4.1. Experiment No. 1

Aim: Data Wrangling I

Perform the following operations using Python on any open source dataset (eg. data.csv)
1. Import all the required Python Libraries.
2. Locate an open source data from the web (eg. 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.

Objective:

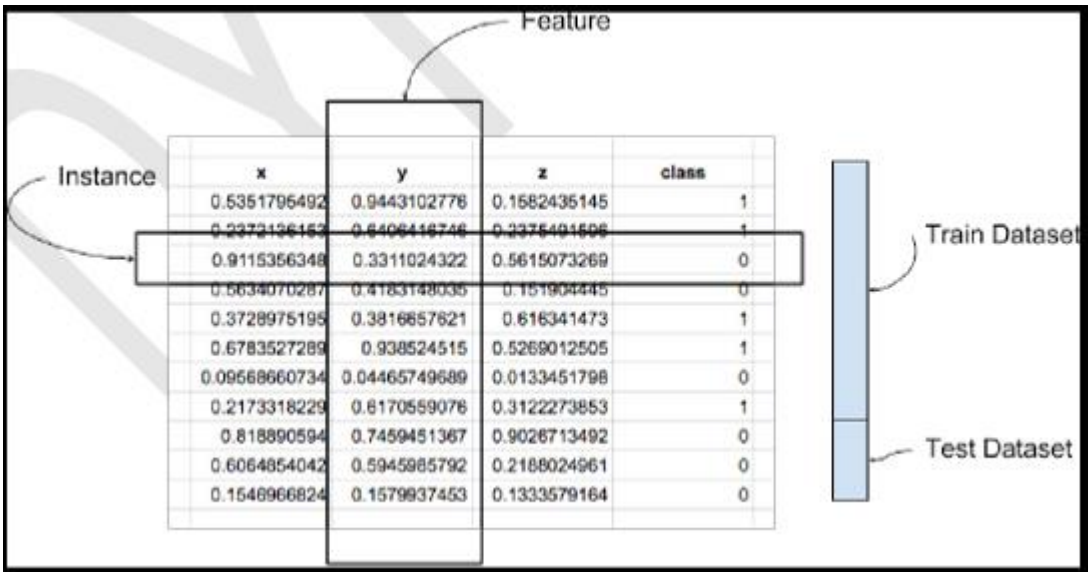
To Study:

1. Import all the required libraries
2. Load the dataset into pandas data frame
3. Data Preprocessing, data formatting, data normalization and handling missing values

Theory:

Introduction to Dataset

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



Instance: A single row of data is called an instance. It is an observation from the domain.

Feature: A single column of data is called a feature. It is a component of an observation and is also called an attribute of a data instance. Some features may be inputs to a model (the predictors) and others may be outputs or the features to be predicted.

Data Type: Features have a data type. They may be real or integer-valued or may have a categorical or ordinal value. You can have strings, dates, times, and more complex types, but typically they are reduced to real or categorical values when working with traditional machine learning methods.

Datasets: A collection of instances is a dataset and when working with machine learning methods we typically need a few datasets for different purposes.

Training Dataset: A dataset that we feed into our machine learning algorithm to train our model.

Testing Dataset: A dataset that we use to validate the accuracy of our model but is not used to train the model. It may be called the validation dataset.

Data Represented in a Table: Data should be arranged in a two-dimensional space made up of rows and columns. This type of data structure makes it easy to understand the data and pinpoint any problems. An example of some raw data stored as a CSV (comma separated values).

```
1., Avatar, 18-12-2009, 7.8
2., Titanic, 18-11-1997,
3., Avengers Infinity War, 27-04-2018, 8.5
```

The representation of the same data in a table is as follows:

S.No	Movie	Release Date	Ratings (IMDb)
1.	Avatar	18-12-2009	7.8
2.	Titanic	18-11-1997	*Na*
3.	Avengers Infinity War	27-04-2018	8.5

Python Data Types

A data type is essentially an internal construct that a programming language uses to understand how to store and manipulate data

Pandas dtype	Python type	NumPy type	Usage
object	str or mixed	string_, unicode_, mixed types	Text or mixed numeric and non-numeric values
int64	Int	int_, int8, int16, int32, int64, uint8, uint16, uint32, uint64	Integer numbers
float64	Float	float_, float16, float32, float64	Floating point numbers
bool	Bool	bool_	True/False values
datetime64	NA	datetime64[ns]	Date and time values
timedelta[ns]	NA	NA	Differences between two datetimes
category	NA	NA	Finite list of text values

Python Libraries for Data Science

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.

(Write brief introduction)

What can you do with Pandas?

Indexing, manipulating, renaming, sorting, merging data frame

1. Update, Add, Delete columns from a data frame

2. Impute missing files, handle missing data or NANs

3. Plot data with histogram or box plot

b) 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.

(Write brief introduction)

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

c) Matplotlib

(Write brief introduction)

What can you do with Matplotlib?

Histogram, bar plots, scatter plots, area plot to pie plot, Matplotlib can depict a wide range of visualizations. With a bit of effort and tint of visualization capabilities, with Matplotlib, you can create just any visualizations:

● Line plots	● Pie charts
● Scatter plots	● Stem plots
● Area plots	● Contour plots
● Bar charts and Histograms	● Quiver plots
	● Spectrograms

d) Seaborn

(Write brief introduction)

What can you do with Seaborn?

1. Determine relationships between multiple variables (correlation)

2. Observe categorical variables for aggregate statistics

3. Analyze univariate or bi-variate distributions and compare them between different data subsets

4. Plot linear regression models for dependent variables

5. Provide high-level abstractions, multi-plot grids

6. Seaborn is a great second-hand for R visualization libraries like corrplot and ggplot.

e) Scikit Learn

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.

(Write brief introduction)

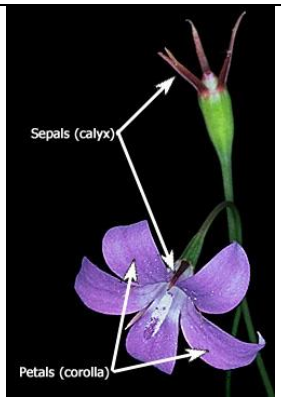
What can you do with Scikit Learn?

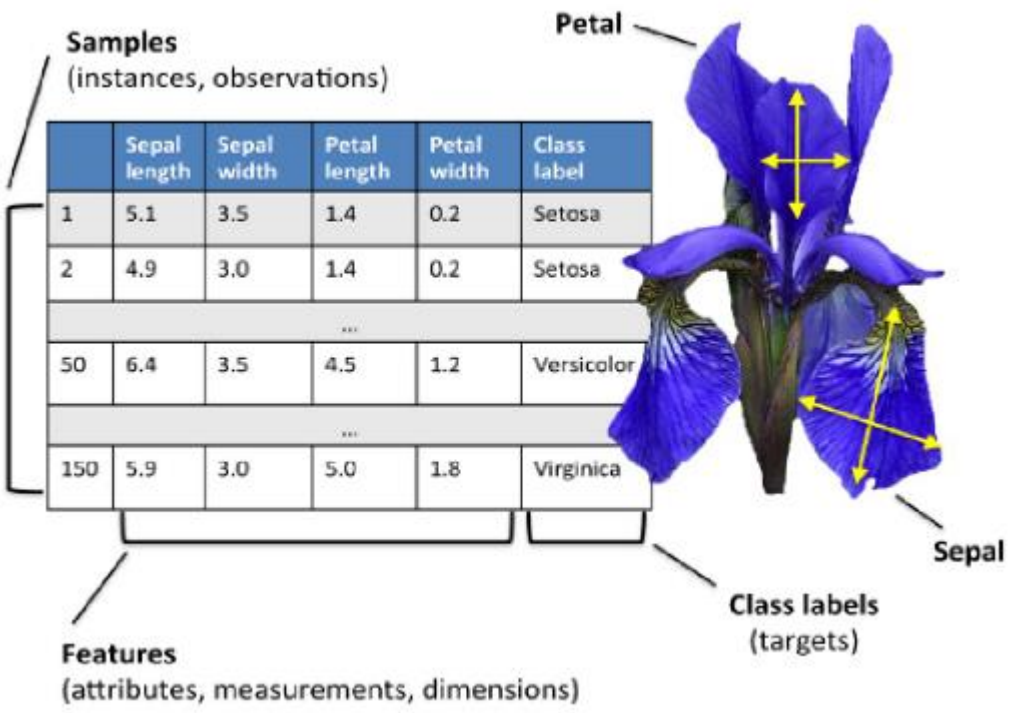
1.Classification: Spam detection, image recognition
2.Clustering: Drug response, Stock price
3.Regression: Customer segmentation, Grouping experiment outcomes 4.Dimensionality reduction: Visualization, Increased efficiency
5.Model selection: Improved accuracy via parameter tuning
6.Pre-processing: Preparing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

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 3 Different Types of Species each contain 50 Sample	



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Panda Data frame functions for Load Dataset:

The columns of the resulting DataFrame have different dtypes.

1. The dataset is downloads from Kaggle repository.(Ref: https://www.kaggle.com)

2. Load all the required libraries

```
import os
import pandas as pd
import numpy as np
os.chdir("D:\Pandas")
print(pd.__version__)
# 2.0.3
```

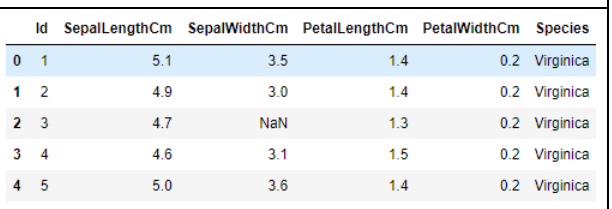
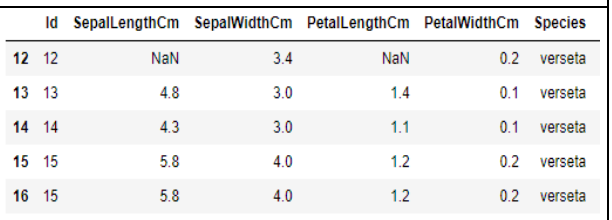
3. Now Read CSV File as a Dataframe in Python from from path where you saved the same The Iris data set is stored in .csv format. '.csv' stands for comma separated values. It is easier to load .csv files in Pandas data frame and perform various analytical operations on it.

Load Iris.csv into a Pandas data frame —

Syntax-

```
df = pd.read_csv('Iris1.csv')
```

Panda Data frame functions for Data Preprocessing and Data frame Operations:

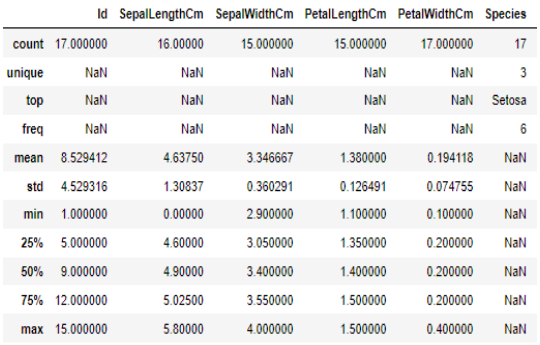
Sr. No.	Data Frame Function(Syntax and example)	Description	Output
1	df.head(5) `df.head(5)`	Return the first n rows.	
2	df.tail(5) `df.tail(5)`	Return the last n rows.	
3	df.index `df.index`	Returns the range index (row labels) of the dataset	`RangeIndex(start=0, stop=17, step=1)`

4	df.column
	`df.columns`
		Returns the column labels of the Dataset.	`Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'Peta`
			`'Species', 'order'],`
			`dtype='object')`

5	df.shape `df.shape`	Return a tuple representing the dimensionality of the Dataset.	`(17, 7)`

6	df.dtypes
	`df.dtypes`
		Return the dtypes in the Dataset. This returns a Series with the data type of each column. The result's index is the original Dataset's columns. Columns with mixed types are stored with the object dtype.	`Id int64`
			`SepalLengthCm float64`
			`SepalWidthCm float64`
			`PetalLengthCm float64`
			`PetalWidthCm float64`
			`Species object`
			`order int64`
			`dtype: object`

7	dataset.columns.values	Return the columns values in the Dataset in array format	`array(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',` `'PetalWidthCm', 'Species', 'order'], dtype=object)`

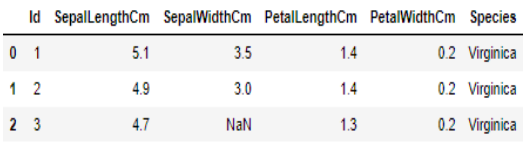
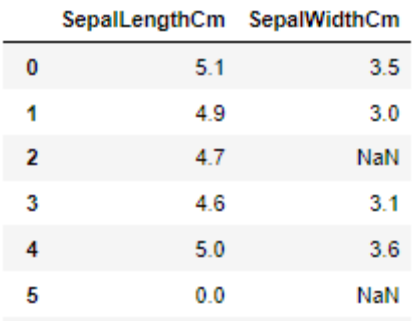
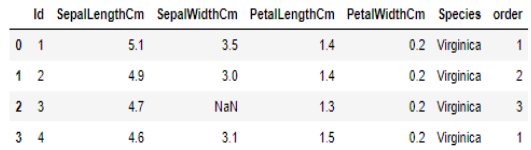
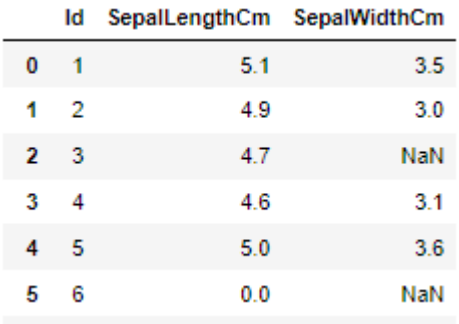
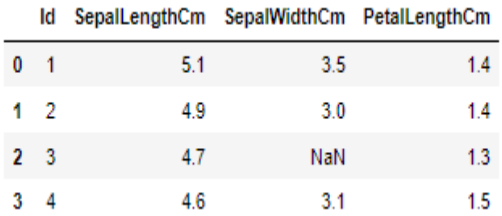
	`df.columns.values`		
8	dataset.describe(include='all') `df.describe(include='all')`	Generate descriptive statistics. To view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values. Analyzes both numeric and object series, as well as Dataset column sets of mixed data types.	

9	dataset['Column name]
	`df['SepalLengthCm']`
		Read the Data Column wise.	`0 5.1`
			`1 4.9`
			`2 4.7`
			`3 4.6`
			`4 5.0`
			`5 0.0`

10	dataset.sort_index(axis=1, ascending=False) `df.sort_index(axis=1, ascending=False)`	Sort object by labels (along an axis).	Write your own output
11	dataset.sort_values(by="C olu mn name") `df.sort_values(by="SepalLengthCm")`	Sort values by column name.	Write your own output

12	dataset.iloc[5]
	`df.iloc[6]`
		Purely integer-location based indexing for selection by position.	`Id 7`
			`SepalLengthCm 4.6`
			`SepalWidthCm 3.4`
			`PetalLengthCm 1.4`
			`PetalWidthCm 0.3`
			`Species Setosa`
			`order 1`
			`Name: 6, dtype: object`

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	17.000000	16.00000	15.000000	15.000000	17.000000	17
unique	NaN	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	NaN	Setosa
freq	NaN	NaN	NaN	NaN	NaN	6
mean	8.529412	4.63750	3.346667	1.380000	0.194118	NaN
std	4.529316	1.30837	0.360291	0.126491	0.074755	NaN
min	1.000000	0.00000	2.900000	1.100000	0.100000	NaN
25%	5.000000	4.60000	3.050000	1.350000	0.200000	NaN
50%	9.000000	4.90000	3.400000	1.400000	0.200000	NaN
75%	12.000000	5.02500	3.550000	1.500000	0.200000	NaN
max	15.000000	5.80000	4.000000	1.500000	0.400000	NaN

13	dataset[0:3] `df[0:3]`	Selecting via [], which slices the rows.	
14	dataset.loc[:, ["Col_name1", "col_name2"]] `df.loc[:,['SepalLengthCm','SepalWidthCm']]`	Selection by label	
15	dataset.iloc[:n, :] `df.iloc[:4, :]`	a subset of the first n rows of the original data	
16	dataset.iloc[:, :n] `df.iloc[:, :3]`	a subset of the first n columns of the original data	
17	dataset.iloc[:m, :n] `df.iloc[:4, :4]`	a subset of the first m rows and the first n columns	

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

● The dataset considered for explanation is:

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

a. is there any missing values in dataframe as a whole

Syntax: DataFrame.isnull()

```
df.isnull()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	order
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	True	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False

b. is there any missing values across each column

Syntax: DataFrame.isnull().any()

```
df.isnull().any()
```

```
Id              False
SepalLengthCm    True
SepalWidthCm     True
PetalLengthCm    True
PetalWidthCm    False
Species         False
order           False
dtype: bool
```

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()

```
df.isnull().sum()
```

```
Id              0
SepalLengthCm   1
SepalWidthCm    2
PetalLengthCm   2
PetalWidthCm    0
Species         0
order           0
dtype: int64
```

d. count of missing values of a specific column.

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

e. groupby count of missing values of a column.

In order to get the count of missing values of the particular column by group in pandas we will be using isnull() and sum() function with apply() and groupby() which performs the group wise count of missing values as shown below.

Syntax: df.groupby(['Gender'])['Score'].apply(lambda x: x.isnull().sum())

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.

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 comparisons later on.It is also known as Min-Max scaling.

Step 1 : Import pandas and sklearn library for preprocessing
Step 2: Load the iris dataset in dataframe object df
Step 3: Print iris dataset.
Step 4: Create x, where x the 'scores' column's values as floats
Step 5: Create a minimum and maximum processor object
Step 6: Create an object to transform the data to fit minmax processor
	`from sklearn import preprocessing`
	`x = df[['SepalLengthCm']].values.astype(float)`
	`min_max_scaler = preprocessing.MinMaxScaler()`
	`x_scaled = min_max_scaler.fit_transform(x)`
	`df_normalized = pd.DataFrame(x_scaled)`
	`df_normalized`

Step 7:Run the normalizer on the dataframe Step 8: View the dataframe	

	0
0	0.879310
1	0.844828
2	0.810345
3	0.793103
4	0.862069
5	0.000000
6	0.793103
7	0.862069
8	0.758621
9	0.844828
10	0.844828

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 three most used methods are discussed.

a. 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 label encoding, the Height column is converted into

Height	Height

Tall	0
Medium	1
Small	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.

Step 1 : Import pandas and sklearn library for preprocessing

Step 2: Load the iris dataset in dataframe object df

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

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

Step 5: Encode labels in column 'species'.

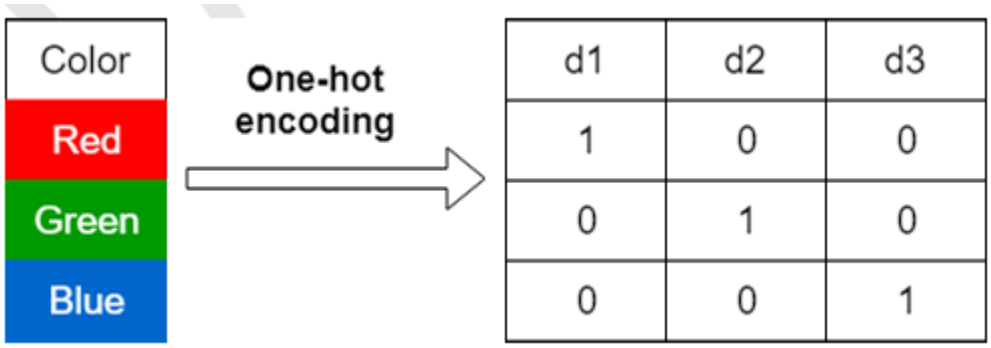
Step 6: Observe the unique values for the Species column.

```
from sklearn import preprocessing
df['Species'].unique()
label_encoder = preprocessing.LabelEncoder()
df['Species']= label_encoder.fit_transform(df['Species'])
df
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	1
1	2	4.9	3.0	1.4	0.2	1
2	3	4.7	NaN	1.3	0.2	1
3	4	4.6	3.1	1.5	0.2	1
4	5	5.0	3.6	1.4	0.2	1
5	6	0.0	NaN	NaN	0.4	0
6	7	4.6	3.4	1.4	0.3	0
7	8	5.0	3.4	1.5	0.2	0
8	9	4.4	2.9	1.4	0.2	0
9	10	4.9	3.1	1.5	0.1	0
10	10	4.9	3.1	1.5	0.1	0
11	11	5.4	3.7	1.5	0.2	2

Limitation: Label encoding converts the data in machine-readable form, but it assigns a unique number(starting from 0) to each class of data. This may lead to the generation of priority issues in the data sets. A label with a high value may be considered to have high priority than a label having a lower value.

b. One-Hot Encoding: In one-hot encoding, we create a new set of dummy (binary) variables that is equal to the number of categories (k) in the variable. For example, let's say we have a categorical variable Color with three categories called "Red", "Green" and "Blue", we need to use three dummy variables to encode this variable using one-hot encoding. A dummy (binary) variable just takes the value 0 or 1 to indicate the exclusion or inclusion of a category.



"Red" color is encoded as [1 0 0] vector of size 3.

“Green” color is encoded as [0 1 0] vector of size 3.

“Blue” color is encoded as [0 0 1] vector of size 3.

One-hot 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 One-hot Encoding:

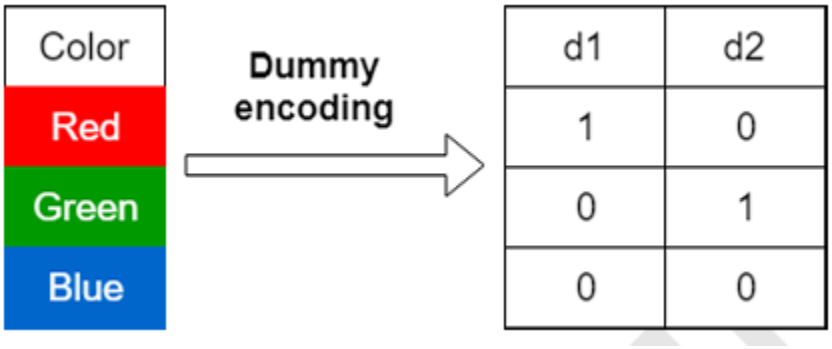
● sklearn.preprocessing.OneHotEncoder(): Encode categorical integer features using a one-hot aka one-of-K scheme

Algorithm:	df['Species'].unique()
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. Step 4: Apply label_encoder object for label encoding the Observe the unique values for the Species column. Step 5: Remove the target variable from dataset Step 6: Apply one_hot encoder for Species column. Step 7: Join the encoded values with Features variable Step 8: Observe the merge dataframe Step 9: Rename the newly encoded columns. Step 10: Observe the merge dataframe	output: array(['Iris-setosa', 'Irisversicolor', 'Iris-virginica'], dtype=object) df['Species'].unique() Output: array([0, 1, 2], dtype=int64) features_df=df.drop(columns=['Species']) enc = preprocessing.OneHotEncoder() enc_df=pd.DataFrame(enc.fit_transform(df[['Species']])).toarray() df_encode = features_df.join(enc_df) df_encode df_encode.rename(columns = {0:'Iris-Setosa', 1:'Iris-Versicolor',2:'Iris-virginica'}, inplace = True) df_encode

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	0	1	2
0	5.1	3.5	1.4	0.2	1.0	0.0	0.0
1	4.9	3.0	1.4	0.2	1.0	0.0	0.0
2	4.7	3.2	1.3	0.2	1.0	0.0	0.0
3	4.6	3.1	1.5	0.2	1.0	0.0	0.0
4	5.0	3.6	1.4	0.2	1.0	0.0	0.0

c. Dummy Variable Encoding

Dummy encoding also uses dummy (binary) variables. Instead of creating a number of dummy variables that is equal to the number of categories (k) in the variable, dummy encoding uses k-1 dummy variables. To encode the same Color variable with three categories using the dummy encoding, we need to use only two dummy variables



n dummy encoding,

“Red” color is encoded as [1 0] vector of size 2.

“Green” color is encoded as [0 1] vector of size

“Blue” color is encoded as [0 0] vector of size

Dummy encoding removes a duplicate category present in the one-hot encoding.

Pandas Functions for One-hot Encoding with dummy variables:

pandas.get_dummies(data, prefix=None, prefix_sep='_', dummy_na=False, columns=None, sparse=False, drop_first=False, dtype=None): Convert categorical variable into dummy/indicator variables.

Parameters:

data:array-like, Series, or DataFrame

Data of which to get dummy indicators.

prefixstr: list of str, or dict of str, default None

String to append DataFrame column names.

prefix_sep: str, default '_'

If appending prefix, separator/delimiter to use. Or pass a list or dictionary as

with prefix.

dummy_nabool: default False

Add a column to indicate NaNs, if False NaNs are ignored.

columns: list:like, default None

Column names in the DataFrame to be encoded. If columns is None then all the

columns with object or category dtype will be converted.

Whether the dummy-encoded columns should be backed by a SparseArray (True) or a regular NumPy array (False).

drop_first:bool, default False

Whether to get k-1 dummies out of k categorical levels by removing the first level.

dtype: dtype, default np.uint8

Data type for new columns. Only a single dtype is allowed.

● Return : DataFrame with Dummy-coded data.

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	

Step 4: Apply label_encoder object for label encoding the Observe the unique values for the Species column. Step 5: Apply one_hot encoder with dummy variables for Species column. Step 6: Apply one_hot encoder with dummy variables for Species column.	

Applications:

(Write down any 3 applications of related practical)

Conclusion:

(Write conclusion in detail)

Outcome:

Upon completion of this experiment, students will be able to:

1.Import the required libraries for data wrangling

2.Load the dataset into pandas data frame

3.Perform data preprocessing, data formatting and data normalization and convert the categorical data into quantitative variables

Questions:

1.

2.

3.

4.

5.