Fundamentals of Machine Learning

Lab Assignment – 8

Name: Niranjana J
USN: 22BTRAD027
Branch: CSE- AI & DE

Implementing Pandas

Pandas is a data manipulation and analysis library present in python. It is used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Pandas library can be installed into a machine using the “pip install pandas” command in anaconda prompt or command prompt. It is an open source library. Given that Pandas is based on the Numpy package, Numpy should function with Pandas.

Pandas has two main data structures including series and dataframe. A pandas series has a 1-d array like structure while data frame resembles a data table. Main features of pandas include indexing & slicing, grouping & aggregation, data manipulation and analysis, reading & writing operations etc.

To make use of the pandas library, first we have to import it.
In the following code, a pandas series is created and printed out.

```
[ ]  import pandas as pd

     # Creating a Pandas Series
     series_data = pd.Series([10, 20, 30, 40])
     print(series_data)
```

```
0    10
1    20
2    30
3    40
dtype: int64
```

Along with storing the values in the series, index of the values can also be passed accordingly.

The elements stored in the series can be retrieved using the indices just like python lists/ arrays.

```
import pandas as pd
x = pd.Series([1,2,3],index = ['a','b','c'])
#retrieve the first element
print(x)
print ('first element=',x[0])
```

```
a    1
b    2
c    3
dtype: int64
first element= 1
```

Now, a pandas dataframe is created and values are stored.

```
# Creating a Pandas DataFrame
data = {'Name': ['Wohn', 'Son', 'Pyung'],
        'Age': [25, 30, 22]}

df = pd.DataFrame(data)
print(df)
```

```
    Name  Age
0   Wohn   25
1    Son   30
2  Pyung   22
```

To select a certain column, the below code can be used. Here, the age column is retrieved in the first line of code. In the second part, the details of people with age less than 30 is retrieved.

```
# Selecting a column by label
ages = df['Age']

# Selecting rows based on a condition
young_people = df[df['Age'] < 30]
```

When the dataset if quite large and with a lot of missing values, it takes up a lot of time and energy to go through each record and correct it. To make the data cleaning process easier, dropna() function and fillna() function is used. While dropna removes the rows or columns that contain missing values, fillna fills missing values (NaN or None) with specified values.

```
[ ]  # Handling missing values
     df.dropna()  # Drop rows with missing values
     df.fillna(0)  # Fill missing values with a specified value
```

Pandas can also be used to transform the data. Here the age of all the people are incremented by 1 and displayed. This makes the updating process easier.

```
# Applying transformations
df['Age'] = df['Age'] + 1  # Incrementing ages by 1
print(df)
```

```
    Name  Age
0   Wohn   26
1    Son   31
2  Pyung   23
```

Grouping: To group the data according to the 'Name' column, we are utilizing the groupby() method from the pandas package. This separates the dataset's unique names into their own categories.

Aggregation: After the data has been aggregated, we use the mean() method to get the mean of the 'Age' column for each group. This determines the mean age of every distinct name.

```
[31] # Grouping by a column and calculating the mean
     average_age_by_name = df.groupby('Name')['Age'].mean()
     print(average_age_by_name)
```

```
Name
Pyung    23.0
Son      31.0
Wohn     26.0
Name: Age, dtype: float64
```

Other functions include:

Series.index	Defines the index of the Series.
Series.shape	It returns a tuple of shape of the data.
Series.dtype	It returns the data type of the data.
Series.size	It returns the size of the data.
Series.empty	It returns True if Series object is empty, otherwise returns false.
Series.hasnans	It returns True if there are any NaN values, otherwise returns false.
Series.nbytes	It returns the number of bytes in the data.
Series.ndim	It returns the number of dimensions in the data.
Series.itemsize	It returns the size of the datatype of item.

The values from two series with a similar column can be mapped using the map() function. The last column of the first series must match the index column of the second series in order to map the two series, and the values must be distinct.

```
import pandas as pd
import numpy as np
a = pd.Series(['Java', 'C', 'C++', np.nan])
a.map({'Java': 'Core'})
```

```
0    Core
1     NaN
2     NaN
3     NaN
dtype: object
```

External files such as csv files can also be read, analysed and manipulated using pandas.

For instance the iris_csv file is loaded into the code and wrote into a new csv file named new_iris. The .head() function is used to display the top 5 rows of the dataset.

```
# Reading data from a CSV file
df = pd.read_csv('iris_csv.csv')

# Writing data to a CSV file
df.to_csv('new_iris.csv', index=False)
print(df.head())
```

```
   sepallength  sepalwidth  petallength  petalwidth        class
0          5.1         3.5          1.4         0.2  Iris-setosa
1          4.9         3.0          1.4         0.2  Iris-setosa
2          4.7         3.2          1.3         0.2  Iris-setosa
3          4.6         3.1          1.5         0.2  Iris-setosa
4          5.0         3.6          1.4         0.2  Iris-setosa
```

Github link:
https://github.com/niranjana628/ML