Skill 3

Title : Perform Classification on the Glass Dataset

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [19]:

```
#importing dataset
df=pd.read_csv("glass.csv")
df.head()
```

Out[19]:

	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	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

In [21]:

```
df.tail(15)
```

Out[21]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
199	1.51609	15.01	0.0	2.51	73.05	0.05	8.83	0.53	0.0	7
200	1.51508	15.15	0.0	2.25	73.50	0.00	8.34	0.63	0.0	7
201	1.51653	11.95	0.0	1.19	75.18	2.70	8.93	0.00	0.0	7
202	1.51514	14.85	0.0	2.42	73.72	0.00	8.39	0.56	0.0	7
203	1.51658	14.80	0.0	1.99	73.11	0.00	8.28	1.71	0.0	7
204	1.51617	14.95	0.0	2.27	73.30	0.00	8.71	0.67	0.0	7
205	1.51732	14.95	0.0	1.80	72.99	0.00	8.61	1.55	0.0	7
206	1.51645	14.94	0.0	1.87	73.11	0.00	8.67	1.38	0.0	7
207	1.51831	14.39	0.0	1.82	72.86	1.41	6.47	2.88	0.0	7
208	1.51640	14.37	0.0	2.74	72.85	0.00	9.45	0.54	0.0	7
209	1.51623	14.14	0.0	2.88	72.61	0.08	9.18	1.06	0.0	7
210	1.51685	14.92	0.0	1.99	73.06	0.00	8.40	1.59	0.0	7
211	1.52065	14.36	0.0	2.02	73.42	0.00	8.44	1.64	0.0	7
212	1.51651	14.38	0.0	1.94	73.61	0.00	8.48	1.57	0.0	7
213	1.51711	14.23	0.0	2.08	73.36	0.00	8.62	1.67	0.0	7

Data Preprocessing

In [4]:

```
df.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
```

In [5]:

```
df.describe()
```

Out[5]:

	RI	Na	Mg	Al	Si	K	Ca	
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	21
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	
25%	1.516522	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	

In [6]:

```
df.isna().sum()
```

Out[6]:

```
RI      0
Na      0
Mg      0
Al      0
Si      0
K       0
Ca      0
Ba      0
Fe      0
Type    0
dtype: int64
```

In [7]:

```
y = df['Type']
X = df.drop('Type', axis=1)
```

In [8]:

```
#scaling data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X = pd.DataFrame(scaler.fit_transform(X))
```

#Spliting dataset into train and test

In [9]:

```
from sklearn.model_selection import train_test_split
```

In [10]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7)
```

Logistic Regression Model

In [11]:

```
from sklearn.linear_model import LogisticRegression
log_model = LogisticRegression()
log_model.fit(X_train, y_train)
y_pred=log_model.predict(X_test)
```

In [12]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

Out[12]:

```
0.5384615384615384
```

Support Vector Machine Model

In [13]:

```
from sklearn.svm import SVC                 #importing model SVC
model=SVC(kernel="linear",C=1)                 #declaring model
model.fit(X_train,y_train)
ypred=model.predict(X_test)
```

In [14]:

```
from sklearn.metrics import accuracy_score
acc=accuracy_score(y_test,ypred)
```

In [15]:

```
acc
```

Out[15]:

```
0.5384615384615384
```

K-Nearest Neibours

In [16]:

```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier(n_neighbors=5)
model.fit(X_train,y_train)
```

Out[16]:

```
KNeighborsClassifier()
```

In [17]:

```
ypred=model.predict(X_test)
```

In [18]:

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
acc=accuracy_score(y_test,ypred)
acc
```

Out[18]:

```
0.5538461538461539
```

Conlusion

#Accuracy of LRM is 67% #Accuracy of svm is 67% #Accuracy of KNN is 63%

In [ ]:

```
```