

MACHINE LEARNING

Pandas

Trainer : Sujata Mohite
Email: sujata.mohite@sunbeaminfo.com



Pandas

- package developed on top of numpy
- provides functionality to load and process the data to make it ready for machine learning algorithms
- open source and free



Installation

- to install the pandas package on linux/macOS

- `sudo pip3 install pandas`

- to install the pandas package on windows

- `pip install pandas`



Usage

- to import pandas

- `import pandas as pd`



Series

- Series -one dimensional array
- one dimensional array
- uses index and values arrays to store the values
- uses numpy array behind the scene
- e.g.

```
▪ s1 = pd.Series([10, 20, 30, 40, 50])  
▪ print(s1)
```

```
▪ # 0    10  
▪ # 1    20  
▪ # 2    30  
▪ # 3    40  
▪ # 4    50  
▪ # dtype = int64
```

```
s2 = pd.Series([10, 20, 30],  
index=["value1", "value2", "value3"])  
print(s2)
```

```
# value1    10  
# value2    20  
# value3    30  
# dtype = int64
```



Creating series using different collections

- # create series using list
- `s1 = pd.Series([10, 20, 30, 40, 50])`

- # create series using dictionary
- `person = {"name": "person1", "age": 30, "address": "pune"}`
- `s3 = pd.Series(person)`

- # not possible to create series using set
- # as the set is an unordered collection

- # create series using ndarray
- `a1 = np.array([10, 20, 30, 40, 50])`
- `s5 = pd.Series(a1)`



Data Frame

- multi dimensional array
- tabular representation of the data
- collection of multiple series objects [merging multiple series creates a data frame]
- every data frame has horizontal rows and vertical columns
- every column represents a series
- axis
 - 0: rows
 - 1: columns



Data Frame

- # create data frame using one dimensional array
- # data frame with ndim = (5, 1) =(rows,cols)
- `df1 = pd.DataFrame([10, 20, 30, 40, 50])`
- | # | 0 |
|-----|----|
| # 0 | 10 |
| # 1 | 20 |
| # 2 | 30 |
| # 3 | 40 |
| # 4 | 50 |
- # create data frame using multi-dimensional array
- # data frame with ndim = (2, 5)
- `df2 = pd.DataFrame([[10, 20, 30, 40, 50], [60, 70, 80, 90, 100]])`
- | # | 0 | 1 | 2 | 3 | 4 |
|-----|----|----|----|----|-----|
| # 0 | 10 | 20 | 30 | 40 | 50 |
| # 1 | 60 | 70 | 80 | 90 | 100 |



Attributes - columns

- returns the list of columns in the data frame
- e.g.

```
df = pd.DataFrame([{"name": "person1", "age": 30}, {"name": "person2", "age": 40}])  
print(df.columns)                # ['name', 'age']
```



Creating Data Frames

- Creating Empty Data frame

```
import pandas as pd
df = pd.DataFrame()
print (df)
```
- Creating Dataframe using list

```
import pandas as pd
data = [1,2,3,4,5]
df = pd.DataFrame(data)
print (df)
```
- Creating Dataframe by giving column names

```
import pandas as pd
data = [['stud1',10],['stud2',12],['stud3',13]]
df = pd.DataFrame(data,columns=['Name','Age'],dtype=float)
print(df)
```



Reading data using pandas

- #Read csv file
- `df = pd.read_csv("filename.csv")`



Functions

- `head()`
 - by default returns first 5 rows
 - `head(10)` → returns first 10 rows
- `tail()`
 - by default returns last 5 rows
 - `tail()` → returns last 10 rows
- `info()`
 - returns the information about the data frame
 - Includes
 - number of columns
 - data type of columns
 - memory usage
 - number of entries (rows)



Functions

- describe()
 - returns statistical information about the data frame
 - includes
 - number of records
 - different values like
 - mean
 - minimum
 - maximum
 - standard deviation
 - percentile values



Functions

- `isna()`
 - return if the NaN values are present in the data frame
- `dropna()`
 - removes the NaN records
 - it removes entire row even if one of the columns is having NaN value
 - e.g
 - # get the another df wihtout having NaN values
`df_clean = df.dropna()`
 - # modify the original df object
`df.dropna(inplace=True)`



Functions

- `drop()`
 - used to remove one or more columns from the data frame
 - e.g.

```
df = pd.read_csv('./titanic.csv')
```

```
# removing a column
```

```
df_new = df.drop('home.dest', axis=1)  
print(df_new.columns)
```

```
# removing a column
```

```
df.drop('home.dest', axis=1, inplace=True)  
print(df.columns)
```

```
# removing multiple columns
```

```
df.drop(['home.dest', 'name', 'parch', 'embarked'], axis=1, inplace=True)
```



Thank You!!

