EXPERIMENT 1

TITLE: DATA WRANGLING I

iris dataset

PROBLEM STATEMENT: -

Perform the following operations using Python on any open source dataset (e.g., data.csv)

Import all the required Python Libraries.

1. Locate open source data from the web (e.g. https://www.kaggle.com).
2. Provide a clear description of the data and its source (i.e., URL of the web site).
3. Load the Dataset into the 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.

OBJECTIVE:

1. To Learn and understand the concepts of Python Libraries.
2. To learn and understand the Data Science for the analysis of real time problems
3. To understand and practice Data Preprocessing & Data Normalization.

PREREQUISITE:-

1 Basic of Python Programming

2 Concept of Data Preprocessing, Data Formatting , Data Normalization and Data Cleaning

THEORY:

1. **Introduction to Big Data**

Big data means really a big data, it is a collection of large datasets that cannot be processed using traditional computing techniques. Big data is not merely a data, rather it has become a complete subject, which involves various tools, technqiues and frameworks. Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.

2. 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.

3. Python Libraries for Data Science

a. NumPy

One of the most fundamental packages in Python, NumPy is a general-purpose array-processing package. It provides high-performance multidimensional array objects and tools to work with the arrays. NumPy is an efficient container of generic multi-dimensional data. NumPy's main object is the homogeneous multidimensional array. It is a table of elements or numbers of the same datatype, indexed by a tuple of positive integers. In NumPy, dimensions are called axes and the number of axes is called rank. NumPy's array class is called ndarray aka array.

What can you do with NumPy?

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

b. 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.

c. Scikit Learn

Introduced to the world as a Google Summer of Code project, Scikit Learn is a robust machine learning library for Python. It features ML algorithms like SVMs, random forests, k-means clustering, spectral clustering, mean shift, cross-validation and more... Even NumPy, SciPy and related scientific operations are supported by Scikit Learn with Scikit Learn being a part of the SciPy Stack.

What can you do with Scikit Learn?

1. Classification: Spam detection, image recognition
2. Clustering: Drug response, Stock price
3. Regression: Customer segmentation, Grouping experiment outcomes
4. .Dimensionality reduction: Visualization, Increased efficiency
5. .Model selection: Improved accuracy via parameter tuning
6. .Pre-processing: Preparing input data as a text for processing with machine learning algorithms.

4. Description of Dataset

The Iris dataset was used in R.A. Fisher's classic 1936 paper, The Use of Multiple

Measurements in Taxonomic Problems, and can also be found on the UCI Machine Learning Repository. It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

Total Sample- 150

The columns in this dataset are:

1. Id

2. SepalLengthCm

3. SepalWidthCm

4. PetalLengthCm

5. PetalWidthCm

6. Species

-

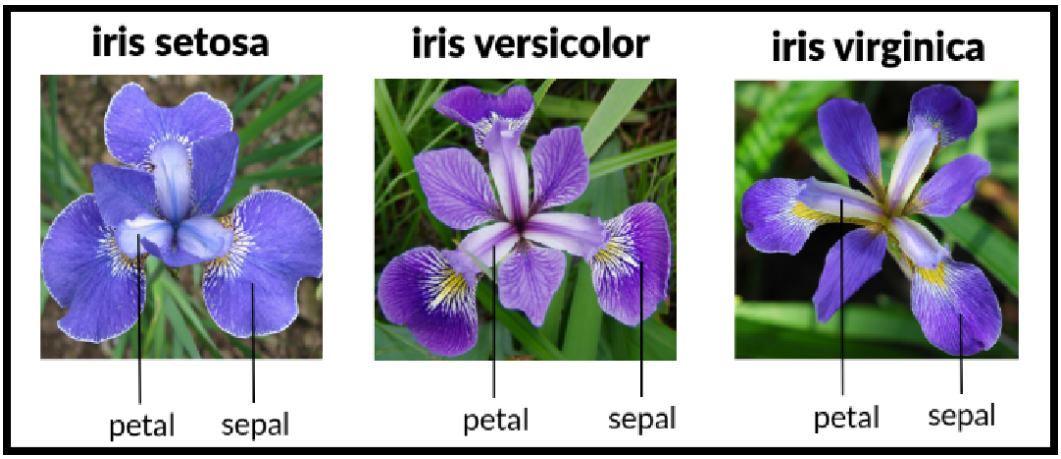

Fig 1 Three Different Types of Species each contain 50 Sample

5. Panda Dataframe functions for Load Dataset

The columns of the resulting DataFrame have different dtypes. iris.dtypes

1.The dataset is downloads from UCI repository.

csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

2. Now Read CSV File as a Dataframe in Python from from path where you saved the same The Iris data set is stored in .csv format. '.csv' stands for comma separated values. It is easier to load .csv files in Pandas data frame and perform various analytical operations on it.

Load Iris.csv into a Pandas data frame —

Syntax-

iris = pd.read_csv(csv_url, header = None)

3.The csv file at the UCI repository does not contain the variable/column names. They are located in a separate file.

col_names = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Species']

4.read in the dataset from the UCI Machine Learning Repository link and specify column names to use

iris = pd.read_csv(csv_url, names = col_names)

Fig.2 Sample Dataset

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

6. Panda Data frame functions for Data Preprocessing :

Sr. No	Data Frame Function	Description
1	**dataset.head(n=5)**	**Return the first n rows.**
2	**dataset.tail(n=5)**	**Return the last n rows.**
3	**dataset.index**	The index (row labels) of the Dataset.
4	**dataset.columns**	The column labels of the Dataset.
5	**dataset.shape**	Return a tuple representing the dimensionality of the Dataset.
6	**dataset.dtypes**	Return the dtypes in the Dataset.
7	**dataset['Column name]**	Read the Data Column wise.
8	**dataset.iloc[5]**	Purely integer-location based indexing for selection by position.
9	**dataset[0:3]**	Selecting via [], which slices the rows.
10	**dataset.loc[:, ["Col_name1", "col_name2"]]**	Selection by label
11	**dataset.iloc[:n, :]**	a subset of the first n rows of the original data
12	**dataset.iloc[:, :n]**	a subset of the first n columns of the original data
13	**dataset.iloc[:m, :n]**	a subset of the first m rows and the first n columns

Table 1. Panda Data frame functions for Data Preprocessing

Checking of Missing Values in Dataset:

- isnull() is the function that is used to check missing values or null values in pandas python.
- isna() function is also used to get the count of missing values of column and row wise count of missing values

a. Is there any missing values in data frame as a whole

Syntax: DataFrame.isnull()

b. Is there any missing values across each column

Syntax: DataFrame.isnull().any()

c.count of missing values across each column using isna() and isnull()

In order to get the count of missing values of the entire dataframe isnull() function is used. sum() which does the column wise sum first and doing another sum() will get the count of missing values of the entire dataframe.

Syntax: dataframe.isnull().sum().sum()

d. count row wise missing value using isnull()

Syntax: dataframe.isnull().sum(axis = 1)

7. Panda functions for Data Formatting and Normalization

The Transforming data stage is about converting the data set into a format that can be analyzed or modelled effectively, and there are several techniques for this process.

A.Data Formatting: Ensuring all data formats are correct (e.g. object, text, floating number, integer, etc.) is another part of this initial 'cleaning' process. If you are working with dates in Pandas, they also need to be stored in the exact format to use special date-time functions.

Sr.	Data Frame	Description	Output

No	Function		

1.	**df.dtypes**	To check the data type	`df.dtypes`
			
			`sepal length (cm)  float64`
			`sepal width (cm)  float64`
			`petal length (cm)  float64`
			`petal width (cm)  float64`
			`dtype: object`

2.	**df['petal length (cm)']= df['petal length (cm)'].astype("int ")**	To change the data type (data type of 'petal length (cm)'changed to int)	`df.dtypes`
			
			`sepal length (cm)  float64`
			`sepal width (cm)  float64`
			`petal length (cm)  int64`
			`petal width (cm)  float64`
			`dtype: object`

B Data normalization:- Mapping all the nominal data values onto a uniform scale (e.g. from 0 to 1) is involved in data normalization. Making the ranges consistent across variables helps with statistical analysis and ensures better comparisonslater on.It is also known as Min-Max scaling.

Algorithm:

Step 1 : Import pandas and sklearn library for preprocessing

from sklearn import preprocessing

Step 2: Load the iris dataset in dataframe object df

iris = load_iris()

df = pd.DataFrame(iris.data, columns=iris.feature_names)

Step 3: Print iris dataset.

df.head()

Step 4: Create x, where x the 'scores' column's values as floats

x = df[['score']].values.astype(float)

Step 5: Create a minimum and maximum processor object

min_max_scaler = preprocessing.MinMaxScaler()

Step 6: Create an object to transform the data to fit minmax processor

x_scaled = min_max_scaler.fit_transform(x)

Step 7:Run the normalizer on the dataframe

df_normalized = pd.DataFrame(x_scaled)

Step 8: View the dataframe

df_normalized

8. Panda Functions for handling categorical variables

- Categorical variables have values that describe a ‘quality’ or ‘characteristic’ of a data unit, like ‘what type’ or ‘which category’.
- Categorical variables fall into mutually exclusive (in one category or in another) and exhaustive (include all possible options) categories. Therefore, categorical variables are qualitative variables and tend to be represented by a non-numeric value.
- Categorical features refer to string type data and can be easily understood by human beings. But in case of a machine, it cannot interpret the categorical data directly. Therefore, the categorical data must be translated into numerical data that can be understood by machine.

There are many ways to convert categorical data into numerical data. Here the most used method is

Label Encoding: Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form. It is an important preprocessing step for the structured dataset in supervised learning.

Example : Suppose we have a column Height in some dataset. After applying labelencoding, the Height column is converted into:

Height
Tall
Medium
Short

Height
0
1
2

Where 0 is the label for tall, 1 is the label for medium, and 2 is a label for short height.

Label Encoding on iris dataset: For iris dataset the target column which is Species. It contains three species Iris-setosa, Iris-versicolor, Iris-virginica.

Sklearn Functions for Label Encoding:

- **preprocessing.LabelEncoder** : It Encode labels with value between 0 and n_classes-1.
- **fit_transform(y):**

  Parameters: yarray-like of shape (n_samples,) Target values.

  Returns: yarray-like of shape (n_samples,)

  Encoded labels.

This transformer should be used to encode target values, and not the input

Algorithm:

Step 1 : Import pandas and sklearn library for preprocessing

```
from sklearn import preprocessing
```

Step 2: Load the iris dataset in dataframe object df

Step 3: Observe the unique values for the Species column.

```
df['Species'].unique()
```

```
output:array(['Iris-setosa','Iris-versicolor','Iris-
virginica'], dtype=object)
```

Step 4: define label_encoder object knows how to understand word labels.

```
label_encoder = preprocessing.LabelEncoder()
```

Step 5: Encode labels in column 'species'.

```
df['Species']= label_encoder.fit_transform(df['Species'])
```

Step 6: Observe the unique values for the Species column.

```
df['Species'].unique()
```

Output: array([0, 1, 2], dtype=int64)

Q.3 : data normalization is the process of organizing data in a database according to rules designed to protect the data and make the database more flexible by eliminating redundancy and inconsistent dependency1. The need of data normalization is to improve the overall database organization, ensure data consistency, reduce redundancy, minimize data modification errors, and simplify the query process2. Data normalization also helps to scale the data to a specific range, such as [0.0, 1.0], which can provide better results for some machine learning algorithms3.

Q.4 isnull() is the function that is used to check missing values or null values in pandas python.
isna() function is also used to get the count of missing values of column and row wise count of missing values
-Deletion:- removing the rows or columns that contain missing values. simple and fast method, but loss of information and reduced sample size.
-Imputation:- filling the missing values with some estimated values based on the available data. There are different methods of imputation, such as mean, median, mode, regression, k-nearest neighbors, etc. preserve the data structure and variability, but bias and uncertainty.
-Indicator variable:- creating a new binary variable that indicates whether the value is missing or not for each observation. help to capture the pattern of missingness and avoid bias, but increase the dimensionality and complexity of the data.
-Model-based methods:- using statistical models to account for the missing data mechanism and estimate the missing values.

CONCLUSION: In this way we have explored the functions of the python library for Data Preprocessing, Data Wrangling Techniques and How to handle missing values on Iris Dataset.

ASSIGNMENT QUESTION

1. Explain Data Frame with Suitable example.
2. What is the limitation of the label encoding method?
3. What is the need of data normalization?
4. What are the different Techniques for Handling the Missing Data?

Q.1 a data frame is a data structure that organizes data into a two-dimensional table of rows and columns, much like a spreadsheet. they are flexible and intuitive for storing and working with data12. Data frames can have different data types for each column and can have named indexes for each row.

An example of a data frame in Python using the pandas library is:

```
import pandas as pd
data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}
df = pd.DataFrame(data)
print(df)
```

Q.2 the limitation of the label encoding method is that it assigns a unique number (starting from 0) to each class of data, which may imply an ordinal relationship among the classes that does not exist.
* For example, if we have a categorical feature with values Mexico, Paris, and Dubai, and we label encode them as 0, 1, and 2 respectively, this may suggest that Dubai is greater than Paris, and Paris is greater than Mexico, which is not true1. This may lead to biased results