Pytorch

Pandas

Create dataframe

```
import pandas as pd
import numpy as np

s = pd.Series([1,3,6,np.nan,44,1])

dates = pd.data_range('20200101',periods=6)

# Create a 6x4 dataframe
df = pd.DataFrame(np.random.randn(6,4),index=dates,column=['a','b','c','d'

# Use dict to create DataFrame
df2 = pd.DataFrame({'A' : 1.,
                    'B' : pd.Timestamp('20130102'),
                    'C' : pd.Series(1,index=list(range(4)),dtype='float32'
                    'D' : np.array([3] * 4,dtype='int32'),
                    'E' : pd.Categorical(["test","train","test","train"]),
                    'F' : 'foo'})
```

Attributes of DataFrame

```
df2.dtypes # return the type of DataFrame

df2.index  # return the row index of the DataFrame

df2.columns #return the column index of the DataFrame

df2.values # return the elements of the DataFrame

df2.describe() # return the info of the DataFrame

df2.T # return the transpose DataFrame

# axis=1 sort the DataFrame based on column
# ascending=False, Reverse order
df2.sort_index(axis=1,ascending=False))

# sort the DataFrame based on specific column
df2.sort_values(by='E')
```

DataFrame slicing

```
import pandas as pd
import numpy as np

dates = pd.date_range('20200101',periods=6)
df = pd.DataFrame(np.arange(24).reshape((6,4)),index=dates,columns=['A','B

# print column A
print(['A'])
print(df.A)

# print 0-3 lines
print(df[0:3])
print(df['20200101':'20200103'])
```

Select by label: loc

```
# print specific element by label
print(df.loc['20200101'])

# print all the element in A to C columns
print(df.loc[:,'A':'C'])

# print the element with '20200103' label and in A/B columns
print(df.loc['20200103',['A','B']])
```

Select by position: iloc

```
# print the element which in the 3 line and 1 column
print(df.iloc[3,1])

print(df.iloc[3:5,1:3])
```

Boolean indexing

```
# print the elment when it is larger than 8
print(df[df.A>8])
```

Assign values to elements

```
df.iloc[2,2] = 111

df.loc['20200102','A'] = 222

# When the element in column A is less than 4,
# change the corresponding element in column B to 0
df.B[df.A>4] = 0

# Add a new column all the elements are NaN
df['F'] = np.nan
df['E'] = [1,2,3,4,5,6]
```

Error value handling

```
df.iloc[0,1] = np.nan
df.iloc[1,2] = np.nan

# Delete line with NaN
df.dropna(axis=0,how='any')

df.dropna(axis=1)

# Use specific value to replace the NaN
df.fillna(value=0)

# Use True and False to replace the element in the matrix
# If the element is not NaN use False to replace it
# otherwise use True
df.isnull()

# Judge whether there is any NaN in the matrix
df.any(df.isnull()) ==True
```

Read and save

```
data = pd.read_csv('student.csv')

data.to_pickle('student.pickle')
```

Concatenate

```
df1 = pd.DataFrame(np.ones((3,4))*0,columns=['a','b','c','d'])
df2 = pd.DataFrame(np.ones((3,4))*1,columns=['a','b','c','d'])
df3 = pd.DataFrame(np.ones((3,4))*2,columns=['a','b','c','d'])

# axis=0 change the lines, ignore_index=True means use new index
res = pd.concat([df1,df2,df3],axis=0,ignore_index = True)



df1 = pd.DataFrame(np.ones((3,4))*0,columns=['a','b','c','d'],index=[1,2,3
df2 = pd.DataFrame(np.ones((3,4))*1,columns=['b','c','d','e'],index=[2,3,4

# The default setting for join is outer, which will use NaN to fill the lo
res = pd.concat([df1,df2])

# inner: only keep the same part of the two DataFrame
res = pd.concat([df1,df2],axis=0, join='inner',ignore_index=False)
```

Append

```
df1 = pd.DataFrame(np.ones((3,4))*0,columns=['a','b','c','d'],index=[1,2,3
df2 = pd.DataFrame(np.ones((3,4))*1,columns=['a','b','c','d'],index=[2,3,4
df3 = pd.DataFrame(np.ones((3,4))*2,columns=['a','b','c','d'],index=[2,3,4

df1.append([df2,df3],ignore_index=True)

s1=pd.Series([1,2,3,4],index=['a','b','c','d'])

res=df1.append(s1,ignore_index=True)
```

Merge

```
left = pd.DataFrame({'key':['K0','K1','K2','K3'],
                     'A':['A0','A1','A2','A3'],
                     'B':['B0','B1','B2','B3']})
right = pd.DataFrame({'key':['K0','K1','K2','K3'],
                     'C':['C0','C1','C2','C3'],
                     'D':['D0','D1','D2','D3']})

# merge left and right dataframe based on 'key' column

res = pd.merge(left,right,on='key')
```

how = ['left','right','inner','outer'], the default value of how = 'innner' : merge the same part

```
left = pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
                      'key2': ['K0', 'K1', 'K0', 'K1'],
                      'A': ['A0', 'A1', 'A2', 'A3'],
                      'B': ['B0', 'B1', 'B2', 'B3']})
right = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                       'key2': ['K0', 'K0', 'K0', 'K0'],
                       'C': ['C0', 'C1', 'C2', 'C3'],
                       'D': ['D0', 'D1', 'D2', 'D3']})
res = pd.merge(left,right,how='inner',on=['key1','key2'])



# how = 'outer' merge all part use NaN to fill the missing part

res = pd.merge(left,right,how='outer',on=['key1','key2'])

# how = 'right' merge two DataFrame based on the right key,
# use NaN to fill the missing part

res = pd.merge(left,right,how='right',on=['key1','key2'])
```

Indicator: show how to merge

```
# Indicator: show how to merge

```

```
df1 = pd.DataFrame({'col1':[0,1], 'col_left':['a','b']})
df2 = pd.DataFrame({'col1':[1,2,2],'col_right':[2,2,2]})

res = pd.merge(df1,df2,on='col1',how='outer',indicator=True)
```

Index

```
left = pd.DataFrame({'A': ['A0', 'A1', 'A2'],
                     'B': ['B0', 'B1', 'B2']},
                     index=['K0', 'K1', 'K2'])
right = pd.DataFrame({'C': ['C0', 'C2', 'C3'],
                      'D': ['D0', 'D2', 'D3']},
                     index=['K0', 'K2', 'K3'])

res=pd.merge(left,right,left_index=True,right_index=True,how='outer')
```

solve overlapping problem

```
boys = pd.DataFrame({'k': ['K0', 'K1', 'K2'], 'age': [1, 2, 3]})
girls = pd.DataFrame({'k': ['K0', 'K0', 'K3'], 'age': [4, 5, 6]})

print(boys)
print(girls)
res = pd.merge(boys,girls,on='k',suffixes=['_boy','_girl'],how='outer')
print(res)
```