Data:

```
data=pd.read_csv('seeds_dataset.csv')
data.head()
# print(data.shape)
```

	area	perimeter	compactness	length_of_kernel	width_of_kernel	asymmetry_coefficient	length_of_kernel_groove	type_of_wheat
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	1
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	1
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	1
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	1
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	1

Standard Scaler:

Standard Scaler

```
def standard_scaler(data1):
    for column in data1.columns:
        data1[column]=(data1[column]-mean(data1[column]))/(std(data1[column]))
    return data1

data1=standard_scaler(data1)
data1.head()
```

	area	perimeter	compactness	length_of_kernel	width_of_kernel	asymmetry_coefficient	length_of_kernel_groove
0	0.142098	0.215462	0.000061	0.304218	0.141702	-0.986152	-0.383577
1	0.011188	0.008224	0.428515	-0.168625	0.197432	-1.788166	-0.922013
2	-0.192067	-0.360201	1.442383	-0.763637	0.208048	-0.667479	-1.189192
3	-0.347091	-0.475333	1.039381	-0.688978	0.319508	-0.960818	-1.229983
4	0.445257	0.330595	1.374509	0.066666	0.805159	-1.563495	-0.475356

Categorical Encoding is Done:

categorical encoding

[8]:
```
def categorical_encode(data1):
    label=[]
    for i in range(len(data1.columns)):
        label.append(i)
    print("label ",label,"len ",len(label))
#     print(data1["Age"][0])
    length = len(data1)
#     print(length)
    column_names=[]
    for columns in data1:
        temp=[]
        column_names.append("cat_"+columns)
        for j in (0,length-1):
            temp.append(data1[columns][j])
        temp = np.array(temp)
        data1["cat_"+columns] = pd.cut(data1[columns].values, bins = len(label), labels = label)

    data1 = data1[column_names]
    return data1

data1=categorical_encode(data1)
data1.head()
```

```
label  [0, 1, 2, 3, 4, 5, 6] len  7
```

:[8]:

	cat_area	cat_perimeter	cat_compactness	cat_length_of_kernel	cat_width_of_kernel	cat_asymmetry_coefficient	cat_length_of_kernel_groove
0	3	3	3	3	3	1	2
1	2	3	4	2	3	0	1
2	2	2	6	1	3	1	1
3	2	2	5	1	3	1	0
4	3	3	6	2	4	0	2

`

SVM is implemented using linear, quadratic and rbf kernels with the accuracy for each printed below:

SVM

```
from sklearn import svm
from sklearn import metrics

###linear kernel
clf1 = svm.SVC(kernel='linear')
clf1.fit(x_train, y_train)
y_pred = clf1.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred)*100)
```

```
Accuracy: 85.71428571428571
```

```
###Quadratic Kernel
clf2 = svm.SVC(kernel='poly')
clf2.fit(x_train, y_train)
y_pred = clf2.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred)*100)
```

```
Accuracy: 85.71428571428571
```

```
###Radial Basis Function
clf3 = svm.SVC(kernel='rbf')
clf3.fit(x_train, y_train)
y_pred = clf3.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred)*100)
```

```
Accuracy: 76.19047619047619
```

MLP is built as per the given specifications, The 2 layer MLP gave the best results

```
Accuracy of MLPClassifier with 1 layer:  76.19047619047619
```

```
Accuracy of MLPClassifier  with 2 layers:  83.33333333333334
```

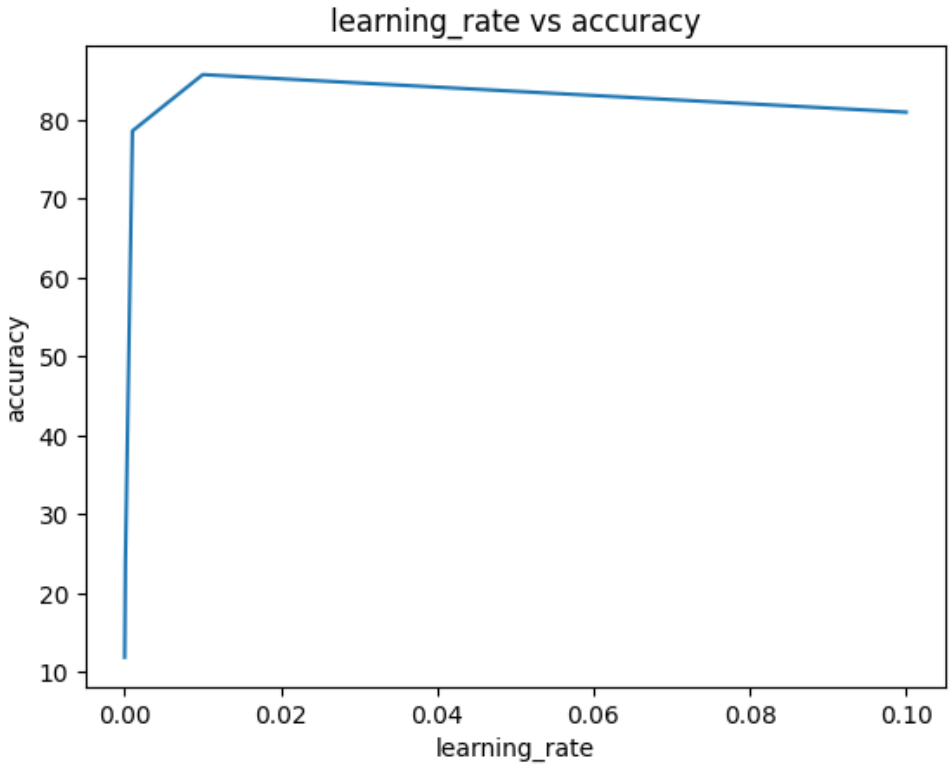
Learning Rate vs Accuracy Graph:

```
Accuracy of MLPClassifier with lr: 0.1  :  80.95238095238095
Accuracy of MLPClassifier with lr: 0.01  :  85.71428571428571
Accuracy of MLPClassifier with lr: 0.001  :  78.57142857142857
Accuracy of MLPClassifier with lr: 0.0001  :  23.809523809523807
Accuracy of MLPClassifier with lr: 1e-05  :  11.904761904761903
accuradcy_list :  [80.95238095238095, 85.71428571428571, 78.57142857142857, 23.809523809523807, 11.904761904761903]
lr_list :  [0.1, 0.01, 0.001, 0.0001, 1e-05]
```

Lr is the learning rate.

The above figure represents the accuracy for different learning rates

`



Features obtained using forward selection:

Forward Feature Selection

```
from sklearn.feature_selection import SequentialFeatureSelector

sfs = SequentialFeatureSelector(classifier2,
                                n_features_to_select = 'auto',
                                direction='forward')
```

```
sfs.fit(x_train, y_train)
print("Top features selected by forward sequential selection:{}"\
      .format(list(X.columns[sfs.get_support()])))
```

Top Features received after forward selection:

```
Top features selected by forward sequential selection:['cat_area', 'cat_width_of_kernel', 'cat_length_of_kernel_groove']
```

Result after ensemble learning:

`

Max-Voting

```
from sklearn.ensemble import VotingClassifier
```

```
eclf1 = VotingClassifier(estimators=[('svm_quad', clf2), ('svm_rbf', clf3), ('mlp', classifier2)], voting='hard')
eclf1 = eclf1.fit(x_train, y_train)
#Comparing the predictions against the actual observations in y_pred
eclf_pred = (eclf1.predict(x_test))
cm = confusion_matrix(eclf_pred, y_test)

print("Accuracy of MLPClassifier :",  '', accuracy(cm) *100)
```

```
Accuracy of MLPClassifier :  83.33333333333334
```

`