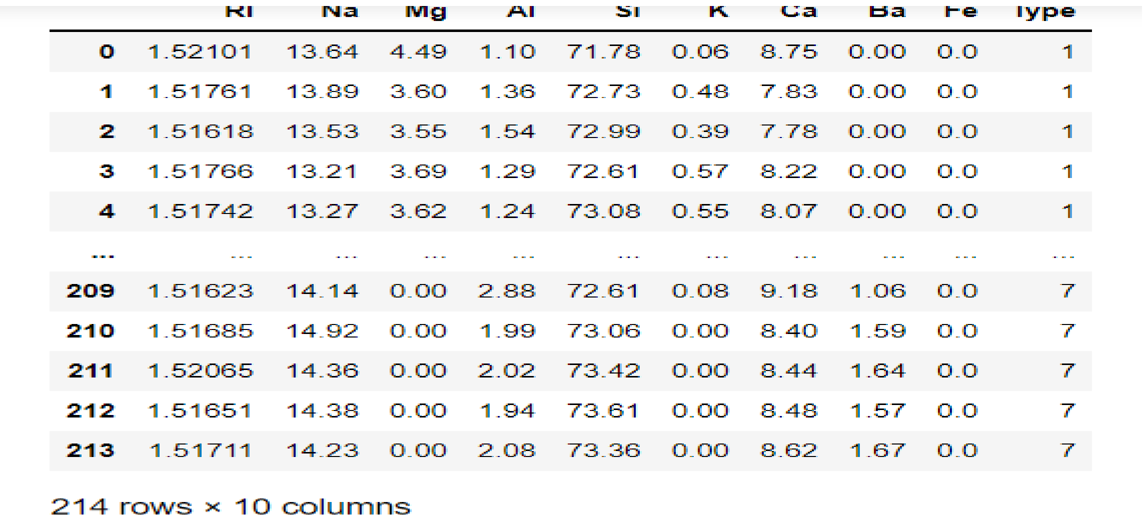
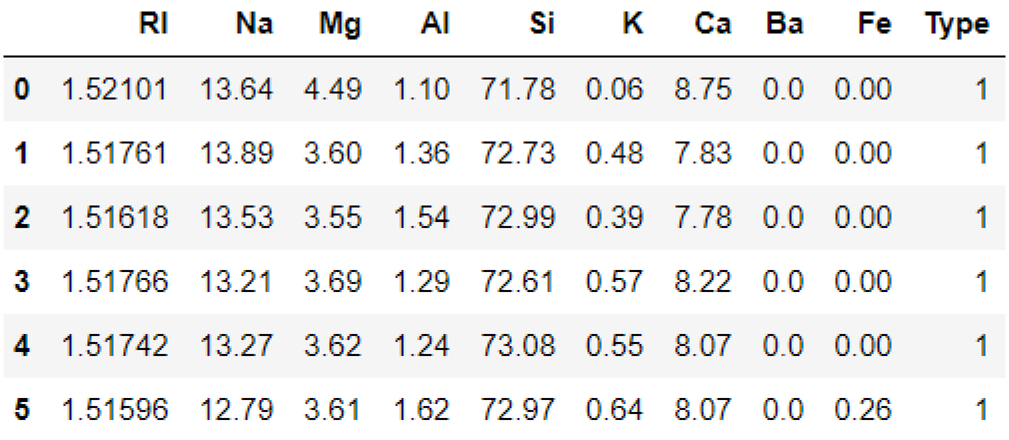
```
—-----------import important libraries—--------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import KFold,cross_val_score,GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,accuracy_score
from sklearn.preprocessing import StandardScaler

—-------read dataset—-------------------
data = pd.read_csv('Downloads/Glass.csv')
data
```



	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0	1
...	...	...	...	...	...	...	...	...	...	...
209	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0	7
210	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0	7
211	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0	7
212	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0	7
213	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0	7

214 rows × 10 columns

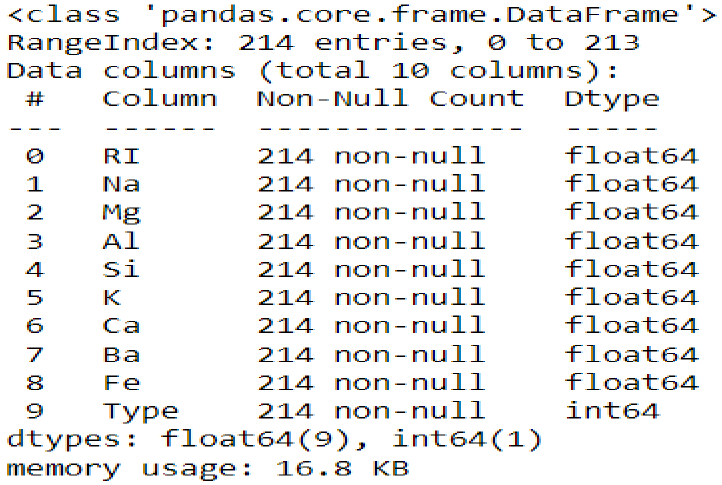
```
—------read rows—--------------
data.head(6)
```



	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.00	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.00	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.00	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.00	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.00	1
5	1.51596	12.79	3.61	1.62	72.97	0.64	8.07	0.0	0.26	1

```
—-------data info—-------------------
```

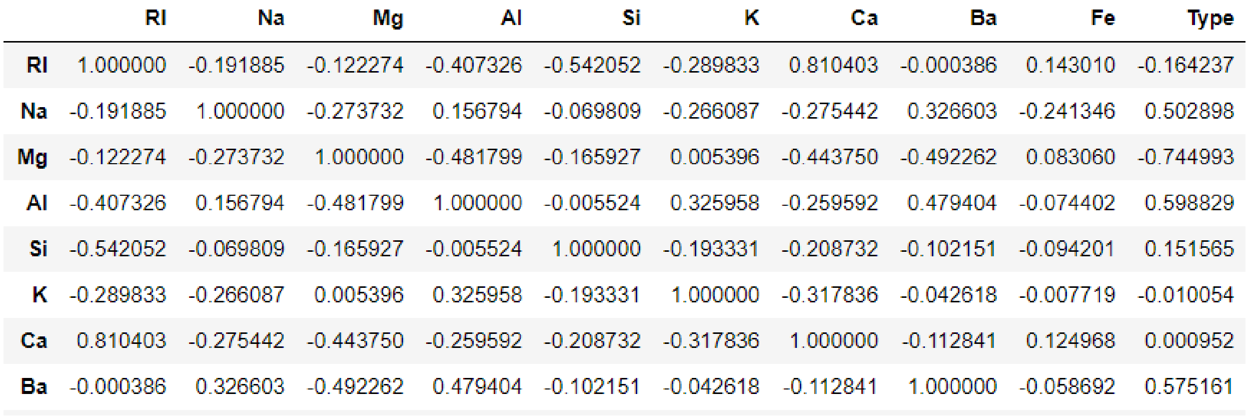
data.info()



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64  
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

—-----correlation—------------------

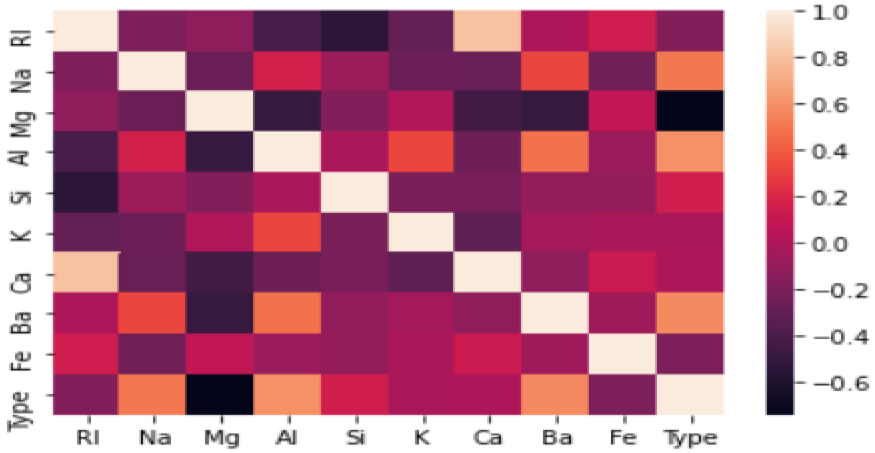
data.corr()



	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
RI	1.000000	-0.191885	-0.122274	-0.407326	-0.542052	-0.289833	0.810403	-0.000386	0.143010	-0.164237
Na	-0.191885	1.000000	-0.273732	0.156794	-0.069809	-0.266087	-0.275442	0.326603	-0.241346	0.502898
Mg	-0.122274	-0.273732	1.000000	-0.481799	-0.165927	0.005396	-0.443750	-0.492262	0.083060	-0.744993
Al	-0.407326	0.156794	-0.481799	1.000000	-0.005524	0.325958	-0.259592	0.479404	-0.074402	0.598829
Si	-0.542052	-0.069809	-0.165927	-0.005524	1.000000	-0.193331	-0.208732	-0.102151	-0.094201	0.151565
K	-0.289833	-0.266087	0.005396	0.325958	-0.193331	1.000000	-0.317836	-0.042618	-0.007719	-0.010054
Ca	0.810403	-0.275442	-0.443750	-0.259592	-0.208732	-0.317836	1.000000	-0.112841	0.124968	0.000952
Ba	-0.000386	0.326603	-0.492262	0.479404	-0.102151	-0.042618	-0.112841	1.000000	-0.058692	0.575161

—-----------plot—---------------

sns.heatmap(data.corr())



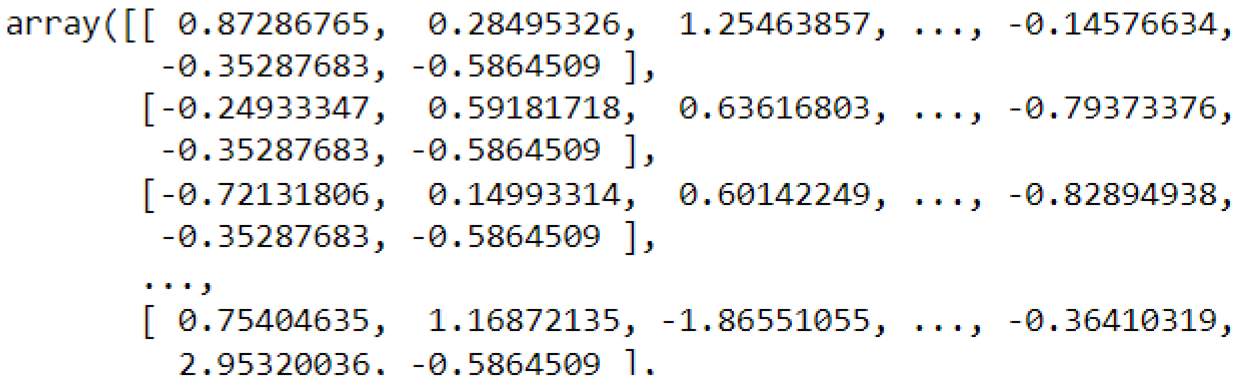
—----------scale—-----------------

scaler = StandardScaler()

scaler.fit(data.drop('Type',axis=1))

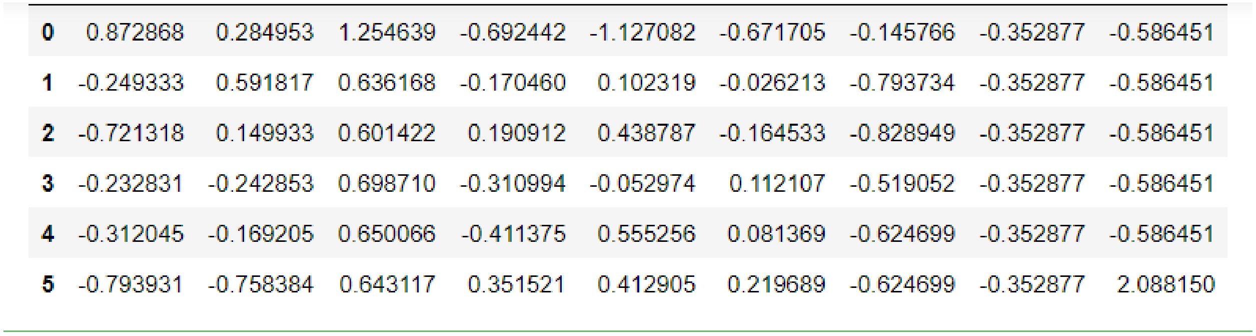
```
StandardScaler()
```

scaled_features = scaler.transform(data.drop('Type',axis=1))
scaled_features



```
array([[ 0.87286765,  0.28495326,  1.25463857, ..., -0.14576634,
        -0.35287683, -0.5864509 ],
       [-0.24933347,  0.59181718,  0.63616803, ..., -0.79373376,
        -0.35287683, -0.5864509 ],
       [-0.72131806,  0.14993314,  0.60142249, ..., -0.82894938,
        -0.35287683, -0.5864509 ],
       ...,
       [ 0.75404635,  1.16872135, -1.86551055, ..., -0.36410319,
         2.95320036, -0.5864509 ],
```

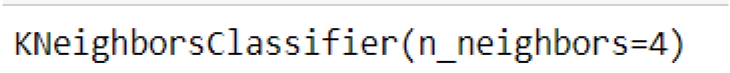
—---------dataframe—-----------------
datafeat = pd.DataFrame(scaled_features)
datafeat.head(6)



0	0.872868	0.284953	1.254639	-0.692442	-1.127082	-0.671705	-0.145766	-0.352877	-0.586451
1	-0.249333	0.591817	0.636168	-0.170460	0.102319	-0.026213	-0.793734	-0.352877	-0.586451
2	-0.721318	0.149933	0.601422	0.190912	0.438787	-0.164533	-0.828949	-0.352877	-0.586451
3	-0.232831	-0.242853	0.698710	-0.310994	-0.052974	0.112107	-0.519052	-0.352877	-0.586451
4	-0.312045	-0.169205	0.650066	-0.411375	0.555256	0.081369	-0.624699	-0.352877	-0.586451
5	-0.793931	-0.758384	0.643117	0.351521	0.412905	0.219689	-0.624699	-0.352877	2.088150

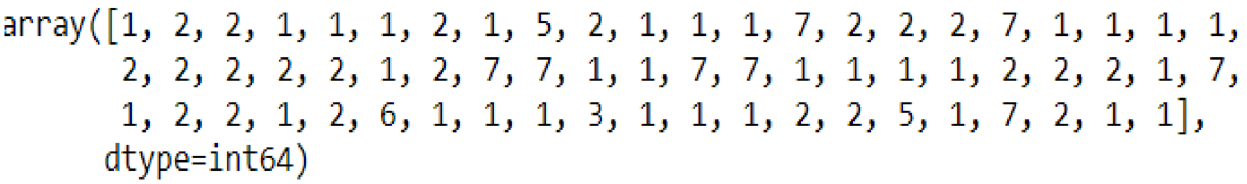
dff = datafeat.drop([6,7],axis=1)
x_train,x_test,y_train,y_test = train_test_split(dff,data['Type'],test_size=0.3)

—-----------knn—-----------------
knn = KNeighborsClassifier(n_neighbors=4)
knn.fit(x_train,y_train)



```
KNeighborsClassifier(n_neighbors=4)
```

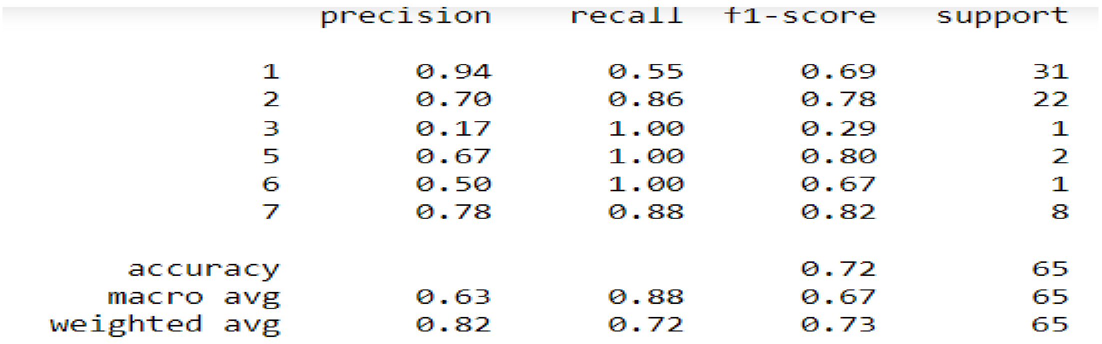
—---------prediction—-----------------
ypredict = knn.predict(x_test)
ypredict



```
array([1, 2, 2, 1, 1, 1, 2, 1, 5, 2, 1, 1, 1, 7, 2, 2, 2, 7, 1, 1, 1, 1,
       2, 2, 2, 2, 2, 1, 2, 7, 7, 1, 1, 7, 7, 1, 1, 1, 1, 2, 2, 2, 1, 7,
       1, 2, 2, 1, 2, 6, 1, 1, 1, 3, 1, 1, 1, 2, 2, 5, 1, 7, 2, 1, 1],
      dtype=int64)
```

—--------classification report—----------------

print(classification_report(ypredict,y_test))

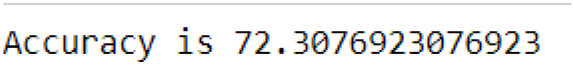


```
              precision    recall  f1-score   support

           1       0.94      0.55      0.69        31
           2       0.70      0.86      0.78        22
           3       0.17      1.00      0.29         1
           5       0.67      1.00      0.80         2
           6       0.50      1.00      0.67         1
           7       0.78      0.88      0.82         8

    accuracy                           0.72        65
   macro avg       0.63      0.88      0.67        65
weighted avg       0.82      0.72      0.73        65
```

—----------accuracy—-----------

a = accuracy_score(ypredict,y_test)*100

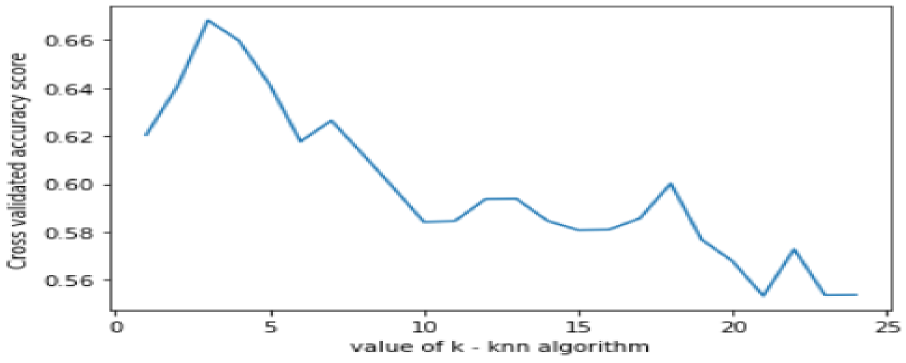
print('Accuracy is',a)



```
Accuracy is 72.3076923076923
```

—------plot—-----------------

```
krange = range[1,25]
kscore = []
for k in krange:
   knn = KNeighborsClassifier(n_neighbors=k)
   score = cross_val_score(knn,x,y,cv=10)
   kscore.append(score_mean())
plt.plot(krange,kscore)
plt.xlabel('value of k - knn algorithm')
plt.ylabel('Cross validated accuracy score')
```


Cross validated accuracy score
value of k - knn algorithm