Homework 1: playing with pandas dataframe

Data source: http://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data
Data location: `4year.arff` at `./data/4year.arff`

Import and read

```python
from scipy.io import arff
import pandas as pd
import matplotlib.pyplot as plt
```

```python
data = arff.loadarff('./data/4year.arff')
df = pd.DataFrame(data[0])
```

Creating Bancruptcy

```python
df['bankruptcy'] = (df['class']==b'1')
```

```python
df.head(3)
```

```css
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10	...	Attr57
0	0.159290	0.46240	0.07773	1.1683	-44.853	0.467020	0.189480	0.82895	1.1223	0.38330	...	0.41557
1	-0.127430	0.46243	0.26917	1.7517	7.597	0.000925	-0.127430	1.16250	1.2944	0.53757	...	-0.23704
2	0.070488	0.23570	0.52781	3.2393	125.680	0.163670	0.086895	2.87180	1.0574	0.67689	...	0.10413

3 rows × 66 columns

```python
df.describe()
```

```css
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9
count	9791.000000	9791.000000	9791.000000	9749.000000	9.771000e+03	9791.000000	9791.000000	9773.000000	9792.000000
mean	0.043019	0.596404	0.130959	8.136600	6.465164e+01	-0.059273	0.059446	19.884016	1.882296
std	0.359321	4.587122	4.559074	290.647281	1.475939e+04	6.812754	0.533344	698.697015	17.674650
min	-12.458000	0.000000	-445.910000	-0.045319	-3.794600e+05	-486.820000	-12.458000	-1.848200	-0.032371
25%	0.001321	0.263145	0.020377	1.047000	-5.121700e+01	-0.000578	0.003004	0.428300	1.006675
50%	0.041364	0.467740	0.199290	1.591800	-5.557600e-02	0.000000	0.048820	1.088700	1.161300
75%	0.111130	0.689255	0.410670	2.880400	5.573200e+01	0.065322	0.126940	2.691000	1.970225
max	20.482000	446.910000	22.769000	27146.000000	1.034100e+06	322.200000	38.618000	53209.000000	1704.800000

8 rows × 64 columns

```python
sum(df.bankruptcy == True)
```

```
515
```

Create a new dataframe

We are going to use the following 4 features: `X1 net profit / total assets`, `X2 total liabilities / total assets`, `X7 EBIT / total assets`, `X10 equity / total assets`, and `class`

Create a new dataframe with only 4 feataures (and and `Bankruptcy`).

```
df1 = df.iloc[:,[0, 1, 6, 9, -1]] # only with X1, X2, X7, X10
```

Properly rename the columns to `X1`, `X2`, `X7`, and `X10`

```
df1.columns = ['X1', 'X2', 'X7', 'X10', 'Bankruptcy']
df1.head(3)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	X1	X2	X7	X10	Bankruptcy
0	0.159290	0.46240	0.189480	0.38330	False
1	-0.127430	0.46243	-0.127430	0.53757	False
2	0.070488	0.23570	0.086895	0.67689	False

Filling missing values

Fill-in the missing values `na` with the mean. (See Ch 4 of `PML`)

```
na_found = df1[df1.isna().any(axis=1)]
na_found
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	X1	X2	X7	X10	Bankruptcy
2898	NaN	NaN	NaN	NaN	False

```
df1 = df1.fillna(df1.mean())
df1.iloc[na_found.index,:]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	X1	X2	X7	X10	Bankruptcy
2898	0.043019	0.596404	0.059446	0.38904	False

Mean and std

Find the mean and std of the 4 features among all, bankrupt and still-operating companies (3 groups).

Generating result

```
df_stat = pd.DataFrame(columns=['Statistics'].append(df1.columns[:-1])) # Constructing a new df for multiindexed statistics
dfs = [df1, df1[df1.Bankruptcy == True], df1[df1.Bankruptcy == False]]  # A list for dfs
status = ['Overall', 'Bankrupt', 'Still-operating'] # A list for descriptions
for i in range(len(dfs)): # using i as list index
    mean = pd.Series({'Bankruptcy':status[i], 'Statistics':'Mean'}).append(dfs[i].iloc[:,:-1].mean()) # calculating the mean and append them to index series
    std  = pd.Series({'Bankruptcy':status[i], 'Statistics':'Std'} ).append(dfs[i].iloc[:,:-1].std())  # calculating the std  and append them to index series
    df_stat = df_stat.append(mean, ignore_index=True)  # append line
    df_stat = df_stat.append(std,  ignore_index=True)  # append line
df_stat = df_stat.set_index(['Bankruptcy', 'Statistics']) # reindex
df_stat = df_stat[['X1', 'X2', 'X7', 'X10']] # resort columns
df_stat
```

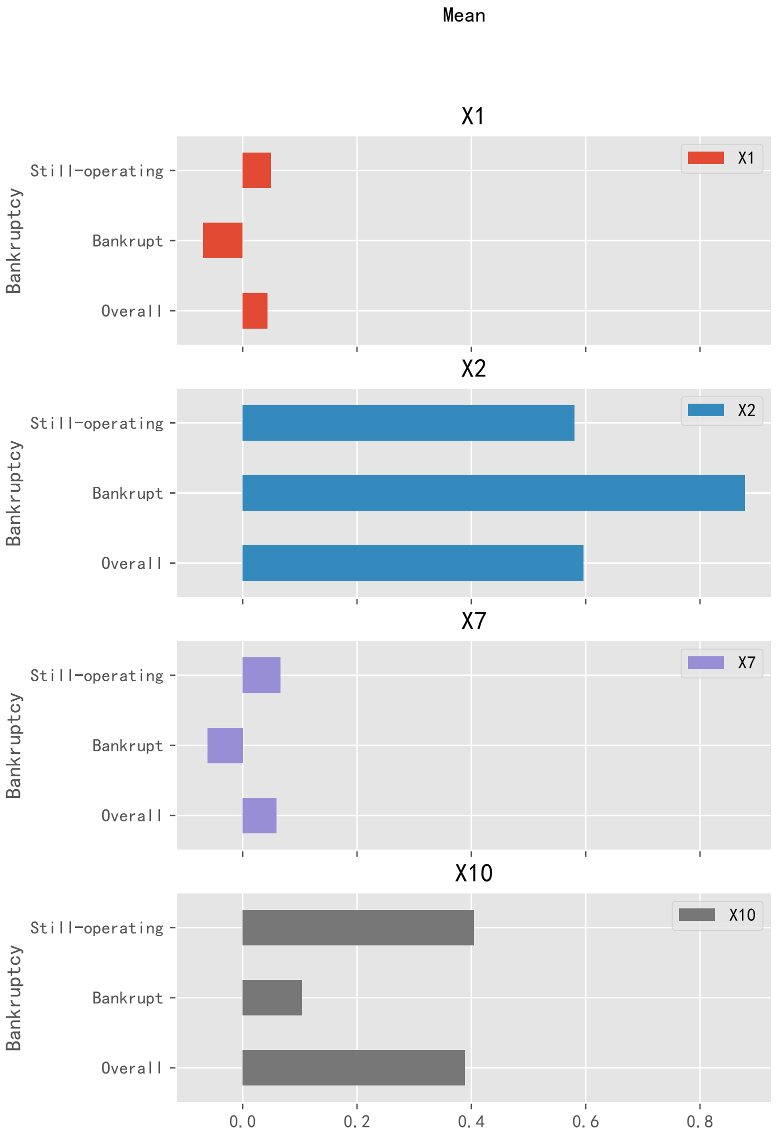
```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

		X1	X2	X7	X10
Bankruptcy	**Statistics**				
Overall	Mean	0.043019	0.596404	0.059446	0.389040
	Std	0.359303	4.586887	0.533317	4.590064
Bankrupt	Mean	-0.068873	0.878355	-0.061538	0.103367
	Std	0.568076	1.945596	0.568432	1.946747
Still-operating	Mean	0.049231	0.580752	0.066162	0.404899
	Std	0.343002	4.689694	0.530524	4.692934

Ploting mean and std

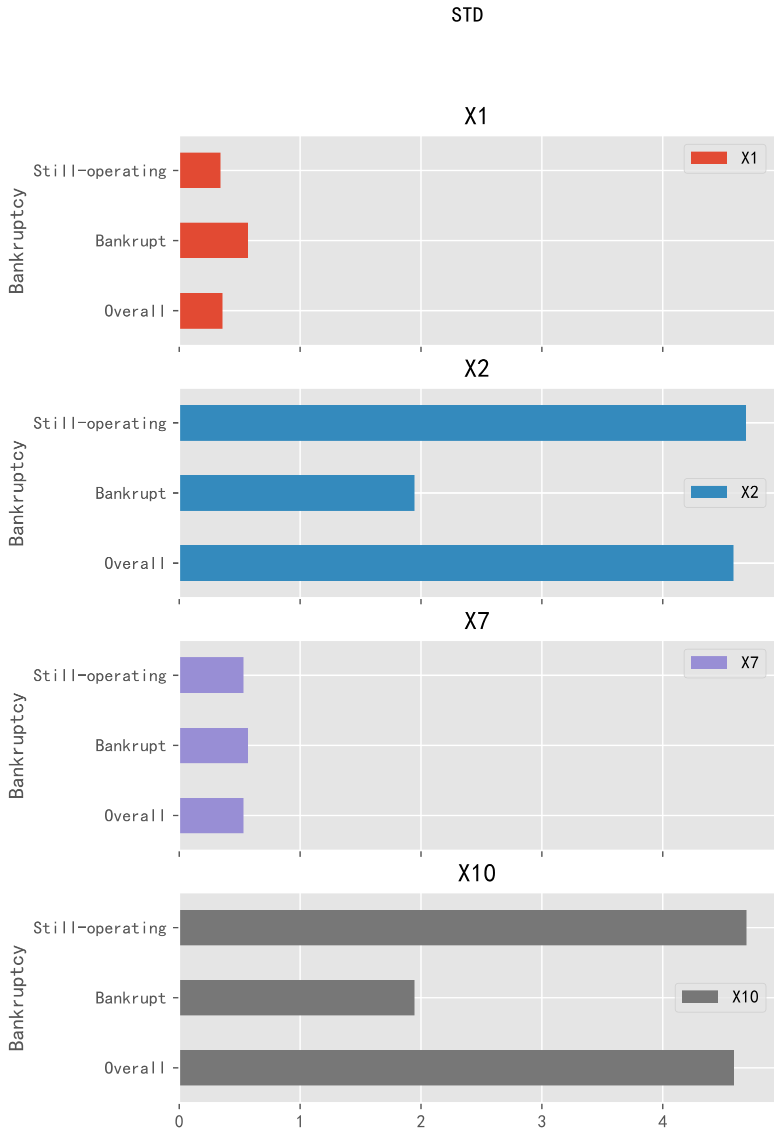
```
plt.style.use('ggplot')
%matplotlib inline
%config InlineBackend.figure_format = 'svg' # for clearer plots
df_stat.xs('Mean', level=1).plot.barh(subplots=True, title='Mean' ,figsize=(6,10), fontsize = 10)
```

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x00000242403C8308>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x000002423FA90988>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x000002423F5FD888>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x000002423F636888>],
      dtype=object)
```



```
df_stat.xs('Std', level=1).plot.barh(subplots=True, title='STD' ,figsize=(6,10), fontsize = 10)
```

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x000002423F9E3848>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x000002423FA20C48>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x000002423FA58D08>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x000002423FAE0608>],
      dtype=object)
```



Companies onsidarable

How many companies have `X1` values 1 std below the mean **AND** `X10` values 1 std below the mean?

Selecting those companies

```
df2 = df1[(df1.X1 < df1.X1.mean() - df1.X1.std()) & (df1.X10 < df1.X10.mean() - df1.X10.std())] # subgroup of  companies have X1 values 1 std below the mean AND X10 values 1 std below the mean
df2
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	X1	X2	X7	X10	Bankruptcy
2312	-1.09270	5.6368	-1.09270	-4.6368	False
2608	-3.72310	11.5300	-3.64240	-10.5300	False
3017	-1.94800	25.0050	-1.94800	-24.0050	False
3739	-0.72685	6.9334	-0.72685	-5.9334	False
4767	-5.96550	6.6818	-5.96550	-5.6818	False
5001	-3.28450	20.4030	-3.28450	-19.4030	False
5259	-0.44000	16.4870	-0.44000	-15.4870	False
5859	-0.32841	6.1187	-0.32841	-5.1187	False
6264	-0.72755	5.2632	-0.72755	-4.2632	False
7846	-1.98410	13.0630	-1.98410	-12.4730	False
8405	-9.29810	9.6992	-9.29810	-8.6992	False
8535	-1.37430	5.7326	-1.37430	-4.7326	False
9584	-4.05060	6.5306	-4.05060	-5.5306	True
9587	-0.65997	40.1570	-0.65997	-39.1560	True
9662	-1.32900	15.0190	-1.32900	-14.0190	True

Sum calculation

```
f'There are {len(df2)} companies.' # Only in Python >= 3.6
```

```
'There are 15 companies.'
```

Ratio mong above

What is the ratio of the bankrupted companies among the sub-groups above?

```
f'The ration is {len(df2[df2.Bankruptcy==True]) / len(df2) * 100}%.'
```

```
'The ration is 20.0%.'
```