VIT UNIVERSITY, ANDHRA PRADESH

School of CSE

CSE3008 - Introduction to Machine Learning

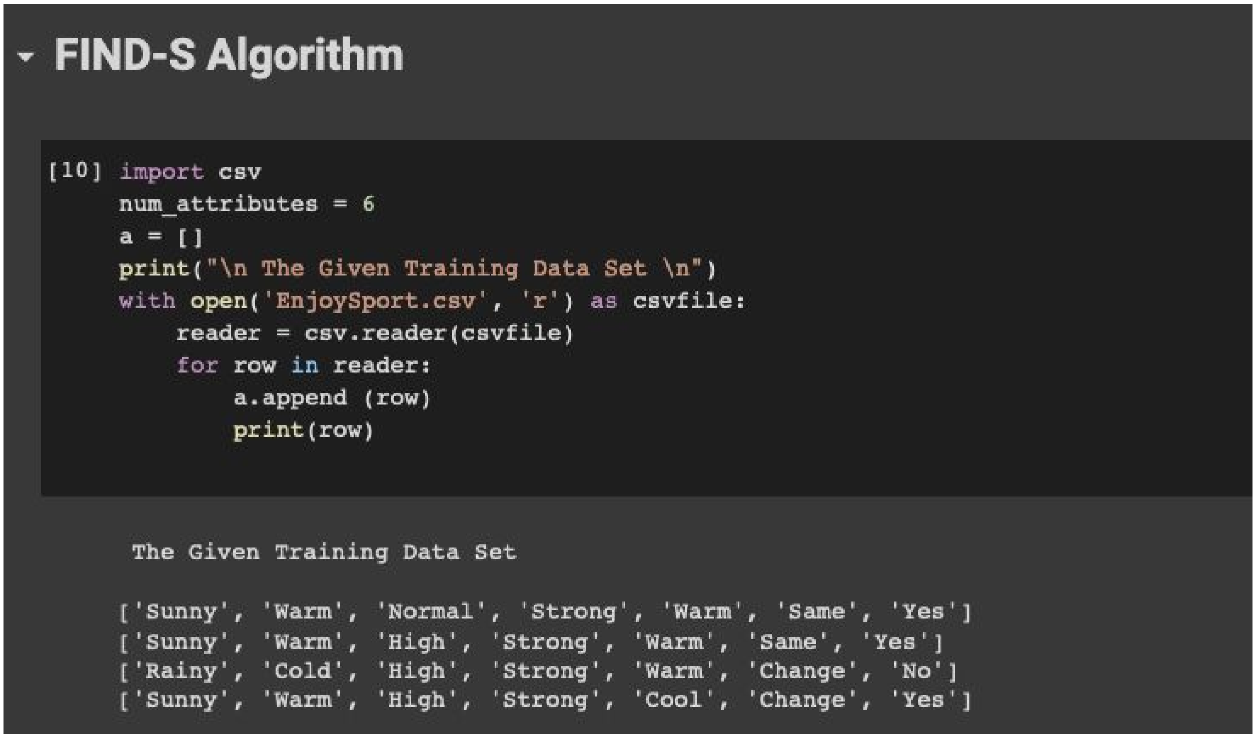
Lab Experiment-1

(Implementation FIND-S algorithm)

Faculty-**Dr. B. SRINIVASA RAO**

Name-Neeraj Guntuku
R.No-18MIS7071
Slot-L55+L56

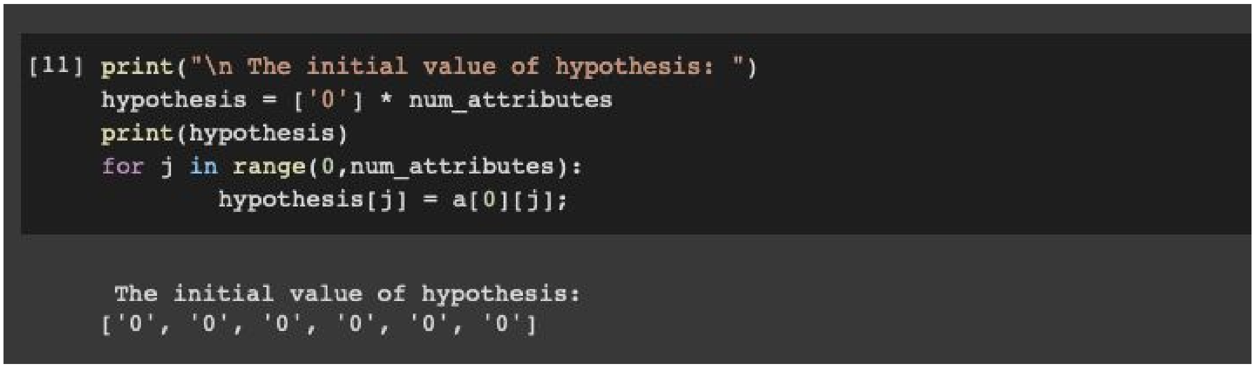
Date-6 February 2021



FIND-S Algorithm

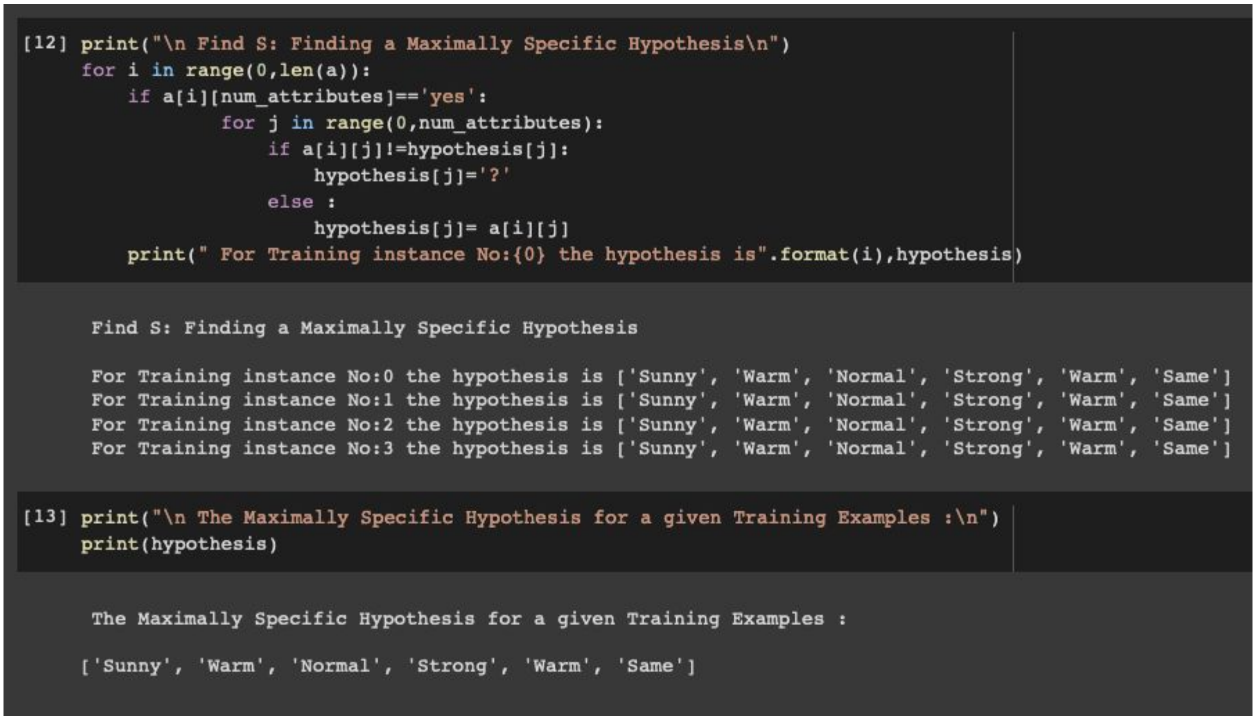
```
[10] import csv
     num_attributes = 6
     a = []
     print("\n The Given Training Data Set \n")
     with open('EnjoySport.csv', 'r') as csvfile:
         reader = csv.reader(csvfile)
         for row in reader:
             a.append (row)
             print(row)
```

```
 The Given Training Data Set 

['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'Yes']
['Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same', 'Yes']
['Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change', 'No']
['Sunny', 'Warm', 'High', 'Strong', 'Cool', 'Change', 'Yes']
```



```
[11] print("\n The initial value of hypothesis: ")
     hypothesis = ['0'] * num_attributes
     print(hypothesis)
     for j in range(0,num_attributes):
             hypothesis[j] = a[0][j];
```

```
 The initial value of hypothesis:
['0', '0', '0', '0', '0', '0']
```



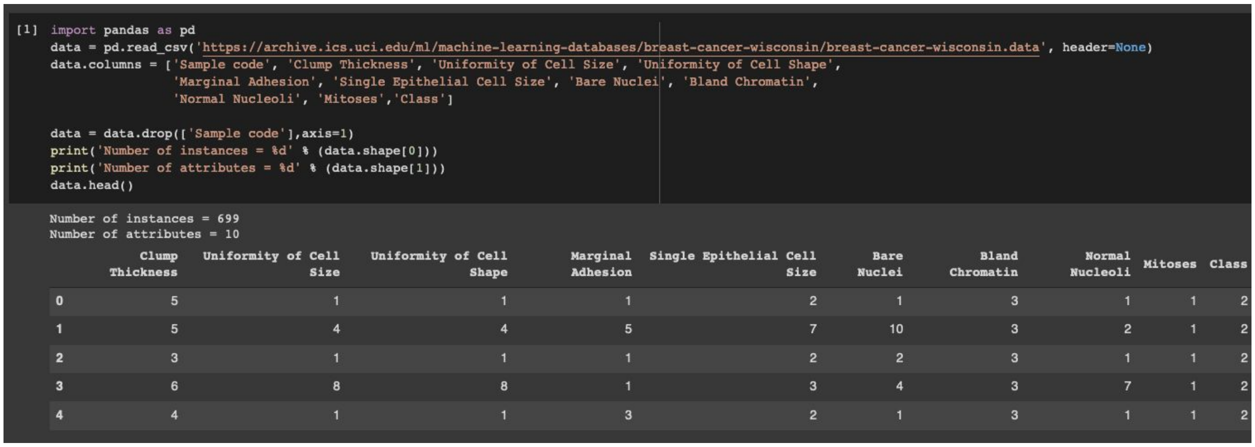
```
[12] print("\n Find S: Finding a Maximally Specific Hypothesis\n")
     for i in range(0,len(a)):
         if a[i][num_attributes]=='yes':
                 for j in range(0,num_attributes):
                     if a[i][j]!=hypothesis[j]:
                         hypothesis[j]='?'
                     else :
                         hypothesis[j]= a[i][j]
         print(" For Training instance No:{0} the hypothesis is".format(i),hypothesis)
```

```
 Find S: Finding a Maximally Specific Hypothesis

 For Training instance No:0 the hypothesis is ['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
 For Training instance No:1 the hypothesis is ['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
 For Training instance No:2 the hypothesis is ['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
 For Training instance No:3 the hypothesis is ['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
```

```
[13] print("\n The Maximally Specific Hypothesis for a given Training Examples :\n")
     print(hypothesis)
```

```
 The Maximally Specific Hypothesis for a given Training Examples :

['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
```

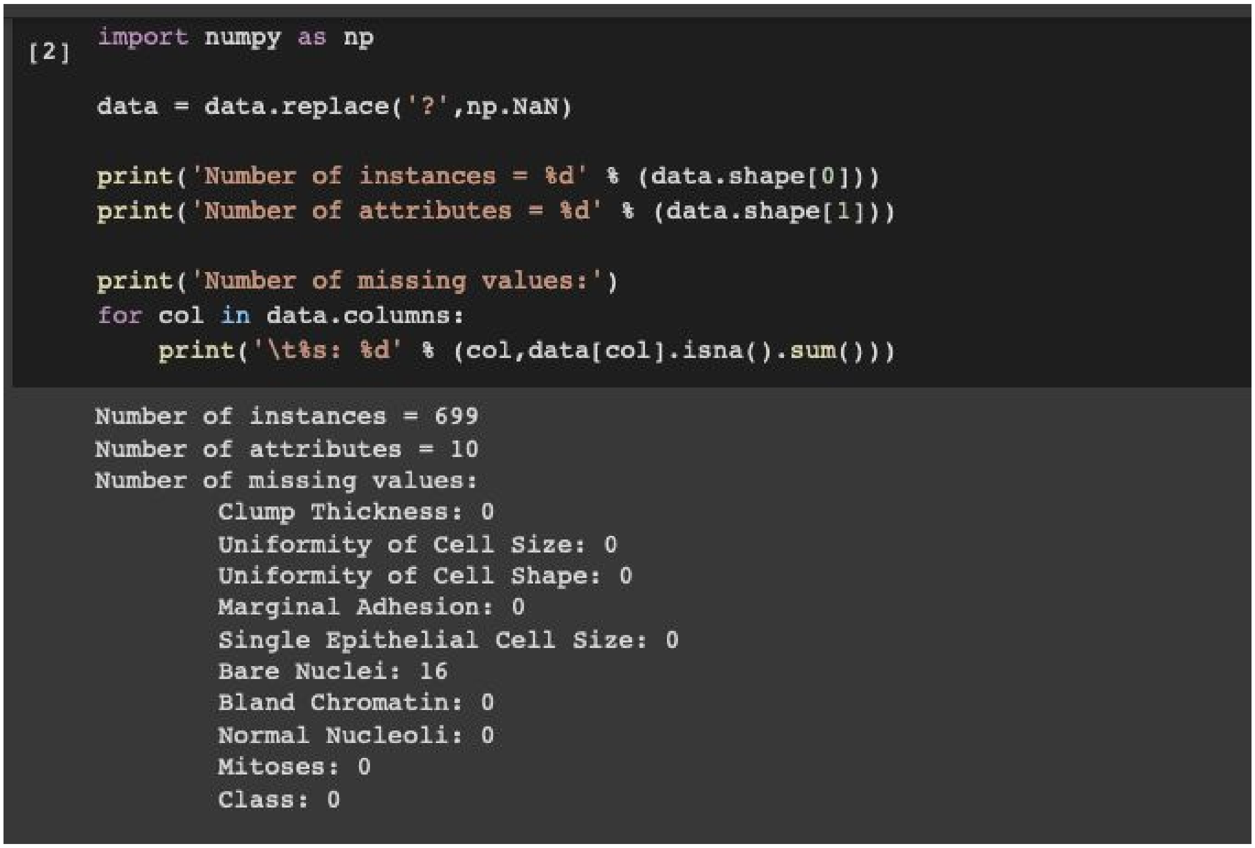
ML Data Preprocessing



```
[1] import pandas as pd
    data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data', header=None)
    data.columns = ['Sample code', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape',
                    'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',
                    'Normal Nucleoli', 'Mitoses','Class']

    data = data.drop(['Sample code'],axis=1)
    print('Number of instances = %d' % (data.shape[0]))
    print('Number of attributes = %d' % (data.shape[1]))
    data.head()
```

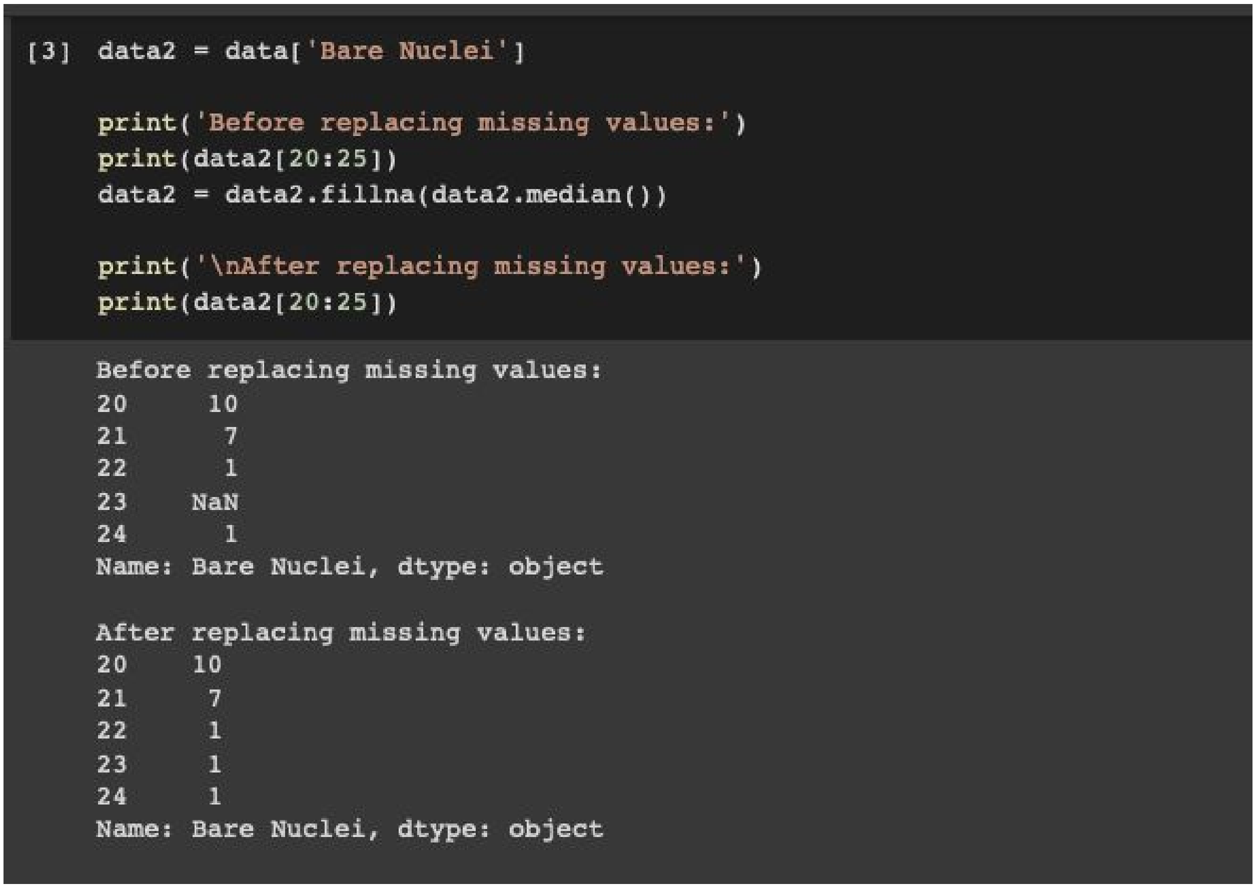
```
Number of instances = 699
Number of attributes = 10
```

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	5	1	1	1	2	1	3	1	1	2
1	5	4	4	5	7	10	3	2	1	2
2	3	1	1	1	2	2	3	1	1	2
3	6	8	8	1	3	4	3	7	1	2
4	4	1	1	3	2	1	3	1	1	2



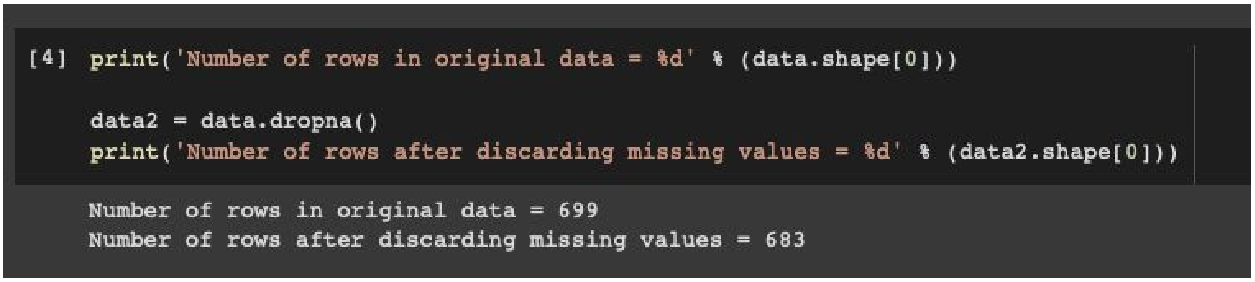
```
[2] import numpy as np

    data = data.replace('?',np.NaN)

    print('Number of instances = %d' % (data.shape[0]))
    print('Number of attributes = %d' % (data.shape[1]))

    print('Number of missing values:')
    for col in data.columns:
        print('\t%s: %d' % (col,data[col].isna().sum()))
```

```
Number of instances = 699
Number of attributes = 10
Number of missing values:
        Clump Thickness: 0
        Uniformity of Cell Size: 0
        Uniformity of Cell Shape: 0
        Marginal Adhesion: 0
        Single Epithelial Cell Size: 0
        Bare Nuclei: 16
        Bland Chromatin: 0
        Normal Nucleoli: 0
        Mitoses: 0
        Class: 0
```



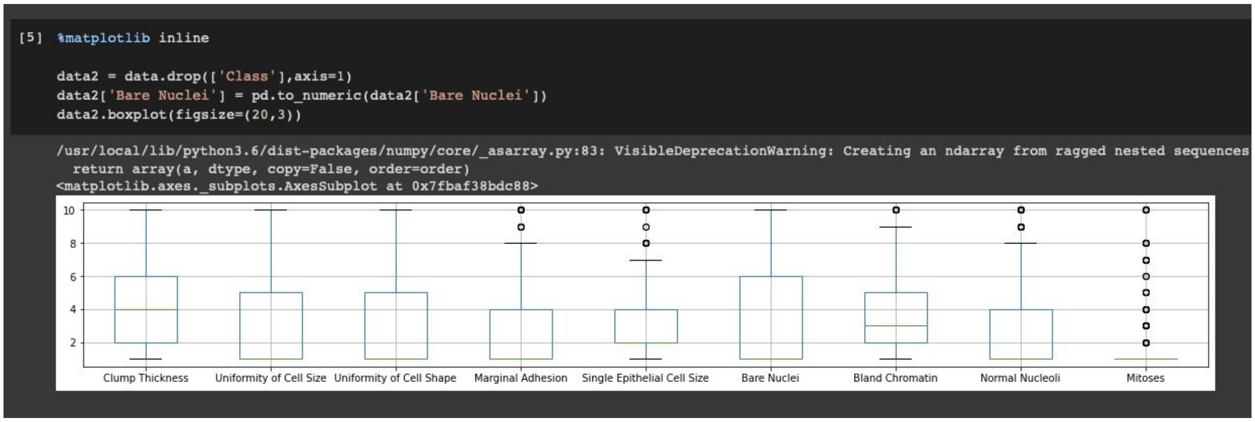
```
[3]  data2 = data['Bare Nuclei']

     print('Before replacing missing values:')
     print(data2[20:25])
     data2 = data2.fillna(data2.median())

     print('\nAfter replacing missing values:')
     print(data2[20:25])
```

```
     Before replacing missing values:
     20      10
     21       7
     22       1
     23     NaN
     24       1
     Name: Bare Nuclei, dtype: object

     After replacing missing values:
     20    10
     21     7
     22     1
     23     1
     24     1
     Name: Bare Nuclei, dtype: object
```



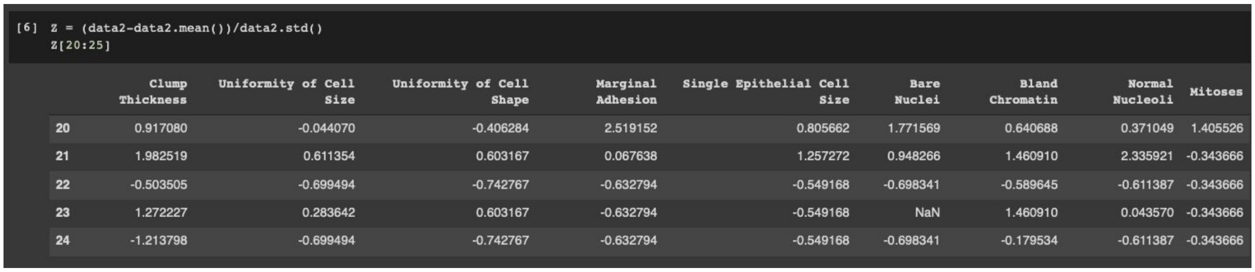
```
[4]  print('Number of rows in original data = %d' % (data.shape[0]))

     data2 = data.dropna()
     print('Number of rows after discarding missing values = %d' % (data2.shape[0]))
```

```
     Number of rows in original data = 699
     Number of rows after discarding missing values = 683
```



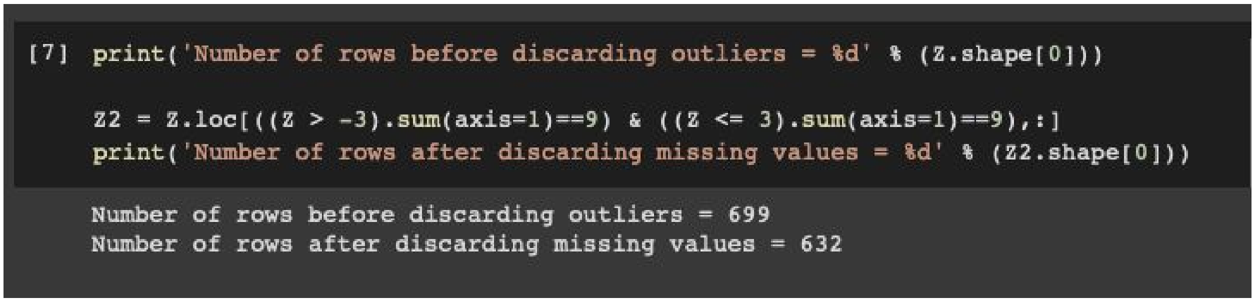
```
[5]  %matplotlib inline

     data2 = data.drop(['Class'],axis=1)
     data2['Bare Nuclei'] = pd.to_numeric(data2['Bare Nuclei'])
     data2.boxplot(figsize=(20,3))
```

```
     /usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
       return array(a, dtype, copy=False, order=order)
     <matplotlib.axes._subplots.AxesSubplot at 0x7fbaf38bdc88>
```

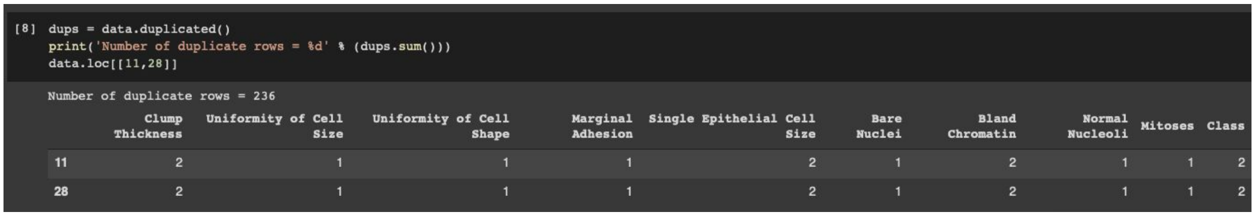


```
[6] Z = (data2-data2.mean())/data2.std()
    Z[20:25]
```

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses
20	0.917080	-0.044070	-0.406284	2.519152	0.805662	1.771569	0.640688	0.371049	1.405526
21	1.982519	0.611354	0.603167	0.067638	1.257272	0.948266	1.460910	2.335921	-0.343666
22	-0.503505	-0.699494	-0.742767	-0.632794	-0.549168	-0.698341	-0.589645	-0.611387	-0.343666
23	1.272227	0.283642	0.603167	-0.632794	-0.549168	NaN	1.460910	0.043570	-0.343666
24	-1.213798	-0.699494	-0.742767	-0.632794	-0.549168	-0.698341	-0.179534	-0.611387	-0.343666

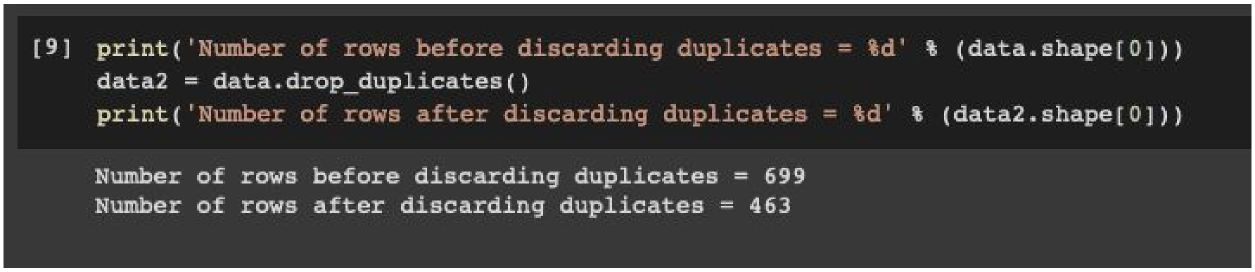


```
[7] print('Number of rows before discarding outliers = %d' % (Z.shape[0]))

    Z2 = Z.loc[((Z > -3).sum(axis=1)==9) & ((Z <= 3).sum(axis=1)==9),:]
    print('Number of rows after discarding missing values = %d' % (Z2.shape[0]))

    Number of rows before discarding outliers = 699
    Number of rows after discarding missing values = 632
```



```
[8] dups = data.duplicated()
    print('Number of duplicate rows = %d' % (dups.sum()))
    data.loc[[11,28]]

    Number of duplicate rows = 236
```

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
11	2	1	1	1	2	1	2	1	1	2
28	2	1	1	1	2	1	2	1	1	2



```
[9] print('Number of rows before discarding duplicates = %d' % (data.shape[0]))
    data2 = data.drop_duplicates()
    print('Number of rows after discarding duplicates = %d' % (data2.shape[0]))

    Number of rows before discarding duplicates = 699
    Number of rows after discarding duplicates = 463
```

***