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

In [2]:
```
df= pd.read_csv("Data_for_UCI_named.csv")
df
```

Out[2]:

	tau1	tau2	tau3	tau4	p1	p2	p3	p4	g1	g2	g3	g4	stab	
0	2.959060	3.079885	8.381025	9.780754	3.763085	-0.782604	-1.257395	-1.723086	0.650456	0.859578	0.887445	0.958034	0.055347	u
1	9.304097	4.902524	3.047541	1.369357	5.067812	-1.940058	-1.872742	-1.255012	0.413441	0.862414	0.562139	0.781760	-0.005957	
2	8.971707	8.848428	3.046479	1.214518	3.405158	-1.207456	-1.277210	-0.920492	0.163041	0.766689	0.839444	0.109853	0.003471	u
3	0.716415	7.669600	4.486641	2.340563	3.963791	-1.027473	-1.938944	-0.997374	0.446209	0.976744	0.929381	0.362718	0.028871	u
4	3.134112	7.608772	4.943759	9.857573	3.525811	-1.125531	-1.845975	-0.554305	0.797110	0.455450	0.656947	0.820923	0.049860	u
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	2.930406	9.487627	2.376523	6.187797	3.343416	-0.658054	-1.449106	-1.236256	0.601709	0.779642	0.813512	0.608385	0.023892	u
9996	3.392299	1.274827	2.954947	6.894759	4.349512	-1.663661	-0.952437	-1.733414	0.502079	0.567242	0.285880	0.366120	-0.025803	
9997	2.364034	2.842030	8.776391	1.008906	4.299976	-1.380719	-0.943884	-1.975373	0.487838	0.986505	0.149286	0.145984	-0.031810	
9998	9.631511	3.994398	2.757071	7.821347	2.514755	-0.966330	-0.649915	-0.898510	0.365246	0.587558	0.889118	0.818391	0.037789	u
9999	6.530527	6.781790	4.349695	8.673138	3.492807	-1.390285	-1.532193	-0.570329	0.073056	0.505441	0.378761	0.942631	0.045263	u

10000 rows × 14 columns

In [6]:
```
df= df.drop("stab",axis=1)
```

In [7]:
```
df
```

Out[7]:

	tau1	tau2	tau3	tau4	p1	p2	p3	p4	g1	g2	g3	g4	stabf
0	2.959060	3.079885	8.381025	9.780754	3.763085	-0.782604	-1.257395	-1.723086	0.650456	0.859578	0.887445	0.958034	unstable
1	9.304097	4.902524	3.047541	1.369357	5.067812	-1.940058	-1.872742	-1.255012	0.413441	0.862414	0.562139	0.781760	stable
2	8.971707	8.848428	3.046479	1.214518	3.405158	-1.207456	-1.277210	-0.920492	0.163041	0.766689	0.839444	0.109853	unstable
3	0.716415	7.669600	4.486641	2.340563	3.963791	-1.027473	-1.938944	-0.997374	0.446209	0.976744	0.929381	0.362718	unstable
4	3.134112	7.608772	4.943759	9.857573	3.525811	-1.125531	-1.845975	-0.554305	0.797110	0.455450	0.656947	0.820923	unstable
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	2.930406	9.487627	2.376523	6.187797	3.343416	-0.658054	-1.449106	-1.236256	0.601709	0.779642	0.813512	0.608385	unstable
9996	3.392299	1.274827	2.954947	6.894759	4.349512	-1.663661	-0.952437	-1.733414	0.502079	0.567242	0.285880	0.366120	stable
9997	2.364034	2.842030	8.776391	1.008906	4.299976	-1.380719	-0.943884	-1.975373	0.487838	0.986505	0.149286	0.145984	stable
9998	9.631511	3.994398	2.757071	7.821347	2.514755	-0.966330	-0.649915	-0.898510	0.365246	0.587558	0.889118	0.818391	unstable
9999	6.530527	6.781790	4.349695	8.673138	3.492807	-1.390285	-1.532193	-0.570329	0.073056	0.505441	0.378761	0.942631	unstable

10000 rows × 13 columns

Getting Accuracy with RandomForestClassifier()

In [61]:
```
np.random.seed(42)

#Create data
x= df.drop("stabf",axis=1)
y= df["stabf"]

#Split into training and test data
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test= train_test_split(x,y, test_size=0.2)

from sklearn.ensemble import RandomForestClassifier
clf= RandomForestClassifier()

clf.fit(x_train, y_train)
y_preds= clf.predict(x_test)
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score

print("Classifier metrics on the test set")
print(f" Accuracy:{accuracy_score(y_test,y_preds)*100:.4f}%")
```

```
Classifier metrics on the test set
 Accuracy:91.1500%
```

In [ ]:
```
```

Getting Accuracy With XGBoost Classifier()

Getting Acurracy With XGBoost Classifier()

In [22]:
```
!pip install xgboost
```

```
Collecting xgboost
  Downloading xgboost-1.6.2-py3-none-win_amd64.whl (125.4 MB)
Requirement already satisfied: scipy in c:\users\aboya\anaconda3\lib\site-packages (from xgboost) (1.7.3)
Requirement already satisfied: numpy in c:\users\aboya\anaconda3\lib\site-packages (from xgboost) (1.21.5)
Installing collected packages: xgboost
Successfully installed xgboost-1.6.2
```

In [51]:
```
from xgboost import XGBClassifier



model = XGBClassifier(objective='reg:squarederror')

clf.fit(x_train,y_train)

y_preds= clf.predict(x_test)

print("Classifier metrics on the test set")
print(f" Accuracy:{accuracy_score(y_test,y_preds)*100:.4f}%")
```

```
Classifier metrics on the test set
 Accuracy:91.6000%
```

In [ ]:
```

```

Getting Acurracy With LGBM Classifier()

In [39]:
```
!pip install lightgbm
```

```
Collecting lightgbm
  Downloading lightgbm-3.3.2-py3-none-win_amd64.whl (1.0 MB)
Requirement already satisfied: scikit-learn!=0.22.0 in c:\users\aboya\anaconda3\lib\site-packages (from lightgb
m) (1.1.2)
Requirement already satisfied: scipy in c:\users\aboya\anaconda3\lib\site-packages (from lightgbm) (1.7.3)
Requirement already satisfied: numpy in c:\users\aboya\anaconda3\lib\site-packages (from lightgbm) (1.21.5)
Requirement already satisfied: wheel in c:\users\aboya\anaconda3\lib\site-packages (from lightgbm) (0.37.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\aboya\anaconda3\lib\site-packages (from scikit-
learn!=0.22.0->lightgbm) (2.2.0)
Requirement already satisfied: joblib>=1.0.0 in c:\users\aboya\anaconda3\lib\site-packages (from scikit-learn!=
0.22.0->lightgbm) (1.1.0)
Installing collected packages: lightgbm
Successfully installed lightgbm-3.3.2
```

In [40]:
```
from lightgbm import LGBMClassifier

np.random.seed(42)

model = LGBMClassifier()

model.fit(x_train,y_train)

y_preds= model.predict(x_test)

print("Classifier metrics on the test set")
print(f" Accuracy:{accuracy_score(y_test,y_preds)*100:.2f}%")
```

```
Classifier metrics on the test set
 Accuracy:93.70%
```

In [ ]:
```

```

Using the ExtraTreesClassifier as your estimator

In [58]:
```
from sklearn.tree import ExtraTreeClassifier
from sklearn.ensemble import BaggingClassifier

extra_tree = ExtraTreeClassifier()

cls = BaggingClassifier(extra_tree, random_state=1).fit(x_train, y_train)

# ExtraTreeClassifier does not have the following parameters: cv, n_iter, scoring, n_jobs, verbose
y_preds= cls.predict(x_test)

print("Classifier metrics on the test set")
print(f" Accuracy:{accuracy_score(y_test,y_preds)*100:.4f}%")
```

```
Classifier metrics on the test set
 Accuracy:86.7000%
```

In [55]:
```
extra_tree.get_params()
```

Out[55]:
```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'sqrt',
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'random'}
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: