In [109]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [110]:
```
df=pd.read_csv('wine.csv')
```

In [111]:
```
df.head()
```

Out[111]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

In [112]:
```
#data quality check
df.isnull()
```

Out[112]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alc
0	False	False	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	F

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alc
...	...	...	...	...	...	...	...	...	...	...	
1594	False	False	False	False	False	False	False	False	False	False	F
1595	False	False	False	False	False	False	False	False	False	False	F
1596	False	False	False	False	False	False	False	False	False	False	F
1597	False	False	False	False	False	False	False	False	False	False	F
1598	False	False	False	False	False	False	False	False	False	False	F

1599 rows × 12 columns

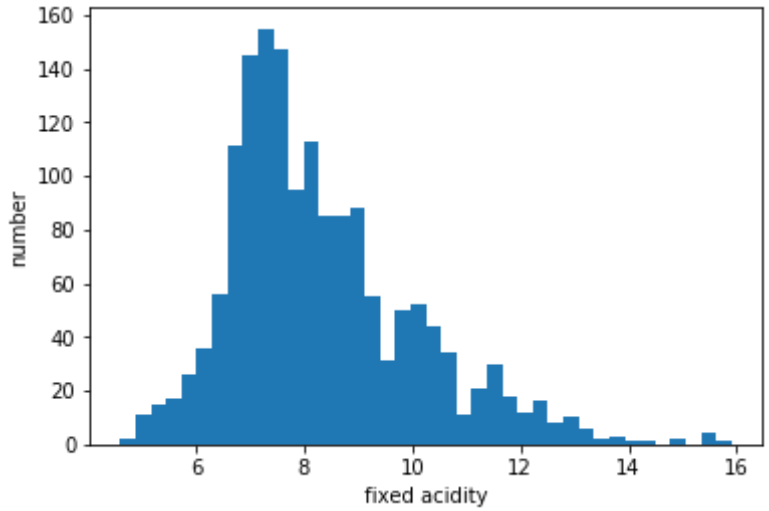
In [113]:
```
#there are 0 null value
```

In [114]:
```
fixedacidity=df["fixed acidity"]
volatileacidity=df["volatile acidity"]
citricacid=df["citric acid"]
residualsugar=df["residual sugar"]
chlorides=df["chlorides"]
fs=df["free sulfur dioxide"]
ts=df["total sulfur dioxide"]
dens=df["density"]
ph=df["pH"]
sulph=df["sulphates"]
alc=df["alcohol"]
qua=df["quality"]
import numpy as np
np.sqrt(1599)
```

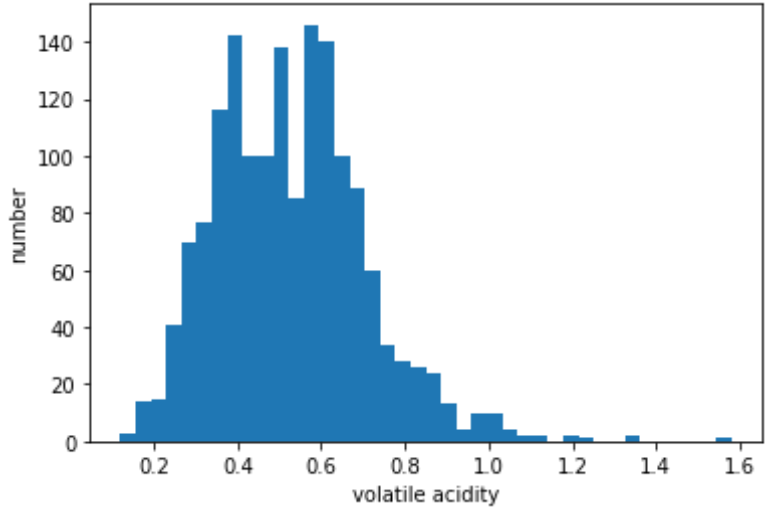
Out[114]:
```
39.98749804626441
```

In [115]:
```
plt.hist(fixedacidity,bins=40)
plt.xlabel("fixed acidity")
```

```
plt.ylabel("number")
plt.show()
```

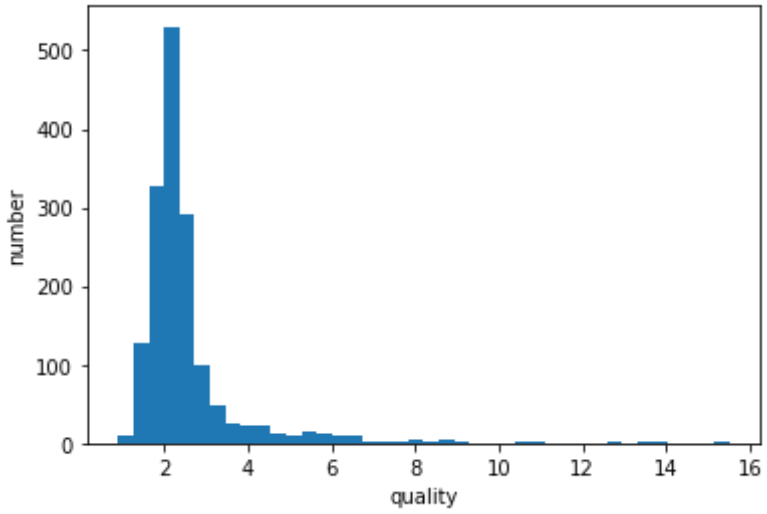


In [116]:
```
plt.hist(volatileacidity,bins=40)
plt.xlabel("volatile acidity")
plt.ylabel("number")
plt.show()
```

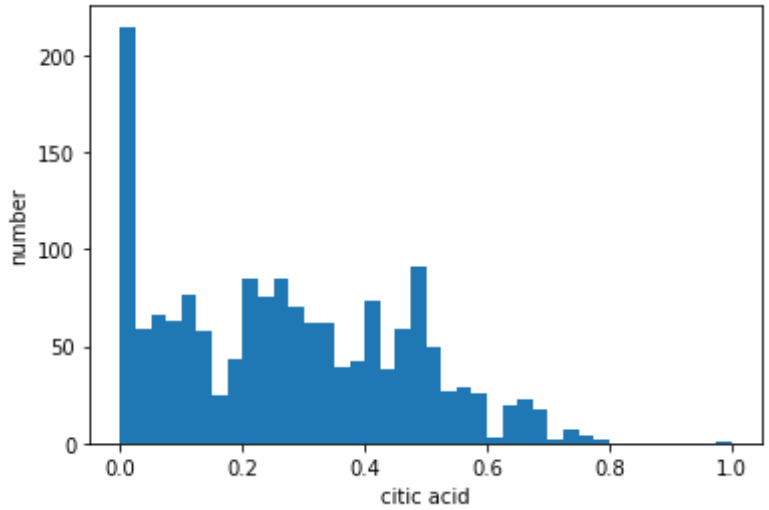


In [117]:
```
plt.hist(residualsugar,bins=40)
plt.xlabel("quality")
plt.ylabel("number")
```

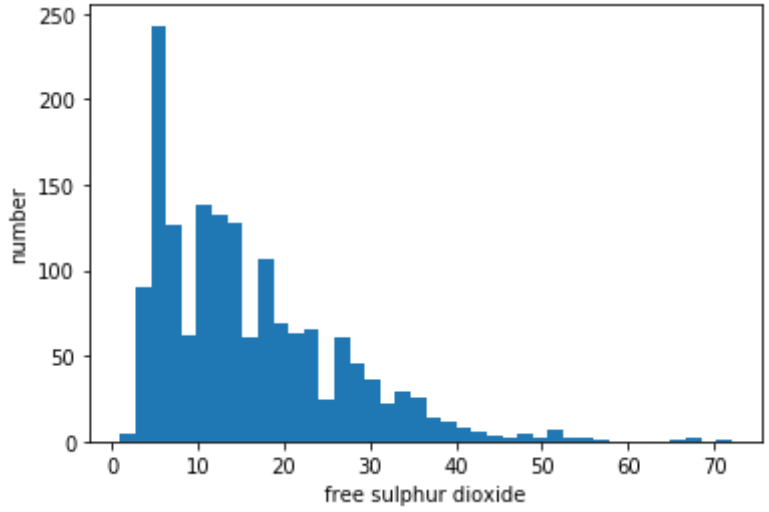
Out[117]:
```
Text(0, 0.5, 'number')
```



In [118]:
```
plt.hist(citricacid,bins=40)
plt.xlabel("citic acid")
plt.ylabel("number")
plt.show()
```

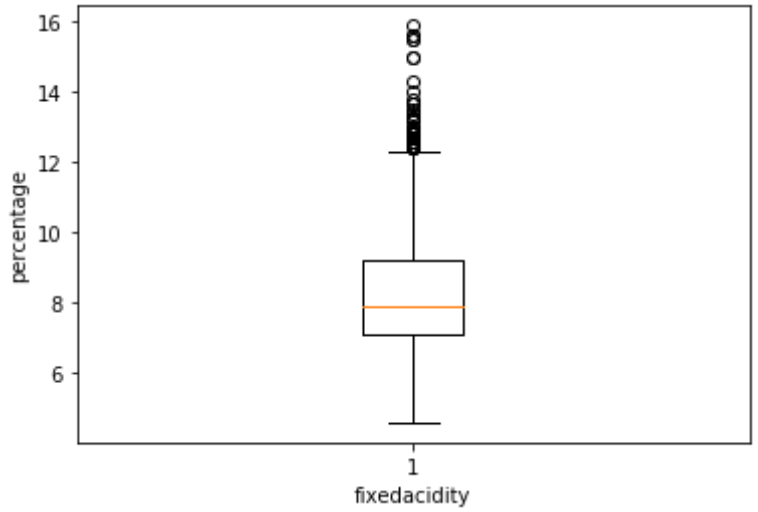


In [119]:
```
plt.hist(fs,bins=40)
plt.xlabel("free sulphur dioxide")
plt.ylabel("number")
plt.show()
```

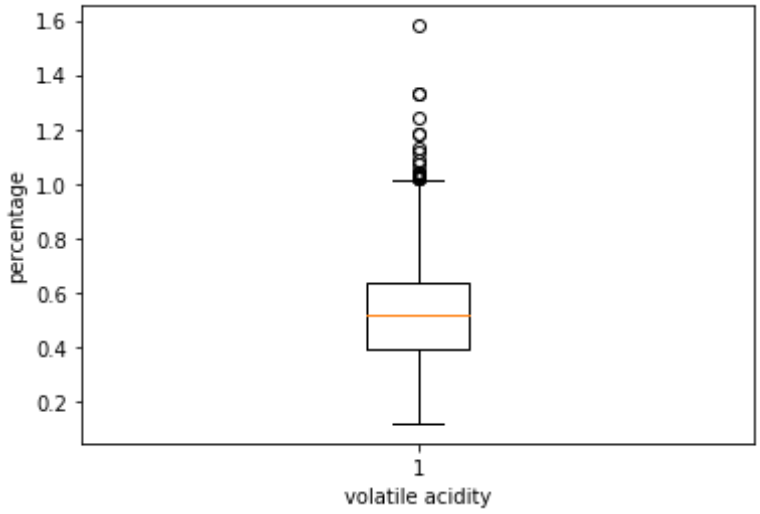


In [120]:
```
plt.boxplot(fixedacidity)
```

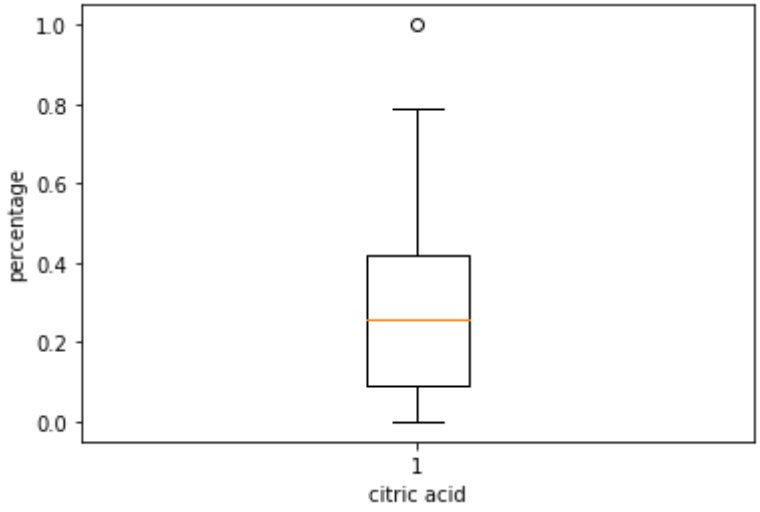
```
plt.xlabel("fixedacidity")
plt.ylabel("percentage")
plt.show()
```



In [121]:
```
plt.boxplot(volatileacidity)
plt.xlabel("volatile acidity")
plt.ylabel("percentage")
plt.show()
```

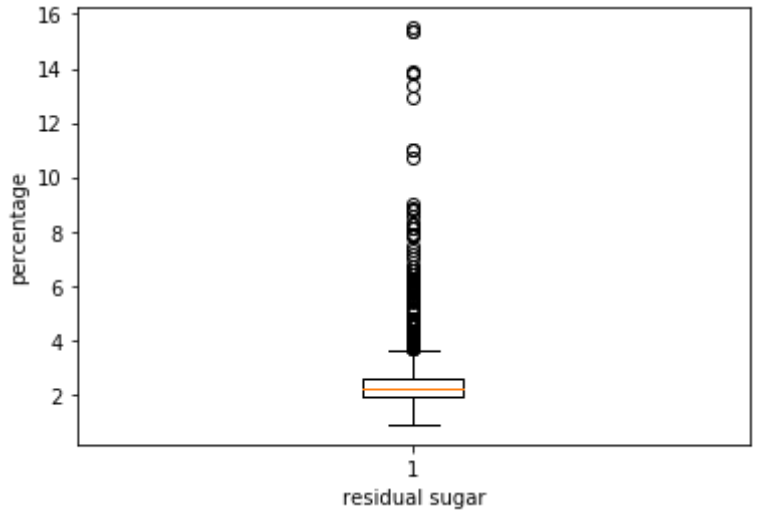


In [122]:
```
plt.boxplot(citricacid)
plt.xlabel("citric acid")
plt.ylabel("percentage")
plt.show()
```

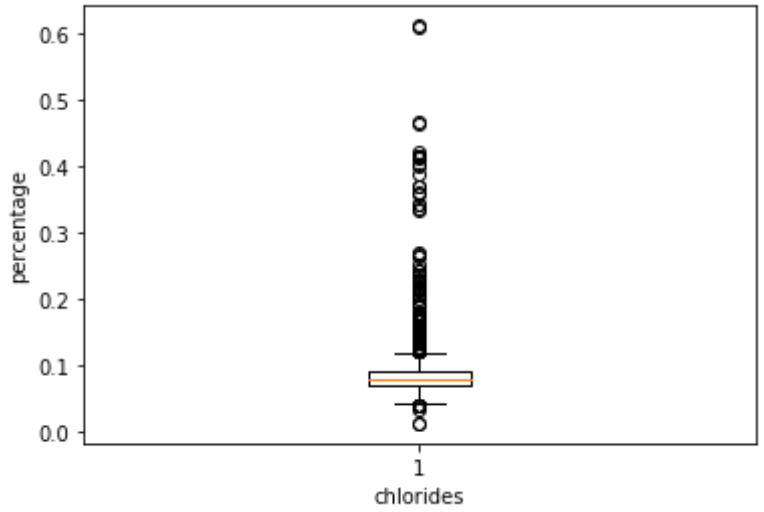


In [123]:
```
plt.boxplot(residualsugar)
```

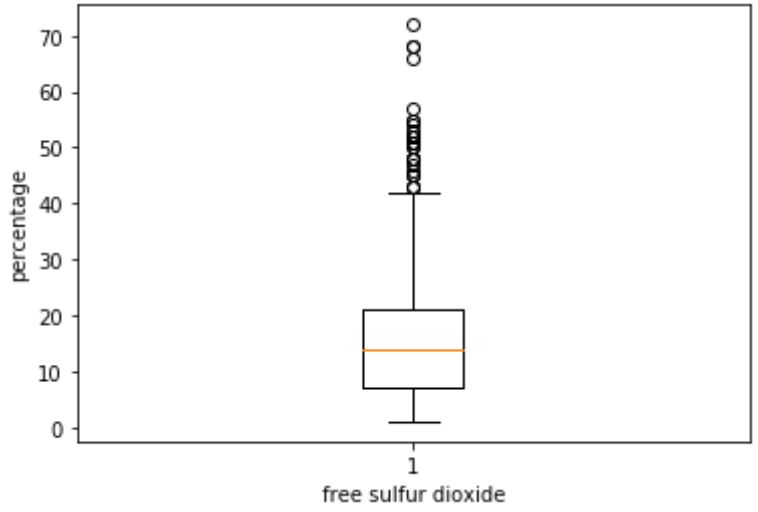
```
plt.xlabel("residual sugar")
plt.ylabel("percentage")
plt.show()
```



In [124]:
```
plt.boxplot(chlorides)
plt.xlabel("chlorides")
plt.ylabel("percentage")
plt.show()
```

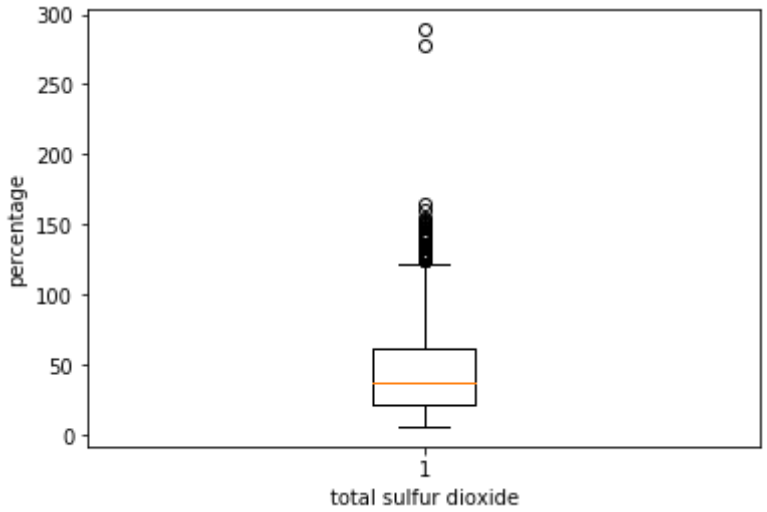


In [125]:
```
plt.boxplot(fs)
plt.xlabel("free sulfur dioxide")
plt.ylabel("percentage")
plt.show()
```

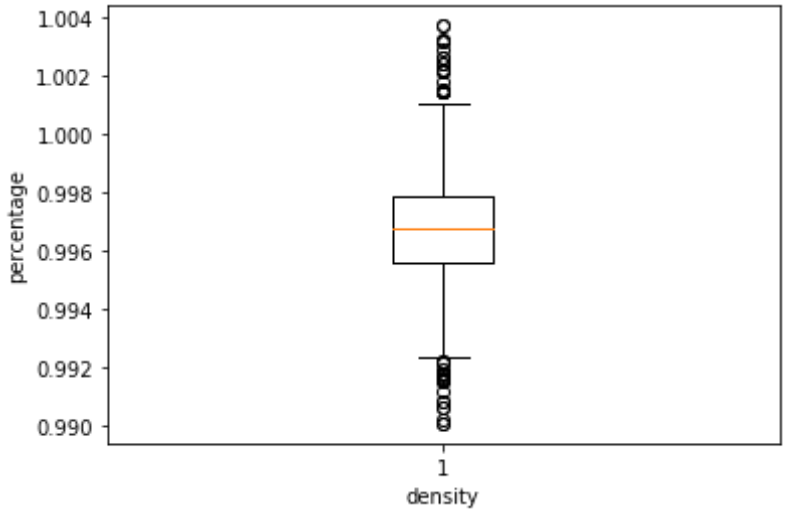


In [126]:
```
plt.boxplot(ts)
```

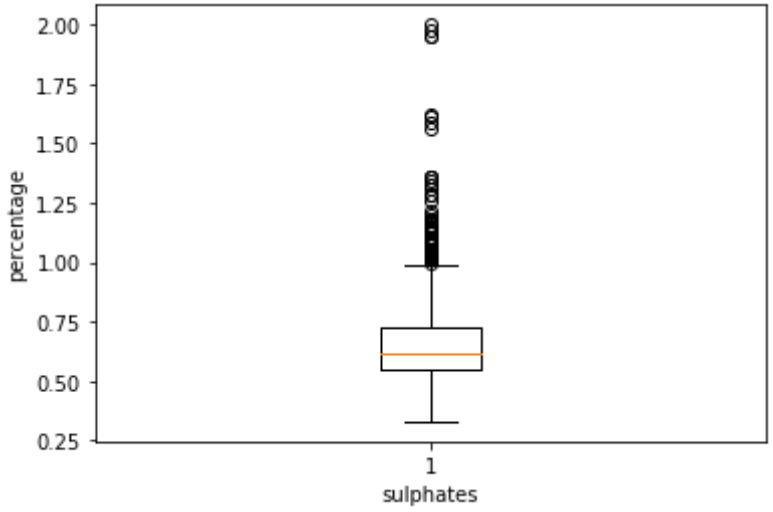
```
plt.xlabel("total sulfur dioxide")
plt.ylabel("percentage")
plt.show()
```



In [127]:
```
plt.boxplot(dens)
plt.xlabel("density")
plt.ylabel("percentage")
plt.show()
```

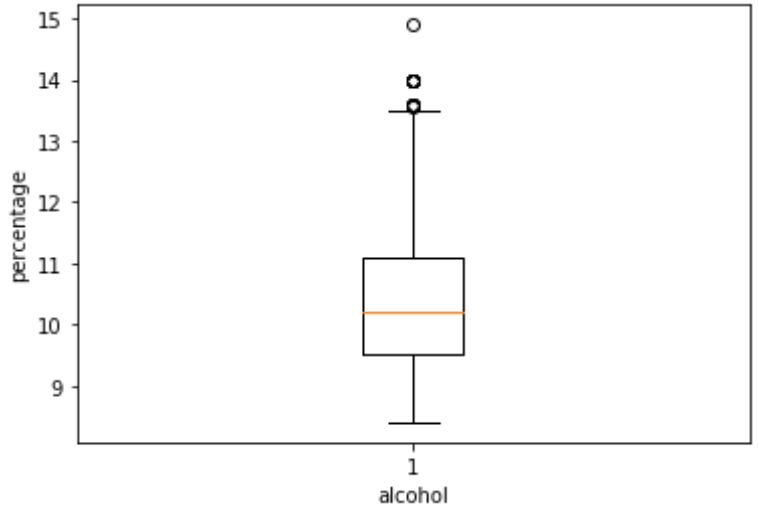


In [128]:
```
plt.boxplot(sulph)
plt.xlabel("sulphates")
plt.ylabel("percentage")
plt.show()
```



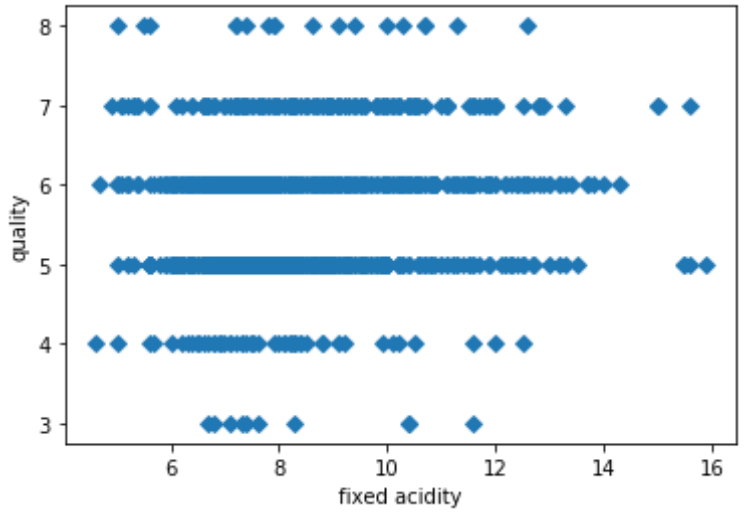
plt.boxplot(ph) plt.xlabel("sulphates") plt.ylabel("percentage") plt.show()

In [129]:
```
plt.boxplot(alc)
plt.xlabel("alcohol")
plt.ylabel("percentage")
plt.show()
```

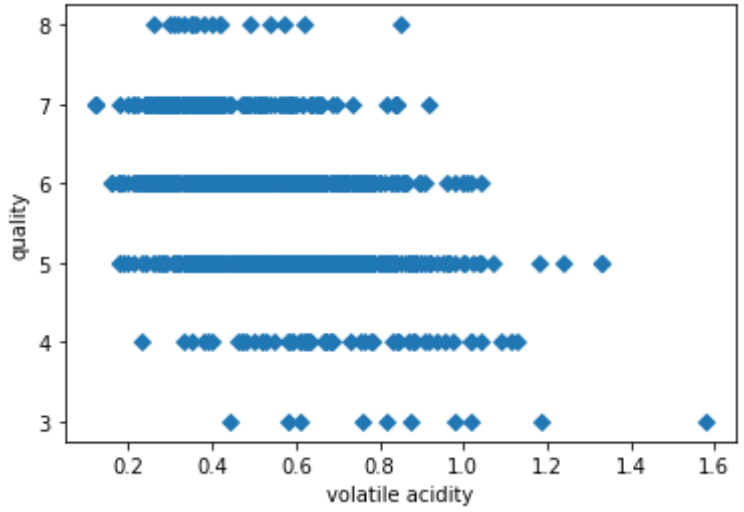


In [130]:
```
#draw a minimum of 5 scatter plot
```

In [131]:
```
_=plt.plot(fixedacidity,qua,marker='D',linestyle='none')
_=plt.xlabel("fixed acidity")
_=plt.ylabel("quality")
plt.show()
```

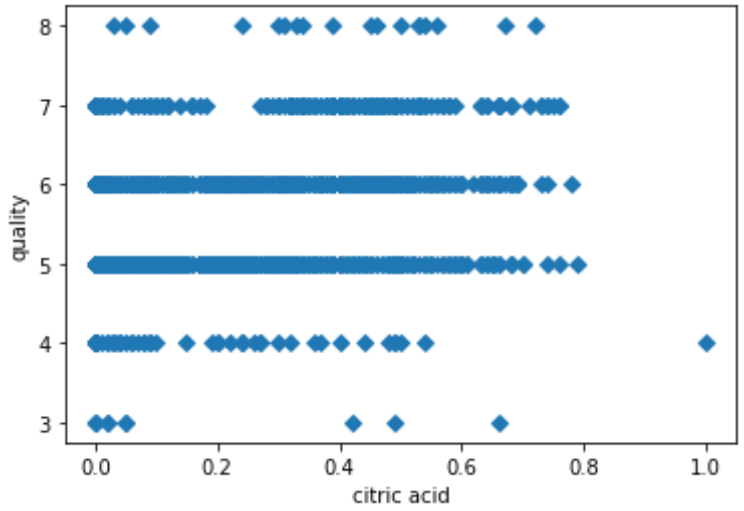


In [132]:
```
_=plt.plot(volatileacidity,qua,marker='D',linestyle='none')
_=plt.xlabel("volatile acidity")
_=plt.ylabel("quality")
_=plt.show()
```

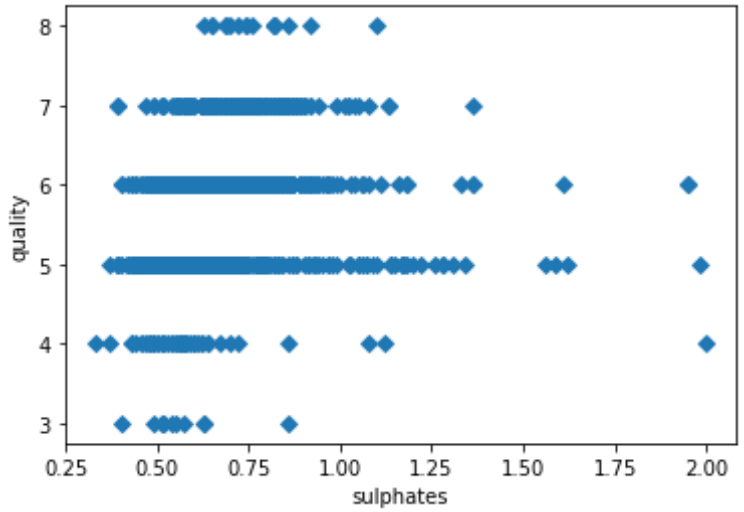


In [133]:
```
_=plt.plot(citricacid,qua,marker='D',linestyle='none')
```

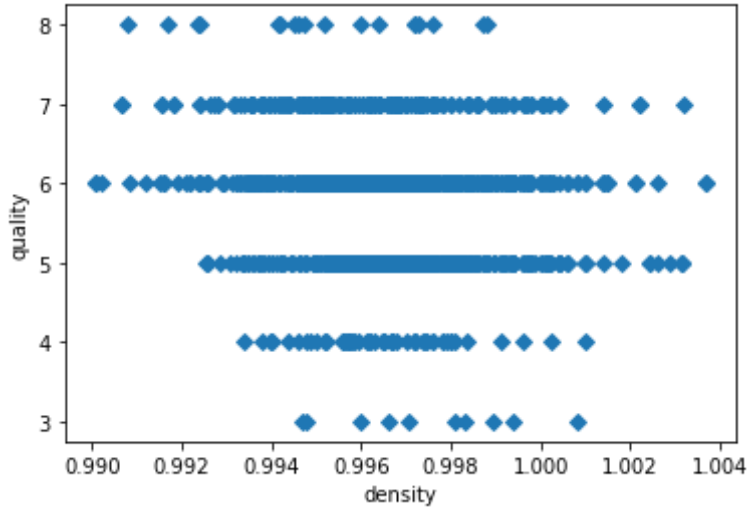
```
_=plt.xlabel("citric acid")
_=plt.ylabel("quality")
plt.show()
```



In [134]:
```
_=plt.plot(sulph,qua,marker='D',linestyle='none')
_=plt.xlabel("sulphates")
_=plt.ylabel("quality")
plt.show()
```



In [135]:
```
_=plt.plot(dens,qua,marker='D',linestyle='none')
plt.xlabel("density")
plt.ylabel("quality")
plt.show()
```



In [136]:
```
#modelling
```

In [137]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
```

In [138]:
```
y=df["quality"]
```

In [139]:
```
x=df.drop('quality',axis=1)
```

In [140]:
```
x
```

Out[140]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcc
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
...	...	...	...	...	...	...	...	...	...	...	
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	

1599 rows × 11 columns

In [141]:
```
y
```

Out[141]:
```
0       5
1       5
```

```
2       5
3       6
4       5
       ..
1594    5
1595    6
1596    6
1597    5
1598    6
Name: quality, Length: 1599, dtype: int64
```

In [142]:
```
#split the datasets using train test
```

In [143]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_sta
te=32)
```

In [144]:
```
#apply KNN classification
```

In [145]:
```
kn=KNeighborsClassifier(n_neighbors=2)
```

In [146]:
```
kn.fit(x,y)
```

Out[146]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
                     metric_params=None, n_jobs=None, n_neighbors=2, p=
2,
                     weights='uniform')
```

In [147]:
```
#calculate accuracy
```

In [148]:
```
print(kn.score(xtrain,ytrain))
```

```
0.8150134048257373
```

In [149]:
```
print(kn.score(xtest,ytest))
```

```
0.8
```

In [150]:
```
#there is no overfitting as well as underfitting
```

In [151]:
```
kn.predict(xtest)#predict on test set
```

Out[151]:
```
array([6, 7, 5, 5, 5, 6, 5, 5, 5, 5, 4, 3, 4, 6, 5, 5, 5, 7, 5, 5, 5,
6,
       4, 5, 6, 5, 5, 5, 5, 5, 6, 7, 7, 5, 5, 6, 5, 5, 6, 5, 7, 6, 6,
5,
       5, 6, 5, 4, 6, 5, 5, 5, 5, 6, 7, 6, 5, 4, 6, 6, 6, 6, 5, 5, 6,
5,
       7, 5, 5, 6, 5, 6, 5, 6, 5, 5, 5, 6, 5, 6, 6, 5, 5, 6, 6, 5, 6,
5,
       6, 5, 5, 5, 6, 4, 4, 5, 6, 5, 6, 5, 5, 5, 6, 5, 5, 6, 4, 6, 6,
5,
       6, 6, 6, 6, 5, 5, 5, 5, 6, 5, 5, 6, 5, 6, 5, 5, 4, 6, 5, 5, 5,
5,
       5, 4, 6, 6, 5, 5, 6, 6, 5, 6, 5, 6, 5, 5, 6, 5, 5, 5, 5, 5, 5,
6,
       5, 5, 5, 3, 5, 4, 4, 6, 6, 6, 6, 6, 5, 6, 5, 6, 6, 5, 6, 5, 5,
6,
       6, 5, 5, 8, 5, 7, 5, 6, 6, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 7, 6,
5,
       6, 5, 6, 5, 6, 6, 4, 5, 5, 7, 7, 7, 7, 6, 5, 5, 4, 6, 6, 6, 6,
5,
       6, 3, 6, 5, 5, 5, 5, 6, 5, 7, 4, 5, 5, 6, 6, 7, 6, 6, 5, 5, 7,
6,
       6, 5, 5, 6, 6, 6, 5, 7, 7, 5, 6, 5, 5, 6, 6, 5, 3, 7, 5, 5, 6,
5,
       6, 5, 6, 6, 6, 6, 5, 7, 5, 5, 6, 5, 6, 5, 6, 5, 6, 5, 6, 6, 6,
5,
       5, 4, 7, 5, 6, 5, 5, 5, 5, 5, 5, 6, 5, 6, 5, 5, 6, 6, 5, 6, 6,
4,
       5, 5, 5, 6, 5, 5, 5, 5, 6, 5, 5, 5, 6, 5, 5, 6, 7, 7, 5, 5, 5,
5,
       5, 5, 5, 6, 5, 7, 6, 5, 6, 6, 6, 5, 5, 6, 5, 5, 5, 5, 6, 5, 5,
6,
       5, 6, 6, 5, 5, 5, 5, 5, 5, 5, 6, 5, 5, 7, 6, 7, 5, 5, 5, 6, 6,
```

```
6,
       6, 7, 5, 5, 7, 6, 6, 5, 6, 5, 5, 7, 6, 6, 6, 6, 6, 5, 5, 6, 6,
5,
       5, 6, 5, 5, 6, 5, 5, 5, 6, 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 5,
6,
       5, 4, 6, 4, 5, 6, 6, 5, 6, 6, 5, 5, 6, 5, 5, 6, 5, 5, 5, 6, 4,
6,
       6, 5, 5, 5, 5, 5, 5, 6, 5, 5, 5, 5, 5, 5, 5, 8, 7, 6, 5, 5, 5,
5,
       5, 6, 5, 5, 5, 5, 6, 5, 6, 5, 5, 5, 5, 5, 7, 5, 5, 7], dtype=int
64)
```

In [152]:
```
print("fetures",df.quality)#fetures
```

```
fetures 0       5
1       5
2       5
3       6
4       5
       ..
1594    5
1595    6
1596    6
1597    5
1598    6
Name: quality, Length: 1599, dtype: int64
```

logistic regression

In [153]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
logreg=LogisticRegression()
```

In [154]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_sta
te=42)
```

In [155]:
```
logreg.fit(xtrain,ytrain)
```

```
C:\Users\USER\Anaconda3\lib\site-packages\sklearn\linear_model\logisti
c.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\USER\Anaconda3\lib\site-packages\sklearn\linear_model\logisti
c.py:469: FutureWarning: Default multi_class will be changed to 'auto'
in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

Out[155]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=
True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbos
e=0,
                   warm_start=False)
```

In [156]:
```
pred=logreg.predict(xtest)
```

In [157]:
```
from sklearn.metrics import confusion_matrix
```

In [158]:
```
confusion_matrix(ytest,pred)
```

Out[158]:
```
array([[  0,   0,   1,   0,   0,   0],
       [  0,   0,  13,   4,   0,   0],
       [  0,   0, 147,  47,   1,   0],
       [  0,   0,  78, 120,   2,   0],
       [  0,   0,   3,  55,   3,   0],
       [  0,   0,   0,   5,   1,   0]], dtype=int64)
```

In [159]:
```
from sklearn.metrics import classification_report
```

In [160]:
```
classification_report(ytest,pred)
```

```
C:\Users\USER\Anaconda3\lib\site-packages\sklearn\metrics\classificatio
n.py:1437: UndefinedMetricWarning: Precision and F-score are ill-define
```

```
d and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

Out[160]:
```
'              precision    recall  f1-score   support\n\n           3
      0.00      0.00      0.00         1\n           4       0.00
0.00      0.00        17\n           5       0.61      0.75      0.67
     195\n           6       0.52      0.60      0.56       200\n
     7       0.43      0.05      0.09        61\n           8       0.0
0      0.00      0.00         6\n\n    accuracy
   0.56       480\n   macro avg       0.26      0.23      0.22       48
0\nweighted avg       0.52      0.56      0.52       480\n'
```

In [ ]:

Submitted by:JOSWIN V JAISON

Currently pursuing:Btech 2nd year

Karunya University

Coimbatore

TamilNadu

Email:joswinvjaison@karunya.edu.in

Contact no:9400249831