KJ's Educational Institutes

Trinity Academy of Engineering, Pune

Department of Computer Engineering

LABORATORY MANUAL

S.E. (Computer Engineering)

DATA SCIENCE AND BIG DATA ANALYTICS

Staff: Mrs. Nilufar Zaman

(2019 course)

Teaching Scheme:	Examination Scheme:	Credit:
Practical: 4Hrs/ Week/Batch	**Term Work: 25 Marks**	**Practical: 4**

DATA SCIENCE AND BIG DATA ANALYTICS

List of Assignments

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.

2. Data Wrangling II

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

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

Reason and document your approach properly.

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.

Provide the codes with outputs and explain everything that you do in this step.

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.

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

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.

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,
3. Recall on the given dataset.

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.

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.

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')

Write observations on the inference from the above statistics.

10. 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.

Compare distributions and identify outliers.

11. Locate dataset (e.g., sample_weather.txt) for working on weather data which reads the textinput files and finds average for temperature, dew point and wind speed.

12. Write a simple program in SCALA using Apache Spark framework.

13. Use the following dataset and classify tweets into positive and negative tweets.
https://www.kaggle.com/ruchi798/data-science-tweets

14 . Develop a movie recommendation model using the scikit-learn library in python.
Refer dataset:
https://github.com/rashida048/Some-NLP-Projects/blob/master/movie_dataset.csv

Assignment 01:

Problem Statement: Title of the Assignment: Data Wrangling, I

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 of the Assignment: Students should be able to perform the data wrangling operation using Python on any open source dataset

---

Prerequisite:

1. Basic of Python Programming
2. Concept of Data Preprocessing, Data Formatting , Data Normalization and Data Cleaning.

Contents for Theory:

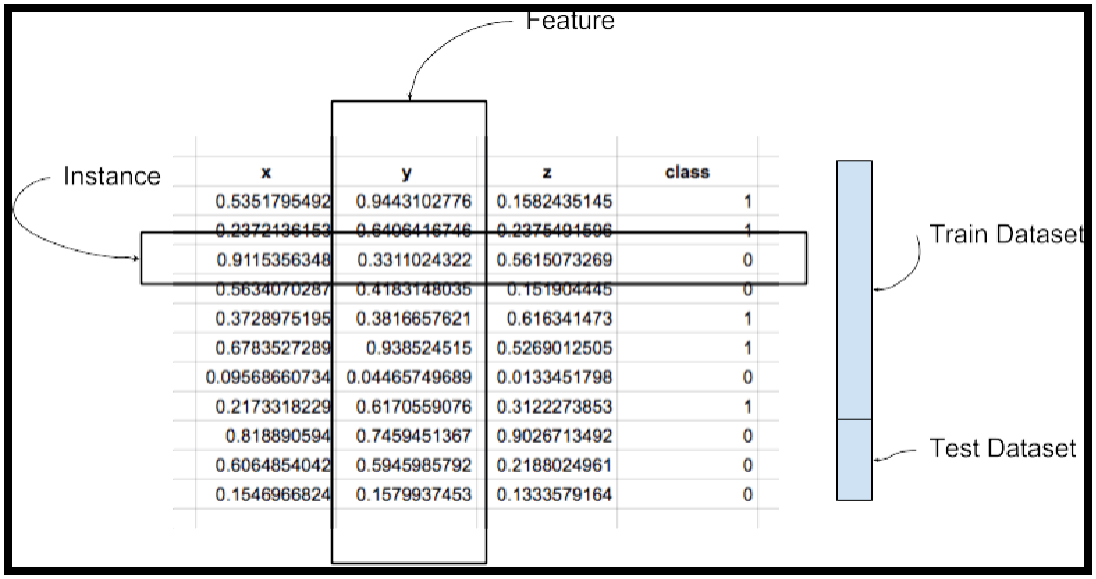
1. **Introduction to Dataset**
2. **Python Libraries for Data Science**
3. **Description of Dataset**
4. **Panda Dataframe functions for load the dataset**
5. **Panda functions for Data Preprocessing**
6. **Panda functions for Data Formatting and Normalization**

7. Panda Functions for handling categorical variables

---

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.



x	y	z	class
0.5351795492	0.9443102776	0.1582435145	1
0.2372136153	0.6406416746	0.2375491596	1
0.9115356348	0.3311024322	0.5615073269	0
0.5634070287	0.4183148035	0.151904445	0
0.3728975195	0.3816657621	0.616341473	1
0.6783527289	0.938524515	0.5269012505	1
0.09568660734	0.04465749689	0.0133451798	0
0.2173318229	0.6170559076	0.3122273853	1
0.818890594	0.7459451367	0.9026713492	0
0.6064854042	0.5945985792	0.2188024961	0
0.1546966824	0.1579937453	0.1333579164	0

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 to used 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

Pandas Data Types

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

A possible confusing point about pandas data types is that there is some overlap between pandas, python and numpy. This table summarizes the key points:

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

2. 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.

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

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

c. Matplotlib

This is undoubtedly my favorite and a quintessential Python library. You can create stories with the data visualized with Matplotlib. Another library from the SciPy Stack, Matplotlib plots 2D figures.

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

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

Matplotlib also facilitates labels, grids, legends, and some more formatting entities with Matplotlib.

d. Seaborn

So when you read the official documentation on Seaborn, it is defined as the data visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics. Putting it simply, seaborn is an extension of Matplotlib with advanced features.

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.

5. 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?

7. Classification: Spam detection, image recognition
8. Clustering: Drug response, Stock price
9. Regression: Customer segmentation, Grouping experiment outcomes
10. Dimensionality reduction: Visualization, Increased efficiency
11. Model selection: Improved accuracy via parameter tuning
12. Pre-processing: Preparing input data as a text for processing with machine learning algorithms.

3. Panda Dataframe functions to Load Dataset

Syntax-

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

4. Panda Dataframe functions for Data Preprocessing :

Dataframe Operations:

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.
		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.
7	**dataset.columns.values**	Return the columns values in the Dataset in array format
8	**dataset.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]**	Read the Data Column wise.
10	**dataset.sort_index(axis=1, ascending=False)**	Sort object by labels (along an axis).
11	**dataset.sort_values(by="Column name")**	Sort values by column name.
12	**dataset.iloc[5]**	Purely integer-location based indexing for selection by position.

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

	Name	State	Gender	Score
0	George	Arizona	M	63.0
1	Andrea	Georgia	F	48.0
2	micheal	Newyork	M	56.0
3	maggie	Indiana	F	75.0
4	Ravi	Florida	M	NaN
5	Xien	California	M	77.0
6	Jalpa	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN

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

Function: DataFrame.isnull()

Output:

	Name	State	Gender	Score
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	True
5	False	False	False	False
6	False	True	True	True
7	True	True	True	True

2. **is there any missing values across each column**

   Function: DataFrame.isnull().any()

   Output:

```
Name      True
State     True
Gender    True
Score     True
dtype: bool
```

3. **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.

   Function: dataframe.isnull().sum().sum()

   Output : 8

4. **count row wise missing value using isnull()**

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

   Output:

```
0    0
1    0
2    0
3    0
4    1
5    0
6    3
7    4
dtype: int64
```

5. **count Column wise missing value using isnull()**

   Method 1:

   Function: dataframe.isnull().sum()

   Output:

```
Name      1
State     2
Gender    2
Score     3
dtype: int64
```

   Method 2:

   unction: dataframe.isna().sum()

```
Name      1
State     2
Gender    2
Score     3
dtype: int64
```

6. **count of missing values of a specific column.**

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

   df1.Gender.isnull().sum()

   Output: 2

7. **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.

   Function: df1.groupby(['Gender'])['Score'].apply(lambda x: x.isnull().sum())

   Output:

```
Gender
F    0
M    1
Name: Score, dtype: int64
```

5. 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.

1. **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.

   Functions used for data formatting

Sr. No	Data Frame Function	Description	Output

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. **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.

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: Print iris dataset.

```
df.head()
```

Step 5: Create a minimum and maximum processor object

```
min_max_scaler = preprocessing.MinMaxScaler()
```

Step 6: Separate the feature from the class label

```
x=df.iloc[:,:4]
```

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
```

Output: After Step 3:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Output after step 8:

	0	1	2	3
0	0.222222	0.625000	0.067797	0.041667
1	0.166667	0.416667	0.067797	0.041667
2	0.111111	0.500000	0.050847	0.041667
3	0.083333	0.458333	0.084746	0.041667
4	0.194444	0.666667	0.067797	0.041667

6. **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 t**end 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.**

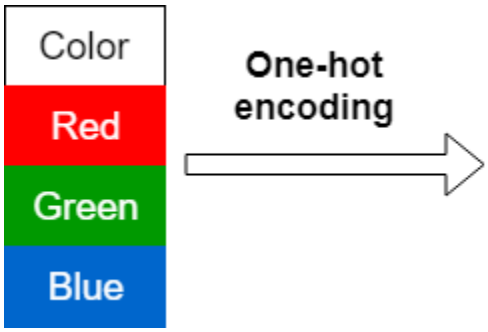
There are many ways to convert categorical data into numerical data. Here the three most used methods are discussed.

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:

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.



d1	d2	d3
1	0	0
0	1	0
0	0	1

In one-hot encoding,

“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:

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: Apply label_encoder object for label encoding the Observe the unique values for the Species column.

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

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

Step 5: Remove the target variable from dataset

```
features_df=df.drop(columns=['Species'])
```

Step 6: Apply one_hot encoder for Species column.

```
enc = preprocessing.OneHotEncoder()
enc_df=pd.DataFrame(enc.fit_transform(df[['Species']])).toarray()
```

Step 7: Join the encoded values with Features variable

```
df_encode = features_df.join(enc_df)
```

Step 8: Observe the merge dataframe

```
df_encode
```

Step 9: Rename the newly encoded columns.

```
df_encode.rename(columns = {0:'Iris-Setosa',
1:'Iris-Versicolor',2:'Iris-virginica'}, inplace = True)
```

Step 10: Observe the merge dataframe

```
df_encode
```

Output after Step 8:

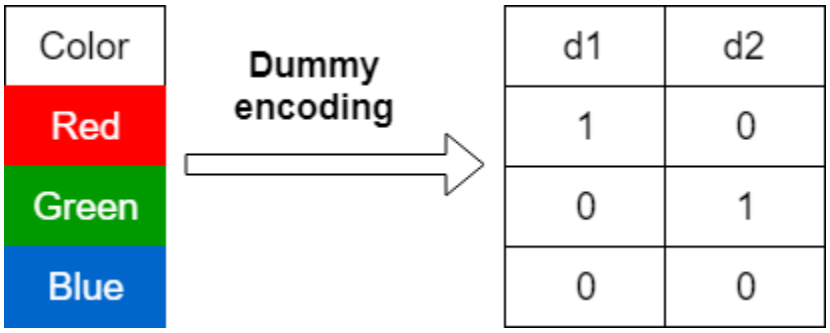
	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	0	1	2
0	5.1	3.5	1.4	0.2	1.0	0.0	0.0
1	4.9	3.0	1.4	0.2	1.0	0.0	0.0
2	4.7	3.2	1.3	0.2	1.0	0.0	0.0
3	4.6	3.1	1.5	0.2	1.0	0.0	0.0
4	5.0	3.6	1.4	0.2	1.0	0.0	0.0

Output after Step 10:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Iris-Setosa	Iris-Versicolor	Iris-virginica
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.



In dummy encoding,

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

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

“Blue” color is encoded as **[0 0]** vector of size 2.

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.

sparse: bool: default False

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.

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

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

Step 4: Apply label_encoder object for label encoding the Observe the unique values for the Species column.

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

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

Step 6: Apply one_hot encoder with dummy variables for Species column.

```
one_hot_df = pd.get_dummies(df, prefix="Species", columns=['Species'], drop_first=True)
```

Step 7: Observe the merge dataframe

```
one_hot_df
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species_1	Species_2
0	5.1	3.5	1.4	0.2	0	0
1	4.9	3.0	1.4	0.2	0	0
2	4.7	3.2	1.3	0.2	0	0
3	4.6	3.1	1.5	0.2	0	0
4	5.0	3.6	1.4	0.2	0	0
...	...	...	...	...	...	...

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. **Explain one hot encoding.**
4. **What is the need of data normalization?**
5. **What are the different Techniques for Handling the Missing Data?**