jupyter WINEQUALITY PREDICTION Last Checkpoint: 01/25/2023 (autosaved)

Trusted | Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help

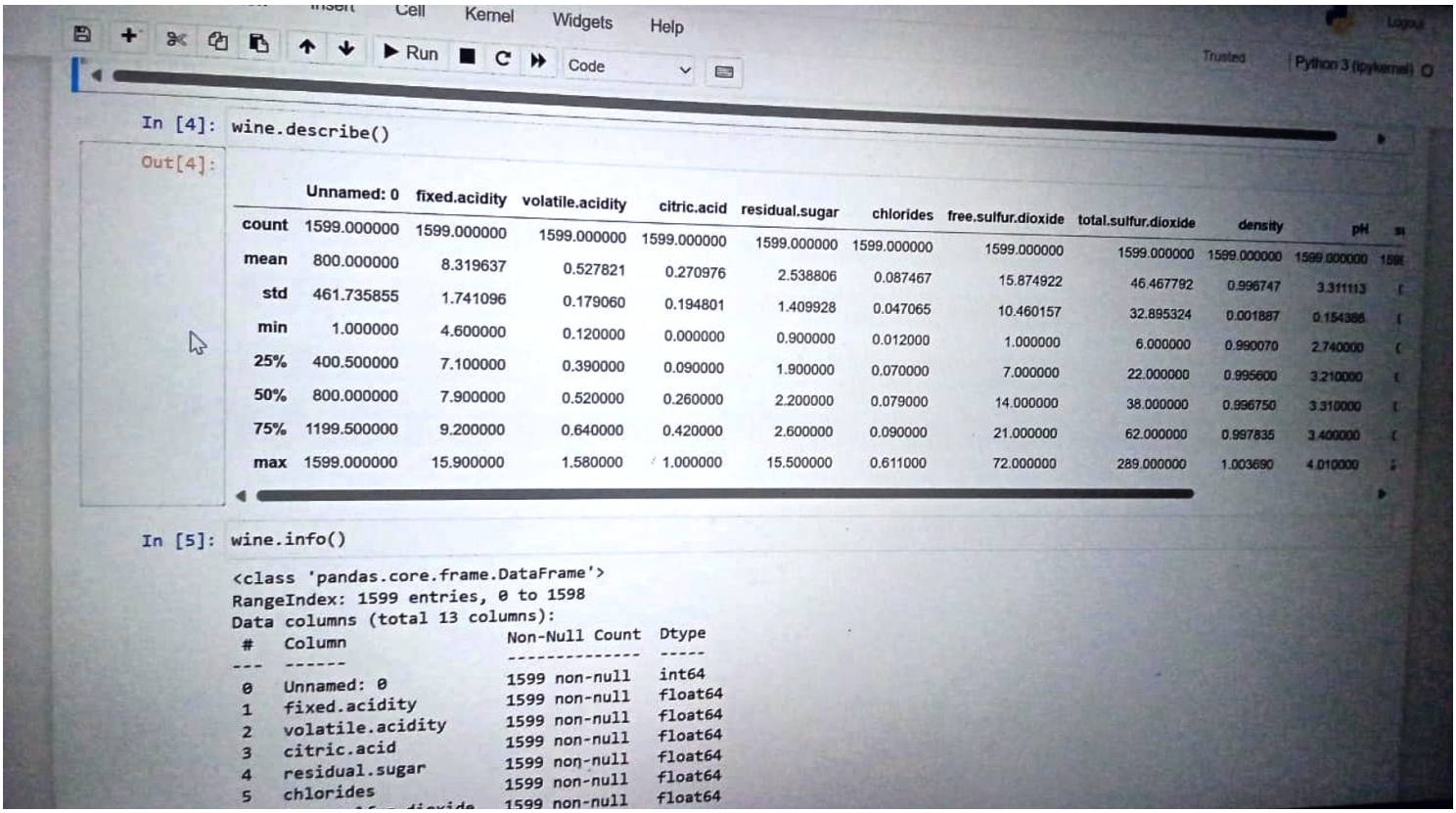
▶ Run Code

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from warnings import filterwarnings
filterwarnings(action='ignore')
```

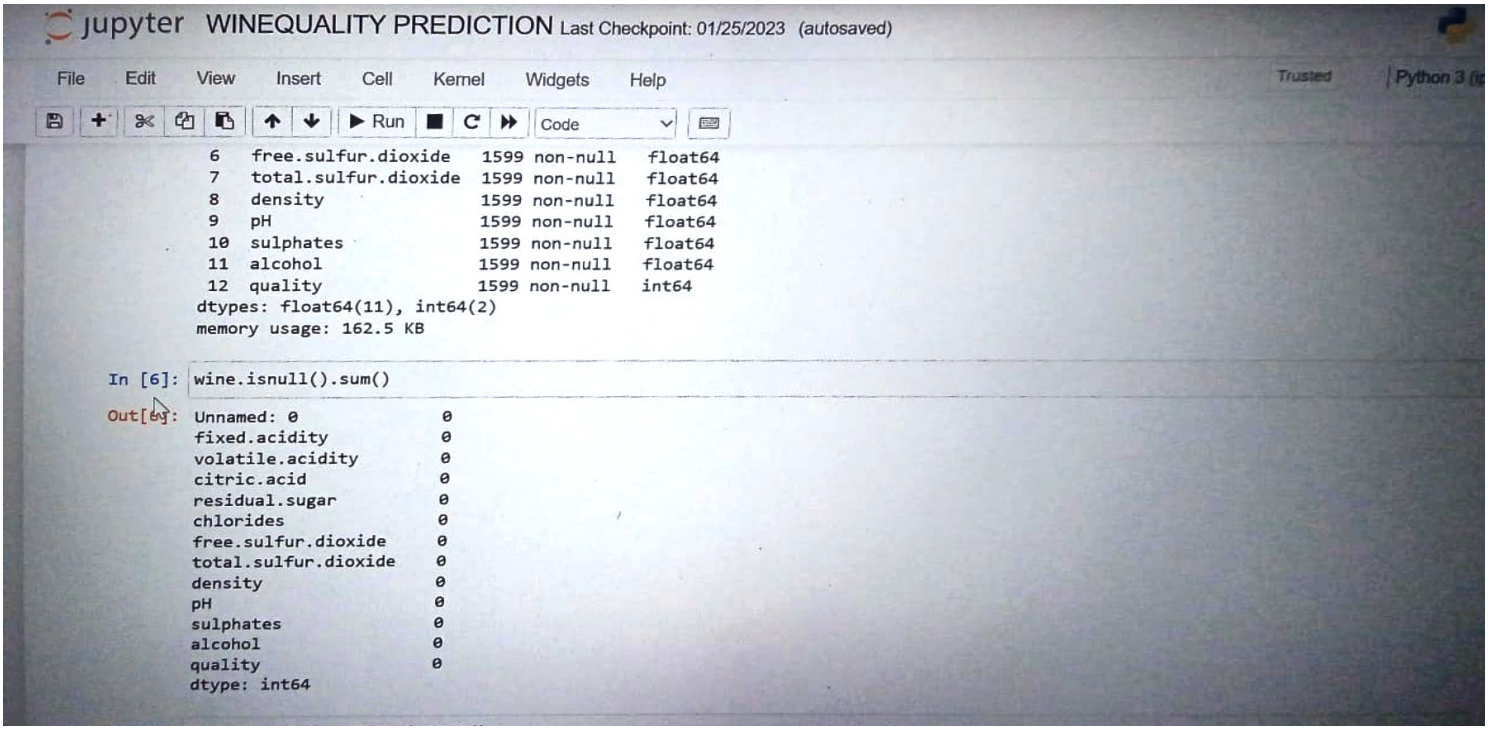
In [3]:
```
wine=pd.read_csv('wineQualityReds.csv')
wine.sample(25)
```

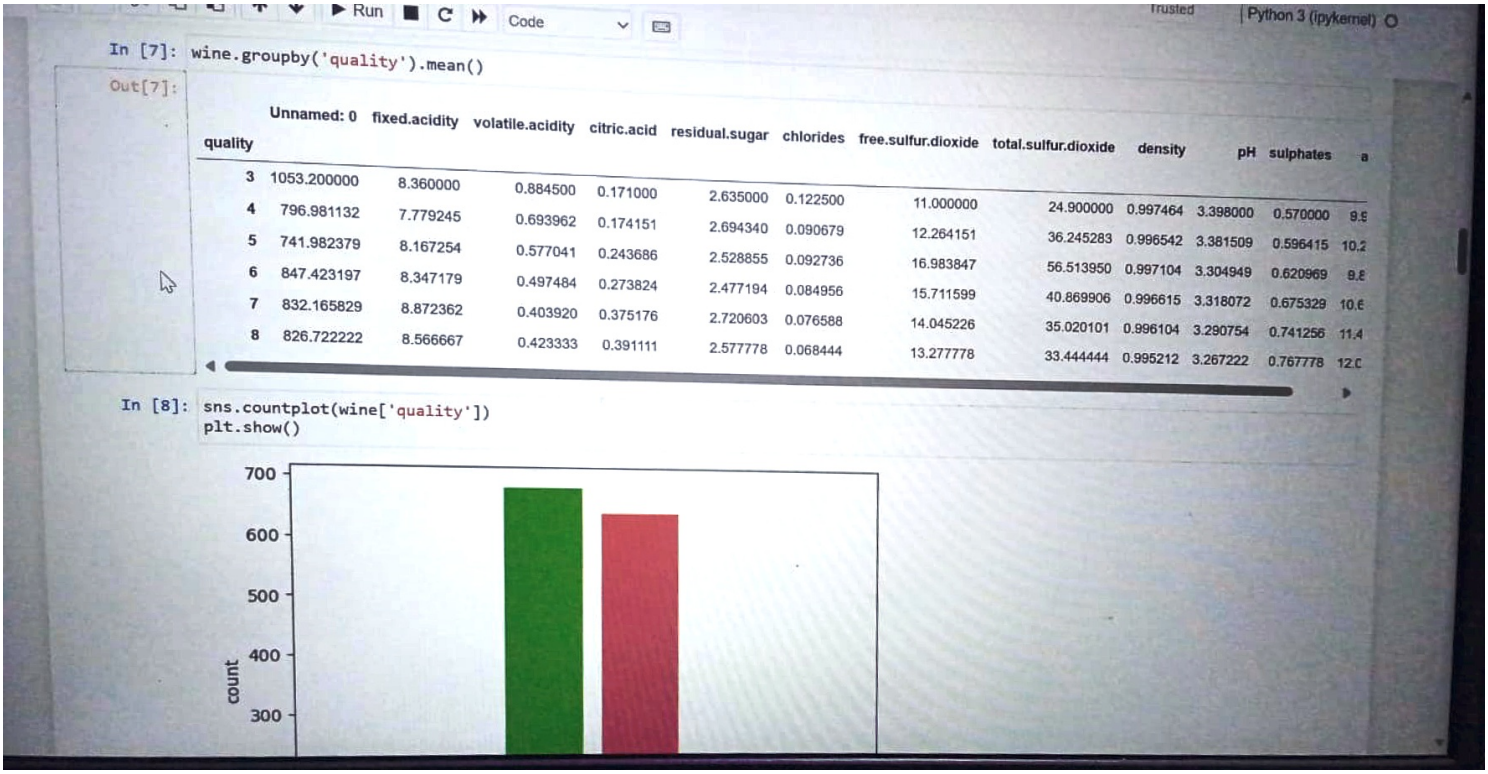
Out[3]:

	Unnamed: 0	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sulphates	alcohol	qual
1553	1554	7.3	0.735	0.00	2.2	0.080	18.0	28.0	0.99765	3.41	0.60	9.4	
1196	1197	7.9	0.580	0.23	2.3	0.076	23.0	94.0	0.99686	3.21	0.58	9.5	
653	654	9.4	0.330	0.59	2.8	0.079	9.0	30.0	0.99760	3.12	0.54	12.0	
107	108	6.2	0.630	0.31	1.7	0.088	15.0	64.0	0.99690	3.46	0.79	9.3	
1469	1470	7.3	0.980	0.05	2.1	0.061	20.0	49.0	0.99705	3.31	0.55	9.7	
1229	1230	7.6	0.430	0.29	2.1	0.075	19.0	66.0	0.99718	3.40	0.64	9.5	
547	548	10.6	0.310	0.49	2.5	0.067	6.0	21.0	0.99870	3.26	0.86	10.7	
1272	1273	5.9	0.460	0.00	1.9	0.077	25.0	44.0	0.99385	3.50	0.53	11.2	
795	796	10.8	0.890	0.30	2.6	0.132	7.0	60.0	0.99786	2.99	1.18	10.2	
1088	1089	11.6	0.410	0.54	1.5	0.095	22.0	41.0	0.99735	3.02	0.76	8.9	
341	342	10.6	0.420	0.48	2.7	0.065	5.0	18.0	0.99720	3.21	0.87	11.3	
1464	1465	6.8	0.590	0.10	1.7	0.063	34.0	53.0	0.99580	3.41	0.67	9.7	
804	805	8.4	0.520	0.22	2.7	0.084	4.0	18.0	0.99682	3.26	0.57	9.9	



	Unnamed: 0	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	su
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	159
mean	800.000000	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	(
std	461.735855	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	(
min	1.000000	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	(
25%	400.500000	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	(
50%	800.000000	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	(
75%	1199.500000	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	(
max	1599.000000	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	;


jupyter WINEQUALITY PREDICTION Last Checkpoint: 01/25/2023 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
6 free.sulfur.dioxide 1599 non-null float64
7 total.sulfur.dioxide 1599 non-null float64
8 density 1599 non-null float64
9 pH 1599 non-null float64
10 sulphates 1599 non-null float64
11 alcohol 1599 non-null float64
12 quality 1599 non-null int64
dtypes: float64(11), int64(2)
memory usage: 162.5 KB
In [6]: wine.isnull().sum()
Out[6]: Unnamed: 0 0
fixed.acidity 0
volatile.acidity 0
citric.acid 0
residual.sugar 0
chlorides 0
free.sulfur.dioxide 0
total.sulfur.dioxide 0
density 0
pH 0
sulphates 0
alcohol 0
quality 0
dtype: int64



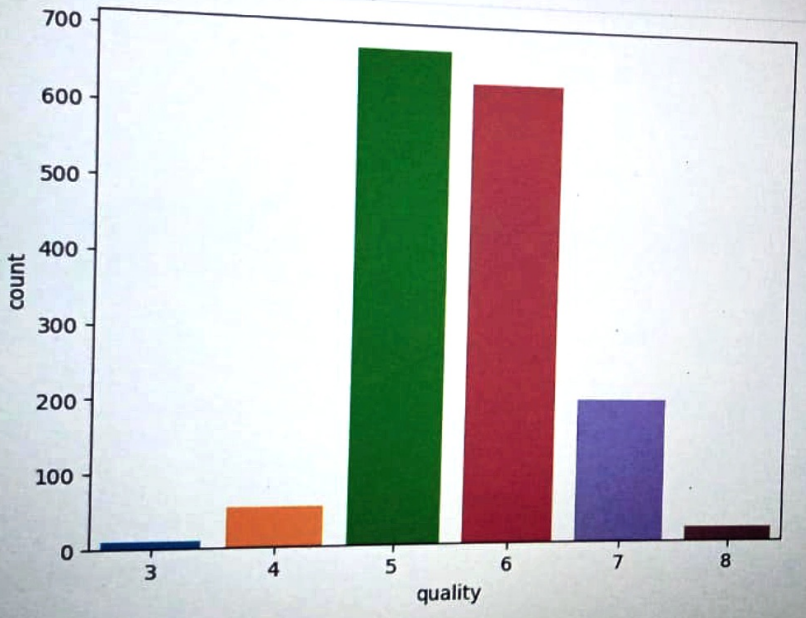
```
In [7]: wine.groupby('quality').mean()
```

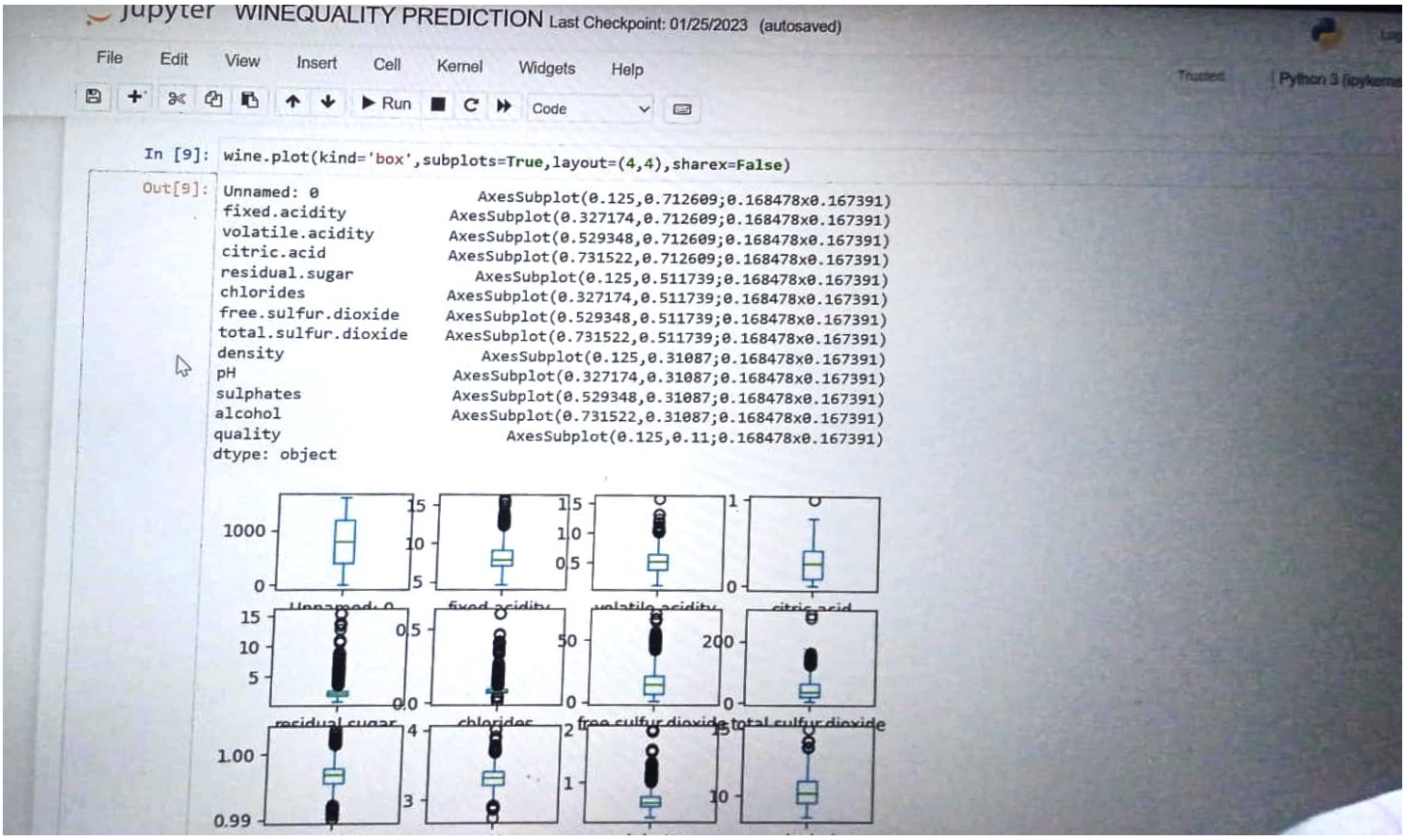
Out[7]:

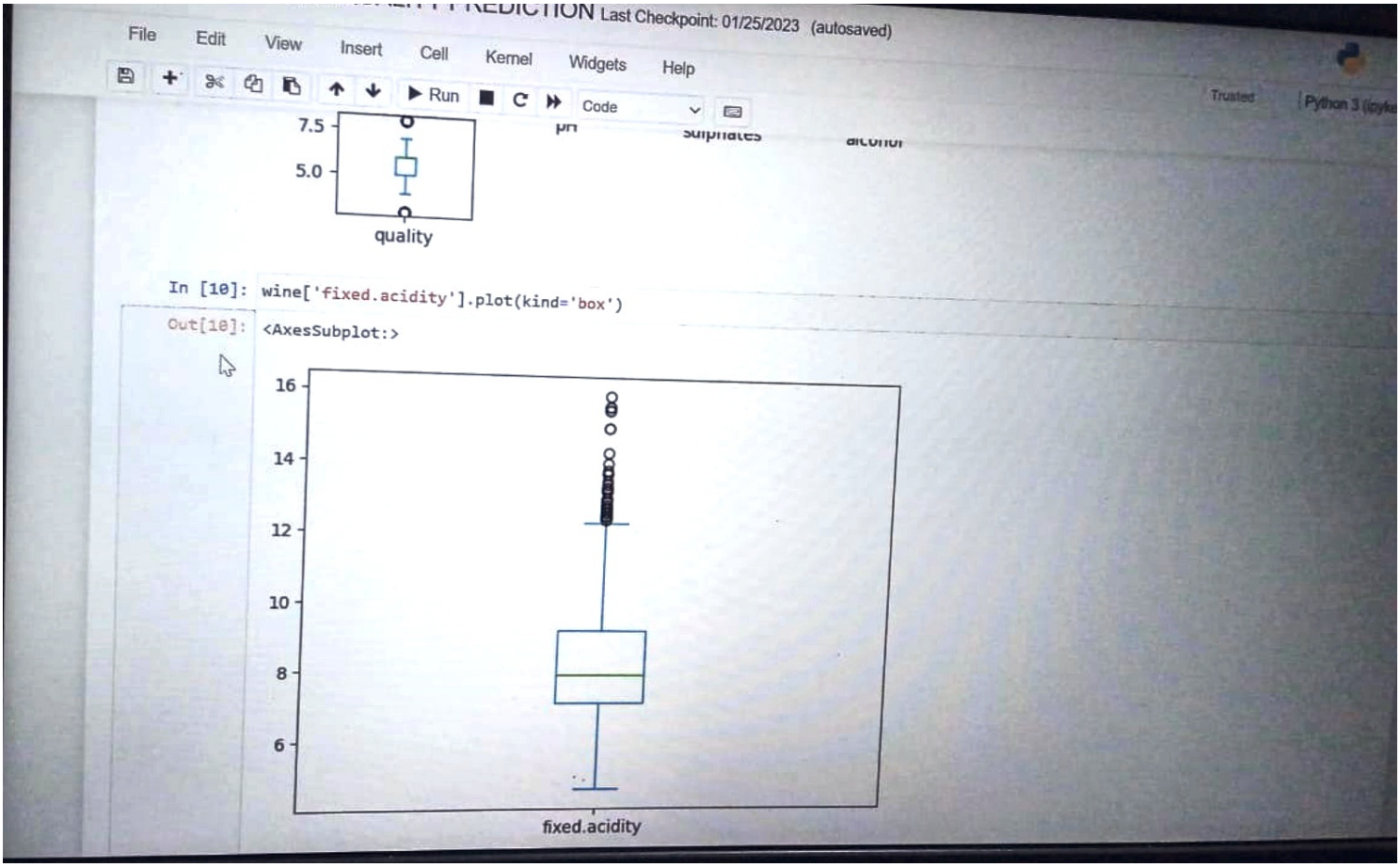
quality	Unnamed: 0	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sulphates	a
3	1053.200000	8.360000	0.884500	0.171000	2.635000	0.122500	11.000000	24.900000	0.997464	3.398000	0.570000	9.9
4	796.981132	7.779245	0.693962	0.174151	2.694340	0.090679	12.264151	36.245283	0.996542	3.381509	0.596415	10.2
5	741.982379	8.167254	0.577041	0.243686	2.528855	0.092736	16.983847	56.513950	0.997104	3.304949	0.620969	9.8
6	847.423197	8.347179	0.497484	0.273824	2.477194	0.084956	15.711599	40.869906	0.996615	3.318072	0.675329	10.6
7	832.165829	8.872362	0.403920	0.375176	2.720603	0.076588	14.045226	35.020101	0.996104	3.290754	0.741256	11.4
8	826.722222	8.566667	0.423333	0.391111	2.577778	0.068444	13.277778	33.444444	0.995212	3.267222	0.767778	12.0

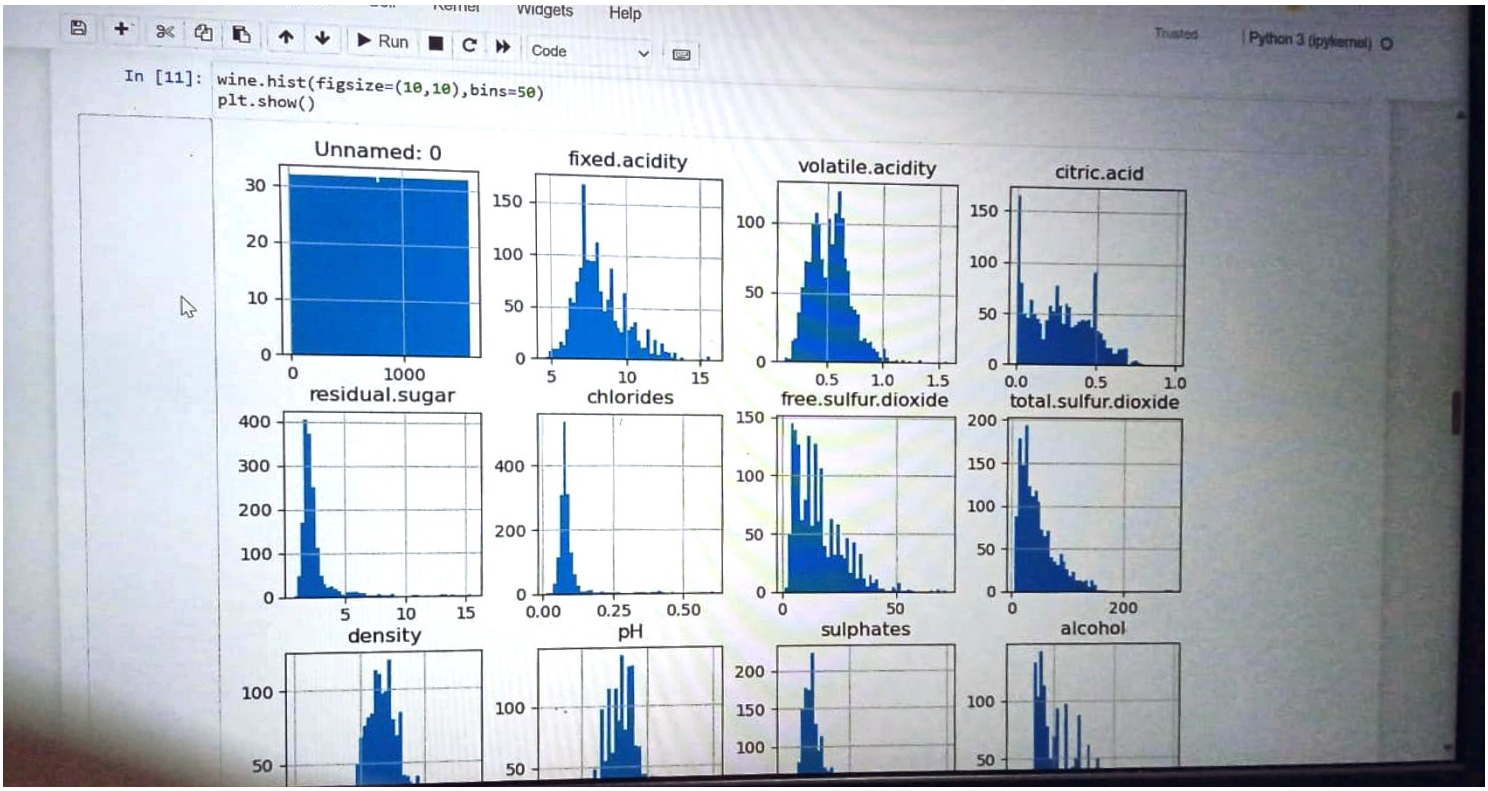
```
In [8]: sns.countplot(wine['quality'])
        plt.show()
```

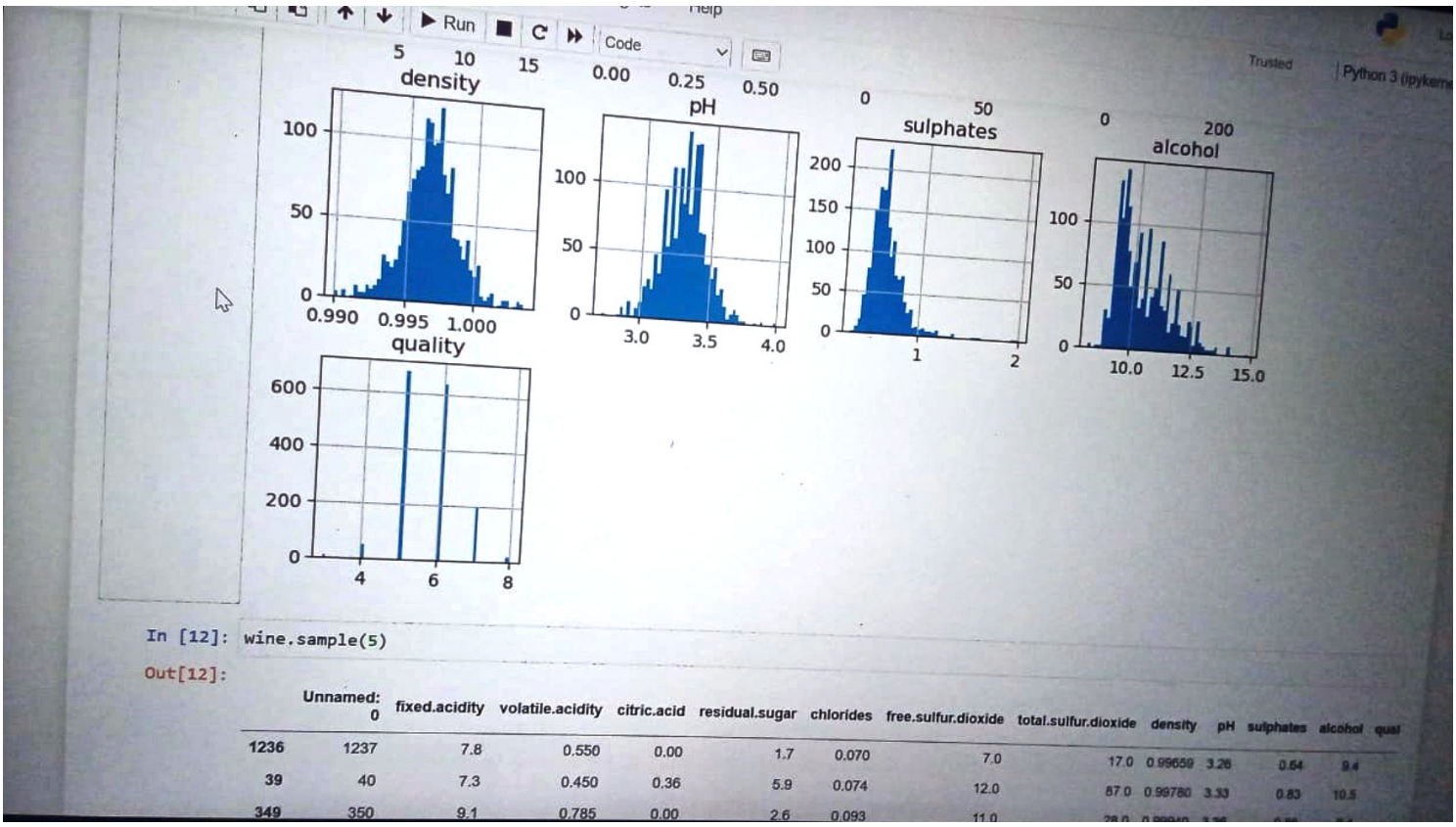
In [8]:
```
sns.countplot(wine['quality'])
plt.show()
```




WINEQUALITY PREDICTION Last Checkpoint: 01/25/2023 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run Code
In [9]: wine.plot(kind='box',subplots=True,layout=(4,4),sharex=False)
Out[9]: Unnamed: 0 AxesSubplot(0.125,0.712609;0.168478x0.167391)
fixed.acidity AxesSubplot(0.327174,0.712609;0.168478x0.167391)
volatile.acidity AxesSubplot(0.529348,0.712609;0.168478x0.167391)
citric.acid AxesSubplot(0.731522,0.712609;0.168478x0.167391)
residual.sugar AxesSubplot(0.125,0.511739;0.168478x0.167391)
chlorides AxesSubplot(0.327174,0.511739;0.168478x0.167391)
free.sulfur.dioxide AxesSubplot(0.529348,0.511739;0.168478x0.167391)
total.sulfur.dioxide AxesSubplot(0.731522,0.511739;0.168478x0.167391)
density AxesSubplot(0.125,0.31087;0.168478x0.167391)
pH AxesSubplot(0.327174,0.31087;0.168478x0.167391)
sulphates AxesSubplot(0.529348,0.31087;0.168478x0.167391)
alcohol AxesSubplot(0.731522,0.31087;0.168478x0.167391)
quality AxesSubplot(0.125,0.11;0.168478x0.167391)
dtype: object


File Edit View Insert Cell Kernel Widgets Help
Last Checkpoint: 01/25/2023 (autosaved)
Run
Code
Trusted
7.5
5.0
quality
In [10]: wine['fixed.acidity'].plot(kind='box')
Out[10]: <AxesSubplot:>
16
14
12
10
8
6
fixed.acidity

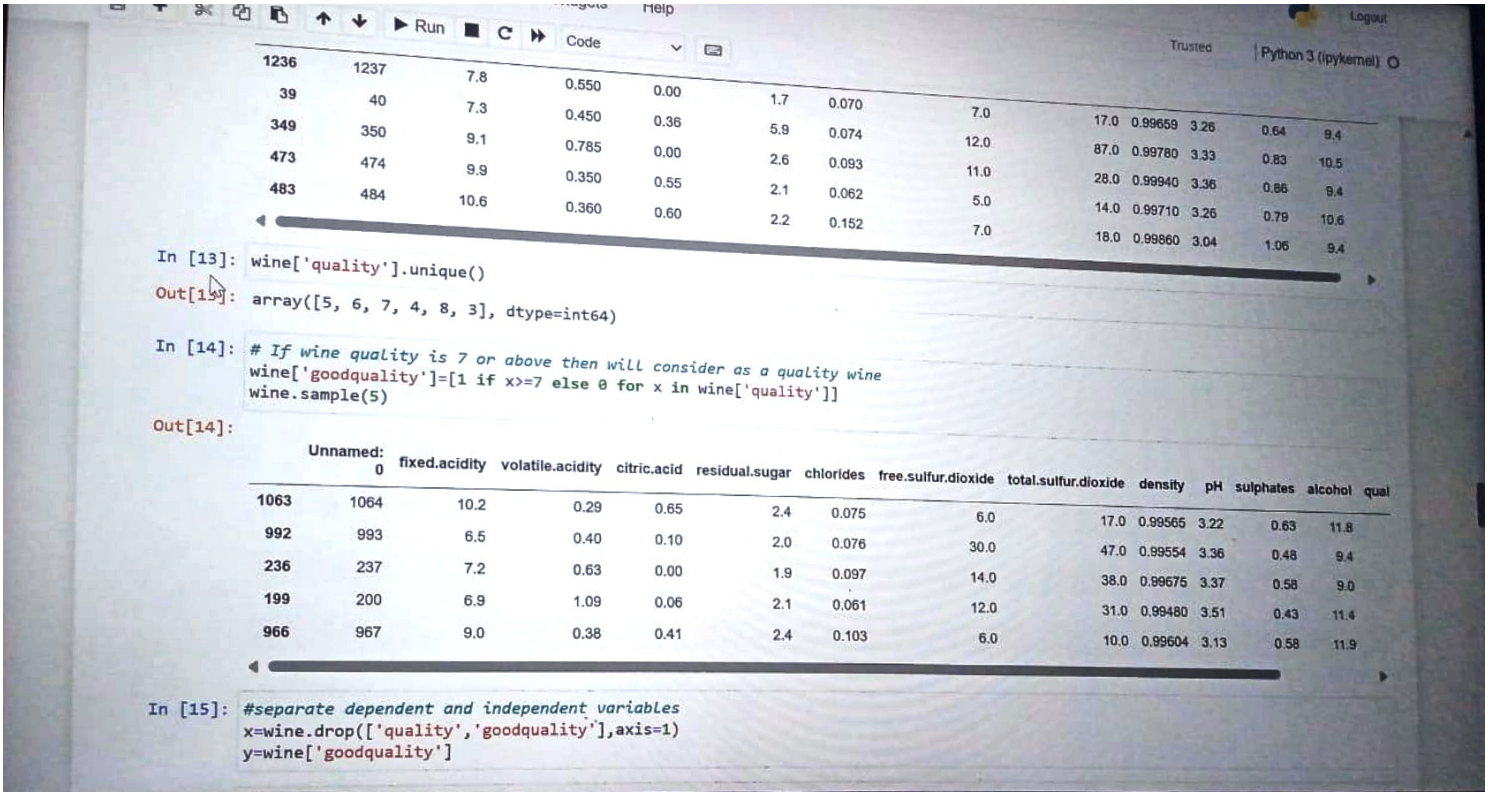

Widgets
Help
Run
Code
Trusted
Python 3 (ipykernel)
In [11]:
wine.hist(figsize=(10,10),bins=50)
plt.show()
Unnamed: 0
fixed.acidity
volatile.acidity
citric.acid
residual.sugar
chlorides
free.sulfur.dioxide
total.sulfur.dioxide
density
pH
sulphates
alcohol

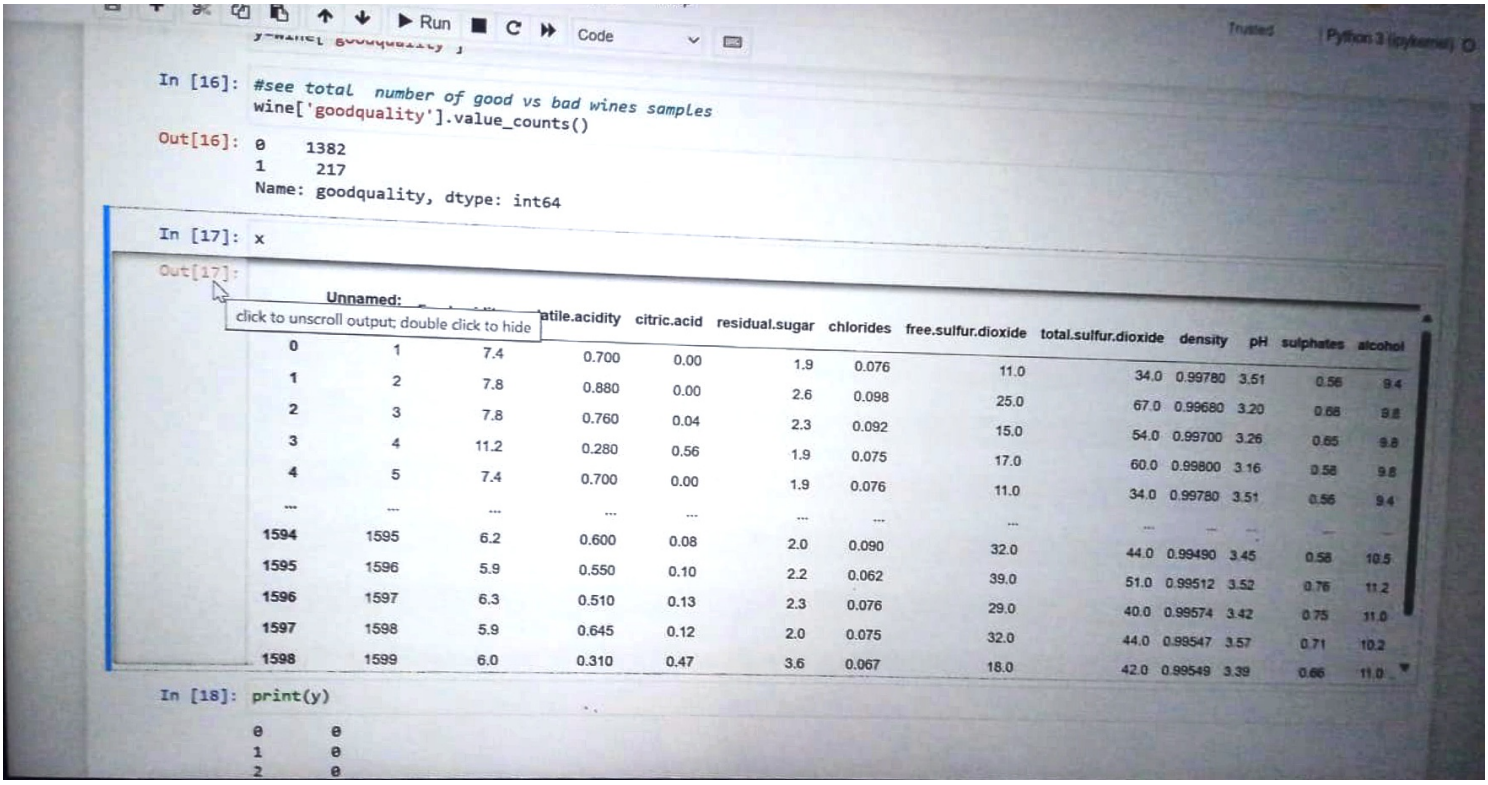


In [12]: `wine.sample(5)`

Out[12]:

	Unnamed: 0	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sulphates	alcohol	qual
1236	1237	7.8	0.550	0.00	1.7	0.070	7.0	17.0	0.99659	3.26	0.64	9.4	
39	40	7.3	0.450	0.36	5.9	0.074	12.0	87.0	0.99780	3.33	0.83	10.5	
349	350	9.1	0.785	0.00	2.6	0.093	11.0						


Help
Logout
Run
Code
Trusted
Python 3 (ipykernel)
1236 1237 7.8 0.550 0.00 1.7 0.070 7.0 17.0 0.99659 3.26 0.64 9.4
39 40 7.3 0.450 0.36 5.9 0.074 12.0 87.0 0.99780 3.33 0.83 10.5
349 350 9.1 0.785 0.00 2.6 0.093 11.0 28.0 0.99940 3.36 0.86 9.4
473 474 9.9 0.350 0.55 2.1 0.062 5.0 14.0 0.99710 3.26 0.79 10.6
483 484 10.6 0.360 0.60 2.2 0.152 7.0 18.0 0.99860 3.04 1.06 9.4
In [13]: wine['quality'].unique()
Out[13]: array([5, 6, 7, 4, 8, 3], dtype=int64)
In [14]: # If wine quality is 7 or above then will consider as a quality wine
wine['goodquality']=[1 if x>=7 else 0 for x in wine['quality']]
wine.sample(5)
Out[14]:
Unnamed: 0 fixed.acidity volatile.acidity citric.acid residual.sugar chlorides free.sulfur.dioxide total.sulfur.dioxide density pH sulphates alcohol qual
1063 1064 10.2 0.29 0.65 2.4 0.075 6.0 17.0 0.99565 3.22 0.63 11.8
992 993 6.5 0.40 0.10 2.0 0.076 30.0 47.0 0.99554 3.36 0.48 9.4
236 237 7.2 0.63 0.00 1.9 0.097 14.0 38.0 0.99675 3.37 0.58 9.0
199 200 6.9 1.09 0.06 2.1 0.061 12.0 31.0 0.99480 3.51 0.43 11.4
966 967 9.0 0.38 0.41 2.4 0.103 6.0 10.0 0.99604 3.13 0.58 11.9
In [15]: #separate dependent and independent variables
x=wine.drop(['quality','goodquality'],axis=1)
y=wine['goodquality']



In [16]:
```
#see total  number of good vs bad wines samples
wine['goodquality'].value_counts()
```

Out[16]:
```
0    1382
1     217
Name: goodquality, dtype: int64
```

In [17]:
```
x
```

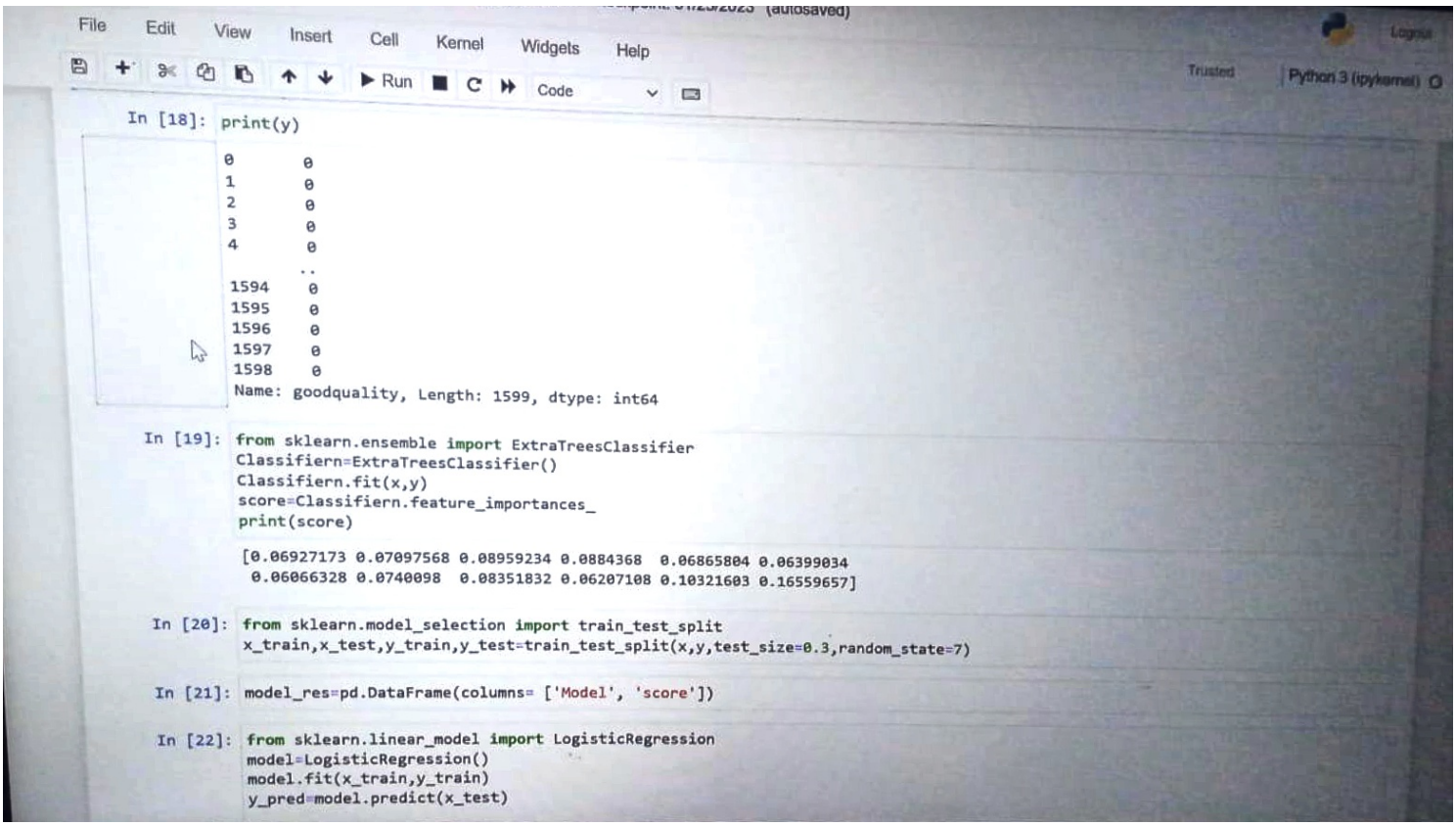
Out[17]:

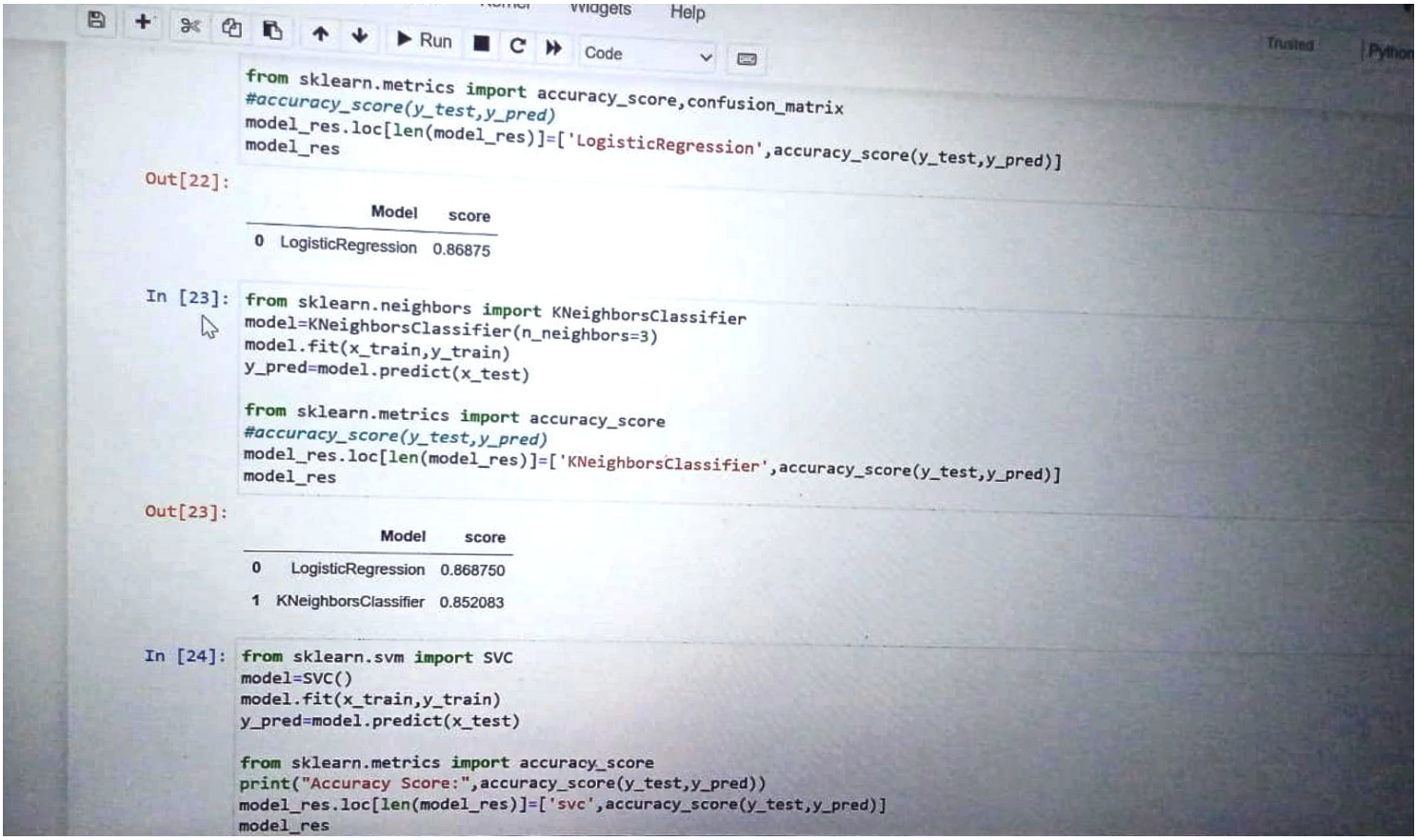
click to unscroll output; double click to hide

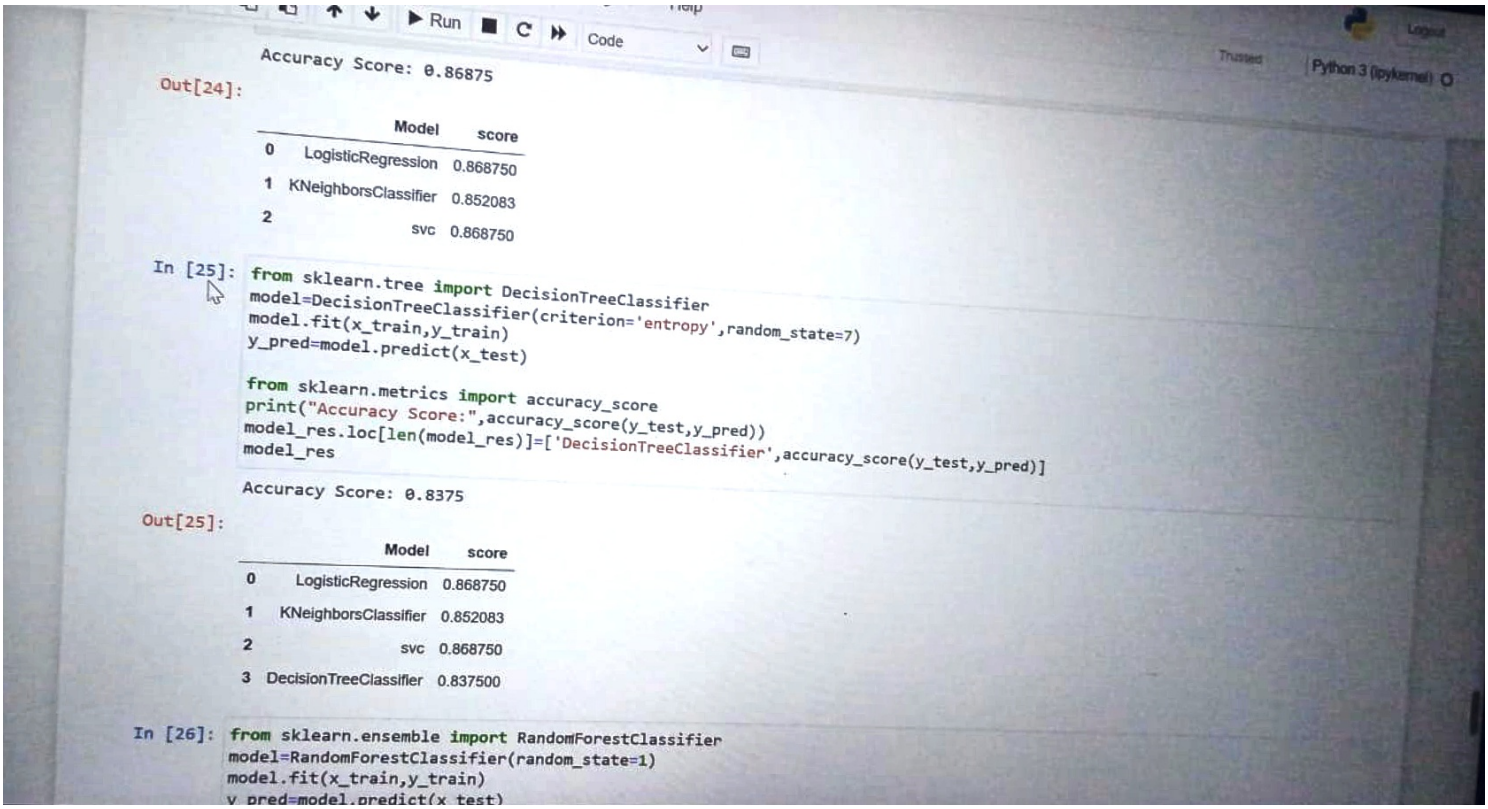
	Unnamed:		atile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sulphates	alcohol
0	1	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	2	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	3	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	4	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	5	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	1595	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1595	1596	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1596	1597	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1597	1598	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
1598	1599	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

In [18]:
```
print(y)
```

```
0       0
1       0
2       0
```


File Edit View Insert Cell Kernel Widgets Help
Run Code
Trusted
Python 3 (ipykernel)
In [18]: print(y)
0 0
1 0
2 0
3 0
4 0
..
1594 0
1595 0
1596 0
1597 0
1598 0
Name: goodquality, Length: 1599, dtype: int64
In [19]: from sklearn.ensemble import ExtraTreesClassifier
Classifiern=ExtraTreesClassifier()
Classifiern.fit(x,y)
score=Classifiern.feature_importances_
print(score)
[0.06927173 0.07097568 0.08959234 0.0884368 0.06865804 0.06399034
0.06066328 0.0740098 0.08351832 0.06207108 0.10321603 0.16559657]
In [20]: from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=7)
In [21]: model_res=pd.DataFrame(columns= ['Model', 'score'])
In [22]: from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)


Widgets
Help
Run
Code
Trusted
Python
from sklearn.metrics import accuracy_score,confusion_matrix
#accuracy_score(y_test,y_pred)
model_res.loc[len(model_res)]=['LogisticRegression',accuracy_score(y_test,y_pred)]
model_res
Out[22]:
Model score
0 LogisticRegression 0.86875
In [23]:
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier(n_neighbors=3)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
from sklearn.metrics import accuracy_score
#accuracy_score(y_test,y_pred)
model_res.loc[len(model_res)]=['KNeighborsClassifier',accuracy_score(y_test,y_pred)]
model_res
Out[23]:
Model score
0 LogisticRegression 0.868750
1 KNeighborsClassifier 0.852083
In [24]:
from sklearn.svm import SVC
model=SVC()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(y_test,y_pred))
model_res.loc[len(model_res)]=['svc',accuracy_score(y_test,y_pred)]
model_res



Run Code Trusted Python 3 (ipykernel)

```
Accuracy Score: 0.86875
```

Out[24]:

	Model	score
0	LogisticRegression	0.868750
1	KNeighborsClassifier	0.852083
2	svc	0.868750

In [25]:

```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier(criterion='entropy',random_state=7)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(y_test,y_pred))
model_res.loc[len(model_res)]=['DecisionTreeClassifier',accuracy_score(y_test,y_pred)]
model_res
```

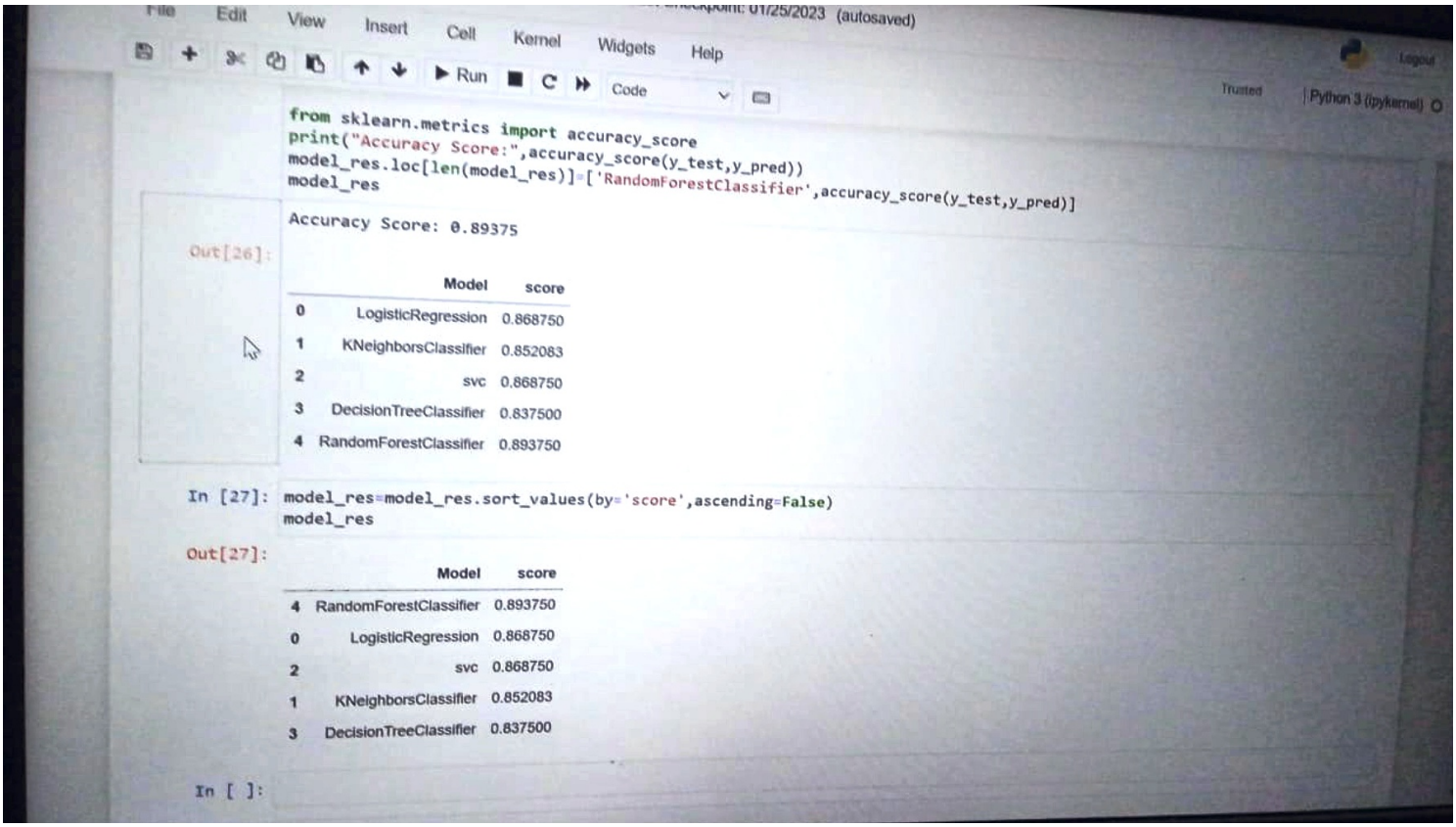
```
Accuracy Score: 0.8375
```

Out[25]:

	Model	score
0	LogisticRegression	0.868750
1	KNeighborsClassifier	0.852083
2	svc	0.868750
3	DecisionTreeClassifier	0.837500

In [26]:

```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(random_state=1)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
```


File Edit View Insert Cell Kernel Widgets Help
(autosaved)
Run
Code
Trusted
Python 3 (ipykernel)
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(y_test,y_pred))
model_res.loc[len(model_res)]=['RandomForestClassifier',accuracy_score(y_test,y_pred)]
model_res
Out[26]:
Accuracy Score: 0.89375
Model score
0 LogisticRegression 0.868750
1 KNeighborsClassifier 0.852083
2 svc 0.868750
3 DecisionTreeClassifier 0.837500
4 RandomForestClassifier 0.893750
In [27]:
model_res=model_res.sort_values(by='score',ascending=False)
model_res
Out[27]:
Model score
4 RandomForestClassifier 0.893750
0 LogisticRegression 0.868750
2 svc 0.868750
1 KNeighborsClassifier 0.852083
3 DecisionTreeClassifier 0.837500
In [ ]: