Kmit

SONET

Data Science

(Level-I)

Ajeet K. Jain, M. Narsimlu

(ML TEAM)- SONET, KMIT, Hyderabad

Session - 21

This session deals with

Pandas Methods

Exercises on Pandas

Control statements on Dataset

Exploration of Data Analysis

Data Preprocessing

Methods

head() : To display top five records in the dataset

tail() : To display last five records in the dataset

isnull(): Detect missing values.

fillna([]):Fill NA/NaN values using the specified method

dropna():Remove missing values.

info()-It displays general information about the dataframe

astype() -It can change the column type

DataFrame

Methods :

describe() - statistical characteristics of each numerical feature such as number of non-missing values, mean, standard deviation, range, median, 0.25 and 0.75 quartiles.

isnull().sum(): Detect missing values in each feature.

Append(): adding features to data set.

DataFrame

```
In [63]: import pandas as pd
data={'one':[1,2,3],'two':[None,5,6],'three':[7,8,9]}
df=pd.DataFrame(data)
print(df.isnull())
print( df.fillna((df.mean())))
print(df.dropna())
print(df.append(df1))
```

```
     one    two  three
0  False   True  False
1  False  False  False
2  False  False  False
   one  two  three
0    1  5.5      7
1    2  5.0      8
2    3  6.0      9
   one  two  three
1    2  5.0      8
2    3  6.0      9
   five  four  one  six  three  two
0   NaN   NaN  1.0  NaN    7.0  NaN
1   NaN   NaN  2.0  NaN    8.0  5.0
2   NaN   NaN  3.0  NaN    9.0  6.0
0   NaN   1.0  NaN  7.0    NaN  NaN
1   5.0   2.0  NaN  7.0    NaN  NaN
2   5.0   3.0  NaN  8.0    NaN  NaN
```

Methods

The Pandas I/O API is a set of top level reader functions accessed like pd.read_csv() that generally return a Pandas object.

```
import pandas as pd
data_loan=pd.read_csv("E:\\KMIT\\SONET\\NPTEL_Python_DS\\datasets\\loan.csv")
df_loan=pd.DataFrame(data_loan)
print(df_loan.head())
print(df_loan.tail())
print(df_loan.isnull())
print(df_loan.isnull().sum())
df_loan["Credit Score"].fillna(df_loan["Credit Score"].mean(),inplace=True)
print(df_loan["Credit Score"].head())
```

Methods

```
import pandas as pd
data_loan=pd.read_csv("E:\\KMIT\\SONET\\NPTEL_Python_DS\\datasets\\loan.csv")
df_loan=pd.DataFrame(data_loan)
df_loan.dropna(inplace=True)
print(df_loan.info())
print(df_loan.describe())
print(df_loan.dtypes)
df_cat=df_loan.select_dtypes(include=['object']).copy()
print(df_loan["Term"].head())
df_loan["Term"]=df_loan["Term"].astype("category")
print(df_loan["Term"].head())
```

Indexing and selecting Data

Indexing	Description
.loc()	Label based
.iloc()	Integer based

```
In [11]: import pandas as pd
         import numpy as np

         df = pd.DataFrame(np.random.randn(4, 4),
         index = ['a','b','c','d'], columns = ['A', 'B', 'C', 'D'])
         print(df)
         print(df.loc[:,'A'])
         print(df.iloc[:,1])
         print(df.ix[:,'A'])
```

```
          A         B         C         D
a -0.903447  0.349539 -2.079854  2.141442
b  0.487952  0.512143  1.788810  1.288846
c  0.136634  0.674832 -1.900180 -0.828820
d  0.134507 -0.458892  0.667469  1.074890
a   -0.903447
b    0.487952
c    0.136634
d    0.134507
Name: A, dtype: float64
a    0.349539
b    0.512143
c    0.674832
d   -0.458892
Name: B, dtype: float64
a   -0.903447
b    0.487952
c    0.136634
d    0.134507
Name: A, dtype: float64
```

Control Structures

Conditional Statement in Python perform different computations or actions depending on whether a specific Boolean constraint evaluates to true or false.

while loop Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.

for loop Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

Exercise-1

1.Read Toyota cars data and add a new feature to Dataframe as Price_Class
based on price value conditions:
a. if price<=8450 then add "Low"
b. if price>11950 then add "High"
c. otherwise add "Medium“
Use for Loop to add each row value to column

Solution

```
import pandas as pd
data_cars=pd.read_csv("ToyotaCorolla.csv")
print(data_cars)
data_cars.insert(10,"Price_Class","")
for i in range(0,len(data_cars["Price"]),1):
    if(data_cars["Price"][i]<=8450):
        data_cars["Price_Class"][i]="Low"
    elif(data_cars["Price"][i]>11950):
        data_cars["Price_Class"][i]="High"
    else:
        data_cars["Price_Class"][i]="Medium"
print(data_cars["Price_Class"].value_counts())
print(data_cars.columns)
```

Output

```
Medium     751
Low        369
High       316
Name: Price_Class, dtype: int64
Index(['Id', 'Model', 'Price', 'Age_08_04', 'Mfg_Month',
'Mfg_Year', 'KM',
       'Fuel_Type', 'HP', 'Met_Color', 'Price_Class',
'Automatic', 'cc',
       'Doors', 'Cylinders', 'Gears', 'Quarterly_Tax',
'Weight',
       'Mfr_Guarantee', 'BOVAG_Guarantee',
'Guarantee_Period', 'ABS',
       'Airbag_1', 'Airbag_2', 'Airco', 'Automatic_airco',
'Boardcomputer',
       'CD_Player', 'Central_Lock', 'Powered_Windows',
```

Exercise-2

1.Read Toyota cars data and add a new feature to Data frame as Price_Class
based on price value conditions:
a. if price<=8450 then add "Low"
b. if price>11950 then add "High"
c. otherwise add "Medium“
Use While Loop to add each row value to column

Solution


```
import pandas as pd
data_cars=pd.read_csv("ToyotaCorolla.csv")
data_cars.insert(10,"Price_Class","")
i=0
while(i<len(data_cars["Price"])):
    if(data_cars["Price"][i]<=8450):
        data_cars["Price_Class"][i]="Low"
    elif(data_cars["Price"][i]>11950):
        data_cars["Price_Class"][i]="High"
    else:
        data_cars["Price_Class"][i]="Medium"
    i=i+1
print(data_cars["Price_Class"].value_counts())
print(data_cars.columns)
```

Output

```
Medium     751
Low        369
High       316
Name: Price_Class, dtype: int64
Index(['Id', 'Model', 'Price', 'Age_08_04', 'Mfg_Month',
'Mfg_Year', 'KM',
       'Fuel_Type', 'HP', 'Met_Color', 'Price_Class',
'Automatic', 'cc',
       'Doors', 'Cylinders', 'Gears', 'Quarterly_Tax',
'Weight',
       'Mfr_Guarantee', 'BOVAG_Guarantee',
'Guarantee_Period', 'ABS',
       'Airbag_1', 'Airbag_2', 'Airco', 'Automatic_airco',
'Boardcomputer',
       'CD_Player', 'Central_Lock', 'Powered_Windows',
```

Exploration of Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyzing datasets to summarize their main characteristics.

It is used to understand data, get some context regarding it, understand the variables and the relationships between them

It is also useful in formulate hypotheses that could be useful when building predictive models.

The objective is to group records in your data set that have similar categorical attributes and then perform some calculation (count, sum, mean, etc.)

Exploration of Data Analysis

Exploratory Data Analysis (EDA) is actually getting in there, exploring the data, and discovering insights.

One of the fundamental ways to extract insights from a data set is to reduce the size of the data so that you can look at just a piece of it at a time.

There are two ways to do this: *filtering* and *aggregating*.

Essentially removing either rows or columns (or both rows and columns) in order to focus on a subset of the data that interests.

Exploration of Data Analysis

Analyzing the relation between feature variables

Frequency Table

Two-Way tables

Two way table – Marginal Probability

Correlation

One Way Table

Create frequency tables (also known as crosstabs) in pandas using the pd.crosstab() function.

The function takes one or more array-like objects as indexes or columns and then constructs a new DataFrame of variable counts based on the supplied arrays.

Exercise-01

create a frequency table from farms dataset on bimas feature which displays index(it also displays each category count) and columns

```
import pandas as pd
data_farms=pd.read_csv("farms.csv")
bimas_tab = pd.crosstab(index=data_farms["bimas"],
                        columns="count")
print(bimas_tab)
```

Output

```
col_0   count
bimas
mixed     162
no        779
yes        85
```

Exercise -2

Let's make a couple more crosstabs to explore other variables: such as status feature and varieties features

```
import pandas as pd
data_farms=pd.read_csv("farms.csv")
status_tab = pd.crosstab(index=data_farms["status"],
                         columns="count")
varieties_tab = pd.crosstab(index=data_farms["varieties"],
                         columns="count")
print(varieties_tab)
print(status_tab)
```

Output

```
col_0        count
varieties
high           294
mixed           50
trad           682
col_0    count
status
mixed      211
owner      736
share       79
```

One of the most useful aspects of frequency tables is that they allow you to extract the proportion of the data that belongs to each category.

Two Way Table

Two-way frequency tables, also called contingency tables, are tables of counts with two dimensions where each dimension is a different variable.

Two-way tables can give you insight into the relationship between two variables.

To create a two way table, pass two variables to the pd.crosstab() function instead of one

Exercise -3

DATA SCIENCE

create a frequency table between **status VS varieties** features, which shows the relation between two variables.

```
import pandas as pd
data_farms=pd.read_csv("farms.csv")
status_varie=pd.crosstab(index=data_farms["status"],
                         columns=data_farms["varieties"])
print(status_varie)
print("status wise varities of farming")
status_varie.index=["mixed","owner","share"]
print(status_varie)
```

Output

```
varieties  high  mixed  trad
status
mixed        33      7   171
owner       227     41   468
share        34      2    43
status wise varities of farming
varieties  high  mixed  trad
mixed        33      7   171
owner       227     41   468
share        34      2    43
```

Exercise -4

create a frequency table between status VS bimas features,
Perform the following tasks
1.find the indexes in the data frame
2.find the columns in the data frame
3.find the marginal counts (totals for each row and column) by including the argument margins=True

Solution

```
import pandas as pd
data_farms=pd.read_csv("farms.csv")
status_bimas=pd.crosstab(index=data_farms["status"],
                         columns=data_farms["bimas"],margins=True)
print(status_bimas.index)
print(status_bimas.columns)
status_bimas.columns=['mixed', 'no', 'yes',"rowtotal"]
status_bimas.index=['mixed', 'owner', 'share',"coltotal"]
print(status_bimas)
```

Output

```
Index(['mixed', 'owner', 'share', 'All'], dtype='object',
name='status')
Index(['mixed', 'no', 'yes', 'All'], dtype='object', name='bimas')
          mixed   no  yes  rowtotal
mixed        36  146   29       211
owner       119  564   53       736
share         7   69    3        79
coltotal    162  779   85      1026
```

Higher Dimensional Tables

The crosstab() function lets you create tables out of more than two categories.

Higher dimensional tables can be a little confusing to look at, but they can also yield

finer-grained insight into interactions between multiple variables.

Exercise -5

Let's create a 3-way table inspecting status, bimas and region on farming dataset

```
import pandas as pd
data_farms=pd.read_csv("farms.csv")
status_varie_bim=pd.crosstab(index=data_farms["status"],
                             columns=[data_farms["varieties"],
                                      data_farms["bimas"]],
                             margins=True)
print(status_varie_bim)
```

Conclusion

You are aware of

Pandas

EDA

We will proceed with

Data Visualization

Kmit

SONET

DATA SCIENCE

THANK YOU