

Assignment No: 1

Aim/Problem Statement:- Data Wrangling I

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

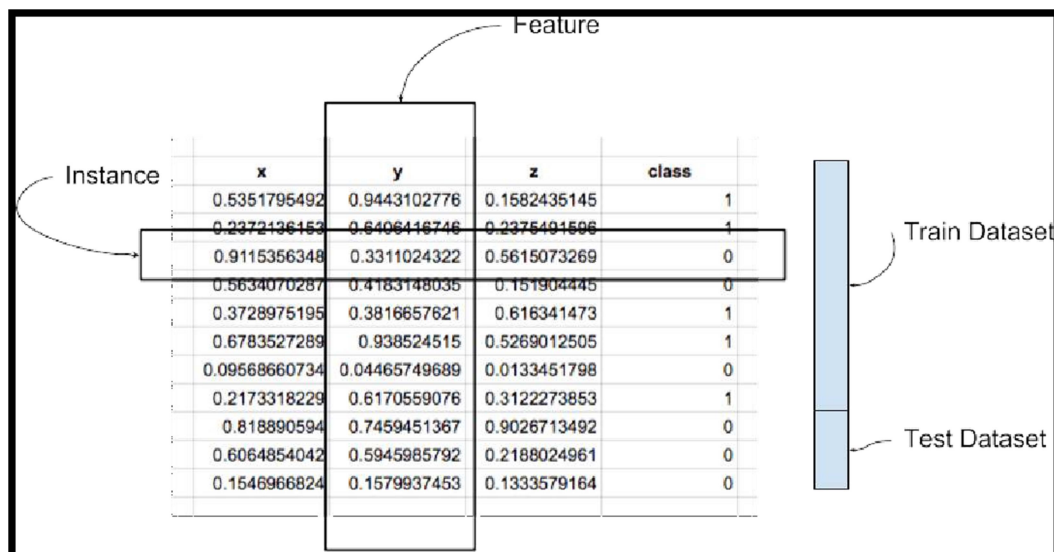
1. Import all the required Python Libraries.
2. Locate an open source data from the web (eg. <https://www.kaggle.com>). Provide a clear description of the data and its source (i.e. URL of the web site).
3. Load the Dataset into pandas dataframe.
4. Data Preprocessing: check for missing values in the 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.

Theory:-

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.



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

Artificial Intelligence & Data Science

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

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 data types	NumPy dtype	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.

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.

Artificial Intelligence & Data Science

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. 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

Artificial Intelligence & Data Science

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

c. 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
1. Plot linear regression models for dependent variables
2. Provide high-level abstractions, multi-plot grids
3. Seaborn is a great second-hand for R visualization libraries like corrplot and ggplot.

d. 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.

Artificial Intelligence & Data Science

Dataset Description

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:

Id

SepalLengthCm

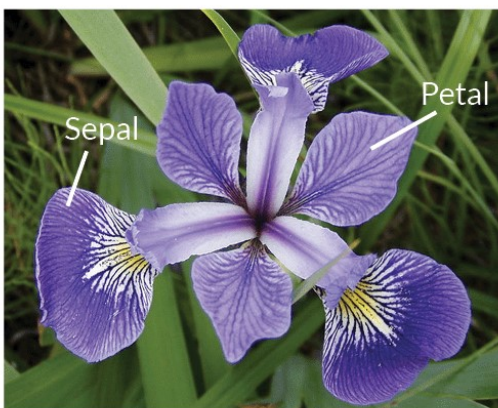
SepalWidthCm

PetalLengthCm

PetalWidthCm

Species

3 Different Types of Species each contain 50 Sample-



Iris Versicolor

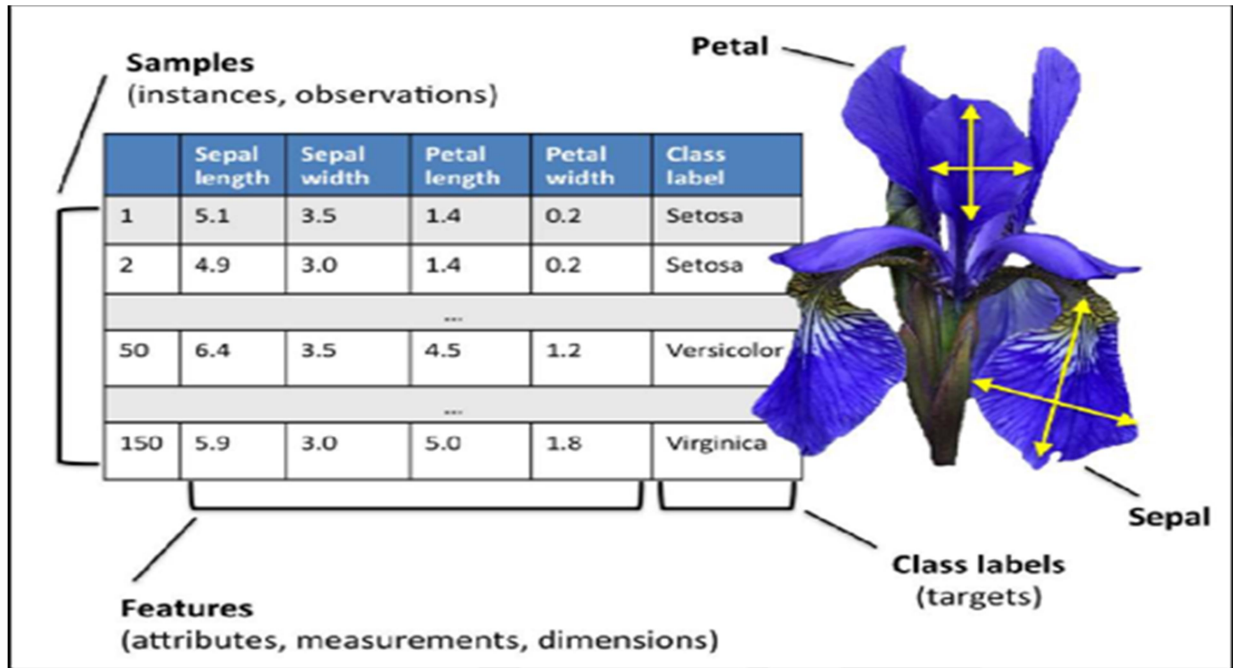


Iris Setosa



Iris Virginica

Artificial Intelligence & Data Science



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.

Artificial Intelligence & Data Science

```
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 columnnames to use- `iris = pd.read_csv(csv_url, names = col_names)`

	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

5. Panda Dataframe functions for Data Preprocessing: Dataframe Operations:

Data Frame Function	Description
<code>dataset.head(n=5)</code>	Return the first n rows.
<code>dataset.tail(n=5)</code>	Return the last n rows.
<code>dataset.index</code>	The index (row labels) of the Dataset.
<code>dataset.columns</code>	The column labels of the Dataset.
<code>dataset.shape</code>	Return a tuple representing the dimensionality of the Dataset.
<code>dataset.dtypes</code>	Return the dtypes in the Dataset.

Artificial Intelligence & Data Science

		<p>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.</p>
	dataset.columns.values	<p>Return the columns values in the Dataset in array format</p>
	dataset.describe(include='all')	<p>Generate descriptive statistics.</p> <p>to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values.</p> <p>Analyzes both numeric and object series, as well as Dataset column sets of mixed data types.</p>
	dataset['Column name']	<p>Read the Data Column wise.</p>
	dataset.sort_index(axis=1, ascending=False)	<p>Sort object by labels (along an axis).</p>
	dataset.sort_values(by="col")	<p>Sort values by column name.</p>
	dataset.iloc[5]	<p>Purely integer-location based indexing for selection</p>
	dataset[0:3]	<p>Selecting via [], which slices the rows.</p>

Artificial Intelligence & Data Science

dataset.iloc[:n, :]	a subset of the first n rows of the original data
dataset.iloc[:, :n]	a subset of the first n columns of the original data
dataset.iloc[:m, :n]	a subset of the first m rows and the first n columns

Few Examples of iLoc to slice data for iris Dataset

Data Frame Function	Description	Output																		
dataset.iloc[3:5,0:2]	Slice the data	<table><tr><th></th><th>Id</th><th>SepalLengthCm</th></tr><tr><td>3</td><td>4</td><td>4.6</td></tr><tr><td>4</td><td>5</td><td>5.0</td></tr></table>		Id	SepalLengthCm	3	4	4.6	4	5	5.0									
	Id	SepalLengthCm																		
3	4	4.6																		
4	5	5.0																		
dataset.iloc[[1,2,4], [0, 2]]	By list integer position locations, similar to the NumPy/Python style:	<table><tr><th></th><th>Id</th><th>SepalWidthCm</th></tr><tr><td>1</td><td>2</td><td>3.0</td></tr><tr><td>2</td><td>3</td><td>3.2</td></tr><tr><td>4</td><td>5</td><td>3.6</td></tr></table>		Id	SepalWidthCm	1	2	3.0	2	3	3.2	4	5	3.6						
	Id	SepalWidthCm																		
1	2	3.0																		
2	3	3.2																		
4	5	3.6																		
dataset.iloc[1:3,:]	For slicing rows explicitly:	<table><tr><th></th><th>Id</th><th>SepalLengthCm</th><th>SepalWidthCm</th><th>PetalLengthCm</th><th>PetalWidthCm</th></tr><tr><td>1</td><td>2</td><td>4.9</td><td>3.0</td><td>1.4</td><td>0.2</td></tr><tr><td>2</td><td>3</td><td>4.7</td><td>3.2</td><td>1.3</td><td>0.2</td></tr></table>		Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	1	2	4.9	3.0	1.4	0.2	2	3	4.7	3.2	1.3	0.2
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm															
1	2	4.9	3.0	1.4	0.2															
2	3	4.7	3.2	1.3	0.2															

Artificial Intelligence & Data Science

<code>dataset.iloc[:, 1:3]</code>	For slicing Column explicitly:	<table> <thead> <tr> <th></th><th>SepalLengthCm</th><th>SepalWidthCm</th></tr> </thead> <tbody> <tr> <td>0</td><td>5.1</td><td>3.5</td></tr> <tr> <td>1</td><td>4.9</td><td>3.0</td></tr> <tr> <td>2</td><td>4.7</td><td>3.2</td></tr> <tr> <td>3</td><td>4.6</td><td>3.1</td></tr> </tbody> </table>		SepalLengthCm	SepalWidthCm	0	5.1	3.5	1	4.9	3.0	2	4.7	3.2	3	4.6	3.1
	SepalLengthCm	SepalWidthCm															
0	5.1	3.5															
1	4.9	3.0															
2	4.7	3.2															
3	4.6	3.1															

	<code>dataset.iloc[1,1]</code>	For getting a value explicitly:	<div>4,5</div>																		
	<code>dataset['SepalLengthCm'].iloc[5]</code>	Accessing Column and Rows by position	<div>5,4</div>																		
	<code>cols_2_4=dataset.columns[2:4]</code> <code>dataset[cols_2_4]</code>	Get Column Name then get data from column	<table><thead><tr><th></th><th>SepalWidthCm</th><th>PetalLengthCm</th></tr></thead><tbody><tr><td>0</td><td>3.5</td><td>1.4</td></tr><tr><td>1</td><td>3.0</td><td>1.4</td></tr><tr><td>2</td><td>3.2</td><td>1.3</td></tr><tr><td>3</td><td>3.1</td><td>1.5</td></tr></tbody></table>		SepalWidthCm	PetalLengthCm	0	3.5	1.4	1	3.0	1.4	2	3.2	1.3	3	3.1	1.5			
	SepalWidthCm	PetalLengthCm																			
0	3.5	1.4																			
1	3.0	1.4																			
2	3.2	1.3																			
3	3.1	1.5																			
	<code>dataset[dataset.columns[2:4]].iloc[5:10]</code>	In one Expression answer for the above	<table><thead><tr><th></th><th>SepalWidthCm</th><th>PetalLengthCm</th></tr></thead><tbody><tr><td>5</td><td>3.9</td><td>1.7</td></tr><tr><td>6</td><td>3.4</td><td>1.4</td></tr><tr><td>7</td><td>3.4</td><td>1.5</td></tr><tr><td>8</td><td>2.9</td><td>1.4</td></tr><tr><td>9</td><td>3.1</td><td>1.5</td></tr></tbody></table>		SepalWidthCm	PetalLengthCm	5	3.9	1.7	6	3.4	1.4	7	3.4	1.5	8	2.9	1.4	9	3.1	1.5
	SepalWidthCm	PetalLengthCm																			
5	3.9	1.7																			
6	3.4	1.4																			
7	3.4	1.5																			
8	2.9	1.4																			
9	3.1	1.5																			

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:

Artificial Intelligence & Data Science

	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

a.is there any missing values in dataframe as a wholeFunction:

`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

b.is there any missing values across each column Function:

`DataFrame.isnull().any()`

Output:

Name	True
State	True
Gender	True
Score	True
dtype:	bool

a.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

b.count row wise missing value using `isnull()`

Function:

`dataframe.isnull().sum(axis = 1)` Output:

c.count Column wise missing value using `isnull()`

Method 1:Function: `dataframe.isnull().sum()`

Artificial Intelligence & Data Science

Output:

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

Method 2:

function: dataframe.isna().sum()

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

a.count of missing values of a specific column.

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

```
df1.Gender.isnull().sum()
```

Output: 2

a.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:

6.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.

Artificial Intelligence & Data Science

Functions used for data formatting

Sr. No	Data Frame Function	Description	Output
1.	df.dtypes	To check the data type	<pre>df.dtypes sepal length (cm) float64 sepal width (cm) float64 petal length (cm) float64 petal width (cm) float64 dtype: object</pre>
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)	<pre>df.dtypes sepal length (cm) float64 sepal width (cm) float64 petal length (cm) int64 petal width (cm) float64 dtype: object</pre>

a.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

```
iris = load_iris()
```

```
df =
```

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

Artificial Intelligence & Data Science

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
```

	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

Artificial Intelligence & Data Science

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

a.Categorical variables have values that **describe a ‘quality’ or ‘characteristic’**

of a data unit, like **‘what type’ or ‘which category’**.

b.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.**

c.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:

Artificial Intelligence & Data Science

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.

Artificial Intelligence & Data Science

```
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()
```

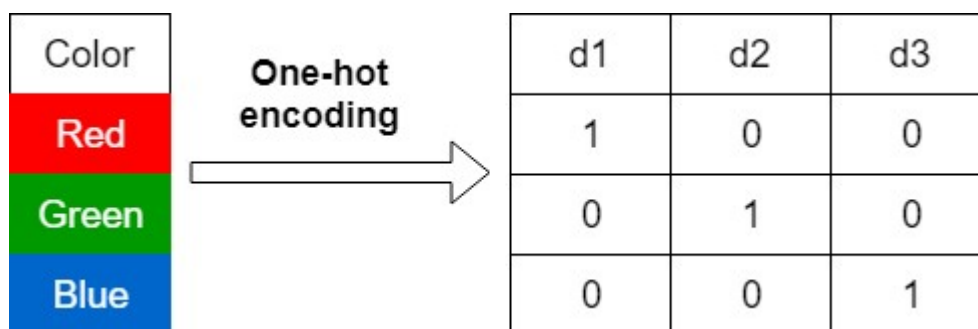
Output: `array([0, 1, 2], dtype=int64)`

Use LabelEncoder when there are only two possible values of a categorical feature. For example, features having value such as yes or no. Or, maybe, gender features when there are only two possible values including male or female.

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.

a. 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.



In one-hot encoding,

Artificial Intelligence & Data Science

“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

Artificial Intelligence & Data Science

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 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

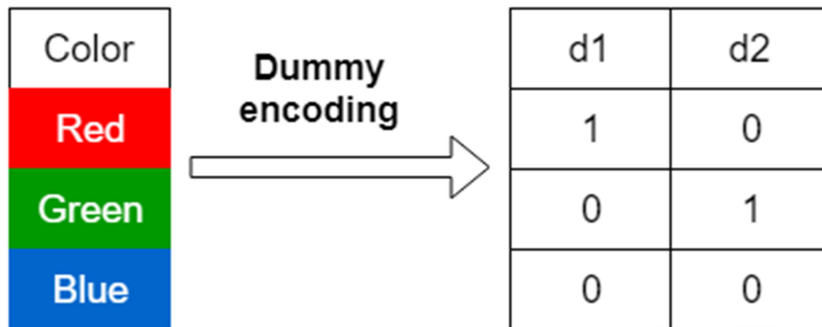
Output step-10

	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

a.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.

Artificial Intelligence & Data Science



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 NoneString 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_na: bool, default False

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

Artificial Intelligence & Data Science

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 NumPyarray (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)
```

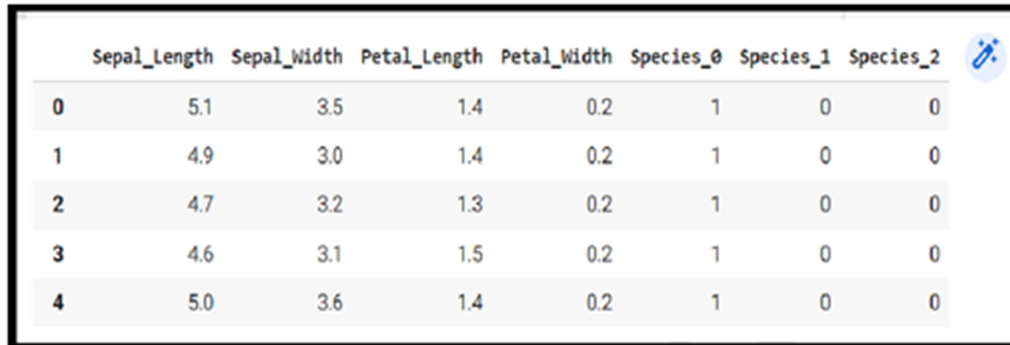
Step 6: Apply one_hot encoder with dummy variables for Species column.

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

Step 7: Observe the merge dataframe

```
one_hot_df
```

Artificial Intelligence & Data Science



	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species_0	Species_1	Species_2
0	5.1	3.5	1.4	0.2	1	0	0
1	4.9	3.0	1.4	0.2	1	0	0
2	4.7	3.2	1.3	0.2	1	0	0
3	4.6	3.1	1.5	0.2	1	0	0
4	5.0	3.6	1.4	0.2	1	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.What is the need of data normalization?
- 4.What are the different Techniques for Handling the Missing Data?