# Scikit Learn Notes

1. **Create synthetic data:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | from sklearn.datasets import make\_classification  X, y = make\_classification(n\_samples=1000, n\_features=20, n\_informative=15, n\_redundant=5, random\_state=6)  print(X.shape, y.shape)  Output will be (1000,2) (1000,) |

v

# test regression dataset

from sklearn.datasets import make\_regression

X, y = make\_regression(n\_samples=1000, n\_features=20, n\_informative=15, noise=0.1, random\_state=6)

1. **Repeated Startified K Fold:**

We will evaluate the model using [repeated stratified k-fold cross-validation](https://machinelearningmastery.com/k-fold-cross-validation/), with three repeats and 10 folds. We will report the mean and standard deviation of the accuracy of the model across all repeats and folds.

model = AdaBoostClassifier()

cv = RepeatedStratifiedKFold(n\_splits=10, n\_repeats=3, random\_state=1)

n\_scores = cross\_val\_score(model, X, y, scoring='accuracy', cv=cv, n\_jobs=-1, error\_score='raise')

# report performance

print('Accuracy: %.3f (%.3f)' % (mean(n\_scores), std(n\_scores)))

1. **Cross Validation with GridSearchCV :**
2. model = AdaBoostClassifier()
3. # define the grid of values to search
4. grid = dict()
5. grid['n\_estimators'] = [10, 50, 100, 500]
6. grid['learning\_rate'] = [0.0001, 0.001, 0.01, 0.1, 1.0]
7. # define the evaluation procedure
8. cv = RepeatedStratifiedKFold(n\_splits=10, n\_repeats=3, random\_state=1)
9. # define the grid search procedure
10. grid\_search = GridSearchCV(estimator=model, param\_grid=grid, n\_jobs=-1, cv=cv, scoring='accuracy')
11. # execute the grid search
12. grid\_result = grid\_search.fit(X, y)
13. # summarize the best score and configuration
14. print("Best: %f using %s" % (grid\_result.best\_score\_, grid\_result.best\_params\_))
15. # summarize all scores that were evaluated
16. means = grid\_result.cv\_results\_['mean\_test\_score']
17. stds = grid\_result.cv\_results\_['std\_test\_score']
18. params = grid\_result.cv\_results\_['params']
19. for mean, stdev, param in zip(means, stds, params):
20. print("%f (%f) with: %r" % (mean, stdev, param))

**4.) To plot the decision boundaries:**

