In [2]:
```
pip install pandas-profiling
```
```
Collecting pandas-profiling
  Using cached pandas_profiling-3.1.0-py2.py3-none-any.whl (261 kB)
Requirement already satisfied: seaborn>=0.10.1 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (0.11.2)
Requirement already satisfied: pandas!=1.0.0,!=1.0.1,!=1.0.2,!=1.1.0,>=0.25.3 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (1.3.4)
Requirement already satisfied: joblib~=1.0.1 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (1.0.1)
Requirement already satisfied: pydantic>=1.8.1 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (1.8.2)
Requirement already satisfied: PyYAML>=5.0.0 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (5.3)
Requirement already satisfied: htmlmin>=0.1.12 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (0.1.12)
Collecting visions[type_image_path]==0.7.4
  Using cached visions-0.7.4-py3-none-any.whl (102 kB)
Requirement already satisfied: numpy>=1.16.0 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (1.18.1)
Requirement already satisfied: markupsafe~=2.0.1 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (2.0.1)
Requirement already satisfied: missingno>=0.4.2 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (0.5.0)
Requirement already satisfied: tangled-up-in-unicode==0.1.0 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (0.1.0)
Collecting multimethod>=1.4
  Using cached multimethod-1.6-py3-none-any.whl (9.4 kB)
Requirement already satisfied: scipy>=1.4.1 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (1.4.1)
Requirement already satisfied: requests>=2.24.0 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (2.26.0)
Requirement already satisfied: tqdm>=4.48.2 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (4.62.3)
Collecting phik>=0.11.1
  Using cached phik-0.12.0-cp37-cp37m-win_amd64.whl (660 kB)
Requirement already satisfied: matplotlib>=3.2.0 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (3.4.3)
Requirement already satisfied: jinja2>=2.11.1 in c:\users\jhare\anaconda3\lib\site-packages (from pandas-profiling) (2.11.1)
Requirement already satisfied: pytz>=2017.3 in c:\users\jhare\anaconda3\lib\site-packages (from pandas!=1.0.0,!=1.0.1,!=1.0.2,!=1.1.0,>=0.25.3->pandas-profiling) (2019.3)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\jhare\anaconda3\lib\site-packages (from pandas!=1.0.0,!=1.0.1,!=1.0.2,!=1.1.0,>=0.25.3->pandas-profiling) (2.8.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\jhare\anaconda3\lib\site-packages (from pydantic>=1.8.1->pandas-profiling) (3.10.0.2)
Requirement already satisfied: attrs>=19.3.0 in c:\users\jhare\anaconda3\lib\site-packages (from visions[type_image_path]==0.7.4->pandas-profiling) (19.3.0)
Requirement already satisfied: networkx>=2.4 in c:\users\jhare\anaconda3\lib\site-packages (from visions[type_image_path]==0.7.4->pandas-profiling) (2.4)
Requirement already satisfied: Pillow; extra == "type_image_path" in c:\users\jhare\anaconda3\lib\site-packages (from visions[type_image_path]==0.7.4->pandas-profiling) (7.0.0)
Processing c:\users\jhare\appdata\local\pip\cache\wheels\4c\d5\59\5e3e297533ddb09407769762985d134135064c6831e29a914e\imagehash-4.2.1-py2.py3-none-any.whl
Requirement already satisfied: certifi>=2017.4.17 in c:\users\jhare\anaconda3\lib\site-packages (from requests>=2.24.0->pandas-profiling) (2019.11.28)
Requirement already satisfied: charset-normalizer~=2.0.0; python_version >= "3" in c:\users\jhare\anaconda3\lib\site-packages (from requests>=2.24.0->pandas-profiling) (2.0.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\jhare\anaconda3\lib\site-packages (from requests>=2.24.0->pandas-profiling) (1.25.8)
Requirement already satisfied: idna<4,>=2.5; python_version >= "3" in c:\users\jhare\anaconda3\lib\site-packages (from requests>=2.24.0->pandas-profiling) (2.8)
Requirement already satisfied: colorama; platform_system == "Windows" in c:\users\jhare\anaconda3\lib\site-packages (from tqdm>=4.48.2->pandas-profiling) (0.4.3)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\jhare\anaconda3\lib\site-packages (from matplotlib>=3.2.0->pandas-profiling) (2.4.6)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\jhare\anaconda3\lib\site-packages (from matplotlib>=3.2.0->pandas-profiling) (1.1.0)
Requirement already satisfied: cycler>=0.10 in c:\users\jhare\anaconda3\lib\site-packages (from matplotlib>=3.2.0->pandas-profiling) (0.10.0)
Requirement already satisfied: six>=1.5 in c:\users\jhare\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pandas!=1.0.0,!=1.0.1,!=1.0.2,!=1.1.0,>=0.25.3->pandas-profiling) (1.14.0)
Requirement already satisfied: decorator>=4.3.0 in c:\users\jhare\anaconda3\lib\site-packages (from networkx>=2.4->visions[type_image_path]==0.7.4->pandas-profiling) (4.4.1)
Requirement already satisfied: PyWavelets in c:\users\jhare\anaconda3\lib\site-packages (from imagehash; extra == "type_image_path"->visions[type_image_path]==0.7.4->pandas-profiling) (1.1.1)
Requirement already satisfied: setuptools in c:\users\jhare\anaconda3\lib\site-packages (from kiwisolver>=1.0.1->matplotlib>=3.2.0->pandas-profiling) (45.2.0.post20200210)
Installing collected packages: multimethod, imagehash, visions, phik, pandas-profiling
Successfully installed imagehash-4.2.1 multimethod-1.6 pandas-profiling-3.1.0 phik-0.12.0 visions-0.7.4
Note: you may need to restart the kernel to use updated packages.
```
```
ERROR: phik 0.12.0 has requirement scipy>=1.5.2, but you'll have scipy 1.4.1 which is incompatible.
```

Name: Reeti Jha

Reg no: RA1911030010121

In [16]:
```
import tkinter
import matplotlib
import matplotlib.pyplot as plt
matplotlib.use('TkAgg')
```

In [17]:
```
import numpy as np
import pandas as pd



import os
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from pandas_profiling import ProfileReport
from sklearn import metrics
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

In [18]:
```
df = pd.read_csv('winequality-red.csv')
print('The Dataset contains {} rows and {} columns '.format(df.shape[0], df.shape[1]))
```
```
The Dataset contains 1599 rows and 12 columns
```

In [19]:
```
df.head()
```
Out[19]:

	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

Data exploration

In [7]:
```
df.describe()
```
Out[7]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfu dioxide
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000

Pandas Profiling

In [8]:
```
ProfileReport(df)
```

Pandas Profiling Report

Overview

Overview | Alerts 30 | Reproduction

Dataset statistics

Number of variables	12
Number of observations	1599
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	220
Duplicate rows (%)	13.8%
Total size in memory	150.0 KiB
Average record size in memory	96.1 B

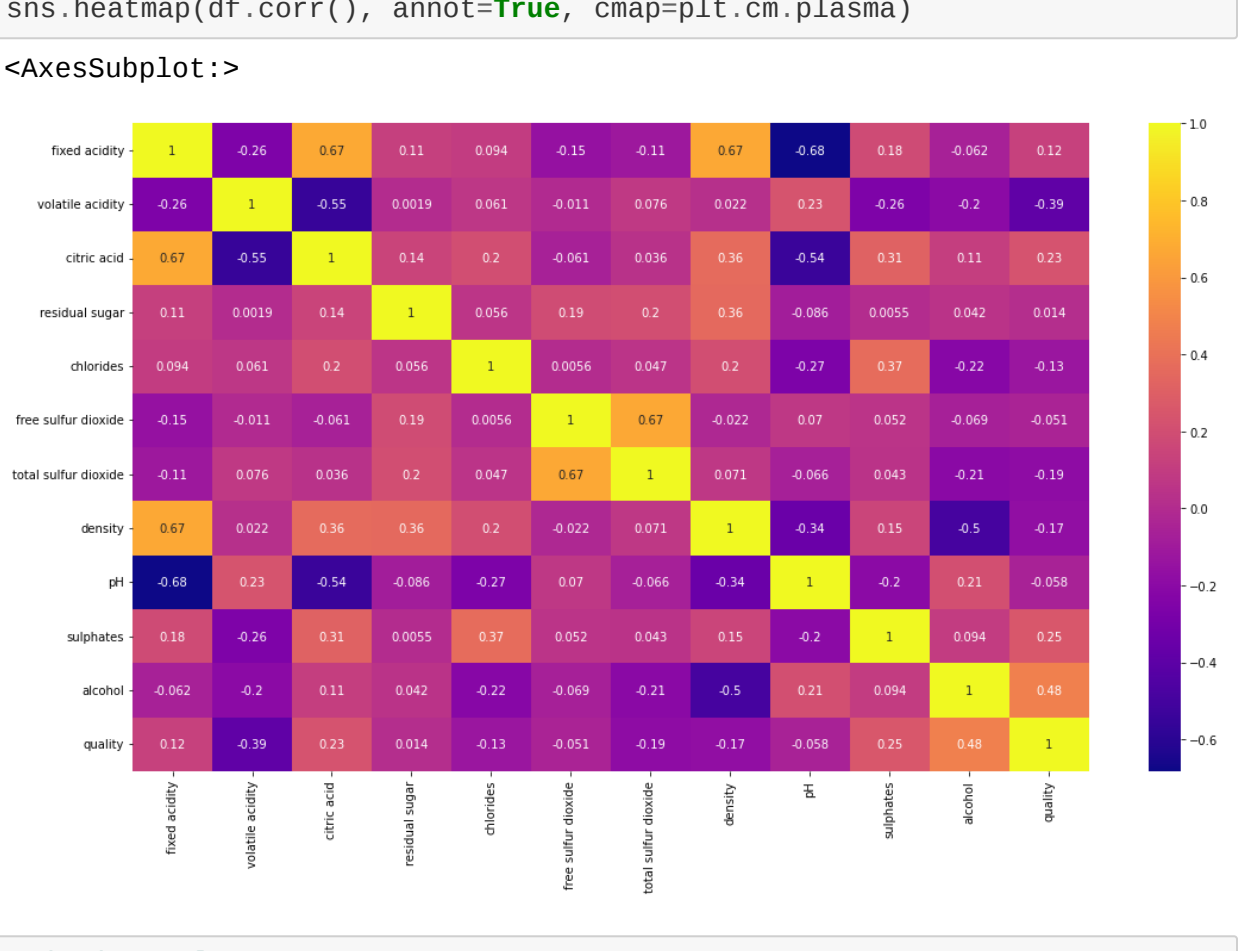
Variable types

Numeric	12

Variables

Out[8]:

In [20]:
```
plt.figure(figsize=(18,10))
sns.heatmap(df.corr(), annot=True, cmap=plt.cm.plasma)
```
Out[20]: <AxesSubplot:>

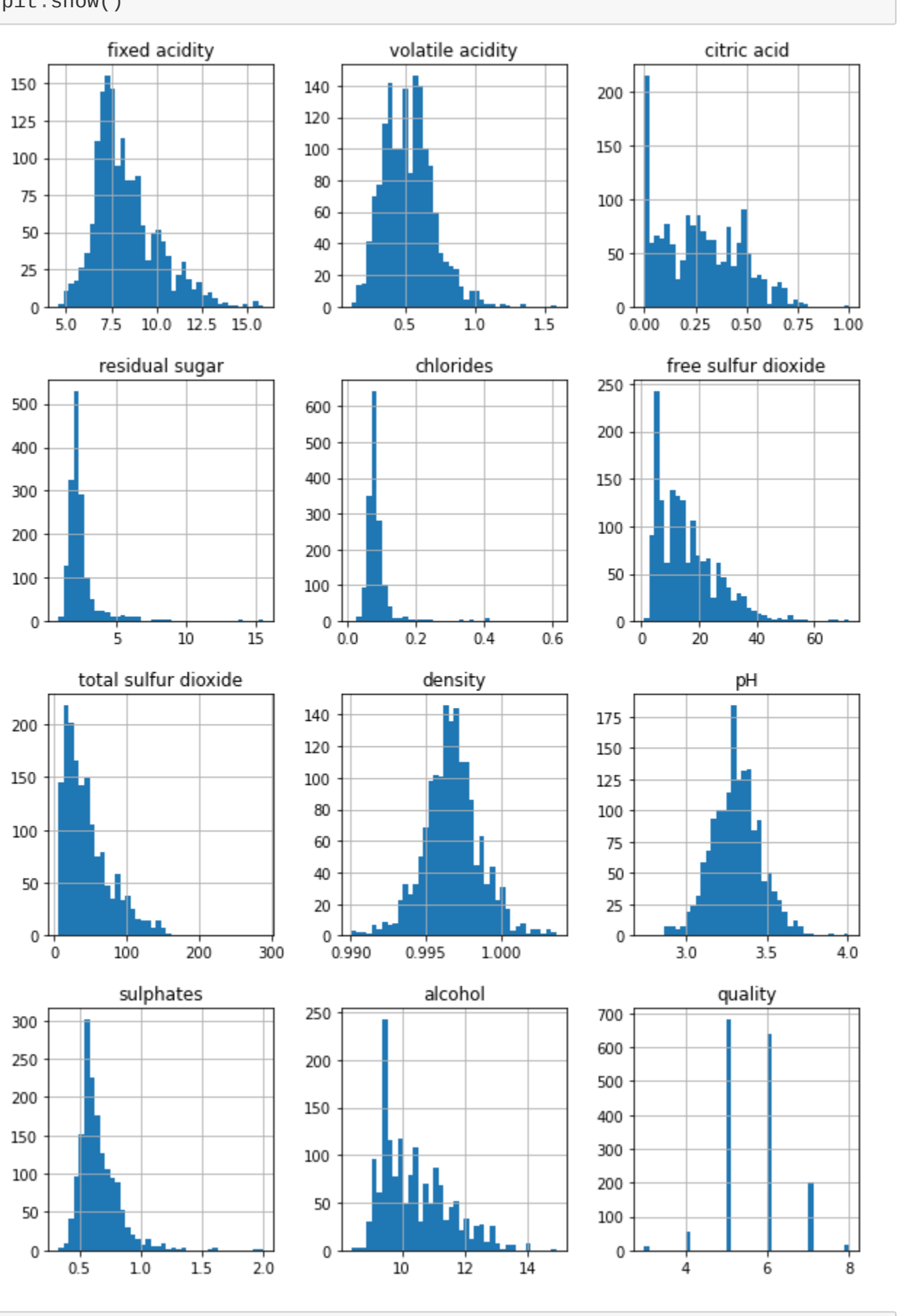


In [12]:
```
#missing values
```

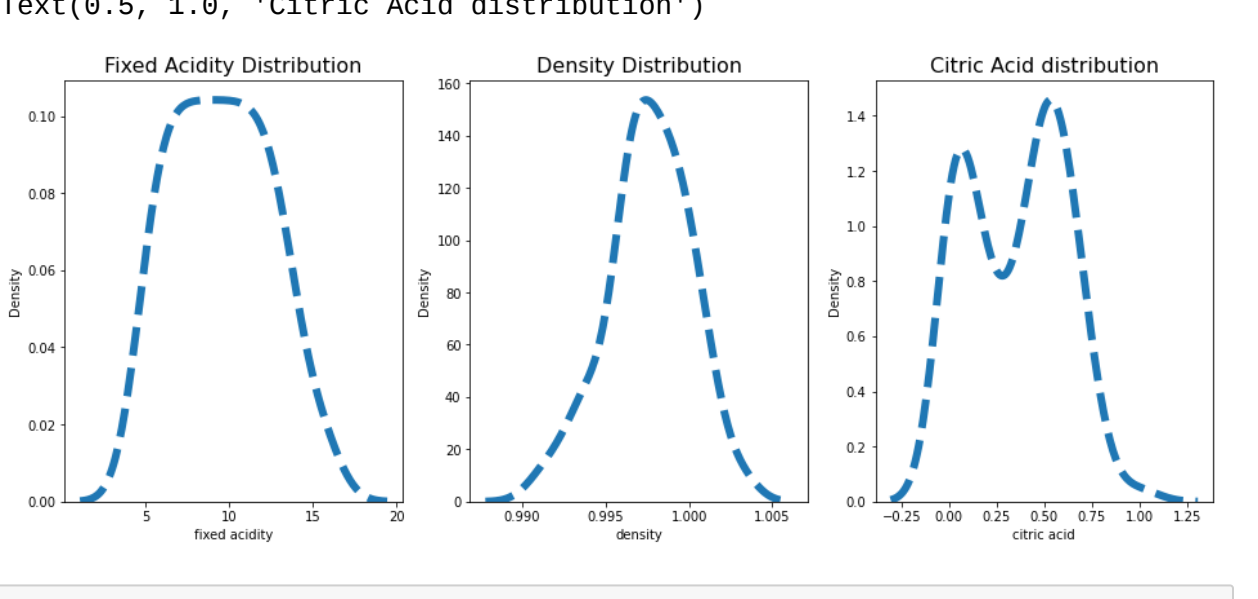
In [13]:
```
df.isnull().sum().sum()
```
Out[13]: 0

In [14]:
```
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

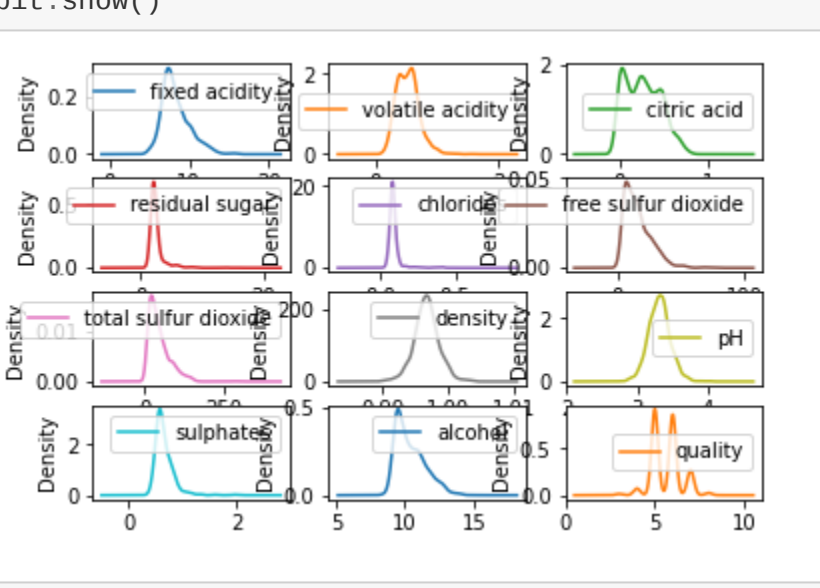
In [21]:
```
df.hist(bins=40, figsize=(10,15))
plt.show()
```



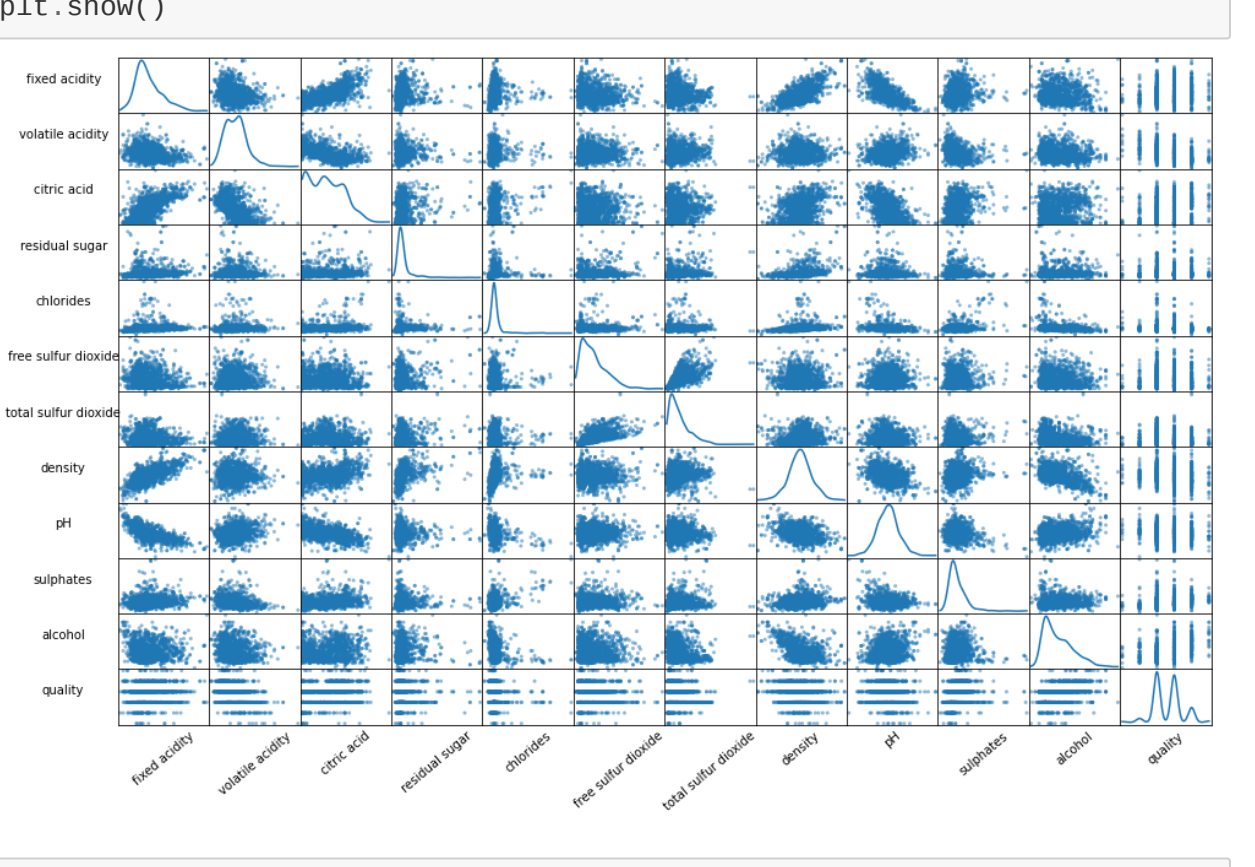
In [23]:
```
data = df.groupby(by="fixed acidity")[["fixed acidity", "density", "citric acid"]].first().reset_index(drop=True)

# Figure
f, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize = (16, 6))

a = sns.distplot(data["fixed acidity"], ax=ax1, hist=False, kde_kws=dict(lw=6, ls="--"))
b = sns.distplot(data["density"], ax=ax2, hist=False, kde_kws=dict(lw=6, ls="--"))
c = sns.distplot(data["citric acid"], ax=ax3, hist=False, kde_kws=dict(lw=6, ls="--"))

a.set_title("Fixed Acidity Distribution", fontsize=16)
b.set_title("Density Distribution", fontsize=16)
c.set_title("Citric Acid distribution", fontsize=16)
```
```
C:\Users\jhare\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\jhare\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\jhare\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```
Out[23]: Text(0.5, 1.0, 'Citric Acid distribution')



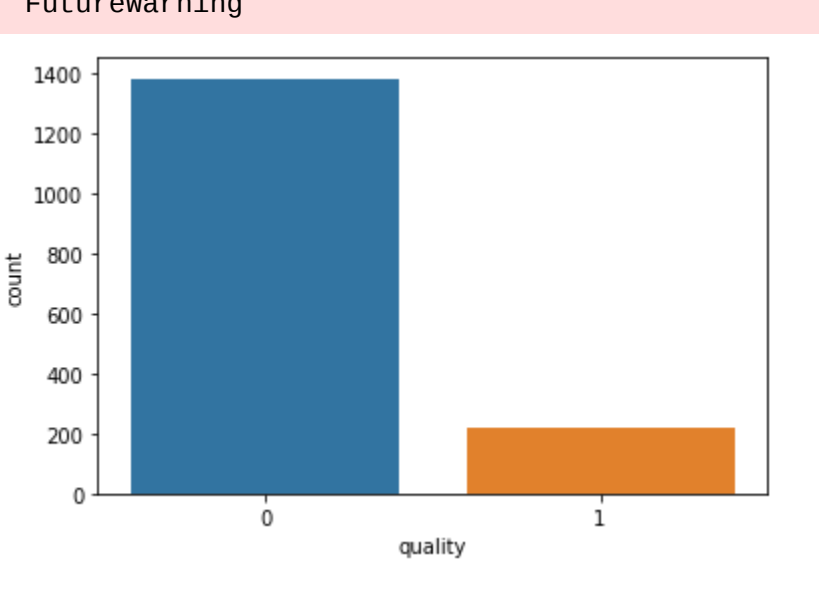
In [22]:
```
df.plot(kind='density', subplots=True, layout=(4,3), sharex=False)
plt.show()
```



In [24]:
```
from pandas.plotting import scatter_matrix

sm = scatter_matrix(df, figsize=(16, 10), diagonal='kde')

[s.xaxis.label.set_rotation(40) for s in sm.reshape(-1)]
[s.yaxis.label.set_rotation(0) for s in sm.reshape(-1)]

#May need to offset label when rotating to prevent overlap of figure

[s.get_yaxis().set_label_coords(-0.6,0.5) for s in sm.reshape(-1)]

#Hide all ticks

[s.set_xticks(()) for s in sm.reshape(-1)]
[s.set_yticks(()) for s in sm.reshape(-1)]
plt.show()
```



In [25]:
```
# Dividing wine as good and bad by giving the limit for the quality

bins = (2, 6, 8)
group_names = ['bad', 'good']
df['quality'] = pd.cut(df['quality'], bins = bins, labels = group_names)
# Now lets assign a labels to our quality variable

label_quality = LabelEncoder()

# Bad becomes 0 and good becomes 1
df['quality'] = label_quality.fit_transform(df['quality'])
print(df['quality'].value_counts())
sns.countplot(df['quality'])
plt.show()
```
```
0    1382
1     217
Name: quality, dtype: int64
```
```
C:\Users\jhare\anaconda3\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



Building model

In [26]:
```
x = df.drop(['quality'], axis=1)
y = df['quality']
```

In [27]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 50)
```

In [28]:
```
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)


cols = ['fixed acidity',
'volatile acidity',
'citric acid',
'residual sugar',
'chlorides',
'free sulfur dioxide',
'total sulfur dioxide',
'density',
'pH',
'sulphates',
'alcohol'
       ]
```

Decision Tree

In [29]:
```
dtc = DecisionTreeClassifier(max_depth=200)
dtc.fit(x_train, y_train)
preds = dtc.predict(x_test)
score = dtc.score(x_test, y_test)
score
```
Out[29]: 0.8675

In [30]:
```
preds[:5]
```
Out[30]: array([0, 0, 0, 0, 0])

In [31]:
```
y_test[:5]
```
Out[31]:
```
453     1
1415    0
1242    0
885     0
488     1
Name: quality, dtype: int32
```

In [32]:
```
Ks = 100
mean_acc = np.zeros((Ks-1))
for n in range(1,Ks):

    #Train Model and Predict
    dtc = DecisionTreeClassifier(max_depth = n).fit(x_train,y_train)
    yhat=dtc.predict(x_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)

mean_acc
```
Out[32]:
```
array([0.8925, 0.8825, 0.8975, 0.8875, 0.895 , 0.89  , 0.88  , 0.8825,
       0.8825, 0.875 , 0.865 , 0.8575, 0.8625, 0.8625, 0.8825, 0.88  ,
       0.8475, 0.8825, 0.855 , 0.86  , 0.8575, 0.8525, 0.865 , 0.8875,
       0.875 , 0.88  , 0.8775, 0.865 , 0.87  , 0.8725, 0.8725, 0.87  ,
       0.85  , 0.8575, 0.88  , 0.8625, 0.8625, 0.86  , 0.865 , 0.88  ,
       0.875 , 0.8575, 0.845 , 0.87  , 0.8625, 0.8825, 0.87  , 0.845 ,
       0.8725, 0.865 , 0.8625, 0.86  , 0.8625, 0.875 , 0.85  , 0.87  ,
       0.875 , 0.855 , 0.8625, 0.855 , 0.8575, 0.85  , 0.8775, 0.8725,
       0.8825, 0.86  , 0.8675, 0.85  , 0.88  , 0.8625, 0.87  , 0.8675,
       0.88  , 0.88  , 0.855 , 0.88  , 0.8625, 0.865 , 0.875 , 0.8525,
       0.8525, 0.8675, 0.8625, 0.8725, 0.8775, 0.87  , 0.87  , 0.8375,
       0.8425, 0.8725, 0.87  , 0.85  , 0.855 , 0.8575, 0.88  , 0.8525,
       0.85  , 0.88  , 0.86  ])
```

In [33]:
```
print( "The best accuracy was with", mean_acc.max(), "with depth =", mean_acc.argmax()+1)
```
```
The best accuracy was with 0.8975 with depth = 3
```

Decision Tree Classification

In [34]:
```
cf = metrics.classification_report(preds,y_test)
print(cf)
```
```
              precision    recall  f1-score   support

           0       0.92      0.93      0.93       350
           1       0.47      0.40      0.43        50

    accuracy                           0.87       400
   macro avg       0.69      0.67      0.68       400
weighted avg       0.86      0.87      0.86       400
```

Random forest Classifier

In [35]:
```
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
preds = rfc.predict(x_test)
score = rfc.score(x_test,y_test)
score
```
Out[35]: 0.925

In [36]:
```
preds[:5]
```
Out[36]: array([0, 0, 0, 0, 0])

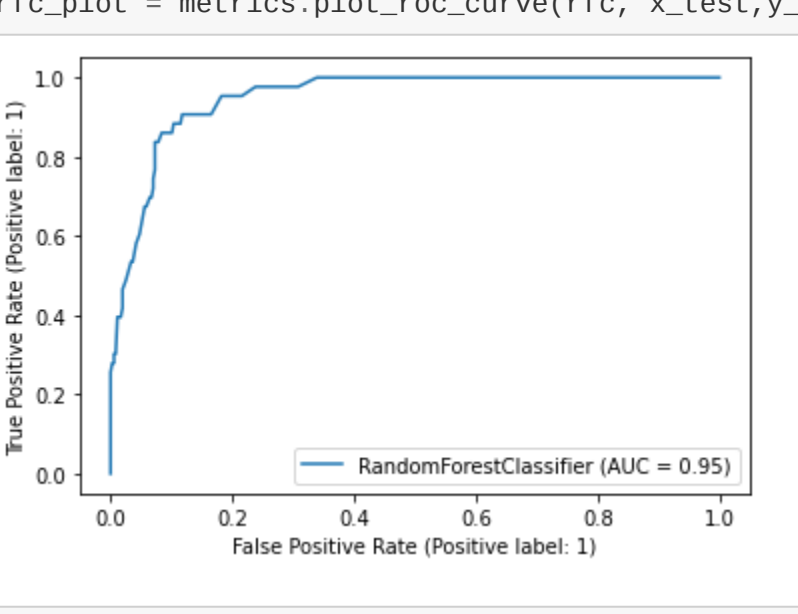
In [37]:
```
y_test[:5]
```
Out[37]:
```
453     1
1415    0
1242    0
885     0
488     1
Name: quality, dtype: int32
```

In [38]:
```
Ks = 100
mean_acc = np.zeros((Ks-1))
for n in range(1,Ks):

    #Train Model and Predict
    rfc = RandomForestClassifier(n_estimators = n).fit(x_train,y_train)
    yhat=dtc.predict(x_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)

mean_acc
```
Out[38]:
```
array([0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86,
       0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86,
       0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86,
       0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86,
       0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86,
       0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86,
       0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86,
       0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86,
       0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.86, 0.8
6])
```

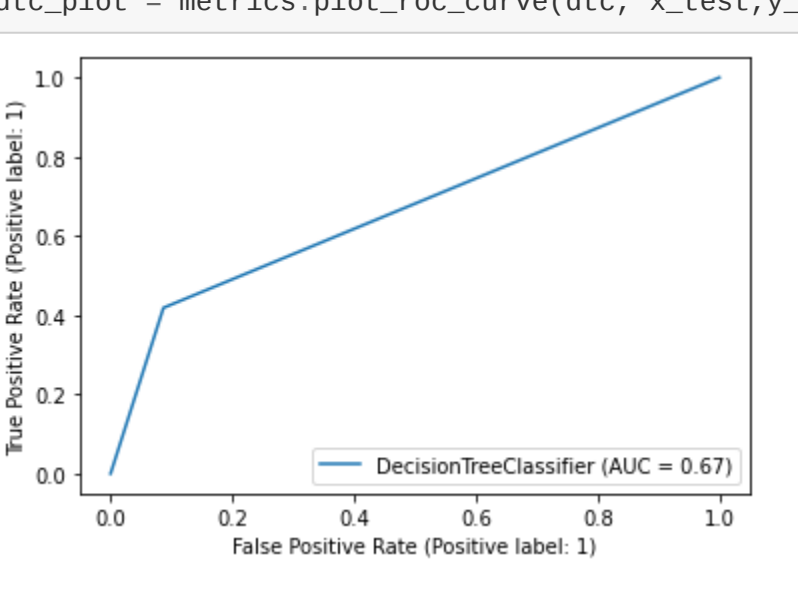
In [39]:
```
print( "The best accuracy was with", mean_acc.max(), "with n_estimator =", mean_acc.argmax()+1)
```
```
The best accuracy was with 0.86 with n_estimator = 1
```

In [40]:
```
cf = metrics.classification_report(preds,y_test)
print(cf)
```
```
              precision    recall  f1-score   support

           0       0.97      0.95      0.96       365
           1       0.56      0.69      0.62        35

    accuracy                           0.93       400
   macro avg       0.76      0.82      0.79       400
weighted avg       0.93      0.93      0.93       400
```

In [41]:
```
rfc_plot = metrics.plot_roc_curve(rfc, x_test,y_test)
```



In [42]:
```
dtc_plot = metrics.plot_roc_curve(dtc, x_test,y_test)
```



Cross Validation

In [43]:
```
dtc_eval = cross_val_score(dtc, x_test, y_test, cv=10)
print('Cross Val Score accuracy is {:.2f}'.format(dtc_eval.mean()))
```
```
Cross Val Score accuracy is 0.85
```

In [44]:
```
rfc_eval = cross_val_score(rfc, x_test, y_test, cv=10)
print('Cross Val Score accuracy is {:.2f}'.format(rfc_eval.mean()))
```
```
Cross Val Score accuracy is 0.90
```

Grid Search for decision tree

In [45]:
```
tree_para = {'criterion':['gini','entropy'],'max_depth':[4,5,6,7,8,9,10,11,12,15,20,30,40,50,70,90,120,150]}
dtc_cv = GridSearchCV(DecisionTreeClassifier(), tree_para, cv=10)
dtc_cv.fit(x_test, y_test)
```
Out[45]:
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [4, 5, 6, 7, 8, 9, 10, 11, 12, 15,
20, 30,
                                       40, 50, 70, 90, 120, 150]})
```

In [46]:
```
dtc_cv.best_params_
```
Out[46]: {'criterion': 'entropy', 'max_depth': 12}

In [47]:
```
dtc_new = DecisionTreeClassifier(criterion='entropy', max_depth = 8)
dtc_new.fit(x_train,y_train)
new_score  = dtc_new.score(x_test, y_test)
new_score
```
Out[47]: 0.9025

For Random forest

In [48]:
```
param_grid = {
    'n_estimators': [200, 500],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [4,5,6,7,8],
    'criterion' :['gini', 'entropy']
}

rfc_cv = GridSearchCV(estimator=rfc, param_grid=param_grid, cv=5)
rfc_cv.fit(x_test, y_test)
```
Out[48]:
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(n_estimators=99),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [4, 5, 6, 7, 8],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [200, 500]})
```

In [49]:
```
rfc_cv.best_params_
```
Out[49]:
```
{'criterion': 'entropy',
 'max_depth': 5,
 'max_features': 'sqrt',
 'n_estimators': 200}
```

In [50]:
```
rfc_new = RandomForestClassifier(criterion='gini', max_depth = 5, max_features='auto', n_estimators=500)
dtc_new.fit(x_train,y_train)
new_score  = dtc_new.score(x_test, y_test)
new_score
```
Out[50]: 0.9025

In [ ]: