2. df1
3. df1_scaled = df1.copy()
   col_names = ['Income', 'Age']
   features = df1_scaled[col_names]
4. features
5. from sklearn.preprocessing import MinMaxScaler
   scaler = MinMaxScaler()
   df1_scaled[col_names] = scaler.fit_transform(features.values)
6. print(df1_scaled[col_names])

theory :

1. What are Outliers?

We all have heard of the idiom ‘odd one out' which means something unusual in comparison to the others in a group.

Similarly, an Outlier is an observation in agivendatasetthatliesfarfromtherest of the observations. That means an outlier is vastly larger or smaller than the remaining values in the set

Z-Score

is also called a standard score. This value/score helps to understand how far is the data point from the mean. And after setting up a threshold value one can utilize z score values of data points to define the outliers. Zscore = (data_point -mean) / std. deviation

Generalization : It converts low-level data attributes to high-level data attributes using concept hierarchy.

Normalization: Data normalization involves converting all data variables into a given range. Some of the techniques that are used for accomplishing normalization are:

○ Min–max normalization:This transforms the originaldata linearly.

○ Z-score normalization: In z-score normalization (or zero-mean normalization) the values of an attribute (A), are normalized based on the mean of A and its standard deviation.

○ Normalization by decimal scaling: It normalizes the values of an attribute by changing the position of their decimal points

3) 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.

```
1. import numpy as np
   import pandas as pd
   import statistics as st
2. df = pd.read_csv("Mall_Customers.csv")
3. df
4. df.mean()  # mean of all columns                    #  1. Summary statistics
5. df.loc[:,'Age'].mean()    # mean of specific column
6. df.mean(axis=1)[0:4]   # mean row wise
7. df.median()     # median of all columns                    # median
8. df.loc[:,'Age'].median()  # median of specific column
9. df.median(axis=1)[0:4]  #median row wise
10. df.mode()    # mode of all columns                     # mode
11. df.loc[:,'Age'].mode()   # mode of a specific column.
12. df.min()   # minimum of all columns                    # minimum
13. df.loc[:,'Age'].min(skipna = False)    # minimum of Specific column
14. df.max()   # Maximum of all columns                     # Maximum
15. df.loc[:,'Age'].max(skipna = False)   # Maximum of Specific column
16. df.std()   # Standard Deviation of all columns          # Standard Deviation
17. df.loc[:,'Age'].std()   # Standard Deviation of specific column
18. df.std(axis=1)[0:4]  # Standard Deviation row wise
```

2. Types of Variables:

A variable is a characteristic that can be measured and that can assume different values. Height, age, income, province or country of birth, grades obtained at school and type of housing are all examples of variables. Variables may be classified into two main categories:
1. Categorical and 2. Numeric.

Each category is then classified in two subcategories: nominal or ordinal for categorical variables, discrete or continuous for numeric variables

3.Summary statistics of income grouped by the age groups

```
1. df.groupby(['Genre'])['Age'].mean()
2. df_u=df.rename(columns= {'Annual Income (k$)':'Income'}, inplace= False)
3. df_u
4. df_u.groupby(['Genre']).Income.mean()
```

To create a list that contains a numeric value for each response to the categorical variable

```
1. from sklearn import preprocessing
   enc = preprocessing.OneHotEncoder()
   enc_df = pd.DataFrame(enc.fit_transform(df[['Genre']]).toarray())
   enc_df
2. df_encode =df_u.join(enc_df)
   df_encode
```

4.Display basic statistical details on the iris dataset.

```
1. import pandas as pd
   import numpy as np
   import matplotlib.pyplot as plt
2. df_iris = pd.read_csv("Iris.csv")
   df_iris.head()
3. print('Iris-setosa')
   setosa = df_iris['Species'] == 'Iris-setosa'
   print(df_iris[setosa].describe())
   print('\nIris-versicolor')
   versicolor = df_iris['Species'] == 'Iris-versicolor'
   print(df_iris[versicolor].describe())
   print('\nIris-virginica')
   virginica = df_iris['Species'] == 'Iris-virginica'
   print(df_iris[virginica].describe())
4. df_iris.dtypes.value_counts()
```

theory ;

What is Statistics?

Statistics is the science of collecting data and analysing them to infer proportions (sample) that are representative of the population. In other words, statistics is interpreting data in order to make predictions for the population.

Branches of Statistics: There are two branches of Statistics.

DESCRIPTIVE STATISTICS : Descriptive Statistics is a statistics or a measure that describes the data.

INFERENTIAL STATISTICS : Using a random sample of data taken from a population to describe and make inferences about the population is called Inferential Statistics.

Mean : Mean is defined as the ratio of the sum of all the observations in the data to the total number of observations

$$Mean = \frac{17+16+21+18+15+17+21+19+11+23}{10} = \frac{178}{10} = 17.8$$

Median : Median is the point which divides the entire data into two equal halves.

$$Median = \frac{5^{th}\ Obs + 6^{th}\ Obs}{2} = \frac{17+18}{2} = 17.5$$

mode : Mode is the number which has the maximum frequency in the entire data set, or in other words,mode is the number that appears the maximum number of times. A data can have one or more than one mode.

Consider the following data points. 17, 16, 21, 18, 15, 17, 21, 19, 11, 23

Mode is given by the number that occurs the maximum number of times. Here, 17 and 21 both occur twice. Hence, this is a Bimodal data and the modes are 17 and 21

Absolute Deviation from Mean — The Absolute Deviation from Mean, also called Mean Absolute Deviation (MAD), describes the variation in the data set, in the sense that it tells the average absolute distance of each data point in the set. It is calculated as

$$Mean\ \ Absolute\ \ Deviation = \frac{1}{N}\sum_{i=1}^{N}\left|X_i - \overline{X}\right|$$

Variance — Variance measures how far are data points spread out from the mean

$$Variance = \frac{1}{N}\sum_{i=1}^{N}(X_i - \bar{X})^2$$

Standard Deviation — The square root of Variance is called the Standard Deviation. It is calculated as

$$Std\ Deviation = \sqrt{Variance} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \bar{X})^2}$$

Range — Range is the difference between the Maximum value and the Minimum value in the data set. It is given as

$$Range = Maximum - Minimum$$

4) **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

```
1. import pandas as pd
   import numpy as np
   import matplotlib.pyplot as plt
   import seaborn as sns
   %matplotlib inline
2. # Importing DataSet and take a look at Data
   Boston = pd.read_csv("C:\\Users\\Rahul 9422253987\\Desktop\\csv\\Boston.csv")
   Boston.head()
3. Boston.info()
   Boston.describe()
4. Boston.plot.scatter('RM', 'MEDV', figsize=(6, 6))
5. plt.subplots(figsize=(12,8))
   sns.heatmap(Boston.corr(), cmap = 'RdGy', annot = True, fmt = '.1f')
6. X = Boston[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE',
   'DIS','RAD','TAX','PTRAT']]
   Y = Boston['MEDV']
7. from sklearn.model_selection import train_test_split
   from sklearn.linear_model import LinearRegression
8. # Split DataSet
   X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,random_state=0)
9. print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
   print(f'Test Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')
10. # Model Building
    lm = LinearRegression()
    lm.fit(X_train,Y_train)
    predictions = lm.predict(X_test)
11. # Model Visualization
    plt.figure(figsize=(6, 6))
    plt.scatter(Y_test, predictions)
    plt.xlabel('Y Test')
    plt.ylabel('Predicted Y')
    plt.title('Test vs Prediction')
12. plt.figure(figsize=(6, 6))
    sns.regplot(x = X_test['RM'], y = predictions, scatter_kws={'s':5})
    plt.scatter(X_test['RM'], Y_test, marker = '+')
    plt.xlabel('Average number of rooms per dwelling')
```

```
plt.ylabel('Median value of owner-occupied homes')
plt.title('Regression Line Tracing')
```

13.
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_test, predictions))
print('Mean Square Error:', metrics.mean_squared_error(Y_test, predictions))
print('Root Mean Square Error:', np.sqrt(metrics.mean_squared_error(Y_test,
predictions))
```

14.
```
# Model Coefficients
coefficients = pd.DataFrame(lm.coef_.round(2), X.columns)
coefficients.columns = ['coefficients']
coefficients
```

theory :

Linear Regression:

It is a machine learning algorithm based on supervised learning. It targets prediction values on the basis of independent variables.

● It is preferred to find out the relationship between forecasting and variables.

● A linear relationship between a dependent variable (X) is continuous; while independent variable(Y) relationship may be continuous or discrete. A linear relationship should be available in between predictor and target variable so known as Linear Regression.

● Linear regression is popular because the cost function is Mean Squared Error (MSE) which is equal to the average squared difference between an observation’s actual and predicted values.

● It is shown as an equation of line like : Y = m*X + b + e Where : b is intercepted, m is slope of the line and e is error term. This equation can be used to predict the value of target variable Y based on given predictor variable(s) X,

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 numpy as np
   import matplotlib.pyplot as plt
   import pandas as pd
   import seaborn as sns
   df = pd.read_csv('C:\\Users\\Rahul 9422253987\\Desktop\\csv\\Social_Network_Ads.csv')
   df.head()
2. df.info()
3. df.describe()
4. X = df[['Age', 'EstimatedSalary']]
   Y = df['Purchased']
5. from sklearn.model_selection import train_test_split
   from sklearn.preprocessing import StandardScaler
   X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state=0)
   sc_X = StandardScaler()
   X_train = sc_X.fit_transform(X_train)
   X_test = sc_X.transform(X_test)
   print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
   print(f'Test Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')
6. from sklearn.linear_model import LogisticRegression
   lm = LogisticRegression(random_state = 0, solver='lbfgs')
   lm.fit(X_train, Y_train)
   predictions = lm.predict(X_test)
   plt.figure(figsize=(6, 6))
   sns.regplot(x = X_test[:, 1], y = predictions, scatter_kws={'s':5})
   plt.scatter(X_test[:, 1], Y_test, marker = '+')
   plt.xlabel("User's Estimated Salary")
   plt.ylabel('Ads Purchased')
   plt.title('Regression Line Tracing')
7. from sklearn.metrics import confusion_matrix
   from sklearn.metrics import classification_report
   cm = confusion_matrix(Y_test, predictions)
   print(f'''Confusion matrix :\n

```
| Positive Prediction\t| Negative Prediction
---------------+-----------------------+--------------------
Positive Class | True Positive (TP) {cm[0, 0]}\t| False Negative (FN) {cm[0, 1]}
---------------+-----------------------+--------------------
Negative Class | False Positive (FP) {cm[1, 0]}\t| True Negative (TN) {cm[1, 1]}\n''')
cr = classification_report(Y_test, predictions)
print('Classification report : \n', cr)                    #confusion matrix
```

8.
```
# Visualizing the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, Y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() +
1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step =
0.01))
plt.figure(figsize=(9, 7.5))
plt.contourf(X1, X2, lm.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.6, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
   plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
            color = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

9.
```
# Visualizing the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, Y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() +
1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step =
0.01))
plt.figure(figsize=(9, 7.5))
plt.contourf(X1, X2, lm.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.6, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(),X2.max())
for i, j in enumerate(np.unique(y_set)):
   plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
            c = ListedColormap(('red', 'green'))(i), label = j)
```

```
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

theory :

Logistic Regression : can be used for various classification problems such as spam detection. Diabetes prediction, if a given customer will purchase a particular product or will they churn another competitor, whether the user will click on a given advertisement link or not, and many more examples are in the bucket.

Differentiate between Linear and Logistic Regression

Linear regression gives you a continuous output, but logistic regression provides a constant output.

Sigmoid Function : The sigmoid function, also called logistic function, gives an 'S' shaped curve that can take any real-valued number and map it into a value between 0 and 1

Confusion Matrix Evaluation Metrics

Contingency table or Confusion matrix is often used to measure the performance of classifiers. A confusion matrix contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix.

Some Important measures derived from confusion matrix are:

● Number of positive (Pos) : Total number instances which are labelled as positive in a given dataset. ● Number of negative (Neg) : Total number instances which are labelled as negative in a given dataset.

● Number of True Positive (TP) : Number of instances which are actually labelled as positive and the predicted class by classifier is also positive.

● Number of True Negative (TN) : Number of instances which are actually labelled as negative and the predicted class by classifier is also negative.

● Number of False Positive (FP) : Number of instances which are actually labelled as negative and the predicted class by classifier is positive.

● Number of False Negative (FN): Number of instances which are actually labelled as positive and the class predicted by the classifier is negative.

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 numpy as np
   import matplotlib.pyplot as plt
   import pandas as pd
   import seaborn as sns
   df = pd.read_csv('iris.csv')
   df.head()
2. df.info()
3. X = df.iloc[:, :4].values
   Y = df['Species'].values
4. from sklearn.model_selection import train_test_split
   from sklearn.preprocessing import StandardScaler
   X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=0)
   sc_X = StandardScaler()
   X_train = sc_X.fit_transform(X_train)
   X_test = sc_X.transform(X_test)
   print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
   print(f'Test Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')
5. from sklearn.naive_bayes import GaussianNB
   classifier = GaussianNB()
   classifier.fit(X_train, Y_train)
   predictions = classifier.predict(X_test)
   mapper = {'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2}
   predictions_ = [mapper[i] for i in predictions]
   fig, axs = plt.subplots(2, 2, figsize = (12, 10), constrained_layout = True)
   fig.suptitle('Regression Line Tracing')
   for i in range(4):
       x, y = i // 2, i % 2
       sns.regplot(x = X_test[:, i], y = predictions_, ax=axs[x, y])
       axs[x, y].scatter(X_test[:, i][::-1], Y_test[::-1], marker = '+', color="white")
       axs[x, y].set_xlabel(df.columns[i + 1][:-2])
6. from sklearn.metrics import confusion_matrix
   from sklearn.metrics import classification_report
   cm = confusion_matrix(Y_test, predictions)

```
print(f'''Confusion matrix :\n
| Positive Prediction\t| Negative Prediction
---------------+-----------------------+--------------------
Positive Class | True Positive (TP) {cm[0, 0]}\t| False Negative (FN) {cm[0, 1]}
---------------+-----------------------+--------------------
Negative Class | False Positive (FP) {cm[1, 0]}\t| True Negative (TN) {cm[1, 1]}\n''')
cm = classification_report(Y_test, predictions)
print('Classification report : \n', cm)
```

theory :

1. Concepts used in Naïve Bayes classifier
   ● Naïve Bayes Classifier can be used for Classification of categorical data. ○ Let there be a ‘j’ number of classes. C={1,2,….j} ○ Let, input observation is specified by ‘P’ features. Therefore input observation x is given , x = {F1,F2,…..Fp} ○
   The Naïve Bayes classifier depends on Bayes' rule from probability theory. ●

2. Prior probabilities: Probabilities which are calculated for some event based on no other information are called Prior probabilities

$$P(A \cap B) = P\left(\frac{A}{B}\right).P(B) = P\left(\frac{B}{A}\right).P(A)$$

$$\therefore \qquad P\left(\frac{A}{B}\right) = \frac{P\left(\frac{B}{A}\right).P(A)}{P(B)}$$

7) Text Analytics

1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.

2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.

1. #Download the required packages
   import nltk
   nltk.download('punkt')
   nltk.download('stopwords')
   nltk.download('wordnet')
   nltk.download('averaged_perceptron_tagger')
2. #Initialize the text
   #Sentence Tokenization
   text= "Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentences is called Tokenization."

   from nltk.tokenize import sent_tokenize
   tokenized_text= sent_tokenize(text)
   print(tokenized_text)
3. #Word Tokenization
   from nltk.tokenize import word_tokenize
   tokenized_word=word_tokenize(text)
   print(tokenized_word)
4. # print stop words of English
   from nltk.corpus import stopwords
   stop_words=set(stopwords.words("english"))
   print(stop_words)
5. #Removing Punctuations and Stop Word
   text= "How to remove stop words with NLTK library in Python?"
   word_tokens= word_tokenize(text.lower())
   filtered_sentence = []
   for w in word_tokens:
   if w not in stop_words:
   filtered_sentence.append(w)
   print("Tokenized Sentence:",word_tokens)
   print("Filterd Sentence:",filtered_sentence)
6. #Perform Stemming
   from nltk.stem import PorterStemmer

```
e_words= ["wait", "waiting", "waited", "waits"]
ps =PorterStemmer()
for w in e_words:
   rootWord=ps.stem(w)
  print(rootWord
```

7.
```
#Perform Lemmatization
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)
for w in tokenization:
   print("Lemma for {} is {}".format(w, wordnet_lemmatizer.lemmatize(w)))
```

8.
```
#Apply POS Tagging to text
from nltk.tokenize import word_tokenize
data="The pink sweater fit her perfectly"
words=word_tokenize(data)
for word in words:
   print(nltk.pos_tag([word]))
```

theory :

6. Text Analysis Operations using natural language toolkit

7. NLTK(natural language toolkit) is a leading platform for building Python programs to work with human language data

Tokenization:

Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentences is called Tokenization. Token is a single entity that is the building blocks for a sentence or paragraph.

● Sentence tokenization : split a paragraph into list of sentences using sent_tokenize() method

Stop words removal

Stopwords considered as noise in the text. Text may contain stop words such as is, am, are, this, a, an, the, etc. In NLTK for removing stopwords, you need to create a list of stopwords and filter out your list of tokens from these words

Stemming is a normalization technique where lists of tokenized words are converted into shortened root words to remove redundancy

Lemmatization in NLTK is the algorithmic process of finding the lemma of a word depending on its meaning and context

Lemmatization Vs Stemming

stemming algorithm works by cutting the suffix from the word. In a broader sense cuts either the beginning or end of the word.

On the contrary, Lemmatization is a more powerful operation, and it takes into consideration morphological analysis of the words

POS (Parts of Speech) tell us about grammatical information of words of the sentence by assigning specific token (Determiner, noun, adjective , adverb , verb,Personal Pronoun etc.) as tag (DT,NN ,JJ,RB,VB,PRP etc) to each words

Term frequency–inverse document frequency(TFIDF) , is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus

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 numpy as np
   import matplotlib.pyplot as plt
   import seaborn as sns
   import warnings
   warnings.filterwarnings('ignore')
2. data = pd.read_csv('titanic_data.csv')
   data.head()
3. data.shape
4. data.describe()
5. data.describe(include = 'object')
6. data.isnull().sum()
7. data['Age'] = data['Age'].fillna(np.mean(data['Age']))
8. data['Cabin'] = data['Cabin'].fillna(data['Cabin'].mode()[0])
9. data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])
10. data.isnull().sum()
11. sns.countplot(x='Survived',data=data)
12. sns.countplot(x='Pclass',data=data)
13. sns.countplot(x='Embarked',data=data)
14. sns.countplot(x='Sex',data=data)
15. sns.boxplot(data['Age'])
16. sns.boxplot(data['Fare'])
17. sns.boxplot(data['Pclass'])
18. sns.boxplot(data['SibSp'])
19. sns.catplot(x= 'Pclass', y = 'Age', data=data, kind = 'box')
20. sns.catplot(x= 'Pclass', y = 'Fare', data=data, kind = 'strip')
21. sns.catplot(x= 'Sex', y = 'Fare', data=data, kind = 'strip')
22. sns.catplot(x= 'Sex', y = 'Age', data=data, kind = 'strip')
23. sns.pairplot(data)
24. sns.scatterplot(x = 'Survived', y = 'Fare', data = data)
25. sns.scatterplot(x = 'Survived', y = 'Fare', data = data)
26. sns.distplot(data['Age'])
27. sns.distplot(data['Fare'])

28. sns.jointplot(x = "Survived", y = "Fare", kind = "scatter", data = data)
29. tc = data.corr()
    sns.heatmap(tc, cmap="YlGnBu")
    plt.title('Correlation')
30. sns.catplot(x='Pclass', y='Fare', data=data, kind='bar')
31. import matplotlib.pyplot as plt
32. plt.hist(data['Fare'])

theory :

Data Visualisation plays a very important role in Data mining. Various data scientists spent their time exploring data through visualisation. To accelerate this process we need to have a welldocumentation of all the plots

Countplot

The countplot is used to represent the occurrence(counts) of the observation present in the categorical variable.

Boxplot

A boxplot is a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile [Q1], median, third quartile [Q3] and “maximum”). It can tell you about your outliers and what their values are.

catplot

The Seaborn catplot() function provides a figure-level interface for creating categorical plots. This means that the function allows you to map to a figure, rather than an axes object.

pairplot

To plot multiple pairwise bivariate distributions in a dataset, you can use the .pairplot() function. The diagonal plots are the univariate plots, and this displays the relationship for the (n, 2) combination of variables in a DataFrame as a matrix of plots.

Scatterplot

Scatter plots are the graphs that present the relationship between two variables in a data-set. It represents data points on a twodimensional plane or on a Cartesian system

distplot

These plots help us to visualise the distribution of data. We can use these plots to understand the mean, median, range, variance, deviation, etc of the data.

jointplot

The joint plot is a way of understanding the relationship between two variables and the distribution of individuals of each variable.

corr()

Pandas dataframe.corr() is used to find the pairwise correlation of all columns in the Pandas Dataframe in Python.

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 seaborn as sns
   dataset = sns.load_dataset('titanic')
   dataset.head()
2. sns.boxplot(x='sex' ,y='age' ,data=dataset)
3. sns.boxplot(x='sex' ,y='age' ,data=dataset ,hue='survived')
```

theory :

An introduction to seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them

Import seaborn

import seaborn as sns Seaborn is the only library we need to import for this simple example. By convention, it is imported with the shorthand sns.

Load an example

dataset tips = sns.load_dataset("tips")

Most code in the docs will use the load_dataset() function to get quick access to an example dataset. There's nothing special about these datasets: they are just pandas dataframes, and we could have loaded them with pandas.read_csv() or built them by hand. Most of the

examples in the documentation will specify data using pandas dataframes, but seaborn is very flexible about the data structures that it accepts.

10) 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 box plot for each feature in the dataset

. 4. Compare distributions and identify outliers.

1. import numpy as np
   import matplotlib.pyplot as plt
   import pandas as pd
   import seaborn as sns
   import warnings
   warnings.filterwarnings('ignore')
   df = pd.read_csv('iris.csv')
   df.head()
2. df.isnull().sum()
3. df['PetalLengthCm']=df['PetalLengthCm'].fillna(np.mean(df['PetalLengthCm']))
4. df.isnull().sum()
5. df.info() #1. List down the features and their types (e.g., numeric, nominal) available in the dataset #Hence the dataset contains 4 numerical columns and 1 object column
6. np.unique(df["Species"])
7. df.describe()
8. fig, axes = plt.subplots(2, 2, figsize=(12, 6), constrained_layout = True)
   for i in range(4):
     x, y = i // 2, i % 2
     axes[x, y].hist(df[df.columns[i + 1]])
     axes[x, y].set_title(f"Distribution of {df.columns[i + 1][:-2]}") #2. Create a histogram for each feature in the dataset to illustrate the feature distributions
9. data_to_plot = [df[x] for x in df.columns[1:-1]]
   fig, axes = plt.subplots(1, figsize=(12,8))
   bp = axes.boxplot(data_to_plot) #3. Create a boxplot for each feature in the dataset.
10. 4. Compare distributions and identify outliers.
    If we observe closely for the box 2, interquartile distance is roughly around 0.75 hence the values lying beyond this range of (third quartile + interquartile distance)

i.e. roughly around 4.05 will be considered as outliers. Similarly outliers with other boxplots can be found.

Theory :

Input:

Structured Dataset: Iris

Dataset File: iris.csv

Output:

1. Display Dataset Details.

2. Calculate Min, Max, Mean, Variance value and Percentiles of probabilities also Display Specific use quantile.

3. Display the Histogram using Hist Function. 4. Display the Boxplot using Boxplot Function.

Application:

1. The histogram is suitable for visualizing distribution of numerical data over a continuous interval, or a certain time period. The histogram organizes large amounts of data, and produces visualization quickly, using a single dimension.

2. The box plot allows quick graphical examination of one or more data sets. Box plots may seem more primitive than a histogram but they do have some advantages. They take up less space and are therefore particularly useful for comparing distributions between several groups or sets of data. Choice of number and width of bins techniques can heavily influence the appearance of a histogram, and choice of bandwidth can heavily influence the appearance of a kernel density estimate.

3. Data Visualization Application lets you quickly create insightful data visualizations, in minutes.

Data visualization tools allow anyone to organize and present information intuitively. They enables users to share data visualizations with others.