Title: 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 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.

```
import pandas as pd
data=pd.read_csv("ass1.csv")
df=pd.DataFrame(data)
df

df.head()
df.tail()
df.isnull()
df.describe()

#Reshaping
df['Variable']=df['Variable'].map({'Seasonally adjusted':0,'Actual':1}).astype(float)
df
#Filtering
df=df[df['Data_value']>=12567]
df=df.drop(['Gas'],axis=1)
df
```

#Wrangling data by removing Duplication:

```
non_duplicate = df[~df.duplicated('Anzsic_description')]
non_duplicate
```

#Wrangling Data Using Merge Opration

```
stud_detail=pd.DataFrame({
   'Id':['101','102','103','104','105','106','107','108','109','110'],
   'Name':['Sanket','Priya','Prajwal','Rushikesh','Diya','Vivek','Athrva','Rutuja','Pooja','Ravi'],
   'Gender':['M','F','M','M','F','M','M','F','F','M']
})
stud_marks=pd.DataFrame({
   'Id':['101','102','103','104','105','106','107','108','109','110'],
   'Marks':[50,67,70,84,45,77,67,85,96,79]
})
print(pd.merge(stud_detail, stud_marks, on='Id')) #merge
```

#Wrangling Data Using Grouping Method

```
stud_db={
   'Name':['Sanket','Sanket','Rushikesh','Vivek','Vivek','Rushikesh','Sanket','Prajwal','Prajwal','Athrav'],
   'Subject':['DSBDA','WT','WT','DSBDA','AI','DSBDA','AI','DSBDA','AI','DSBDA'],
   'Mark':[78,64,83,42,90,58,76,55,40,88]
}
df=pd.DataFrame(stud_db)
df
grouped = df.groupby('Subject')
print(grouped.get_group('DSBDA'))
```

#Dealing with missing values

```
stud_db={
   'Name':['Sanket','Sanket','Rushikesh','Vivek','Vivek','Rushikesh','Sanket','Prajwal','Prajwal','Athrav'],
   'Subject':['DSBDA','WT','WT','DSBDA','AI','DSBDA','AI','DSBDA','AI','DSBDA'],
   'Mark':[78,64,'NaN',42,90,'NaN',76,55,40,'NaN']
}
df=pd.DataFrame(stud_db)
df
```

```
c = avg = 0
for ele in df['Mark']:
    if str(ele).isnumeric():
        c += 1
        avg += ele
    avg /= c
df = df.replace(to_replace="NaN",value=avg)
df
```

Theory:

Data Wrangling:

Data Wrangling is the process of gathering, collecting, and transforming Raw data into another format for better understanding, decision-making, accessing, and analysis in less time.

Reshaping data

in the Variable column, we can reshape the data by categorizing them into different numbers.

Merge operation is used to merge raw data and into the desired format.

Filtering data:

Some times datasets are comprised of unwanted rows or columns which are required to be removed or filtered

Title: Create an “Academic performance” dataset of students and perform the given operations using Python.

```
import pandas as pd
import numpy as np
import seaborn as sns

data=pd.read_csv("stud_db.csv")
df=pd.DataFrame(data)
df
df.info()
df.describe()
df.head(10)
```

#Filling Null Values

```
df['DSBDA'] = df['DSBDA'].fillna(df['DSBDA'].mean())
df['AI'] = df['AI'].fillna(df['AI'].mean())
df['WT'] = df['WT'].fillna(df['WT'].mean())
df['Total'] = df['Total'].fillna(df['Total'].mean())
df['Average'] = df['Average'].fillna(df['Average'].mean())
df.head(5)
```

#BoxPlot Method

```
sns.boxplot(df['Age'])
```

#Scatter Plot Method

```
x=df.Age
y=df.RollNo
plt.scatter(x,y,c="Green",s=200)
plt.show()
```

#Z-score Method

```
data=df.Age
m=np.mean(data)
sd=np.std(data)
print("Mean is :",m)
print("Standard Deviation is:",sd)
```

```
threshold = 3
outlier = []
for i in data:
   z = (i-m)/sd
   if z > threshold:
      outlier.append(i)
print('Outlier in dataset is', outlier)
```

#IQR (Inter Quartile Range)

```
data=np.sort(df.Age)
Q1 = np.percentile(data, 25, interpolation = 'midpoint')
Q2 = np.percentile(data, 50, interpolation = 'midpoint')
Q3 = np.percentile(data, 75, interpolation = 'midpoint')
print('Q1 25 percentile of the given data is, ', Q1)
print('Q1 50 percentile of the given data is, ', Q2)
print('Q1 75 percentile of the given data is, ', Q3)
IQR = Q3 - Q1
print('Interquartile range is', IQR)
```

#Removing the Outliers

```
df.drop(10, inplace = True)
df
```

Theory

Boxplot method

It captures the summary of the data effectively and efficiently with only a simple box

Scatter Plot Method

It is used when you have paired numerical data, or when your dependent variable has multiple values for each reading independent variable, or when trying to determine the relationship between the two variables.

Z-score Method

It is also called a standard score. This value/score helps to understand that 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

IQR

Inter Quartile Range approach to finding the outliers is the most commonly used and most trusted approach used in the research field

MinMax Scaler It just scales all the data between 0 and 1.

Standard Scaler scales the values such that the mean is 0 and the standard deviation is 1

Log Transform convert a skewed distribution to a normal distribution

Title: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.

```
import pandas as pd
data=pd.read_csv("iris.csv")
df=pd.DataFrame(data)
df
print(df.dtypes)
#Mean
print("Mean of Iris dataset \n",df.groupby('species').mean(),"\n")
#Median
print("Median of Iris dataset \n",df.groupby('species').median(),"\n")
#Mode
print("Mode of Iris dataset \n",df.mode(axis=1, numeric_only=True),"\n") #column
print("Mode of Iris dataset \n",df.mode(axis=0, numeric_only=False),"\n") #row
#Standard Deviation
print("STD of Iris dataset \n",df.groupby('species').std(),"\n")
#Varience
print("Variance of Iris dataset \n",df.groupby('species').var(),"\n")
#Maximum
print("Max of Iris dataset \n",df.groupby('species').max(),"\n")
#Minimum
print("Min of Iris dataset \n",df.groupby('species').min(),"\n")
```

Theory:

Statistics is a branch of mathematics that deals with collecting, analyzing, interpreting, and visualizing empirical data**.**

Mean sum of all observations divided by the number of observations.

Mode most frequently occurring value in your data

Median middle value of a data.

Standard Deviation measure of how spread out numbers are,square root of Variance

Varience average of the squared differences from the Mean

Title:

Create a Linear Regression Model using Python/R to predict home prices using Boston

Housing Dataset (https://www.kaggle.com/c/boston-housing).

```
import pandas as pd
import numpy as np
boston = pd.read_csv('boston_train.csv')
boston.head()
boston.isnull().sum()

X = pd.DataFrame(np.c_[boston['lstat'], boston['rm']], columns = ['lstat','rm'])
Y = boston['medv']
from sklearn.linear_model import LinearRegression
#Initialize the linear regression model
reg = LinearRegression()

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=42)
reg.fit(x_train, y_train)

#print our price predictions on our test data
y_pred = reg.predict(x_test)
print(y_pred)

#Print the the prediction for the third row of our test data
y_pred[2]
y_train[0]

print(np.mean((y_pred-y_test)**2)) #by numpy

from sklearn.metrics import mean_squared_error #by sklearn
print(mean_squared_error(y_test, y_pred))
```

Linear regression: used to predict the value of a variable based on the value of another variable

Conclusion: Thus, we predicted the value of prices of the house using the given features.

Title:

Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset =pd.read_csv('Social_Network_Ads.csv')
df=pd.DataFrame(dataset)
df
df.head()
```

#X and Y define

```
X = df.iloc[:,[2,3]].values
y = df.iloc[:,4].values
print(X[:3,:])
print('-'*15)
print(y[:3])
```

#splitting dataset into traning set and testing set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
print(X_train[:3])
print('-'*15)
print(y_train[:3])
print(X_test[:3])
print('-'*15)
print(y_test[:3])
```

#feature scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train[:3])
print('-'*15)
```

```
print(X_test[:3])
#Fitting Logistic Regression Model
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(y_pred)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix,classification_report
cm = confusion_matrix(y_test, y_pred)
cl_report=classification_report(y_test,y_pred)
print(cm)
print(cl_report)

TP = cm[1,1]    # True Positive
TN = cm[0,0]    # True Negatives
FP = cm[0,1]    # False Nositives
FN = cm[1,0]    # False Negatives
print(TP)
print(TN)
print(FP)
print(FN)
#Precision
print(TP/TP+FP)
#Recall
print(TP/TP+FN)
#Accuracy
print(TP+TN/TP+FP+TN+FN)
```

Regression: estimating the relationships between a dependent variable and one or more independent variables

Dependent variable: target variable which we are trying to predict.

Independent variables: a variable that stands alone and isn't changed by the other variables

logistic Regression: estimates the probability of an event occurring, such as voted or didn't vote

Precision: Out of all the positive predicted, what percentage is truly positive.

Formula= TP/TP+FP

The precision value lies between 0 and 1.

Recall: Out of the total positive, what percentage are predicted positive.

Formula=TP/TP+FN

Accuracy: it is correct prediction and it tells overall correctness

Formula=TP+TN/TP+FP+TN+FN

confusion matrix: used for finding the errors in prediction

		Actual	
		Positive	**Negative**
Predicted	**Positive**	**TP**	**FP**
	Negative	**FN**	**TN**

True Positive: predicted positive and it's true
True Negative: predicted negative and it's true.
False Positive: predicted positive and it's false.
False Negative: predicted negative and it's false

Title: Data Analytics III

Aim :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.

```
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv("iris.csv")
data.head()
data.tail()
data.shape
data.info()
data.describe()
data.isnull().sum()

X = data.drop(['species'], axis=1)    #Drop species axis=1(column)
y = data.drop(['sepal_length',  'sepal_width',  'petal_length',  'petal_width'], axis=1)
print(X)
print(y)
print(X.shape)
print(y.shape)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
model.score(X_test,y_test)

from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay
print(accuracy_score(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix = cm)
print("Confusion matrix:")
print(cm)

disp.plot()
plt.show()

def get_confusion_matrix_values(y_true, y_pred):
   cm = confusion_matrix(y_true, y_pred)
   return(cm[0][0], cm[0][1], cm[1][0], cm[1][1])

TP, FP, FN, TN = get_confusion_matrix_values(y_test, y_pred)
print("TP: ", TP)
print("FP: ", FP)
print("FN: ", FN)
print("TN: ", TN)

print("The Accuracy is ", (TP+TN)/(TP+TN+FP+FN))
print("The precision is ", TP/(TP+FP))
print("The recall is ", TP/(TP+FN))
```

Naïve Bayes Classifier can be used for Classification of categorical data.

Title: Text Analytics

Aim

1. Extract Sample document and apply following document pre-processing methods:

Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.

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

```
import nltk       (Natural Language Toolkit)
nltk.download('punkt')
```

#Tokenization

```
from nltk.tokenize import sent_tokenize
text = """ I am SANKET """
tokenized_text = sent_tokenize(text)
print(tokenized_text)
from nltk.tokenize import word_tokenize
tokenized_text = word_tokenize(text)
print(tokenized_text)
import nltk
nltk.download('stopwords')
```

#Stop words

```
from nltk.corpus import stopwords
stop_words = set(stopwords.words("english"))
print(stop_words)
```

#Stemming

```
text = """I am SANKET KSHIRSAGAR"""
tokenized_sent = sent_tokenize(text)
filtered_sent=[]
for w in tokenized_sent:
    if w not in stop_words:
        filtered_sent.append(w)
print("Tokenized Sentence:",tokenized_sent)
print("Filtered Sentence:",filtered_sent)
```

#Optional

```
import pkg_resources
import pip
import sys
installedPackages = {pkg.key for pkg in pkg_resources.working_set}
required = {'nltk', 'spacy', 'textblob','gensim' }
missing = required - installedPackages
if missing:
    !pip install nltk==3.4
    !pip install textblob==0.15.3
    !pip install gensim==3.8.2
    !pip install -U SpaCy==2.2.0
    !python -m spacy download en_core_web_lg

from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

ps = PorterStemmer()
stemmed_words=[]
for w in filtered_sent:
    stemmed_words.append(ps.stem(w))
print("Filtered Sentence:",filtered_sent)
print("Stemmed Sentence:",stemmed_words)
```

#Lemmatization

```
text = "This world has a lot of faces "
from textblob import Word
parsed_data= TextBlob(text).words
parsed_data
```

#POS Tagging

```
text = 'My Name Is Sanket'
TextBlob(text).tags
```

Tokenization: The process of breaking down a text paragraph into smaller words or sentence is called Tokenization.

Stopwords: considered as noise in the text.

Stemming: Convert capital letter In small letter

Lemmatization: make the word list

POS Tagging: Part Of Speech it find the noun pronoun verb

Title:

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

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
dataset = sns.load_dataset('titanic')
dataset.head()

sns.distplot(dataset['fare'])
```

kde kernel density estimation. You can remove line

```
sns.distplot(dataset['fare'], kde=False)
sns.jointplot(x='age', y='fare', data=dataset)
sns.rugplot(dataset['fare'])
sns.barplot(x='sex', y='age', data=dataset)
sns.countplot(x='sex', data=dataset)
sns.boxplot(x='sex', y='age', data=dataset)
sns.violinplot(x='sex', y='age', data=dataset)
sns.stripplot(x='sex', y='age', data=dataset)
sns.swarmplot(x='sex', y='age', data=dataset)
```

#Heat map

```
dataset.corr()
corr = dataset.corr()
sns.heatmap(corr)
corr = dataset.corr()
sns.heatmap(corr, annot=True)
corr = dataset.corr()
sns.heatmap(corr)
```

A. Distribution Plots

a. Dist-Plot: represents the univariate analysis

b. Joint Plot: represents the biivariate analysis

d. Rug Plot: draw small bars along the x-axis for each point

B. Categorical Plots

a. Bar Plot: display the mean value

b. Count Plot: displays the count

c. Box Plot: display the distribution of the categorical data in the form of quartiles.

d. Violin Plot: statistical representation of numerical data

C. Advanced Plots

a. Strip Plot: draws a scatter plot

b. Swarm Plot: combination of the strip and the violin plots.

D. Matrix Plots

a. Heat Map:2d representation of data

b. Cluster Map

scatter plot how two variables relate to each other

Title: Data Visualization

Aim: 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.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from seaborn import load_dataset
df=pd.read_csv('titanic.csv')

df.isnull().sum()

df['Age']=df['Age'].fillna(np.mean(df['Age']))
df['Fare']=df['Fare'].fillna(np.mean(df['Fare']))
df['Cabin']=df['Cabin'].fillna(df['Cabin'].mode()[0])

df.isnull().sum()

sns.boxplot(x='Sex',y="Age", data=df)

sns.boxplot(x='Sex',y="Age",data=df, hue="Survived")
plt.show()
sns.barplot(x='Sex',y="Age",data=df, hue="Survived")
df['Sex'].value_counts().plot(kind='pie',autopct="%.2f")
plt.show()
sns.countplot(df['Survived'])
plt.show()
```

Pie Chart: same as the countplot, gives additional info about the percentage

Histogram: value distribution plot of numerical columns.

Scatter Plot: To plot relationship between two numerical variables

Cluster map: understand the relationship between two categorical variables.

Title: 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.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
df=pd.read_csv('iris.csv')

df.head()
df.tail()
df.isnull().sum()
```

#List down the features and their types (e.g., numeric, nominal) available in the dataset.

```
print("\n\nThe features in the dataset are as follows : ")
print("1. Sepal length : ", df['sepal_length'].dtype)
print("2. Sepal width : ", df['sepal_width'].dtype)
print("3. Petal length : ", df['petal_length'].dtype)
print("4. Petal width : ", df['petal_width'].dtype)
print("5. Species : ", df['species'].dtype)
```

#Create a boxplot for each feature in the dataset

```
sns.boxplot(df['sepal_length'])
sns.boxplot(df['sepal_width'])
sns.boxplot(df['petal_length'])
sns.boxplot(df['petal_width'])
```

#Compare distributions and identify outliers.

```
sns.boxplot(df['sepal_length'], df['species'])
sns.boxplot(df['petal_length'], df['species'])
```

Conclusion: In this assignment, we covered how to plot boxplot and Histogram on iris dataset. Also studied how to generate inference from it.