

pulsars_data_visualization

March 17, 2018

```
In [1]: import numpy as np
import os
```

```
DATA_PATH = 'dataset'
```

```
In [3]: from scipy.io import arff
import pandas as pd
```

```
def load_pulsar_csv(path = DATA_PATH):
    csv_path = os.path.join(path, 'HTRU_2.csv')
    return np.loadtxt(csv_path, delimiter=',', dtype=np.float32)
```

```
def load_pulsar_arff(path = DATA_PATH):
    arff_path = os.path.join(path, 'HTRU_2.arff')
    return arff.loadarff(arff_path)
```

```
In [4]: arff_data = load_pulsar_arff()
df = pd.DataFrame(arff_data[0])
```

```
In [5]: df.head()
```

```
Out[5]:
```

	Profile_mean	Profile_stddev	Profile_skewness	Profile_kurtosis	DM_mean	\
0	140.562500	55.683782	-0.234571	-0.699648	3.199833	
1	102.507812	58.882430	0.465318	-0.515088	1.677258	
2	103.015625	39.341649	0.323328	1.051164	3.121237	
3	136.750000	57.178449	-0.068415	-0.636238	3.642977	
4	88.726562	40.672225	0.600866	1.123492	1.178930	

	DM_stddev	DM_skewness	DM_kurtosis	class
0	19.110426	7.975532	74.242225	b'0'
1	14.860146	10.576487	127.393580	b'0'
2	21.744669	7.735822	63.171909	b'0'
3	20.959280	6.896499	53.593661	b'0'
4	11.468720	14.269573	252.567306	b'0'

```
In [6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17898 entries, 0 to 17897
```

```
Data columns (total 9 columns):
Profile_mean      17898 non-null float64
Profile_stdev     17898 non-null float64
Profile_skewness  17898 non-null float64
Profile_kurtosis  17898 non-null float64
DM_mean          17898 non-null float64
DM_stdev         17898 non-null float64
DM_skewness      17898 non-null float64
DM_kurtosis      17898 non-null float64
class            17898 non-null object
dtypes: float64(8), object(1)
memory usage: 1.2+ MB
```

```
In [7]: # Number of negative and positive examples
df.loc[:, 'class'].value_counts()
```

```
Out[7]: b'0'      16259
        b'1'      1639
        Name: class, dtype: int64
```

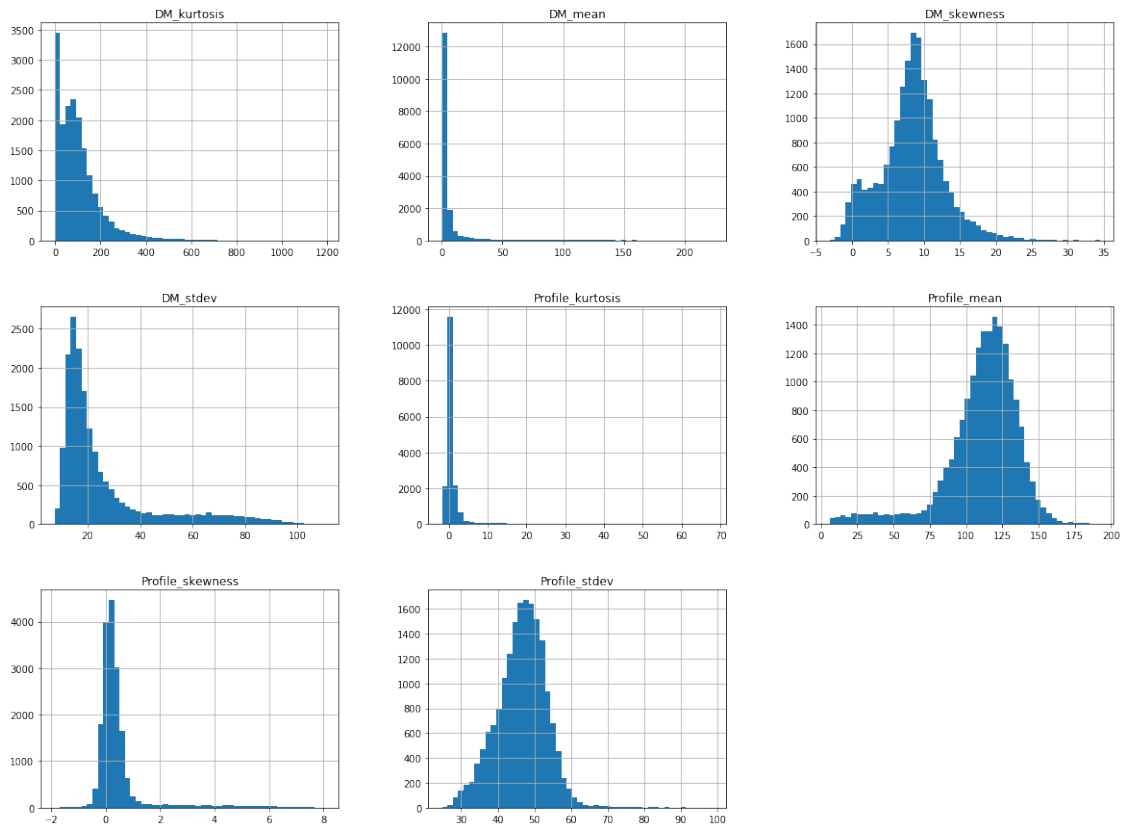
```
In [8]: df.describe()
```

```
Out[8]:
```

	Profile_mean	Profile_stdev	Profile_skewness	Profile_kurtosis	\
count	17898.000000	17898.000000	17898.000000	17898.000000	
mean	111.079968	46.549532	0.477857	1.770279	
std	25.652935	6.843189	1.064040	6.167913	
min	5.812500	24.772042	-1.876011	-1.791886	
25%	100.929688	42.376018	0.027098	-0.188572	
50%	115.078125	46.947479	0.223240	0.198710	
75%	127.085938	51.023202	0.473325	0.927783	
max	192.617188	98.778911	8.069522	68.101622	

	DM_mean	DM_stdev	DM_skewness	DM_kurtosis
count	17898.000000	17898.000000	17898.000000	17898.000000
mean	12.614400	26.326515	8.303556	104.857709
std	29.472897	19.470572	4.506092	106.514540
min	0.213211	7.370432	-3.139270	-1.976976
25%	1.923077	14.437332	5.781506	34.960504
50%	2.801839	18.461316	8.433515	83.064556
75%	5.464256	28.428104	10.702959	139.309330
max	223.392141	110.642211	34.539844	1191.000837

```
In [9]: %matplotlib inline
import matplotlib.pyplot as plt
df.hist(bins=50, figsize=(20,15))
plt.show()
```



```
In [10]: import numpy as np
```

```
def split_train_dataset(data, test_ratio):
    shuffled_indices = np.random.permutation(len(data))
    test_set_size = int(test_ratio * len(data))
    test_indices = shuffled_indices[:test_set_size]
    train_indices = shuffled_indices[test_set_size:]
    return data.iloc[train_indices], data.iloc[test_indices]
```

Use hash of identifier to decide if instance goes into train or test set

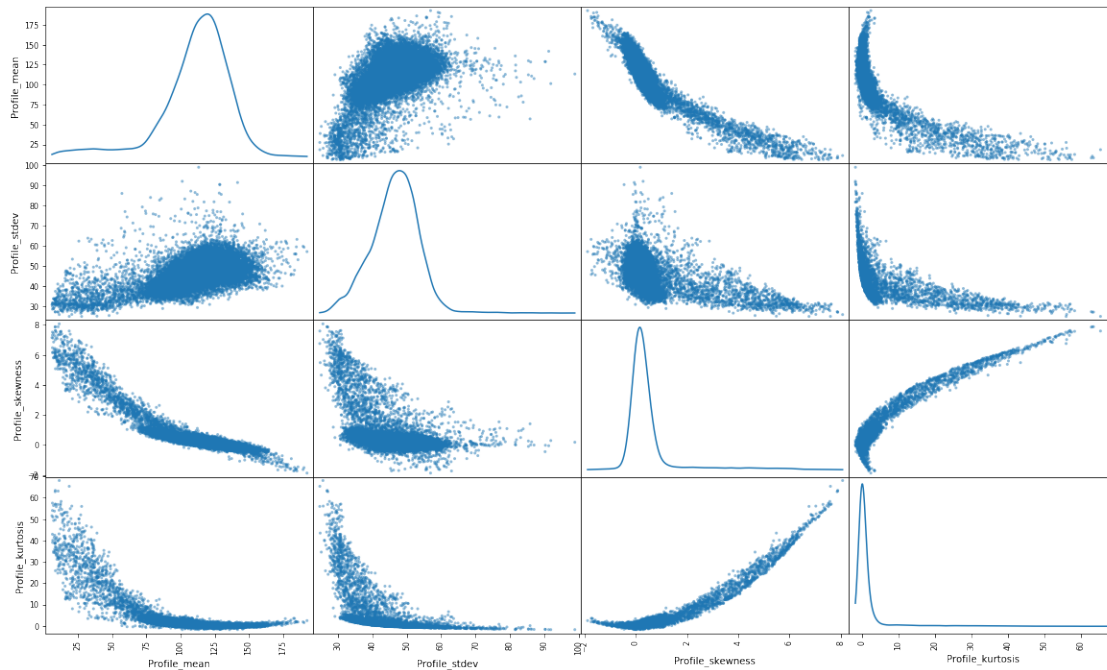
```
In [11]: # Random seed for replicability
np.random.seed(42)
```

```
train_set, test_set = split_train_dataset(df, 0.2)
```

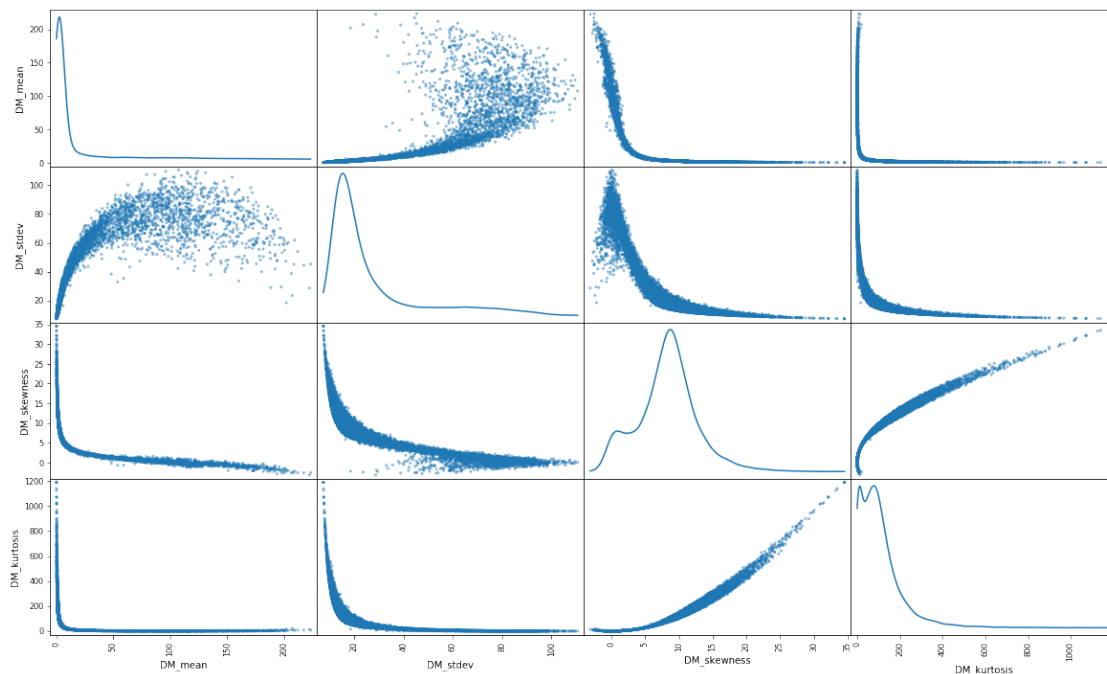
```
In [12]: # For non-linear correlations use panda's scatter_matrix
# (plots scatter graphs between every pair of given attributes)
from pandas.plotting import scatter_matrix
```

```
profile_attributes = df.keys()[:4]
```

```
scatter_matrix(df[profile_attributes], diagonal='kde', figsize=(20,12))
plt.savefig('imgs/profile_scatter_matrix.png')
```



```
In [13]: dm_attributes = df.keys()[4:]
scatter_matrix(df[dm_attributes], diagonal='kde', figsize=(20,12))
plt.savefig('imgs/DM_scatter_matrix.png')
```



```
In [14]: all_attributes = df.keys()
scatter_matrix(df[all_attributes], diagonal='kde', figsize=(20,12))
plt.savefig('imgs/full_scatter_matrix.png')
```

