Load Dataset

In [2]:
```python
import numpy as np
import pandas as pd
df_wine=pd.read_csv("WineQT.csv")
print(df_wine)
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0               7.4             0.700         0.00             1.9      0.076
1               7.8             0.880         0.00             2.6      0.098
2               7.8             0.760         0.04             2.3      0.092
3              11.2             0.280         0.56             1.9      0.075
4               7.4             0.700         0.00             1.9      0.076
...             ...               ...          ...             ...        ...
1138            6.3             0.510         0.13             2.3      0.076
1139            6.8             0.620         0.08             1.9      0.068
1140            6.2             0.600         0.08             2.0      0.090
1141            5.9             0.550         0.10             2.2      0.062
1142            5.9             0.645         0.12             2.0      0.075

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
4                    11.0                  34.0  0.99780  3.51       0.56
...                   ...                   ...      ...   ...        ...
1138                 29.0                  40.0  0.99574  3.42       0.75
1139                 28.0                  38.0  0.99651  3.42       0.82
1140                 32.0                  44.0  0.99490  3.45       0.58
1141                 39.0                  51.0  0.99512  3.52       0.76
1142                 32.0                  44.0  0.99547  3.57       0.71

      alcohol  quality    Id
0         9.4        5     0
1         9.8        5     1
2         9.8        5     2
3         9.8        6     3
4         9.4        5     4
...       ...      ...   ...
1138     11.0        6  1592
1139      9.5        6  1593
1140     10.5        5  1594
1141     11.2        6  1595
1142     10.2        5  1597

[1143 rows x 13 columns]
```

Analyse Dataset

In [3]:
```python
df=df_wine.drop(labels='Id', axis=1)
```

In [4]:
```python
df.shape
```

Out[4]:
```
(1143, 12)
```

In [5]:
```python
df.head()
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

In [6]:
```python
df.describe()
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000
mean	8.311111	0.531339	0.268364	2.532152	0.086933	15.615486	45.914698	0.996730	3.311015	0.657708	10.442111	5.657043
std	1.747595	0.179633	0.196686	1.355917	0.047267	10.250486	32.782130	0.001925	0.156664	0.170399	1.082196	0.805824
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.392500	0.090000	1.900000	0.070000	7.000000	21.000000	0.995570	3.205000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.250000	2.200000	0.079000	13.000000	37.000000	0.996680	3.310000	0.620000	10.200000	6.000000
75%	9.100000	0.640000	0.420000	2.600000	0.090000	21.000000	61.000000	0.997845	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	68.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

In [7]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 107.3 KB
```

In [8]:
```python
df.isnull().sum()
```

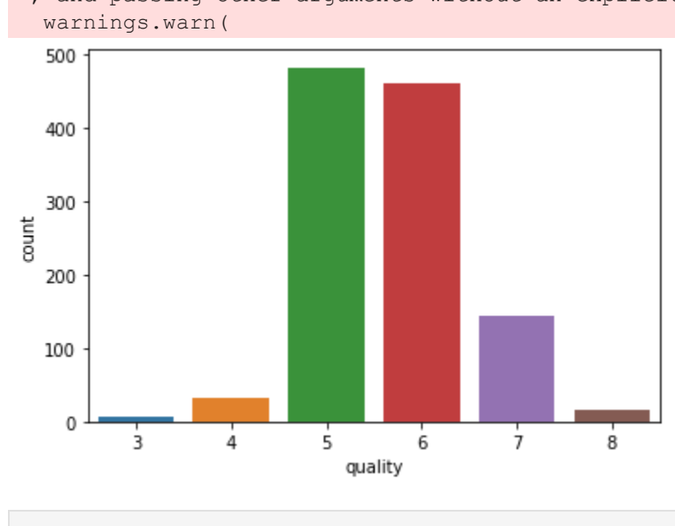
Out[8]:
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

In [9]:
```python
df['quality'].unique()
```

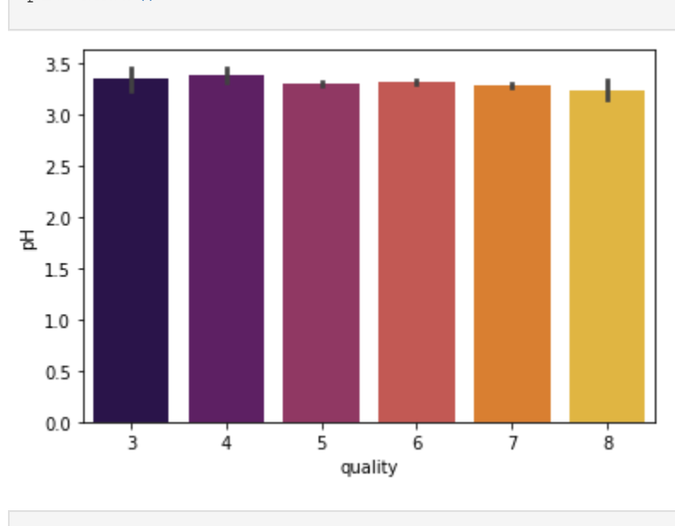
Out[9]:
```
array([5, 6, 7, 4, 8, 3], dtype=int64)
```

In [10]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(6,4))
sns.countplot(df['quality'])
plt.show()
```

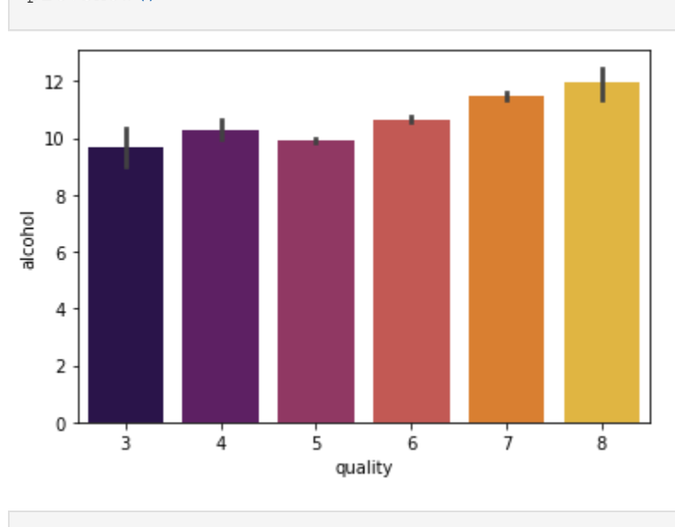
```
c:\users\asus\appdata\local\programs\python\python39\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



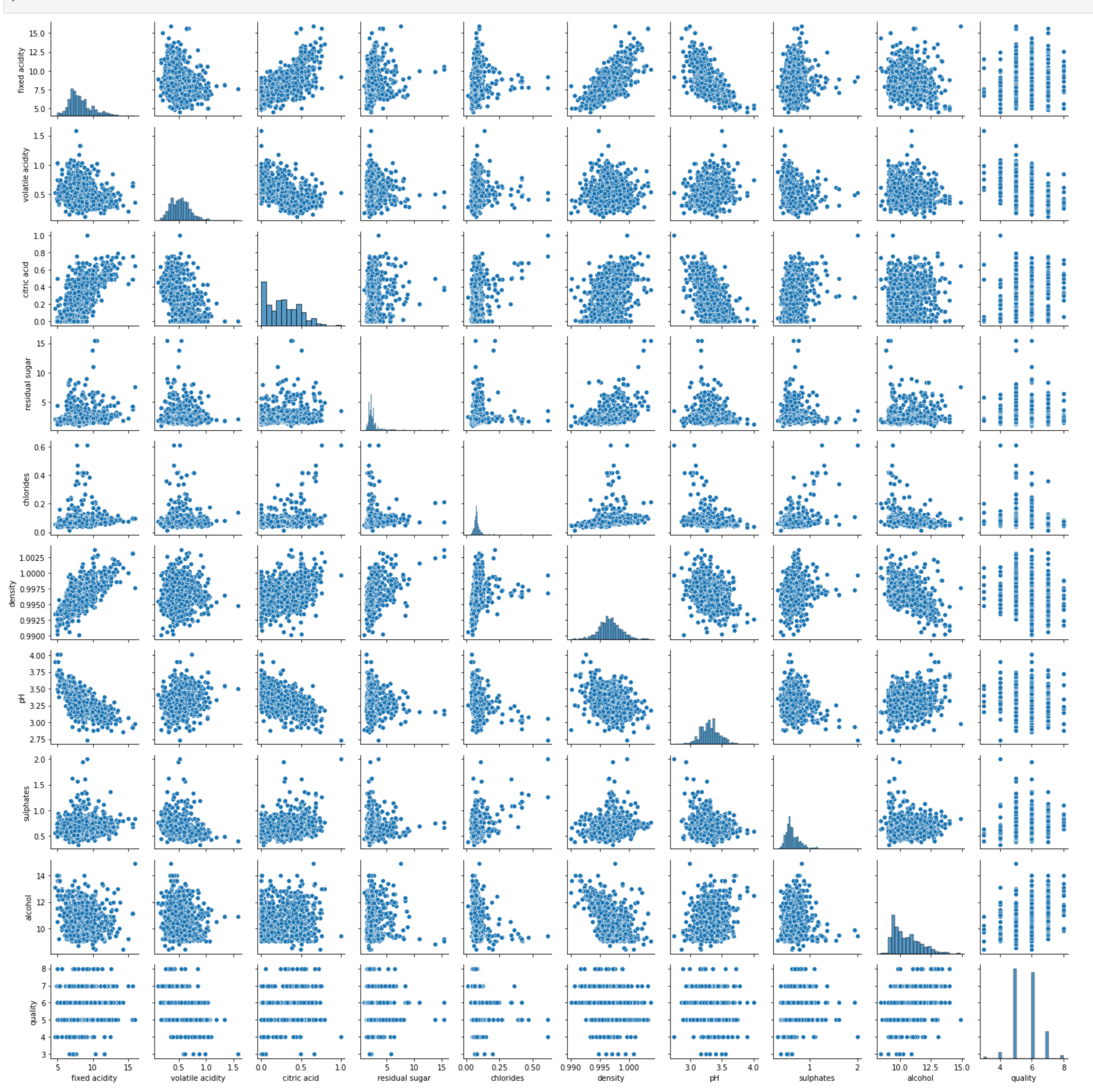
In [11]:
```python
plt.figure(figsize=(6,4))
sns.barplot(x='quality',y='pH',data=df,palette='inferno')
plt.show()
```



In [12]:
```python
plt.figure(figsize=(6,4))
sns.barplot(x='quality',y='alcohol',data=df,palette='inferno')
plt.show()
```



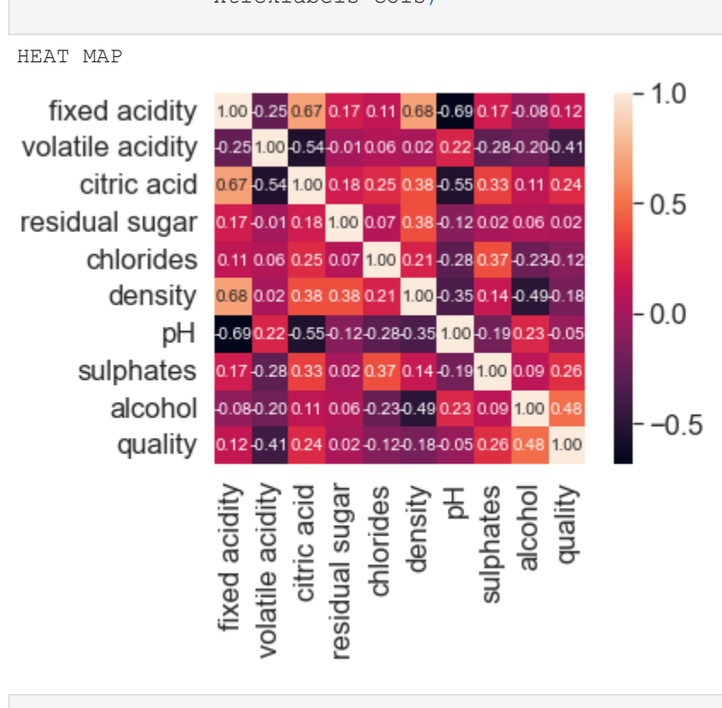
In [13]:
```python
cols=['fixed acidity','volatile acidity','citric acid','residual sugar','chlorides','density','pH','sulphates','alcohol','quality']
sns.pairplot(df[cols],height=2)
plt.tight_layout()
plt.show()
```



In [14]:
```python
#Correlation matrix
print("HEAT MAP")
cm=np.corrcoef(df[cols].values.T)
sns.set(font_scale=1.5)

hm=sns.heatmap(cm,
            cbar=True,
            annot=True,square=True,fmt='.2f',annot_kws={'size':10},
            yticklabels=cols,
            xticklabels=cols)
```

```
HEAT MAP
```



In [15]:
```python
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
print(x)
print(y)
print(x.shape)
print(y.shape)
```

```
[[ 7.4    0.7    0.    ...  3.51   0.56   9.4  ]
 [ 7.8    0.88   0.    ...  3.2    0.68   9.8  ]
 [ 7.8    0.76   0.04  ...  3.26   0.65   9.8  ]
 ...
 [ 6.2    0.6    0.08  ...  3.45   0.58  10.5  ]
 [ 5.9    0.55   0.1   ...  3.52   0.76  11.2  ]
 [ 5.9    0.645  0.12  ...  3.57   0.71  10.2  ]]
[5 5 5 ... 5 6 5]
(1143, 11)
(1143,)
```

In [20]:
```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=0)
print(x_train.shape)
print(x_test.shape)
```

```
(1028, 11)
(115, 11)
```

In [21]:
```python
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train_sc=sc.fit_transform(x_train)
x_test_sc=sc.fit_transform(x_test)
```

In [22]:
```python
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

Implement three Models with hyperParameter tuning

In [24]:
```python
#Random forest

from sklearn.ensemble import RandomForestClassifier
forest=RandomForestClassifier(criterion='entropy',max_features=3,n_estimators=180,n_jobs=3)
forest.fit(x_train_sc,y_train)
print("Accuracy of random forest on training dataset: ",forest.score(x_train_sc,y_train))

y_pred_forest=forest.predict(x_test_sc)
forest_score=accuracy_score(y_test,y_pred_forest)
forest_results = confusion_matrix(y_test, y_pred_forest)
print('Miscalssified samples: %d'%(y_test!=y_pred_forest).sum())
print('Accuracy score of Random forest on test datset: %.2f'%forest_score)
print("Result of Confusion matrix: \n",forest_results )
```

```
Accuracy of random forest on training dataset:  1.0
Miscalssified samples: 36
Accuracy score of Random forest on test datset: 0.69
Result of Confusion matrix:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 39 13  0  0]
 [ 0  0 11 33  1  0]
 [ 0  0  0  5  7  0]
 [ 0  0  0  0  2  0]]
```

In [25]:
```python
#tuning of Hyperparameter-----------------------------------------------------------TUNING

from sklearn.model_selection import GridSearchCV
import numpy as np

max_features_range = np.arange(1,12,1)

n_estimators_range = np.arange(10,210,10)
param_grid   = dict(max_features=max_features_range,n_estimators=n_estimators_range)
```

In [26]:
```python
rf=RandomForestClassifier()
grid = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5)
```

In [27]:
```python
grid.fit(x_train_sc,y_train)
```

Out[27]:
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'max_features': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
                         'n_estimators': array([ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100, 110, 120, 130,
       140, 150, 160, 170, 180, 190, 200])})
```

In [28]:
```python
print("The best parameter are %s with a score of %0.2f"
      % (grid.best_params_,grid.best_score_))
```

```
The best parameter are {'max_features': 2, 'n_estimators': 160} with a score of 0.67
```

In [30]:
```python
grid_results = pd.concat([pd.DataFrame(grid.cv_results_["params"]),pd.DataFrame(grid.cv_results_["mean_test_score"], columns=["Accuracy"])],axis=1)
grid_results.head()
```

Out[30]:

	max_features	n_estimators	Accuracy
0	1	10	0.618707
1	1	20	0.627407
2	1	30	0.641085
3	1	40	0.640066
4	1	50	0.651722

In [31]:
```python
#Decision tree
from sklearn.tree import DecisionTreeClassifier
tree=DecisionTreeClassifier(criterion='entropy',max_depth=3)
tree.fit(x_train_sc,y_train)
```

Out[31]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

In [32]:
```python
y_pred_tree=tree.predict(x_test_sc)

print("Accuracy of Decision tree model on training set: ", tree.score(x_train_sc,y_train))
tree_results=confusion_matrix(y_test,y_pred_tree)
tree_score=accuracy_score(y_test,y_pred_tree)
print('Miscalssified samples: %d'%(y_test!=y_pred_tree).sum())
print('Accuracy score of Decision Tree model on test set: %.2f'%tree_score)
print("Result of Confusion matrix: \n",tree_results )
```

```
Accuracy of Decision tree model on training set:  0.5963035019455253
Miscalssified samples: 49
Accuracy score of Decision Tree model on test set: 0.57
Result of Confusion matrix:
 [[ 0  0  0  1  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 40 12  0  0]
 [ 0  0 15 22  8  0]
 [ 0  0  2  6  4  0]
 [ 0  0  0  1  1  0]]
```

In [33]:
```python
param_grid2   = {"criterion":["gini","entropy"],
                 "max_depth":[1,2,3,4,5,6,7,None]
                }
```

In [34]:
```python
grid2 = GridSearchCV(tree, param_grid=param_grid2, cv=10, n_jobs=-1)
```

In [35]:
```python
grid2.fit(x_train_sc,y_train)
```

```
c:\users\asus\appdata\local\programs\python\python39\lib\site-packages\sklearn\model_selection\_split.py:676: UserWarning: The least populated class in y has only 5 members, which is less than n_splits=10.
  warnings.warn(
```

Out[35]:
```
GridSearchCV(cv=10,
             estimator=DecisionTreeClassifier(criterion='entropy', max_depth=3),
             n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [1, 2, 3, 4, 5, 6, 7, None]})
```

In [36]:
```python
grid2.best_score_
```

Out[36]:
```
0.5875975632971635
```

In [37]:
```python
grid2.best_params_
```

Out[37]:
```
{'criterion': 'entropy', 'max_depth': 3}
```

In [38]:
```python
print("The best parameter are %s with a score of %0.2f"
      % (grid2.best_params_,grid2.best_score_))
```

```
The best parameter are {'criterion': 'entropy', 'max_depth': 3} with a score of 0.59
```

In [39]:
```python
#Knn----------------------------------------------KNN-----------------------------
from sklearn.neighbors import KNeighborsClassifier
knn_scorelist = []
for k in range(21):
    k = k+1
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(x_train_sc,y_train)
    knn_pred=knn.predict(x_test_sc)
    knn_results = confusion_matrix(y_test,knn_pred)

    knnacc_score = accuracy_score(y_test, knn_pred)
    print("The Accuracy of Knn model for k=",k,"is:%0.4f"%knnacc_score)
    print("The confusion matrix is: \n", knn_results)

    knn_scorelist.append(knnacc_score)
```

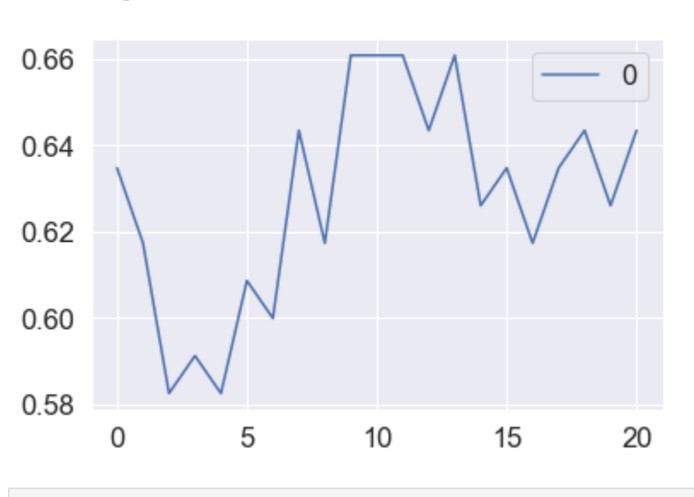
```
The Accuracy of Knn model for k= 1 is:0.6348
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  1 38 13  0  0]
 [ 0  0 13 29  3  0]
 [ 0  0  2  4  6  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 2 is:0.6174
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  2  1  0  0  0]
 [ 0  2 42  8  0  0]
 [ 0  0 19 24  2  0]
 [ 0  0  3  6  3  0]
 [ 0  0  0  2  0  0]]
The Accuracy of Knn model for k= 3 is:0.5826
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  1  0  2  0  0]
 [ 0  0 34 17  1  0]
 [ 0  0 14 28  3  0]
 [ 0  0  2  6  4  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 4 is:0.5913
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  1  1  1  0  0]
 [ 0  0 38 12  2  0]
 [ 0  1 18 24  2  0]
 [ 0  0  2  5  5  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 5 is:0.5826
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  1  2  0  0]
 [ 0  0 32 18  2  0]
 [ 0  0 13 29  3  0]
 [ 0  0  2  4  6  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 6 is:0.6087
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  1  2  0  0]
 [ 0  0 39 11  2  0]
 [ 0  0 15 27  3  0]
 [ 0  0  1  7  4  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 7 is:0.6000
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  1  2  0  0]
 [ 0  1 36 14  1  0]
 [ 0  0 13 28  4  0]
 [ 0  0  1  6  5  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 8 is:0.6435
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  1  2  0  0]
 [ 0  1 39 11  1  0]
 [ 0  0 11 30  4  0]
 [ 0  0  1  6  5  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 9 is:0.6174
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  1  2  0  0]
 [ 0  1 37 14  0  0]
 [ 0  0 13 29  3  0]
 [ 0  0  1  6  5  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 10 is:0.6609
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 38 14  0  0]
 [ 0  0 10 33  2  0]
 [ 0  0  1  6  5  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 11 is:0.6609
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 37 15  0  0]
 [ 0  0  8 35  2  0]
 [ 0  0  1  7  4  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 12 is:0.6609
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 39 13  0  0]
 [ 0  0  9 33  3  0]
 [ 0  0  1  7  4  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 13 is:0.6435
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 35 17  0  0]
 [ 0  0  8 35  2  0]
 [ 0  0  1  7  4  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 14 is:0.6609
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 37 15  0  0]
 [ 0  0  8 34  3  0]
 [ 0  0  1  6  5  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 15 is:0.6261
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 34 18  0  0]
 [ 0  0 10 32  3  0]
 [ 0  0  1  5  6  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 16 is:0.6348
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 35 17  0  0]
 [ 0  0 10 33  2  0]
 [ 0  0  1  6  5  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 17 is:0.6174
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 36 16  0  0]
 [ 0  0 12 30  3  0]
 [ 0  0  0  7  5  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 18 is:0.6348
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 36 16  0  0]
 [ 0  0 12 32  1  0]
 [ 0  0  0  7  5  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 19 is:0.6435
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 37 15  0  0]
 [ 0  0 11 32  2  0]
 [ 0  0  0  7  5  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 20 is:0.6261
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 37 15  0  0]
 [ 0  0 12 31  2  0]
 [ 0  0  0  8  4  0]
 [ 0  0  0  1  1  0]]
The Accuracy of Knn model for k= 21 is:0.6435
The confusion matrix is:
 [[ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0 37 15  0  0]
 [ 0  0 11 33  1  0]
 [ 0  0  0  8  4  0]
 [ 0  0  0  1  1  0]]
```

In [46]:
```python
print("Maximum accuracy using knnmodel is: %0.4f " %max(knn_scorelist))
```

```
Maximum accuracy using knnmodel is: 0.6609
```

In [42]:
```python
curve=pd.DataFrame(knn_scorelist)
curve.plot()
```

Out[42]:
```
<AxesSubplot:>
```



In [47]:
```python
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=10, p=2)
knn.fit(x_train_sc,y_train)
```

Out[47]:
```
KNeighborsClassifier(n_neighbors=10)
```

In [48]:
```python
knn_pred=knn.predict(x_test_sc)

print('misclassified samples: %d'%(y_test!=knn_pred).sum())

from sklearn.metrics import accuracy_score
knn_score=accuracy_score(y_test,knn_pred)
print('Accuracy:%.2f'%knn_score)
```

```
misclassified samples: 39
Accuracy:0.66
```

Ensemble Models

In [49]:
```python
#Ensemble model

#------------------------Adaboost model-----------
from sklearn.ensemble import AdaBoostClassifier
ada=AdaBoostClassifier(n_estimators=100,random_state=0)
ada.fit(x_train_sc,y_train)
```

Out[49]:
```
AdaBoostClassifier(n_estimators=100, random_state=0)
```

In [50]:
```python
ada_pred=ada.predict(x_test_sc)
ada_results=confusion_matrix(y_test,ada_pred)
ada_score=accuracy_score(y_test,ada_pred)
print("Accuracy of Decision tree model on training set: ", ada.score(x_train_sc,y_train))
print('Miscalssified samples: %d'%(y_test!=ada_pred).sum())
print('Accuracy score of Decision Tree model on test set: %.2f'%ada_score)
print("Result of Confusion matrix: \n",ada_results )
```

```
Accuracy of Decision tree model on training set:  0.5369649805447471
Miscalssified samples: 41
Accuracy score of Decision Tree model on test set: 0.64
Result of Confusion matrix:
 [[ 0  0  0  1  0  0]
 [ 0  0  1  2  0  0]
 [ 0  0 45  7  0  0]
 [ 0  0 15 28  0  2]
 [ 0  0  2  6  0  4]
 [ 0  0  0  1  0  1]]
```

In [51]:
```python
#-----------------Bagging model----------------------------------
from sklearn.ensemble import BaggingClassifier
bag = BaggingClassifier(n_estimators=100, random_state=0)
bag.fit(x_train_sc,y_train)
```

Out[51]:
```
BaggingClassifier(n_estimators=100, random_state=0)
```

In [52]:
```python
bag_pred=ada.predict(x_test_sc)
bag_results=confusion_matrix(y_test,bag_pred)
bag_score=accuracy_score(y_test,bag_pred)
print("Accuracy of Decision tree model on training set: ", bag.score(x_train_sc,y_train))
print('Miscalssified samples: %d'%(y_test!=bag_pred).sum())
print('Accuracy score of Decision Tree model on test set: %.2f'%bag_score)
print("Result of Confusion matrix: \n",bag_results )
```

```
Accuracy of Decision tree model on training set:  1.0
Miscalssified samples: 41
Accuracy score of Decision Tree model on test set: 0.64
Result of Confusion matrix:
 [[ 0  0  0  1  0  0]
 [ 0  0  1  2  0  0]
 [ 0  0 45  7  0  0]
 [ 0  0 15 28  0  2]
 [ 0  0  2  6  0  4]
 [ 0  0  0  1  0  1]]
```

Comparing the Accuracy of models on test datasets

In [53]:
```python
names = ["Random_forest","Knn","Decision_tree","Adaboost","Bagging"]
classifiers = [
    rf,knn,tree,ada,bag]

scores = [forest_score,knn_score,tree_score,ada_score,bag_score]
```

In [54]:
```python
df = pd.DataFrame()
df['name'] = names
df['Accuracy score'] = scores
df
```

Out[54]:

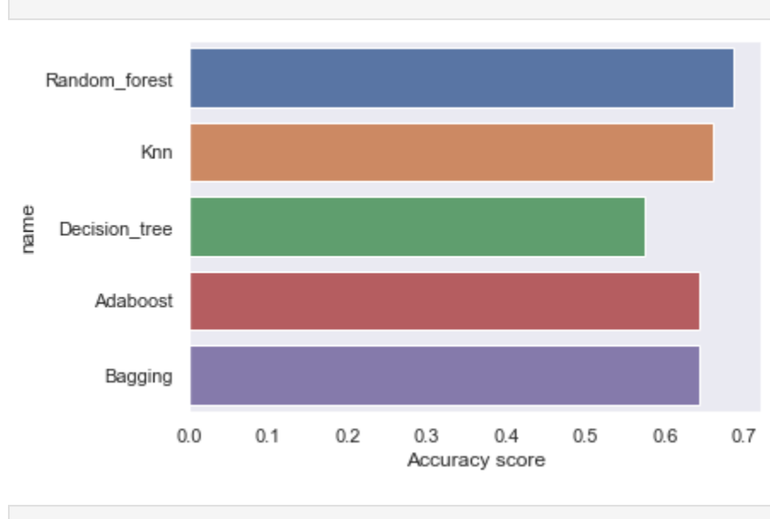
	name	Accuracy score
0	Random_forest	0.686957
1	Knn	0.660870
2	Decision_tree	0.573913
3	Adaboost	0.643478
4	Bagging	0.643478

In [57]:
```python
cm = sns.light_palette("yellow", as_cmap=True)
s = df.style.background_gradient(cmap=cm)
s
```

Out[57]:

	name	Accuracy score
0	Random_forest	0.686957
1	Knn	0.660870
2	Decision_tree	0.573913
3	Adaboost	0.643478
4	Bagging	0.643478

In [59]:
```python
sns.set(style="dark")
ax = sns.barplot(y="name", x="Accuracy score", data=df)
```



In [ ]: