In [1]:

```
```

In [ ]:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
```

Importing Dataset ¶

In [2]:

```
df = pd.read_csv('merged_file1.csv')
df
```

Out[2]:

	FACIES	GR	NPHI	RHOB
0	3.0	31.5743	0.5045	1.7643
1	3.0	39.3396	0.4365	2.0439
2	0.0	46.5190	0.4037	2.2661
3	0.0	52.1829	0.3938	2.3546
4	0.0	56.4486	0.3974	2.3663
...	...	...	...	...
3958	0.0	74.6066	0.5261	2.4379
3959	0.0	76.7127	0.5439	2.4342
3960	0.0	77.0013	0.5283	2.4508
3961	0.0	72.7778	0.5135	2.4784
3962	0.0	68.5550	0.5175	2.4600

3963 rows × 4 columns

In [3]:

```
x = df.iloc[ : , 1:4]
x
```

Out[3]:

	GR	NPHI	RHOB
0	31.5743	0.5045	1.7643
1	39.3396	0.4365	2.0439
2	46.5190	0.4037	2.2661
3	52.1829	0.3938	2.3546
4	56.4486	0.3974	2.3663
...	...	...	...
3958	74.6066	0.5261	2.4379
3959	76.7127	0.5439	2.4342
3960	77.0013	0.5283	2.4508
3961	72.7778	0.5135	2.4784

In [4]:

```
y = df.iloc[ : , 0:1]
y
```

Out[4]:

	FACIES
0	3.0
1	3.0
2	0.0
3	0.0
4	0.0
...	...
3958	0.0
3959	0.0
3960	0.0
3961	0.0
3962	0.0

3963 rows × 1 columns

Doing scaling using StandardScaler

In [5]:

```
scaler = StandardScaler()
```

In [6]:

```
x = scaler.fit_transform(x)
x
```

Out[6]:

```
array([[-1.40607785,  0.41050619, -2.09003135],
       [-1.09851818, -0.22037945, -0.97254357],
       [-0.8141642 , -0.524689  , -0.08446852],
       ...,
       [ 0.39314604,  0.63131617,  0.65372888],
       [ 0.22586618,  0.49400576,  0.76403883],
       [ 0.05861405,  0.53111669,  0.69049886]])
```

Spliting the dataset into train and test

In [7]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_state=5
```

Making the model using SVM

In [9]:

```
cf = svm.SVC()
```

In [10]:

```
cf.fit(x_train, y_train.values.ravel())
```

Out[10]:

```
SVC()
```

In [11]:

```
y_pred = cf.predict(x_test)
y_pred
```

Out[11]:

```
array([0., 1., 0., ..., 3., 0., 1.])
```

accuracy_score of SVM

In [12]:

```
accuracy_score(y_test, y_pred)
```

Out[12]:

```
0.8048780487804879
```

Making the model using KNN

In [13]:

```
knn = KNeighborsClassifier()
knn.fit(x_train, y_train.values.ravel())
```

Out[13]:

```
KNeighborsClassifier()
```

In [14]:

```
y_pred = knn.predict(x_test)
y_pred
```

Out[14]:

```
array([0., 4., 0., ..., 3., 4., 1.])
```

accuracy_score of KNN

In [15]:

```
accuracy_score(y_test, y_pred)
```

Out[15]:

```
0.8200168208578638
```

Making the model using Logistic Regression

In [16]:

```
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train, y_train.values.ravel())
```

Out[16]:

```
LogisticRegression(random_state=0)
```

In [17]:

```
y_pred = classifier.predict(x_test)
y_pred
```

Out[17]:

```
array([0., 1., 0., ..., 3., 0., 1.])
```

accuracy_score of Logistic Regression

In [18]:

```
accuracy_score(y_test, y_pred)
```

Out[18]:

```
0.7872161480235492
```

Making the model using MLP Classifier

In [10]:

```
MLP_clf = MLPClassifier(random_state=1, max_iter=1000).fit(x_train, y_train.values.ravel
```

In [12]:

```
y_pred = MLP_clf.predict(x_test)
y_pred
```

Out[12]:

```
array([0., 1., 0., ..., 3., 0., 1.])
```

accuracy_score of MLP Classifier

In [14]:

```
accuracy_score(y_test, y_pred)
```

Out[14]:

```
0.8233809924306139
```

In [ ]:

In [ ]:

In [ ]:

```
```