Instead of checking which model predicts better, we can use all the models and combine them using an Ensemble method known as “Voting Classifier” because the combined model always gives better accuracy than the individual.

In [11]:

```
import pandas as pd
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
cancer_cells = load_breast_cancer()
cancer_feat = pd.DataFrame(cancer_cells['data'],columns=cancer_cells['feature_names'])
cancer_feat.head()
```

Out[11]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mear symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.241
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.181
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.206
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.259
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.180

5 rows × 30 columns

In [12]:

```
from sklearn.model_selection import train_test_split
X=cancer_feat
y=cancer_cells['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
#instantiate SVM
svm=SVC()
#Fit the model to the training dataset
svm.fit(X_train,y_train)
#Predict using the test set
predictions=svm.predict(X_test)
#instantiate Evaluation matrics
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
```

```
[[ 56  10]
 [  3 102]]
              precision    recall  f1-score   support

           0       0.95      0.85      0.90        66
           1       0.91      0.97      0.94       105

    accuracy                           0.92       171
   macro avg       0.93      0.91      0.92       171
weighted avg       0.93      0.92      0.92       171
```

In [13]:

```
#Instantiate Logistic Regression
lr=LogisticRegression()
#Fit the model to the training set and predict using the test set
lr.fit(X_train,y_train)
predictions=lr.predict(X_test)
#Evaluation matrics
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
```

```
[[ 57   9]
 [  4 101]]
              precision    recall  f1-score   support

           0       0.93      0.86      0.90        66
           1       0.92      0.96      0.94       105

    accuracy                           0.92       171
   macro avg       0.93      0.91      0.92       171
weighted avg       0.92      0.92      0.92       171


C:\Users\ALISHA ANJUM\anaconda3\lib\site-packages\sklearn\linear_model\_logi
stic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

In [14]:

```
#Instantiate Decision tree model
dt=DecisionTreeClassifier()
#Fit and predict the model
dt.fit(X_train,y_train)
predictions=dt.predict(X_test)
#Evaluation matrics
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.92      0.88      0.90        66
           1       0.93      0.95      0.94       105

    accuracy                           0.92       171
   macro avg       0.92      0.92      0.92       171
weighted avg       0.92      0.92      0.92       171
```

In [15]:

```
#import Voting Classifier
from sklearn.ensemble import VotingClassifier
#instantiating three classifiers
logReg= LogisticRegression()
dTree= DecisionTreeClassifier()
svm= SVC()
voting_clf = VotingClassifier(estimators=[('SVC', svm), ('DecisionTree',dTree), ('LogReg',
#fit and predict using training and testing dataset respectively
voting_clf.fit(X_train, y_train)
predictions = voting_clf.predict(X_test)
#Evaluation matrics
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
```

```
[[ 58   8]
 [  3 102]]
              precision    recall  f1-score   support

           0       0.95      0.88      0.91        66
           1       0.93      0.97      0.95       105

    accuracy                           0.94       171
   macro avg       0.94      0.93      0.93       171
weighted avg       0.94      0.94      0.94       171


C:\Users\ALISHA ANJUM\anaconda3\lib\site-packages\sklearn\linear_model\_logi
stic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

In [ ]: