CSC 767 Neural Networks and Deep Learning
Homework # 1
Part 1

Team Members:

`

**Introduction**

The dataset was downloaded from the UCI Machine Learning Repository. There are 2 datasets related to red and white variants of the Portuguese “Vinho Verde” but for this study we’ve selected the white dataset.
In this study we want to search for elements which effects the wine quality by using multiclass decision classification such as Logistic Regression (LR), Linear Discriminant Analysis (LDA), k-Nearest Neighbor (KNN), Classifications and Regression Trees(CART), Gaussian Naive Bayes(NB), Support Vector Machines (SVM).

**Information of the Data**

Attributes are numeric (float values) and they are: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates and alcohol.
The classes are in quality in integer from 0 to 10.
Attributes values are in acidity levels, sugar levels, dioxide level, pH level and alcohol levels.

**Fixed acidity:** Fixed acids include tartaric, malic, citric, and succinic acids which are found in grapes (except succinic).

**Volatile acidity:** These acids are to be distilled out from the wine before completing the production process. Excess of volatile acids are undesirable and lead to unpleasant flavor.

**Citric acid:** This is one of the fixed acids which gives a wine its freshness.

**Residual sugar:** This typically refers to the natural sugar from grapes which remains after the fermentation process stops.

**Chlorides:** Chloride concentration in the wine.

**Free sulfur dioxide:** They are also known as sulfites and too much of it is undesirable and gives a pungent odor.

**Total sulfur dioxide:** This is the sum total of the bound and the free sulfur dioxide. This is mainly added to kill harmful bacteria and preserve quality and freshness.

**Density:** It is generally used as a measure of the conversion of sugar to alcohol.

**pH:** Also known as the potential of hydrogen, this is a numeric scale to specify the acidity of the wine.

**Sulphates:** These are mineral salts containing sulfur. They are connected to the fermentation process and affect the wine aroma and flavor.

**Alcohol:** It's usually measured in % vol

**Quality:** Wine experts graded the wine quality between 0 (very bad) and 10 (very excellent).

Listing 1a: Increase the sizes of output space and Load Libraries

In [53]:
```
%%javascript
IPython.OutputArea.auto_scroll_threshold = 99999;
```

In [54]:
```
# Hello World Classification: White Wine Quality

# Prepare Problem

# Load libraries
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

# Load dataset
filename = 'WhiteWineQuality.csv'
names = ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides' ,
         'free sulfur dioxide' , 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', 'quality']
dataset = read_csv(filename, names=names)
```

Listing 1b: Load Dataset

In [55]:
```
# Summarize Data

# Descriptive statistics
# shape
print('Number of Data Rows = ', dataset.shape[0]);
print('Number of Data columns = ', dataset.shape[1]);

```

```
Number of Data Rows =  4898
Number of Data columns =  12
```

Listing 2: Dimension of the dataset. Peek at the data itself. Statistical summary of all attributes. Breakdown of the data by the class variable

a, Print the shape of the data set

```
In [56]:
# head
print(dataset.head(20))
```

b, Print the first few rows of the data-set (in the first 20 rows)

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol | quality |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 7.0 | 0.27 | 0.36 | 20.70 | 0.045 | 45.0 | 170.0 | 1.0010 | 3.00 | 0.45 | 8.8 | 6 |
| 1 | 6.3 | 0.30 | 0.34 | 1.60 | 0.049 | 14.0 | 132.0 | 0.9940 | 3.30 | 0.49 | 9.5 | 6 |
| 2 | 8.1 | 0.28 | 0.40 | 6.90 | 0.050 | 30.0 | 97.0 | 0.9951 | 3.26 | 0.44 | 10.1 | 6 |
| 3 | 7.2 | 0.23 | 0.32 | 8.50 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 | 6 |
| 4 | 7.2 | 0.23 | 0.32 | 8.50 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 | 6 |
| 5 | 8.1 | 0.28 | 0.40 | 6.90 | 0.050 | 30.0 | 97.0 | 0.9951 | 3.26 | 0.44 | 10.1 | 6 |
| 6 | 6.2 | 0.32 | 0.16 | 7.00 | 0.045 | 30.0 | 136.0 | 0.9949 | 3.18 | 0.47 | 9.6 | 6 |
| 7 | 7.0 | 0.27 | 0.36 | 20.70 | 0.045 | 45.0 | 170.0 | 1.0010 | 3.00 | 0.45 | 8.8 | 6 |
| 8 | 6.3 | 0.30 | 0.34 | 1.60 | 0.049 | 14.0 | 132.0 | 0.9940 | 3.30 | 0.49 | 9.5 | 6 |
| 9 | 8.1 | 0.22 | 0.43 | 1.50 | 0.044 | 28.0 | 129.0 | 0.9938 | 3.22 | 0.45 | 11.0 | 6 |
| 10 | 8.1 | 0.27 | 0.41 | 1.45 | 0.033 | 11.0 | 63.0 | 0.9908 | 2.99 | 0.56 | 12.0 | 5 |
| 11 | 8.6 | 0.23 | 0.40 | 4.20 | 0.035 | 17.0 | 109.0 | 0.9947 | 3.14 | 0.53 | 9.7 | 5 |
| 12 | 7.9 | 0.18 | 0.37 | 1.20 | 0.040 | 16.0 | 75.0 | 0.9920 | 3.18 | 0.63 | 10.8 | 5 |
| 13 | 6.6 | 0.16 | 0.40 | 1.50 | 0.044 | 48.0 | 143.0 | 0.9912 | 3.54 | 0.52 | 12.4 | 7 |
| 14 | 8.3 | 0.42 | 0.62 | 19.25 | 0.040 | 41.0 | 172.0 | 1.0002 | 2.98 | 0.67 | 9.7 | 5 |
| 15 | 6.6 | 0.17 | 0.38 | 1.50 | 0.032 | 28.0 | 112.0 | 0.9914 | 3.25 | 0.55 | 11.4 | 7 |
| 16 | 6.3 | 0.48 | 0.04 | 1.10 | 0.046 | 30.0 | 99.0 | 0.9928 | 3.24 | 0.36 | 9.6 | 6 |
| 17 | 6.2 | 0.66 | 0.48 | 1.20 | 0.029 | 29.0 | 75.0 | 0.9892 | 3.33 | 0.39 | 12.8 | 8 |
| 18 | 7.4 | 0.34 | 0.42 | 1.10 | 0.033 | 17.0 | 171.0 | 0.9917 | 3.12 | 0.53 | 11.3 | 6 |
| 19 | 6.5 | 0.31 | 0.14 | 7.50 | 0.044 | 34.0 | 133.0 | 0.9955 | 3.22 | 0.50 | 9.5 | 5 |

c, Print the statistical descriptions of the data-set

```
In [57]:
# descriptions
print(dataset.describe())

```

```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    4898.000000       4898.000000  4898.000000     4898.000000
mean        6.854788          0.278241     0.334192        6.391415
std         0.843868          0.100795     0.121020        5.072058
min         3.800000          0.080000     0.000000        0.600000
25%         6.300000          0.210000     0.270000        1.700000
50%         6.800000          0.260000     0.320000        5.200000
75%         7.300000          0.320000     0.390000        9.900000
max        14.200000          1.100000     1.660000       65.800000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  4898.000000          4898.000000           4898.000000  4898.000000
mean      0.045772            35.308085            138.360657     0.994027
std       0.021848            17.007137             42.498065     0.002991
min       0.009000             2.000000              9.000000     0.987110
25%       0.036000            23.000000            108.000000     0.991723
50%       0.043000            34.000000            134.000000     0.993740
75%       0.050000            46.000000            167.000000     0.996100
max       0.346000           289.000000            440.000000     1.038980

                pH    sulphates      alcohol      quality
count  4898.000000  4898.000000  4898.000000  4898.000000
mean      3.188267     0.489847    10.514267     5.877909
std       0.151001     0.114126     1.230621     0.885639
min       2.720000     0.220000     8.000000     3.000000
25%       3.090000     0.410000     9.500000     5.000000
50%       3.180000     0.470000    10.400000     6.000000
75%       3.280000     0.550000    11.400000     6.000000
max       3.820000     1.080000    14.200000     9.000000
```

d, Print the class distribution in the data-set.

```
In [58]:
# class distribution
print(dataset.groupby('quality').size())
```

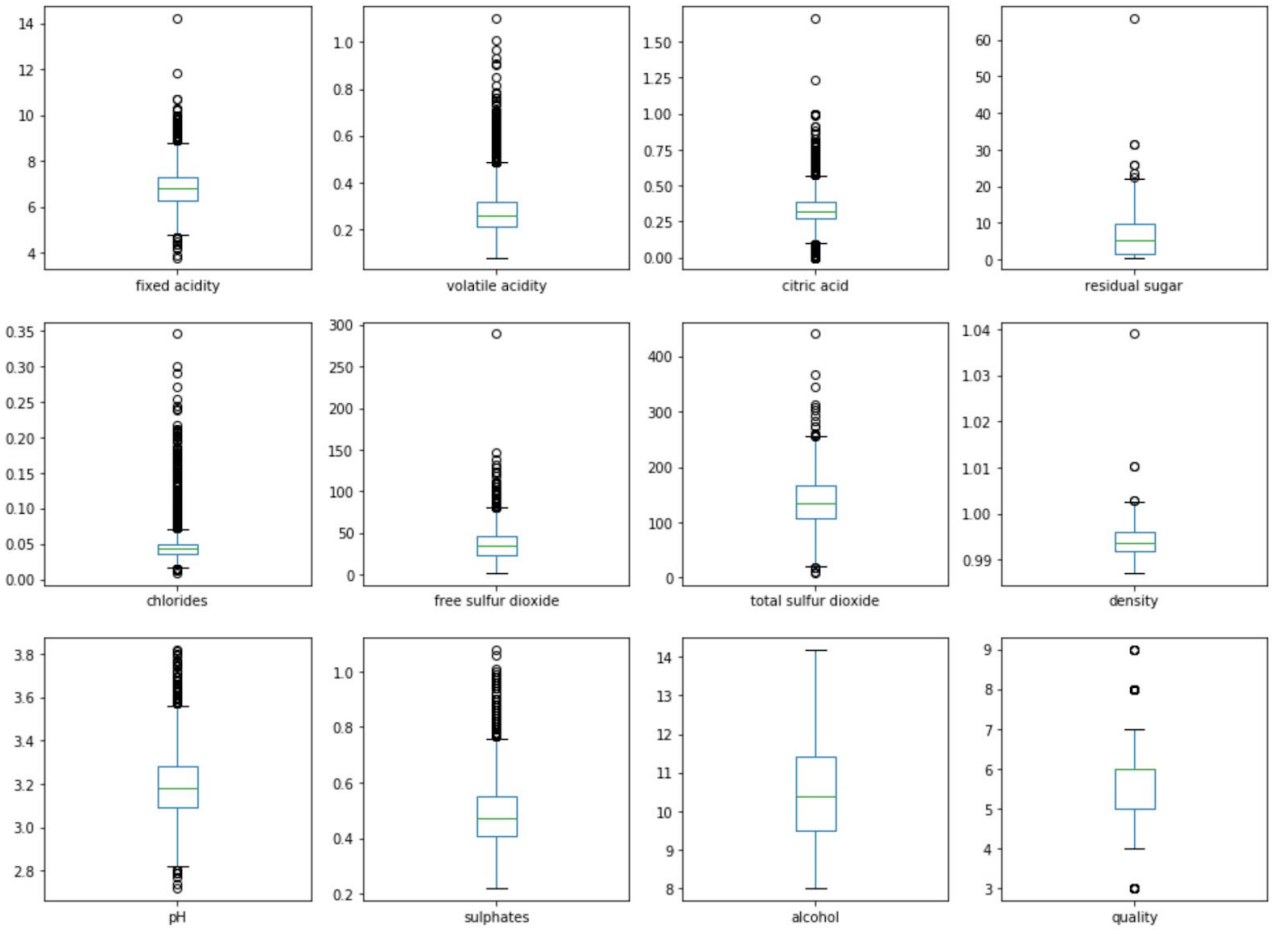
```
quality
3      20
4     163
5    1457
6    2198
7     880
8     175
9       5
dtype: int64
```

Listing 3: Univariable plots to better understand each attribute. Multivariable plots to better understand the relationships between attributes

a) Univariable plot (box and whisker)

```
In [61]:
# Data visualizations

# box and whisker plots
dataset.plot(kind='box', subplots=True, layout=(3,4), sharex=False, sharey=False, figsize=(16,12))
pyplot.show()
```

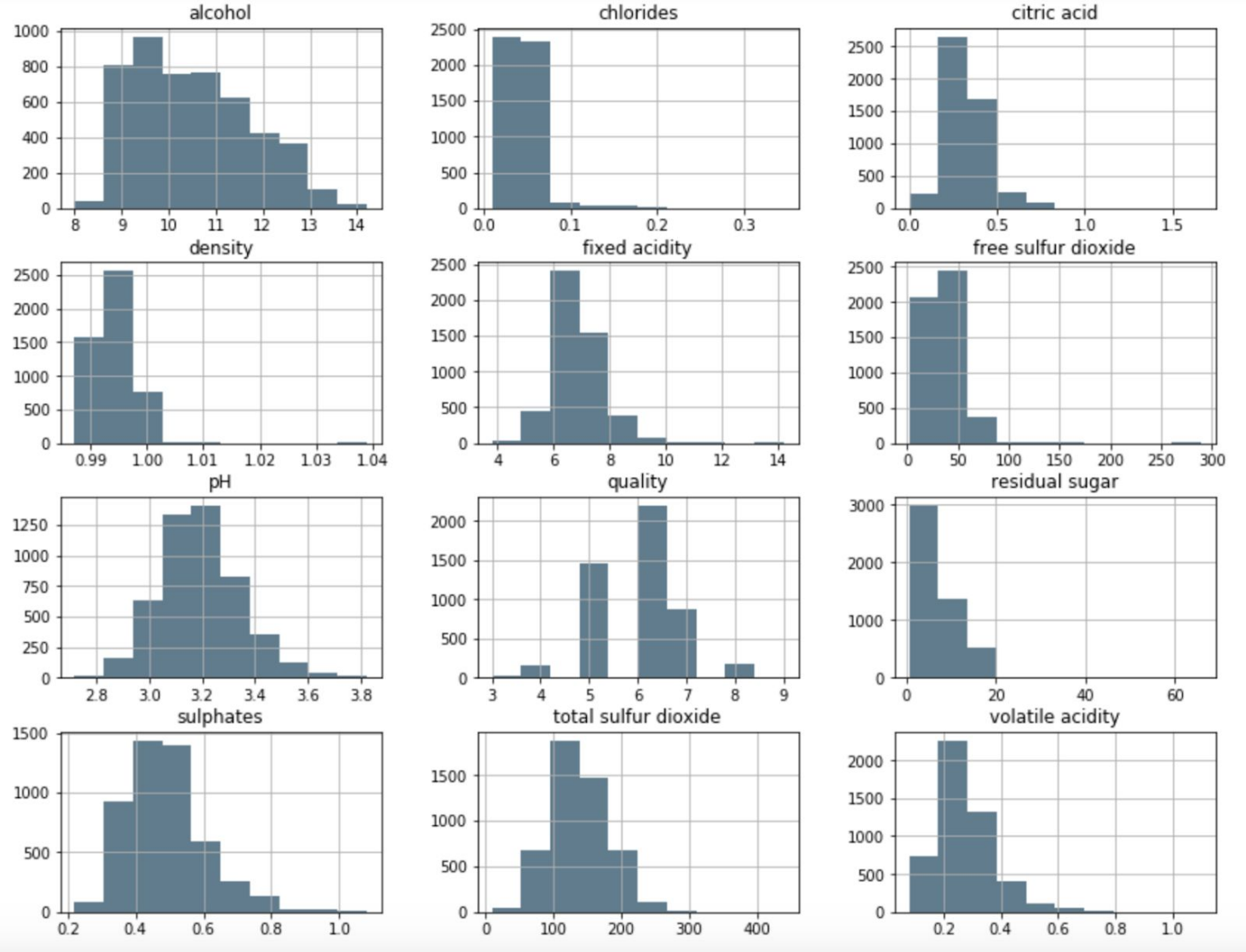



**Assessment:** The quality of more white wine is center around 5-6, putting quality of the wine around 2nd percentage tile. Fixed acidity falls around 25-40% percentile with a lot of out liners from 3-5 and 7-14. Alcohol

seems to very fairly contained without many outliers, but most attributes cannot be grouped very cohesively in a box.

b) Visualize the data-set using histogram plots

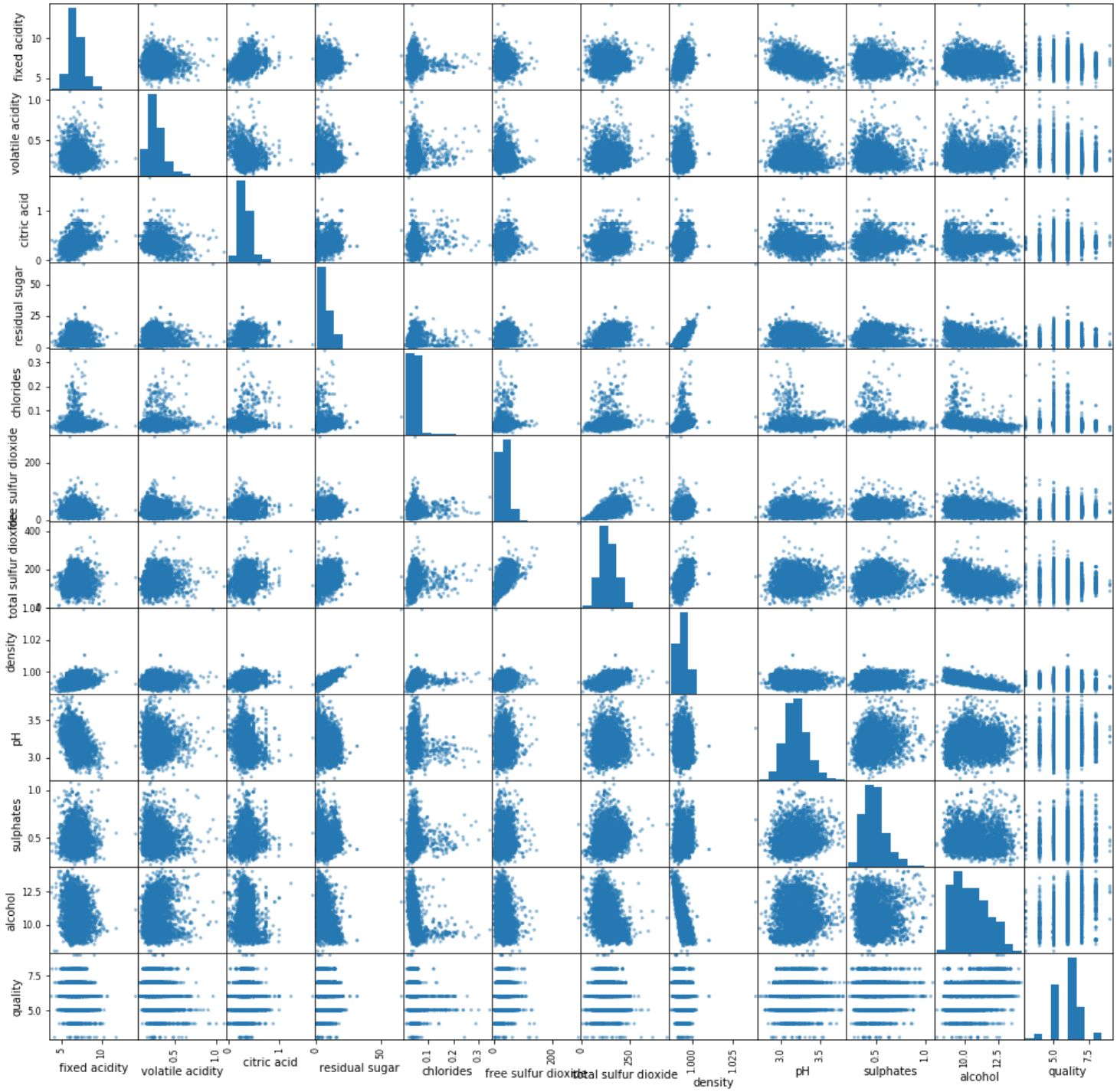
```
In [69]:
# histograms
dataset.hist(color='#607c8e',figsize=(14,11))
pyplot.show()
```




**Assessment:** Histogram plots of alcohol, chlorides, citric acid, density, free sulfur dioxide, total sulfur dioxide and volatile acidity, skew positive. pH histogram plot looks like normal distribution.

c) Visualize the dataset using scatter plots

```
In [84]:
1 # scatter plot matrix
2 scatter_matrix(dataset,figsize=(17,17));
3 pyplot.show()
```




Assessment: The diagonal values are bar plots the x values and y values are the same.

Listing 4: Separate out a validation dataset. Setup the test harness to use 10-fold cross-validation (not in this code, but you might want to include it.) Build 5 different models to predict species from flower measurements. Select the best model.

a) Create validation set

In [74]:
```
# Prepare Data

# Split-out validation dataset
array = dataset.values
X = array[:,0:11]
Y = array[:,11]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=validation_size,
                                                                random_state=seed)

```

b) Build models **(**Logistic Regression (LR), Linear Discriminant Analysis (LDA), k- Nearest Neighbors (KNN), Classifications and Regression Trees (CART), Gaussian Naive Bayes (NB), Support Vector Machines (SVM) and select the best model.

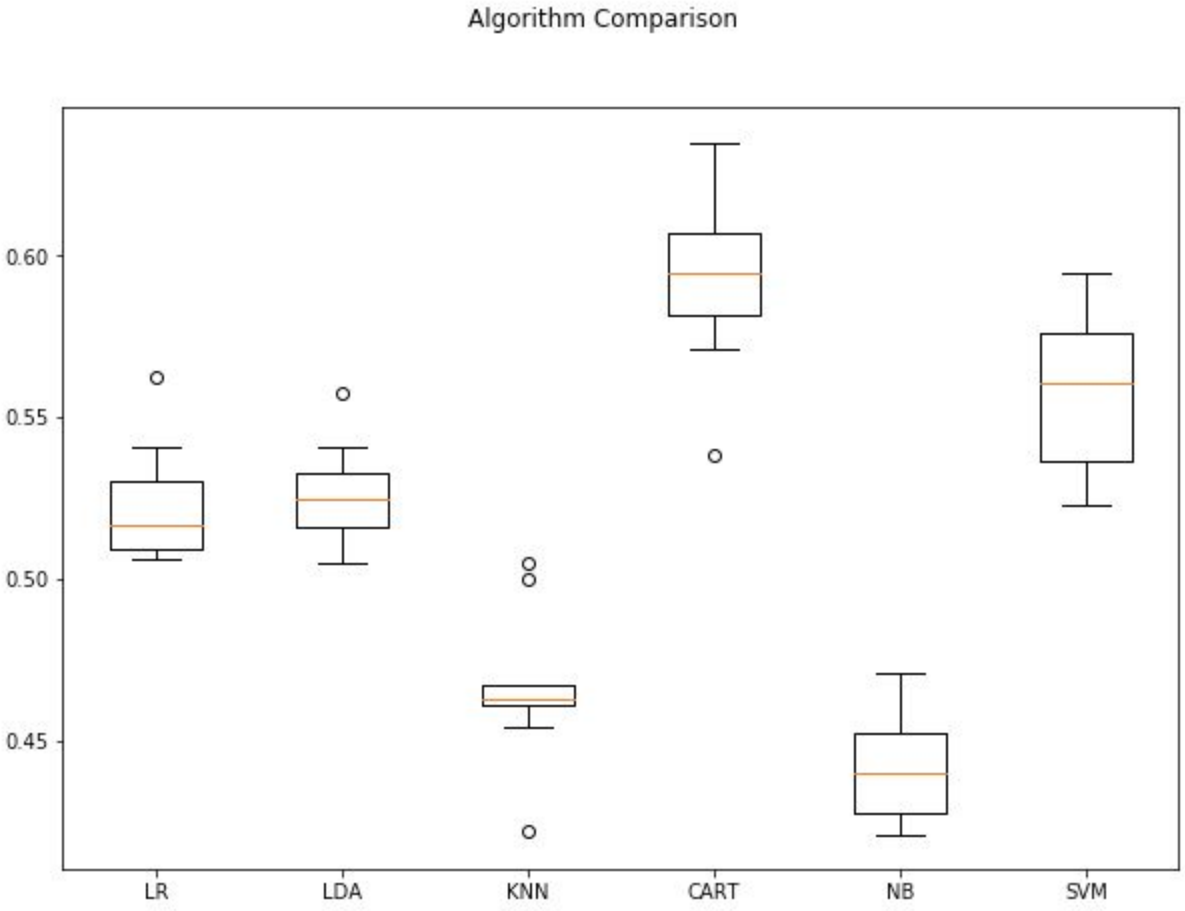
```
# Spot-Check Algorithms
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=10, random_state=seed)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

# Compare Algorithms
fig = pyplot.figure(figsize=(10,7))
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
pyplot.boxplot(results)
ax.set_xticklabels(names)
pyplot.show()
```

```
LR: 0.522467 (0.017132)
LDA: 0.525787 (0.015002)
KNN: 0.466297 (0.021989)
CART: 0.592905 (0.025656)
NB: 0.441816 (0.015709)
```

```
SVM: 0.556400 (0.023142)
```

c) Comparing Algorithms




**Assessment:** The accuracy of each classifier are around 40-60%. One reason the accuracy is low is because there are 10 different outputs which will result in many misclassifications.

## Listing 5: Making Prediction on the validation data-set

### a) Prediction using K-Nearest Neighbor

In [79]:

```
import numpy as np

# Make predictions on validation dataset using K-Nearest Neighbor
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
predictions = knn.predict(X_validation)

print('Prediction on Validation dataset using K-Nearest Neighbor')
print('Validation Accuracy = ', accuracy_score(Y_validation, predictions))
print('Confusion Matrix = '); print(confusion_matrix(Y_validation, predictions));print()
print('Normalized Confusion Matrix = ')
print(np.around((confusion_matrix(Y_validation, predictions) /
                 confusion_matrix(Y_validation, predictions).max()),5));
print();
print(classification_report(Y_validation, predictions))
```

```
Prediction on Validation dataset using K-Nearest Neighbor
Validation Accuracy =  0.46938775510204084
Confusion Matrix =
[[  0   2   0   4   0   0   0]
 [  1   4  22   9   3   0   0]
 [  0   4 142 116  15   3   0]
 [  0   3 130 266  54   3   0]
 [  0   1  37  78  43   3   0]
 [  0   0   4  19   8   5   0]
 [  0   0   0   1   0   0   0]]

Normalized Confusion Matrix =
[[0.      0.00752 0.      0.01504 0.      0.      0.     ]
 [0.00376 0.01504 0.08271 0.03383 0.01128 0.      0.     ]
 [0.      0.01504 0.53383 0.43609 0.05639 0.01128 0.     ]
 [0.      0.01128 0.48872 1.      0.20301 0.01128 0.     ]
 [0.      0.00376 0.1391  0.29323 0.16165 0.01128 0.     ]
 [0.      0.      0.01504 0.07143 0.03008 0.0188  0.     ]
 [0.      0.      0.      0.00376 0.      0.      0.     ]]

              precision    recall  f1-score   support

         3.0       0.00      0.00      0.00         6
         4.0       0.29      0.10      0.15        39
         5.0       0.42      0.51      0.46       280
         6.0       0.54      0.58      0.56       456
         7.0       0.35      0.27      0.30       162
         8.0       0.36      0.14      0.20        36
         9.0       0.00      0.00      0.00         1

    accuracy                           0.47       980
   macro avg       0.28      0.23      0.24       980
weighted avg       0.45      0.47      0.46       980
```

**Assessment:** Assessment: There are a lot of misclassifications (show in normalized confusion matrix, most values on diagonal are less than one), which results in low classification accuracy 47%.

**b) Prediction using decision tree classifier**

In [80]:

```
# Make predictions on validation dataset using Decision Tree Classifier
CART = DecisionTreeClassifier()
CART.fit(X_train, Y_train)
predictions = CART.predict(X_validation)
print('Prediction on Validation dataset using Decision Tree Classifier')
print('Validation Accuracy = ', accuracy_score(Y_validation, predictions))
print('Confusion Matrix = '); print(confusion_matrix(Y_validation, predictions));print()
print('Normalized Confusion Matrix = ')
print(np.around((confusion_matrix(Y_validation, predictions)/confusion_matrix(Y_validation,
                                                                               predictions).max()),5));
print();

print(classification_report(Y_validation, predictions))
```

```
Prediction on Validation dataset using Decision Tree Classifier
Validation Accuracy =  0.6214285714285714
Confusion Matrix =
[[  0   0   3   3   0   0   0]
 [  0   6  16  14   3   0   0]
 [  0  13 179  77  10   1   0]
 [  1   6  80 306  54   9   0]
 [  0   0   7  46 103   6   0]
 [  0   0   4  10   7  15   0]
 [  0   0   0   1   0   0   0]]

Normalized Confusion Matrix =
[[0.      0.      0.0098  0.0098  0.      0.      0.     ]
 [0.      0.01961 0.05229 0.04575 0.0098  0.      0.     ]
 [0.      0.04248 0.58497 0.25163 0.03268 0.00327 0.     ]
 [0.00327 0.01961 0.26144 1.      0.17647 0.02941 0.     ]
 [0.      0.      0.02288 0.15033 0.3366  0.01961 0.     ]
 [0.      0.      0.01307 0.03268 0.02288 0.04902 0.     ]
 [0.      0.      0.      0.00327 0.      0.      0.     ]]

              precision    recall  f1-score   support

         3.0       0.00      0.00      0.00         6
         4.0       0.24      0.15      0.19        39
         5.0       0.62      0.64      0.63       280
         6.0       0.67      0.67      0.67       456
         7.0       0.58      0.64      0.61       162
         8.0       0.48      0.42      0.45        36
         9.0       0.00      0.00      0.00         1

    accuracy                           0.62       980
   macro avg       0.37      0.36      0.36       980
weighted avg       0.61      0.62      0.62       980
```

**Assessment:** There are a lot of misclassifications (shows in normalized confusion matrix, most values on diagonal are less than one), which results in low classification accuracy 62%.

## c) Prediction using Logistic Regression

In [87]:

```
# Make predictions on validation dataset using Logistic Regression
LR = LogisticRegression()
LR.fit(X_train, Y_train)
predictions = LR.predict(X_validation)
print('Prediction on Validation dataset using Logistic Regression')
print('Validation Accuracy = ', accuracy_score(Y_validation, predictions))
print('Confusion Matrix = '); print(confusion_matrix(Y_validation, predictions));print()
print('Normalized Confusion Matrix = ')
print(np.around((confusion_matrix(Y_validation, predictions)/confusion_matrix(Y_validation,
                                                                              predictions).max()),5));
print();

print(classification_report(Y_validation, predictions))

```

```
Prediction on Validation dataset using Logistic Regression
Validation Accuracy =  0.5408163265306123
Confusion Matrix =
[[  0   0   3   3   0   0   0]
 [  0   0  29  10   0   0   0]
 [  0   0 155 125   0   0   0]
 [  0   0  86 359  11   0   0]
 [  0   0   8 138  16   0   0]
 [  0   0   3  31   2   0   0]
 [  0   0   0   1   0   0   0]]

Normalized Confusion Matrix =
[[0.      0.      0.00836 0.00836 0.      0.      0.     ]
 [0.      0.      0.08078 0.02786 0.      0.      0.     ]
 [0.      0.      0.43175 0.34819 0.      0.      0.     ]
 [0.      0.      0.23955 1.      0.03064 0.      0.     ]
 [0.      0.      0.02228 0.3844  0.04457 0.      0.     ]
 [0.      0.      0.00836 0.08635 0.00557 0.      0.     ]
 [0.      0.      0.      0.00279 0.      0.      0.     ]]

              precision    recall  f1-score   support

         3.0       0.00      0.00      0.00         6
         4.0       0.00      0.00      0.00        39
         5.0       0.55      0.55      0.55       280
         6.0       0.54      0.79      0.64       456
         7.0       0.55      0.10      0.17       162
         8.0       0.00      0.00      0.00        36
         9.0       0.00      0.00      0.00         1

    accuracy                           0.54       980
   macro avg       0.23      0.21      0.19       980
weighted avg       0.50      0.54      0.48       980
```

**Assessment:** There are a lot of misclassifications (shows in normalized confusion matrix, most values on diagonal are less than one), which results in low classification accuracy 54%. The misclassifications of are the values greater than one that are not on diagonal of normalized confusion matrix.

## d) Prediction Linear Discriminant Analysis

In [91]:

```
# Make predictions on validation dataset using Linear Discriminant Analysis
LDA = LinearDiscriminantAnalysis()
LDA.fit(X_train, Y_train)
predictions = LDA.predict(X_validation)
print('Prediction on Validation dataset using Linear Discriminant Analysis')
print('Validation Accuracy = ', accuracy_score(Y_validation, predictions))
print('Confusion Matrix = '); print(confusion_matrix(Y_validation, predictions));print()
print('Normalized Confusion Matrix = ')
print(np.around((confusion_matrix(Y_validation, predictions)/confusion_matrix(Y_validation,
                                                                              predictions).max()),5));
print();

print(classification_report(Y_validation, predictions))
```

```
Prediction on Validation dataset using Linear Discriminant Analysis
Validation Accuracy =  0.5530612244897959
Confusion Matrix =
[[  0   1   1   4   0   0   0]
 [  1   9  19   9   1   0   0]
 [  2  10 145 120   3   0   0]
 [  2   2  74 341  36   0   1]
 [  0   0   5 110  47   0   0]
 [  0   0   4  22  10   0   0]
 [  0   0   0   0   1   0   0]]

Normalized Confusion Matrix =
[[0.      0.00293 0.00293 0.01173 0.      0.      0.     ]
 [0.00293 0.02639 0.05572 0.02639 0.00293 0.      0.     ]
 [0.00587 0.02933 0.42522 0.35191 0.0088  0.      0.     ]
 [0.00587 0.00587 0.21701 1.      0.10557 0.      0.00293]
 [0.      0.      0.01466 0.32258 0.13783 0.      0.     ]
 [0.      0.      0.01173 0.06452 0.02933 0.      0.     ]
 [0.      0.      0.      0.      0.00293 0.      0.     ]]

              precision    recall  f1-score   support

         3.0       0.00      0.00      0.00         6
         4.0       0.41      0.23      0.30        39
         5.0       0.58      0.52      0.55       280
         6.0       0.56      0.75      0.64       456
         7.0       0.48      0.29      0.36       162
         8.0       0.00      0.00      0.00        36
         9.0       0.00      0.00      0.00         1

    accuracy                           0.55       980
   macro avg       0.29      0.26      0.26       980
weighted avg       0.52      0.55      0.53       980
```

**Assessment:** Using the Linear Discriminant Analysis and classification accuracy is 55%. There are also a lot of misclassifications, see normalized confusion matrix, where most diagonal values are below 1. The misclassifications of are the values greater than one that are not on diagonal of normalized confusion matrix.

## e) Prediction on Gaussian NB

In [89]:

```
# Make predictions on validation dataset using Gaussian NB
NB = GaussianNB()
NB.fit(X_train, Y_train)
predictions = NB.predict(X_validation)
print('Prediction on Validation dataset using Gaussian NB()')
print('Validation Accuracy = ', accuracy_score(Y_validation, predictions))
print('Confusion Matrix = '); print(confusion_matrix(Y_validation, predictions));print()
print('Normalized Confusion Matrix = ')
print(np.around((confusion_matrix(Y_validation, predictions)/confusion_matrix(Y_validation,
                                                                              predictions).max()),5));
print();

print(classification_report(Y_validation, predictions))
```

```
Prediction on Validation dataset using Gaussian NB()
Validation Accuracy =  0.4377551020408163
Confusion Matrix =
[[  1   2   0   1   2   0   0]
 [  2  12  12   8   5   0   0]
 [  5  16 154  73  32   0   0]
 [  5   5 136 149 156   4   1]
 [  0   0  17  30 112   3   0]
 [  0   0   6  10  19   1   0]
 [  0   0   0   0   1   0   0]]

Normalized Confusion Matrix =
[[0.00641 0.01282 0.      0.00641 0.01282 0.      0.     ]
 [0.01282 0.07692 0.07692 0.05128 0.03205 0.      0.     ]
 [0.03205 0.10256 0.98718 0.46795 0.20513 0.      0.     ]
 [0.03205 0.03205 0.87179 0.95513 1.      0.02564 0.00641]
 [0.      0.      0.10897 0.19231 0.71795 0.01923 0.     ]
 [0.      0.      0.03846 0.0641  0.12179 0.00641 0.     ]
 [0.      0.      0.      0.      0.00641 0.      0.     ]]

              precision    recall  f1-score   support

         3.0       0.08      0.17      0.11         6
         4.0       0.34      0.31      0.32        39
         5.0       0.47      0.55      0.51       280
         6.0       0.55      0.33      0.41       456
         7.0       0.34      0.69      0.46       162
         8.0       0.12      0.03      0.05        36
         9.0       0.00      0.00      0.00         1

    accuracy                           0.44       980
   macro avg       0.27      0.30      0.26       980
weighted avg       0.47      0.44      0.43       980
```

**Assessments:** Using Gaussian NB classifier we arrived at classification accuracy 44% where there was a lot of misclassifications. Confusion matrix shows a lot of misclassifications because a lot of diagonal values are less than one.

## f) Prediction using SVM

In [92]:
```
# Make predictions on validation dataset using SVM
SVM = SVC()
SVM.fit(X_train, Y_train)
predictions = SVM.predict(X_validation)
print('Prediction on Validation dataset using SVM')
print('Validation Accuracy = ', accuracy_score(Y_validation, predictions))
print('Confusion Matrix = '); print(confusion_matrix(Y_validation, predictions));print()
print('Normalized Confusion Matrix = ')
print(np.around((confusion_matrix(Y_validation, predictions)/confusion_matrix(Y_validation,
                                                                              predictions).max()),5));
print();

print(classification_report(Y_validation, predictions))
```

```
Prediction on Validation dataset using SVM
Validation Accuracy =  0.5714285714285714
Confusion Matrix =
[[  0   0   1   5   0   0   0]
 [  0   1  14  22   2   0   0]
 [  0   1 144 132   3   0   0]
 [  0   0  69 358  29   0   0]
 [  0   0  16  95  51   0   0]
 [  0   0   3  24   3   6   0]
 [  0   0   0   1   0   0   0]]

Normalized Confusion Matrix =
[[0.      0.      0.00279 0.01397 0.      0.      0.     ]
 [0.      0.00279 0.03911 0.06145 0.00559 0.      0.     ]
 [0.      0.00279 0.40223 0.36872 0.00838 0.      0.     ]
 [0.      0.      0.19274 1.      0.08101 0.      0.     ]
 [0.      0.      0.04469 0.26536 0.14246 0.      0.     ]
 [0.      0.      0.00838 0.06704 0.00838 0.01676 0.     ]
 [0.      0.      0.      0.00279 0.      0.      0.     ]]

              precision    recall  f1-score   support

         3.0       0.00      0.00      0.00         6
         4.0       0.50      0.03      0.05        39
         5.0       0.58      0.51      0.55       280
         6.0       0.56      0.79      0.66       456
         7.0       0.58      0.31      0.41       162
         8.0       1.00      0.17      0.29        36
         9.0       0.00      0.00      0.00         1

    accuracy                           0.57       980
   macro avg       0.46      0.26      0.28       980
weighted avg       0.58      0.57      0.54       980
```

**Assessment:** Using SVM the classification accuracy is 57% where there are a lot of misclassifications. The normalized confusion matrix shows values below 1 and the diagonal values are all below 1.