In [1]:
```
# Import Libraries
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,  classification_report
from sklearn.preprocessing import StandardScaler
from warnings import filterwarnings
filterwarnings('ignore')
```

In [3]:
```
data = pd.read_csv('C:/Users/saile/Desktop/IoT/datasets/model evaluation_spine dataset.
```

In [4]:
```
data.head()
```

Out[4]:

	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col1
0	63.027817	22.552586	39.609117	40.475232	98.672917	-0.254400	0.744503	12.5661	14.5386	15.3046
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.564259	0.415186	12.8874	17.5323	16.7848
2	68.832021	22.218482	50.092194	46.613539	105.985135	-3.530317	0.474889	26.8343	17.4861	16.6589
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.211523	0.369345	23.5603	12.7074	11.4244
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.918501	0.543360	35.4940	15.9546	8.8723

In [5]:
```
data.shape
```

Out[5]:
```
(310, 14)
```

In [6]:
```
# Data is clean except the "Unnamed: 13" column
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 310 entries, 0 to 309
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Col1         310 non-null    float64
 1   Col2         310 non-null    float64
```

```
 2   Col3         310 non-null    float64
 3   Col4         310 non-null    float64
 4   Col5         310 non-null    float64
 5   Col6         310 non-null    float64
 6   Col7         310 non-null    float64
 7   Col8         310 non-null    float64
 8   Col9         310 non-null    float64
 9   Col10        310 non-null    float64
 10  Col11        310 non-null    float64
 11  Col12        310 non-null    float64
 12  Class_att    310 non-null    object
 13  Unnamed: 13  14 non-null     object
dtypes: float64(12), object(2)
memory usage: 34.0+ KB
```

In [7]:
```
# Type of Backbone Conditions
data.Class_att.unique()
```

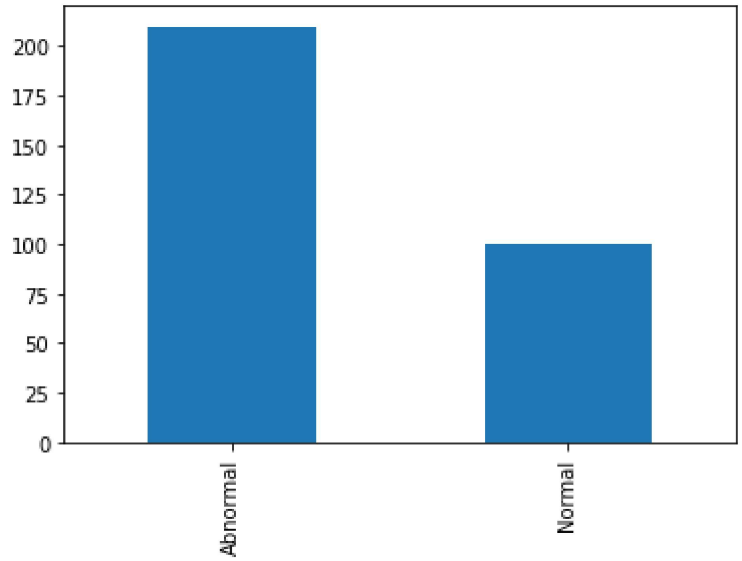
Out[7]:
```
array(['Abnormal', 'Normal'], dtype=object)
```

In [8]:
```
# Remove unwanted column
df = data.drop("Unnamed: 13", axis=1)
```

In [9]:
```
# Change the Column names to be sensable
df.rename(columns = {
    "Col1" : "pelvic_incidence",
    "Col2" : "pelvic_tilt",
    "Col3" : "lumbar_lordosis_angle",
    "Col4" : "sacral_slope",
    "Col5" : "pelvic_radius",
    "Col6" : "degree_spondylolisthesis",
    "Col7" : "pelvic_slope",
    "Col8" : "direct_tilt",
    "Col9" : "thoracic_slope",
    "Col10" :"cervical_tilt",
    "Col11" : "sacrum_angle",
    "Col12" : "scoliosis_slope",
    "Class_att" : "target"}, inplace=True)
```

In [10]:
```
# How skewed is the data?
df["target"].value_counts().sort_index().plot.bar()
```

Out[10]:
```
<AxesSubplot:>
```



In [11]:
```
# Convert categorical to numeric {"Abnormal":0, Normal:1}
df.target = df.target.astype("category").cat.codes
```

In [12]:
```
df.head()
```

Out[12]:

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
0	63.027817	22.552586	39.609117	40.475232	98.672917	-0.254400
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.564259
2	68.832021	22.218482	50.092194	46.613539	105.985135	-3.530317
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.211523
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.918501

In [13]:
```
# 88% Accuracy
dataset = df[["pelvic_incidence","pelvic_tilt","lumbar_lordosis_angle","sacral_slope","
```

In [14]:
```
# Separate input and output
y = dataset.target
X = dataset.drop("target", axis=1)
```

In [15]:
```
# Split data between train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=
```

In [16]:
```
# List models
from sklearn import datasets
from sklearn.cluster import KMeans
models =  [LogisticRegression, DecisionTreeClassifier, RandomForestClassifier,GradientB
```

In [17]:
```
# Train & Predict models
acc_list = []
name_list =[]
for model in models:
    clf = model()
    clf = clf.fit(X_train, y_train)
    predictions = clf.predict(X_test)
    name_list.append((model).__name__)
    acc_list.append(classification_report(y_test,predictions,output_dict=True)["accurac
    print((model).__name__," --> ",classification_report(y_test,predictions,output_dict
```

```
LogisticRegression  -->  0.8640776699029126
DecisionTreeClassifier  -->  0.7378640776699029
RandomForestClassifier  -->  0.8252427184466019
GradientBoostingClassifier  -->  0.7961165048543689
SVC  -->  0.7961165048543689
```

In [18]:
```
# Make a dataframe
team = pd.DataFrame(list(zip(name_list,acc_list)))
team.columns =['Name', "Accuracy"]
team
```

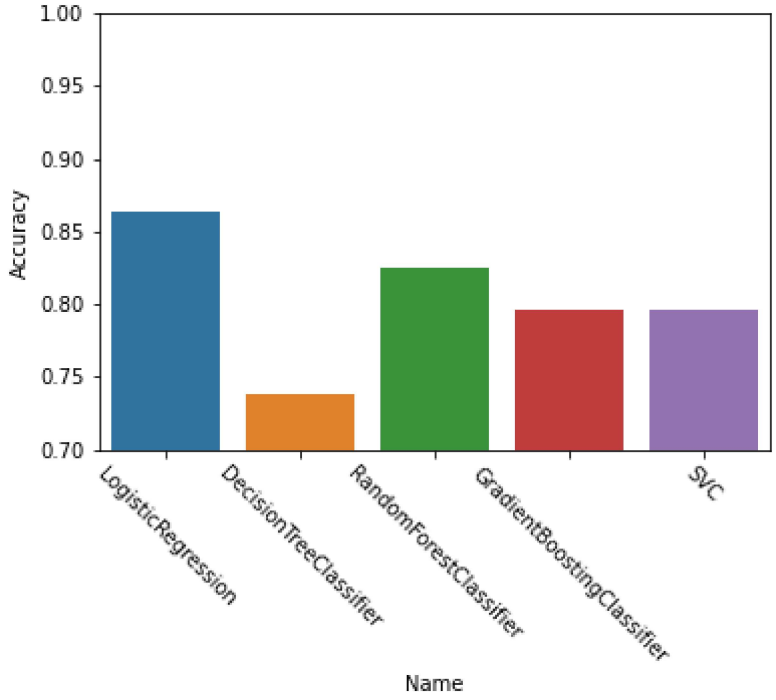
Out[18]:

	Name	Accuracy
0	LogisticRegression	0.864078
1	DecisionTreeClassifier	0.737864
2	RandomForestClassifier	0.825243
3	GradientBoostingClassifier	0.796117
4	SVC	0.796117

In [19]:
```
# Render a Chart
sns.barplot(x=team["Name"], y=team["Accuracy"],data=team)

# Rotate x-labels
plt.xticks(rotation=-45)
plt.ylim(0.7, 1)
```

Out[19]:
```
(0.7, 1.0)
```



In [20]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

In [21]:
```
km = KMeans(n_clusters=2, random_state=42)
#
# Fit the KMeans model
#
km.fit_predict(X_train)
#
# Calculate Silhoutte Score
#
score = silhouette_score(X_train, km.labels_, metric='euclidean')
#
# Print the score
#
print('Silhouetter Score: %.3f' % score)
```

```
Silhouetter Score: 0.478
```

In [22]:
```
pip install yellowbrick
```

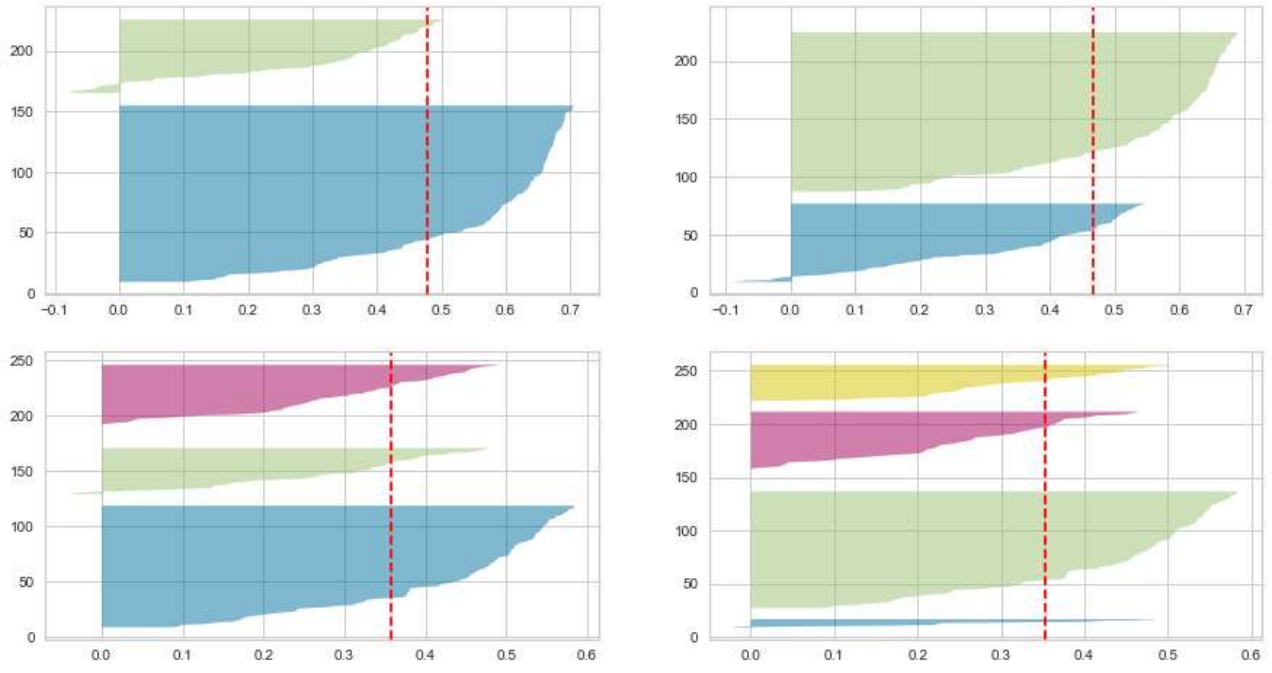
```
Requirement already satisfied: yellowbrick in c:\users\saile\anaconda3\lib\site-packages
(1.5)
Requirement already satisfied: scikit-learn>=1.0.0 in c:\users\saile\anaconda3\lib\site-
packages (from yellowbrick) (1.1.3)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in c:\users\saile\anaconda3\lib
\site-packages (from yellowbrick) (3.4.3)
Requirement already satisfied: scipy>=1.0.0 in c:\users\saile\anaconda3\lib\site-package
s (from yellowbrick) (1.7.1)
Requirement already satisfied: numpy>=1.16.0 in c:\users\saile\anaconda3\lib\site-packag
es (from yellowbrick) (1.20.3)
Requirement already satisfied: cycler>=0.10.0 in c:\users\saile\anaconda3\lib\site-packa
ges (from yellowbrick) (0.10.0)
Requirement already satisfied: six in c:\users\saile\anaconda3\lib\site-packages (from c
```

```
ycler>=0.10.0->yellowbrick) (1.16.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\saile\anaconda3\lib\site-packag
es (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (8.4.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\saile\anaconda3\lib\site
-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\saile\anaconda3\lib\site-pa
ckages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.3.1)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\saile\anaconda3\lib\site-pac
kages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (3.0.4)
Requirement already satisfied: joblib>=1.0.0 in c:\users\saile\anaconda3\lib\site-packag
es (from scikit-learn>=1.0.0->yellowbrick) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\saile\anaconda3\lib\site
-packages (from scikit-learn>=1.0.0->yellowbrick) (2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [23]:

```
from yellowbrick.cluster import SilhouetteVisualizer

fig, ax = plt.subplots(2, 2, figsize=(15,8))
for i in [2,3,4,5]:
    '''
    Create KMeans instance for different number of clusters
    '''
    km = KMeans(n_clusters=i,random_state=42)
    q, mod = divmod(i, 2)
    '''
   # Create SilhouetteVisualizer instance with KMeans instance
    #Fit the visualizer
    '''
    visualizer = SilhouetteVisualizer(km, colors='yellowbrick', ax=ax[q-1][mod])
    visualizer.fit(X_train)
```



In [ ]: