# PythonExercise 2 - Data Preprocessing in Python

## Author: Sanjay Bhargav Siddi

### This document contains Python examples for data preprocessing. Data preprocessing consists of a broad set of techniques for cleaning, selecting, and transforming data to improve data mining analysis.

## 1. Data Quality Issues

### 1.1. Loading the Wisconsin's breast cancer dataset

In [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
```

Out[1]:

| | 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 |

### 1.2. Missing Values

In [3]:
```
import numpy as np

# convert the missing values to NaNs
data = data.replace('?',np.NaN)

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

# count the number of missing values in each column of the data
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
```

In [5]:
```
print('Number of rows in original data = %d' % (data.shape[0]))

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

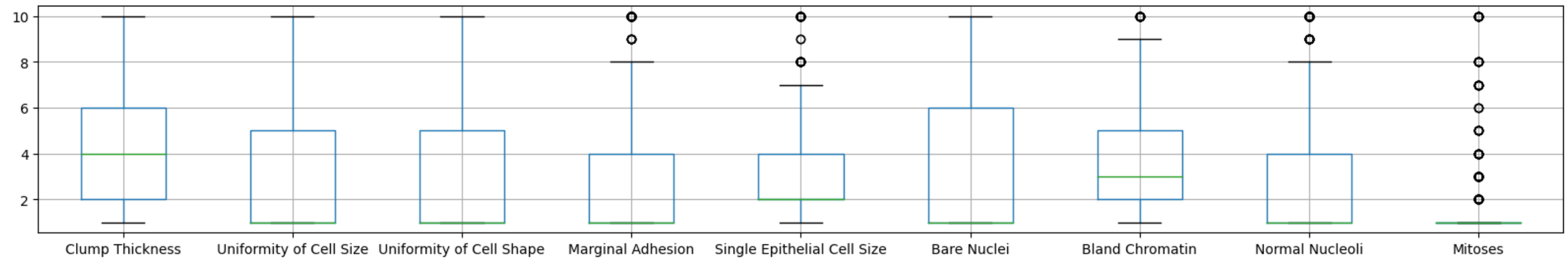
```
Number of rows in original data = 699
Number of rows after discarding missing values = 683
```

### 1.3. Outliers

In [7]:
```
%matplotlib inline

# convert the column into numeric values
data2 = data.drop(['Class'],axis=1)
data2['Bare Nuclei'] = pd.to_numeric(data2['Bare Nuclei'])
data2.boxplot(figsize=(20,3))
```

Out[7]:
```
<Axes: >
```




In [8]:
```
# standardizing the columns of the data
Z = (data2-data2.mean())/data2.std()
Z[20:25]
```

Out[8]:

| | 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 |

In [9]:
```
print('Number of rows before discarding outliers = %d' % (Z.shape[0]))

# discarding columns with Z > 3 or Z <= -3
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
```

### 1.4. Duplicate Data

In [11]:
```
# check for duplicate instances in the breast cancer dataset
dups = data.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))
data.loc[[11,28]]
```

```
Number of duplicate rows = 236
```

Out[11]:

| | 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 |

In [12]:
```
print('Number of rows before discarding duplicates = %d' % (data.shape[0]))

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

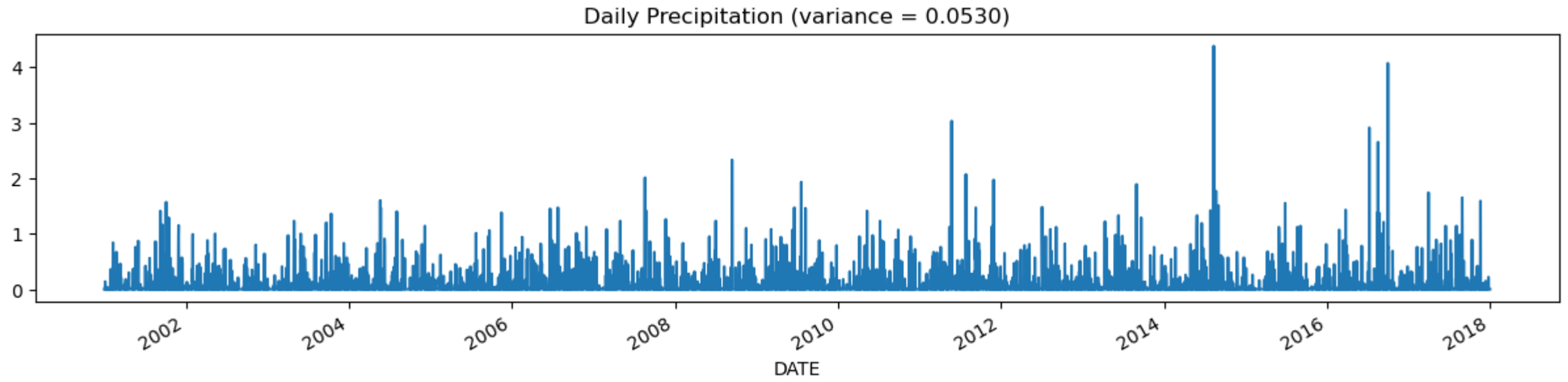
```
Number of rows before discarding duplicates = 699
Number of rows after discarding duplicates = 463
```

## 2. Aggregation

### 2.1. Loading the precipitation time series data

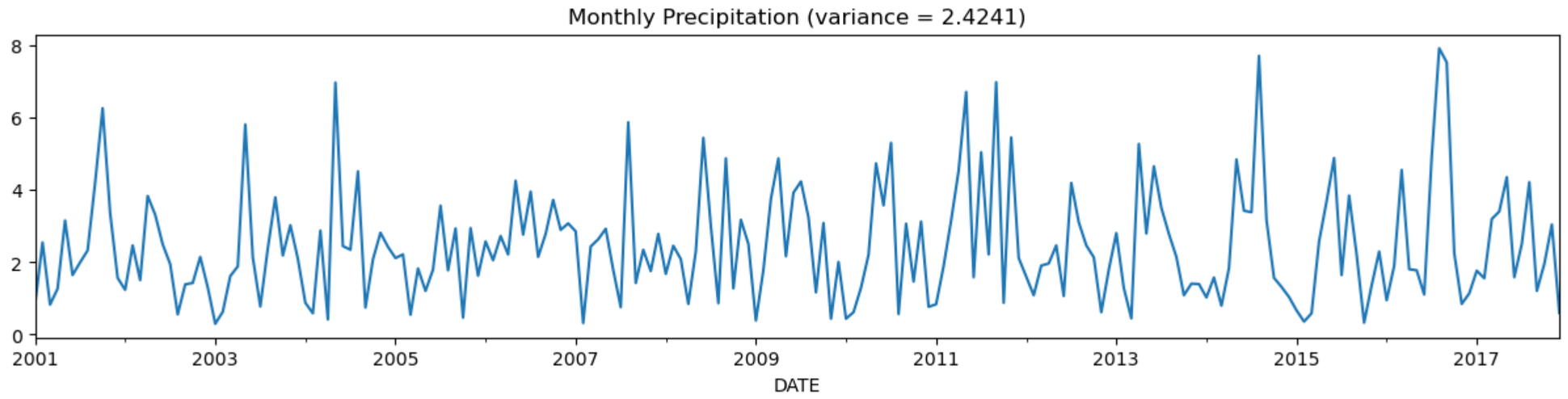
In [13]:
```
daily = pd.read_csv('C:/Users/sanja/OneDrive/Desktop/University of Arizona Classes/INFO 523 - Data Mining/HW/r-python-exercise2-SanjaySiddi/data/DTW_prec.csv', header='infer')
daily.index = pd.to_datetime(daily['DATE'])
daily = daily['PRCP']
ax = daily.plot(kind='line',figsize=(15,3))
ax.set_title('Daily Precipitation (variance = %.4f)' % (daily.var()))
```

Out[13]:
```
Text(0.5, 1.0, 'Daily Precipitation (variance = 0.0530)')
```



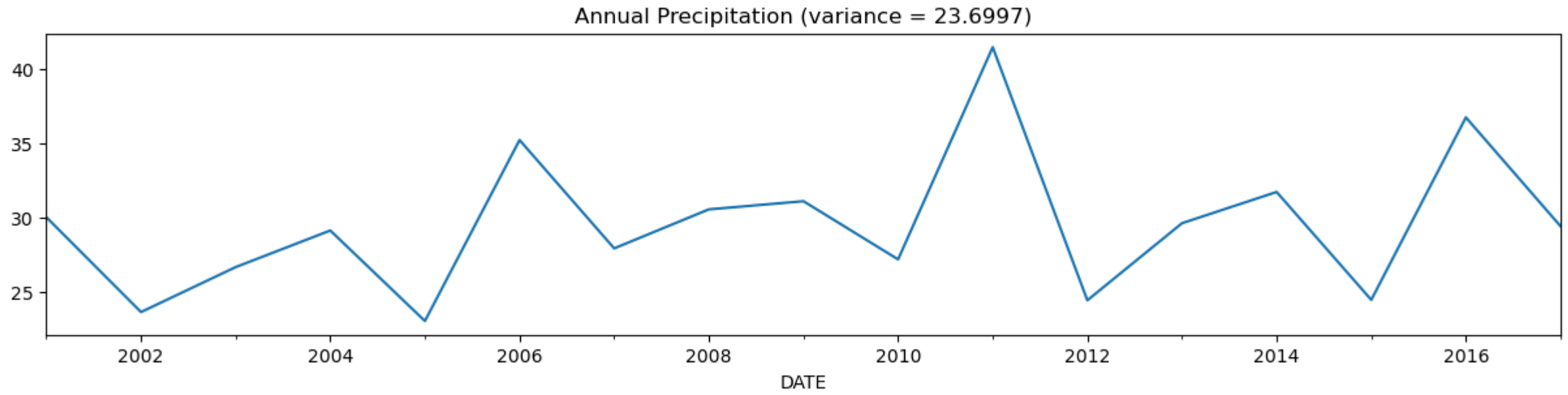

In [14]:
```
# The time series can be grouped and aggregated by month to obtain the total monthly precipitation values.
monthly = daily.groupby(pd.Grouper(freq='M')).sum()
ax = monthly.plot(kind='line',figsize=(15,3))
ax.set_title('Monthly Precipitation (variance = %.4f)' % (monthly.var()))
```

Out[14]:
```
Text(0.5, 1.0, 'Monthly Precipitation (variance = 2.4241)')
```




In [15]:
```
# The daily precipitation time series are grouped and aggregated by year to obtain the annual precipitation values
annual = daily.groupby(pd.Grouper(freq='Y')).sum()
ax = annual.plot(kind='line',figsize=(15,3))
ax.set_title('Annual Precipitation (variance = %.4f)' % (annual.var()))
```

Out[15]:
```
Text(0.5, 1.0, 'Annual Precipitation (variance = 23.6997)')
```




## 3. Sampling

### 3.1. Loading the Wisconsin's breast cancer data

In [20]:
```
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.head()
```

Out[20]:

| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1000025 | 5 | 1 | 1 | 1 | 2 | 1 | 3 | 1 | 1 | 2 |
| **1** | 1002945 | 5 | 4 | 4 | 5 | 7 | 10 | 3 | 2 | 1 | 2 |
| **2** | 1015425 | 3 | 1 | 1 | 1 | 2 | 2 | 3 | 1 | 1 | 2 |
| **3** | 1016277 | 6 | 8 | 8 | 1 | 3 | 4 | 3 | 7 | 1 | 2 |
| **4** | 1017023 | 4 | 1 | 1 | 3 | 2 | 1 | 3 | 1 | 1 | 2 |

In [21]:
```
# sample of size 3 is randomly selected (without replacement) from the original data
sample = data.sample(n=3)
sample
```

Out[21]:

| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **200** | 1214966 | 9 | 7 | 7 | 5 | 5 | 10 | 7 | 8 | 3 | 4 |
| **651** | 1323477 | 1 | 2 | 1 | 3 | 2 | 1 | 2 | 1 | 1 | 2 |
| **498** | 1204558 | 4 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 2 |

In [1]:
```
# randomly select 1% of the data (without replacement) and display the selected samples
sample = data.sample(frac=0.01, random_state=1)
sample
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 2
      1 # randomly select 1% of the data (without replacement) and display the selected samples
----> 2 sample = data.sample(frac=0.01, random_state=1)
      3 sample

NameError: name 'data' is not defined
```

In [23]:
```
# Perform a sampling with replacement to create a sample whose size is equal to 1% of the entire data
sample = data.sample(frac=0.01, replace=True, random_state=1)
sample
```

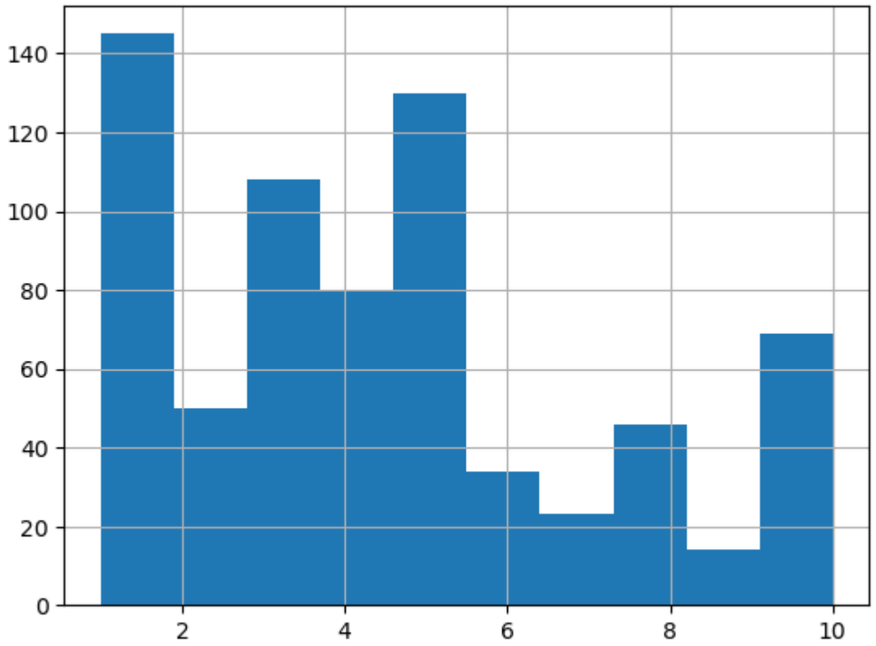
Out[23]:

| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **37** | 1081791 | 6 | 2 | 1 | 1 | 1 | 1 | 7 | 1 | 1 | 2 |
| **235** | 1241232 | 3 | 1 | 4 | 1 | 2 | ? | 3 | 1 | 1 | 2 |
| **72** | 1124651 | 1 | 3 | 3 | 2 | 2 | 1 | 7 | 2 | 1 | 2 |
| **645** | 1303489 | 3 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 2 |
| **144** | 1184241 | 2 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 2 |
| **129** | 1177512 | 1 | 1 | 1 | 1 | 10 | 1 | 1 | 1 | 1 | 2 |
| **583** | 1115762 | 3 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 |

## 4. Discretization

In [24]:
```
# we plot a histogram that shows the distribution of the attribute values
data['Clump Thickness'].hist(bins=10)
data['Clump Thickness'].value_counts(sort=False)
```

Out[24]:
```
5     130
3     108
6      34
4      80
8      46
1     145
2      50
7      23
10     69
9      14
Name: Clump Thickness, dtype: int64
```




In [25]:
```
# For the equal width method, we can apply the cut() function to discretize the attribute into 4 bins of similar interval widths
bins = pd.cut(data['Clump Thickness'],4)
bins.value_counts(sort=False)
```

Out[25]:
```
(0.991, 3.25]    303
(3.25, 5.5]      210
(5.5, 7.75]       57
(7.75, 10.0]     129
Name: Clump Thickness, dtype: int64
```

In [26]:
```
# For the equal frequency method, the qcut() function can be used to partition the values into 4 bins such that each bin has nearly the same number of instances
bins = pd.qcut(data['Clump Thickness'],4)
bins.value_counts(sort=False)
```

Out[26]:
```
(0.999, 2.0]    195
(2.0, 4.0]      188
(4.0, 6.0]      164
(6.0, 10.0]     152
Name: Clump Thickness, dtype: int64
```

## 5. Principal Component Analysis

In [27]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np

numImages = 16
fig = plt.figure(figsize=(7,7))
imgData = np.zeros(shape=(numImages,36963))

# application of PCA to an image dataset
for i in range(1,numImages+1):
    filename = 'C:/Users/sanja/OneDrive/Desktop/University of Arizona Classes/INFO 523 - Data Mining/HW/r-python-exercise2-SanjaySiddi/data/pics/Picture'+str(i)+'.jpg'
    img = mpimg.imread(filename)
    ax = fig.add_subplot(4,4,i)
    plt.imshow(img)
```

```
    plt.axis('off')
    ax.set_title(str(i))
    imgData[i-1] = np.array(img.flatten()).reshape(1,img.shape[0]*img.shape[1]*img.shape[2])
```


In [28]:
```
import pandas as pd
from sklearn.decomposition import PCA

numComponents = 2
pca = PCA(n_components=numComponents)
pca.fit(imgData)

# Using PCA, the data matrix is projected to its first two principal components
projected = pca.transform(imgData)
projected = pd.DataFrame(projected,columns=['pc1','pc2'],index=range(1,numImages+1))
projected['food'] = ['burger', 'burger','burger','burger','drink','drink','drink','drink',
                     'pasta', 'pasta', 'pasta', 'pasta', 'chicken', 'chicken', 'chicken', 'chicken']
projected
```

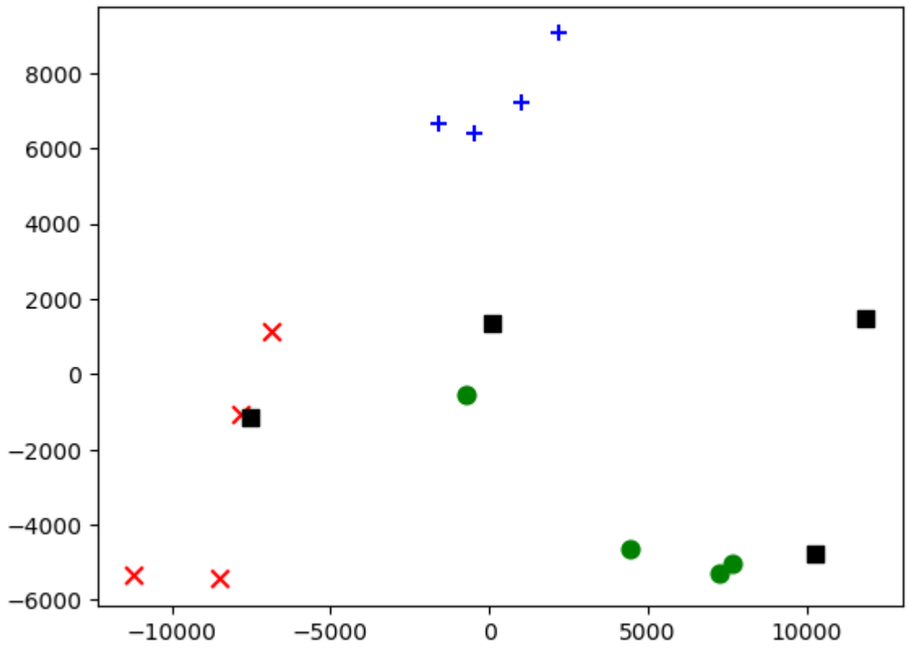
Out[28]:

| | pc1 | pc2 | food |
|---|---|---|---|
| **1** | -1576.713362 | 6641.426119 | burger |
| **2** | -493.812166 | 6399.711927 | burger |
| **3** | 990.090469 | 7235.264063 | burger |
| **4** | 2189.876479 | 9049.733363 | burger |
| **5** | -7843.054234 | -1063.209474 | drink |
| **6** | -8498.431574 | -5439.004991 | drink |
| **7** | -11181.797285 | -5318.605854 | drink |
| **8** | -6851.914533 | 1126.243025 | drink |
| **9** | 7635.133058 | -5044.118082 | pasta |
| **10** | -708.062499 | -528.409960 | pasta |
| **11** | 7236.227159 | -5301.416407 | pasta |
| **12** | 4417.314542 | -4658.923839 | pasta |
| **13** | 11864.519704 | 1473.193927 | chicken |
| **14** | 76.468728 | 1364.682872 | chicken |
| **15** | -7505.665974 | -1164.724404 | chicken |
| **16** | 10249.821488 | -4771.842285 | chicken |

In [29]:
```
import matplotlib.pyplot as plt

colors = {'burger':'b', 'drink':'r', 'pasta':'g', 'chicken':'k'}
markerTypes = {'burger':'+', 'drink':'x', 'pasta':'o', 'chicken':'s'}

# scatter plot to display the projected values
for foodType in markerTypes:
    d = projected[projected['food']==foodType]
    plt.scatter(d['pc1'],d['pc2'],c=colors[foodType],s=60,marker=markerTypes[foodType])
```




In [ ]: