Lesson 10, Pandas

Pandas

- Pandas is all about the DataFrame (DF)
  - As with the R data.frame
- DF has rows and columns
  - Like a table in RDBMS or Excel spreadsheet
- One call select and filter the DF
- DF columns are the variables to plot
- DF columns may be the ML predictors
  - Features -> labels
- DF may be read in or formed via dictionary

Dictionaries

- Immutable, as with tuples
- Key-value pairs
  - If key is a string, consistent with JSON
  - Value can be anything, often a list or other collection
- Very useful for defining data frames in pandas

Examples of dictionaries

```
d1=[{"name":"Tinker","age":23,"pos":"shortstop"},
  {"name":"Evers","age":25,"pos":"second"}]
print(f"{d1[0]['pos']} {d1[0]['name']} threw to
{d1[1]['name']}")
print(f"for the out at {d1[1]['pos']}.")
print(f"Their average age is
{(d1[0]['age']+d1[1]['age'])/2}")

dict=[{"id":1,"base":3, "height":4},{"id":2,"base":5,
"height":12}]
area0=dict[0]['base'] * dict[0]['height']
area1=dict[1]['base'] * dict[1]['height']
print(f"Those areas are {area0} and {area1}.")
```

Package Pandas in Python: DataFrame

- Built on NumPy so always import that
- Python’s Excel or R data frame
- Series, DataFrame
- Handles missing data
- Group by
- Merging, joining, concatenating
- Filter, select, mutate
- Operations
- Input & output

Series in Pandas: basis for data frames

```
import numpy as np
import pandas as pd
labels1=['abc','def','ghi']
list1=[5,10,15]
arr1=np.array(list1)
ser1=pd.Series(data=list1,index=labels1) # using Python list
ser2=pd.Series(data=arr1,index=labels1)  # using NumPy array
# ser2=pd.Series(arr1,labels1)    # note this simplification
print(ser2)                    # note dtype is int32 vs. int64
d1={"ab":10, "cd":20, "ef":30}            # dictionary
ser3=pd.Series(d1)   # key becomes index, value becomes data
print(ser3)
print(ser3['cd'])
```

DataFrames in Pandas

- Each DataFrame column is a Series
- Each DataFrame row has an index
- A DataFrame is a collection of Series that share indices
- Each DataFrame row is also a Series
- Index for rows, 0,1,2,etc. by default, but you can assign this
- Columns have names, too
  - Dictionary is convenient way to establish them

Make a DataFrame

```
import numpy as np
import pandas as pd
idx=['a','b','c','d','e','f','g','h','i']
x=np.arange(0,9)*np.math.pi/4
y=np.cos(x)
d2={'x': x, 'cosine': y}
df2=pd.DataFrame(d2,index=idx)
print(df2)
```

DataFrames part 1

```
import numpy as np
import pandas as pd
np.random.seed(42)      # for repeatability
row_labels=["Canada","USA","Mexico","Brazil","Peru"]
col_labels=["Pop","GDP","Feat1","Feat2"]
df1=pd.DataFrame(np.random.randn(5,4),row_labels,col_labels)
print(df1)
print(df1['GDP'])                       # gets the column
print(df1[['Pop','Feat1']])             # gets two columns
df1['Feat3'] = df1['Feat1'] + df1['Feat2'] # new column
df1.drop('GDP',axis=1,inplace=True)
df1.drop('USA',axis=0,inplace=True)
```

DataFrames part 2

```
print(df1.loc['Canada'])
print(df1.iloc[1])
print(df1.loc['Canada','Feat3'])
print(df1.iloc[0,3])
print(df1.loc[['Canada','Mexico'],['Pop','Feat1']])
print(df1)
print(df1 > 0)
print(df1[df1 > 0])
print(df1[df1['Pop'] > 0]) # only rows whose Pop col > 0
print(df1[df1['Pop'] > 0]['Feat3'])
print(df1[df1['Pop'] > 0][['Feat2','Feat3']])
print(df1[(df1['Feat2']>1) & (df1['Feat3']>2)])
```

Missing values

```
# Drop either rows or columns, fill with 0 or mean
import numpy as np
import pandas as pd
d3={'a':[1,2,np.nan],'b':[5,np.nan,np.nan],'c':[1,2,3]}
df3=pd.DataFrame(d3)
print(df3)
df3a=df3.dropna()        # axis=0 by default
df3b=df3.dropna(axis=1)
print(df3a)
print(df3b)
df3c=df3['a'].fillna(value=df3['a'].mean())
df3d=df3['a'].fillna(value=0)
print(df3c)
print(df3d)
```

25 Pandas tricks

- https://www.kdnuggets.com/2019/08/25-tricks-pandas.html

- Start demo next
  - pan25_2.py