Indexing and Selection

This lesson will focus on how to view, add, and rename columns of a Pandas dataframe.

WE'LL COVER THE FOLLOWING

Indexing is the technique of efficiently retrieving records from data based on some criteria that the data has been arranged by. As we saw in the previous lesson, the data is organized in rows and columns in a dataframe. So, we can index data using the positions and names of these rows and columns. Now let's see how to select rows and columns from the data.

Columns

To view the names of the columns we use `df.columns.values`. We will be using the file *housing.csv*.

housing.csv

> In Machine Learning terminology, a column in a spreadsheet is referred to as a **feature**, while in Statistics it is referred to as a **variable**. It is also referred to as an `attribute`. We will be using all of these terms interchangeably in this course.

```
import pandas as pd
df = pd.read_csv('housing.csv')

# Print Column names
print(df.columns.values)

# Number of columns
num = len(df.columns)
print("number of columns: ",num)
```

Selecting columns

Let's see how we can select the data of a few columns of *housing.csv*.

```
import pandas as pd
df = pd.read_csv('housing.csv')

new_df = df['population']
print(new_df.head())

# Make a list of the columns to select
col_to_select = ['longitude','latitude','population', 'ocean_proximity']

# Collect the specified columns and print them
new_df = df[col_to_select]
print('\n\n',new_df.head())
```

We view the values in a column by simply typing their name as we did in **line 4**. In **line 8**, we create a list of columns that we want to select. In **line 11**, we retrieve those columns out of the dataframe `df` and save it as a new dataframe called `new_df`. **Line 12** prints the `head` of the dataframe.

Changing column names

If we want to change the column names of all the columns or add column names that are not already present in the CSV file, we have to provide a list of column names.

We can also change specific column names as well using the `rename` function. Let's see an example of these

Let's see an example of these.

```
import pandas as pd
df = pd.read_csv('housing.csv')

# change or add column names
df.columns = ['long' , 'lat' ,'housing_med_age', 'total_rooms',
 'total_bedrooms', 'pop', 'households', 'med_income',
 'med_house_value' , 'ocean_proximity']

print(df.columns.values,'\n')

# rename speific column names
df.rename(columns = {'pop':'population',
                     'long':'longitude'},inplace=True)

print(df.columns.values,'\n')

print(df.head())
```

In **line 5**, we add the column names by providing a list of column names. This modifies the names of all columns to what we provide.

In **line 12**, we rename only two specific columns by providing a dictionary with old names and new names. We change the names of two columns, `pop` and `long` to `population` and `longitude` respectively.

Rows

We can select the rows we want by specifying their positions in the dataframe.

```
import pandas as pd
df = pd.read_csv('housing.csv')

# specify rows to select
rows_to_select = [0,2,5,8]
new_df = df.loc[rows_to_select]
print(new_df,'\n')

# specify a range of rows directly
new_df = df.loc[3:7]
print(new_df)
```

In **line 5**, we have specified the rows to select and in the next line, we have retrieved those rows using the `loc` keyword. We can provide the indices of rows to `loc` to get the specified rows.

We can also specify a range of rows to retrieve directly, as done in **line 10**. This gives us consecutive rows from the 4th row to the 7th row, i.e., rows with indices 3, 4, 5 and 6. Remember that the row and column numbers start from 0 in Python.

> A single row is sometimes called a **record**.

Indexing both rows and columns

We can specify rows and columns both to select a subset of the dataframe. Let's look at an example.

```
import pandas as pd
df = pd.read_csv('housing.csv')

# specify the columns
col_to_select = ['longitude','latitude','population']

# pass the rows and columns
new_df = df.loc[0:5,col_to_select]

print(new_df.head())
```

In **line 5**, we have specified the columns that we want. In **line 8** we specify both rows and columns to `loc`. We have written `0:5` to specify that we want the first 5 rows and then a comma, followed by the list of the column names that we want. Then we print the head of the new dataframe in **line 10**.

Setting values

We can set or replace a value in a cell using `loc`. We select the cell by providing the row index and the column name and then provide the new value.

```
import pandas as pd
```

```
import pandas as pd
df = pd.read_csv('housing.csv')

# pass the rows and columns
df.loc[0,'longitude'] = 0

print(df.head())
```

From the output of **line** 7, it can be verified that the value of `longitude` for the first row has been changed to 0 as specified in **line 5**.

Series vs Dataframe

At this point, it is important to differentiate between a *series* and a *dataframe* in pandas. A **series** in pandas is like a one-dimensional array. It can hold data of a single type. Whereas a **dataframe** is like a two-dimensional array that can hold data of multiple types in the form of rows and columns. Rows, as well as columns, can be named in both series and dataframes. You can think of a series as a dataframe with only a single column.

We know that pandas stores a CSV file in a dataframe. But when we select a single column from a dataframe it gives us a series, and when we select multiple columns from a dataframe by giving a list of column names, we get a dataframe.

```
import pandas as pd
df = pd.read_csv('housing.csv')

# this will give a series
ser = df['population']
print(ser.head())

# this will give a dataframe
cols_to_select = ['longitude','latitude','population']
new_df = df[cols_to_select]
print(new_df.head())

# this will give a dataframe
cols_to_select = ['population']
new_df = df[cols_to_select]
print(new_df.head())
```

In **line 5**, we select the column `population` by directly providing `df` the name of the column. This gives us a *series* in `ser`.

In **line 10**, we select multiple columns by providing a list to `df`. This gives us a dataframe in `new_df`.

In **line 15**, we select a single column by providing the column name in a list to `df`. This gives us a dataframe in `new_df`.

The distinction between series and dataframe is necessary because some functionalities can only be used with series, while some can only be used with a dataframe as we will see in the future lessons.

Now that we know how to do indexing and selection, we will look at how to filter data in the next lesson.