Pandas

Mostly used in preprocessing, cleaning and analysis in machine learning

Reading Files

```
File= pd.read_csv('file.csv')
#Fields separatred with white-space
File= pd.read_csv('file.data',sep=r'\s+')
File['column_name']   # retrive all observations for that Column
```

- `sep=r'\s+'` is a regular expression --> more flexible when fields are separated with other characters

Data Frame Operations

Command	Description
`File.shape`	Will return (NumObservations, NumVariables)
`File.dropna`	Removes rows with missing observations
`File.columns`	Will list all the Variables ,columns names

Data Frame indexing

- Usually When we load our Data Frame its indexed using integers (Number of Observations)

```
File_reindex = File.set_index('col_name')
```

- Now the rows are indexed With names
- The column used to index will be removed , `File.columns` to check

- Now we can access rows of the data frame by the `col-name` we indexed by earlier, Using `loc`

```
File_reindex = File.set_index('name')
observations = ['Hady','Hadi']
```

```
File_reindex.loc[observations]
File_reindex.loc['Hady',['grade','job']]
```

- The method `loc` used to retrieve data using labels such as : columns_names
  Inclusive --> the end range is included
- The method `iloc` used to retrieve data using integers
  exclusive --> the end range isn't included (just like python slices)

```
File.iloc[[3,4]] #Selecting the Forth and the Fifith Row
File.iloc[:,[0,2,3]] #Selecting the 1st,3th,4th columns
File.iloc[[3,4],[0,2,3]] #Combining both of em
```

- A condition can be added before using the two methods `loc` , `iloc`

```
index_15 = File['grade']>15 #return  [True, False ,False, True]
foreach observation
File_reindex.loc[index_15,['name','last-name']]
```

- `index_15` is a Boolean array , That get passed in the `loc` method to retrieve depending on the set Boolean array

- A better way to do it is using `lambda` function

```
File.loc[lambda df: df['grade']>15.['name','last-name']]
File.loc[lambda df: (df['grade']>15)&(df['field']="CS"|
df['field']="math"),
['name','last-name']
]
```

- The `lambda` function can include multiple --> Complex Conditions (and ,or,xor..)

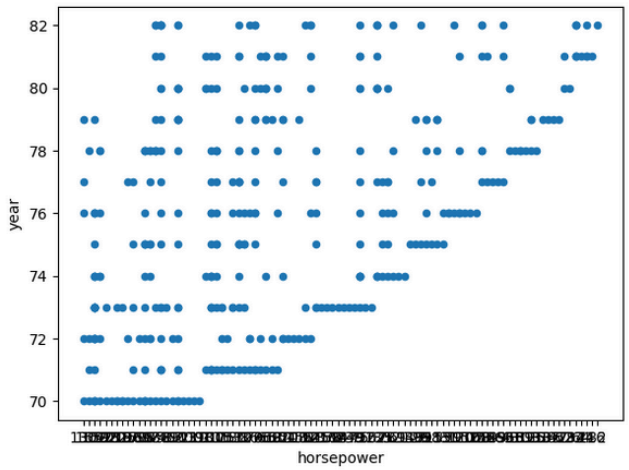
  If we index's the `names` field it can also be used to Select the needed data easily

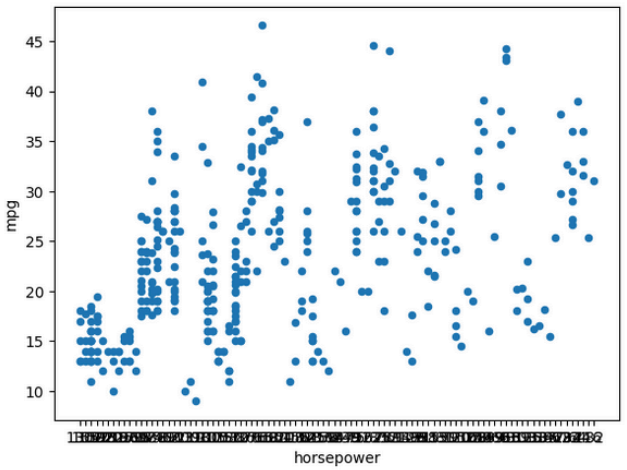
```
File_reindex.loc[lambda df: (df.index.str.contains('Hady'))&
(df.index.str.contains('Hady')), ['garde','birthday']]
```

Data Frame Plotting

- Pandas Relies on Matplotlib for plotting Data Frames and visualize them

```
fig, axes = plt.subplots(ncols=2 ,figsize=(8,8))
File.plot.scatter('horsepower','mpg', ax=axes[1])
File.plot.scatter('horsepower','year',ax=axes[0])
```



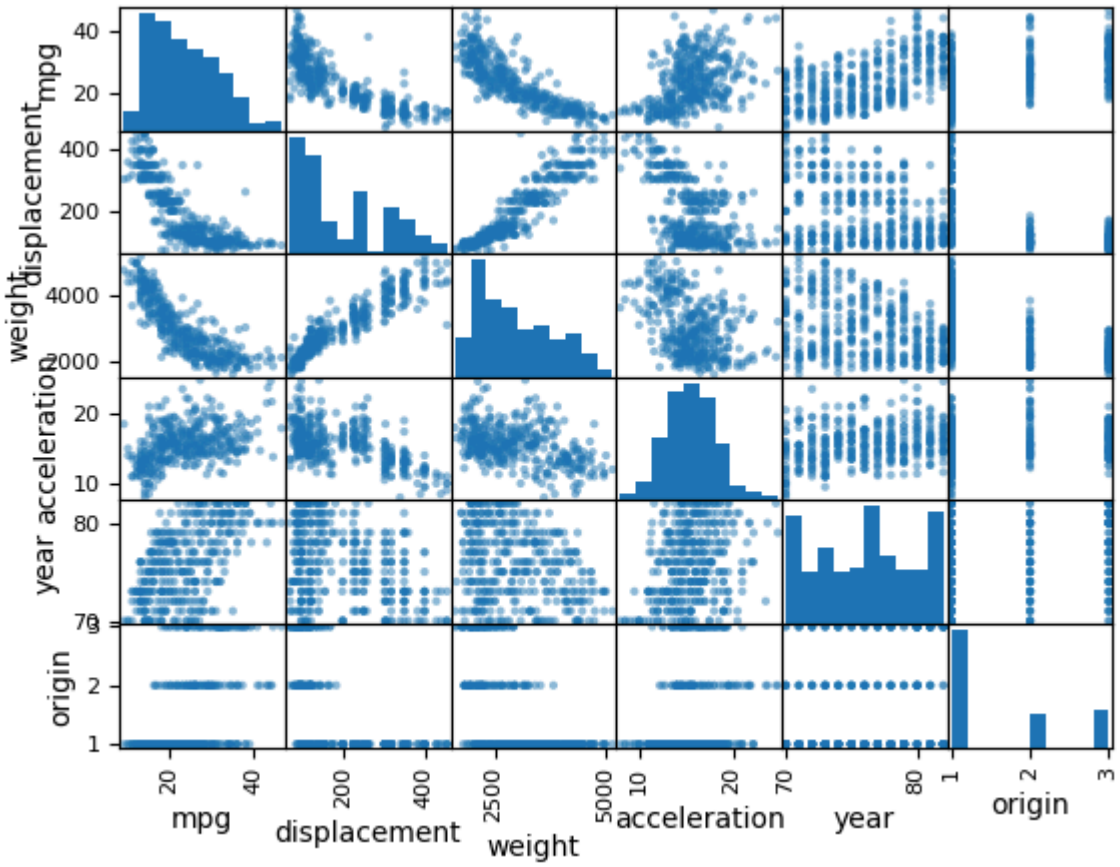


- Sometimes a quantitative variable `column` only have a small of possible numbers that it can be treated as a qualitative value and be categorize Using `pd.Series()`:

```
File.ages = pd.Series(File.ages,dtype='category')
File.ages.dtype # return dtype: category
```

- We can use Pandas to plot a matrix of the relationships between the columns of the Data Frame :

```
pd.ploting.scatter_matrix(File)
```



- Another helpful function in Pandas is `describe()` method it will list --> count, mean, std..etc of that said variable `column`

```
File['col_name1','col_name2'].describe()
```

Ref :[Introduction to Statistical Learning Book]

tags: #islp #machine-learning #numpy #matplotlib