```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.datasets import load_breast_cancer
        from sklearn.model_selection import RandomizedSearchCV,train_test_split,cross_val_sc
        from sklearn.metrics import mean_squared_error,accuracy_score,make_scorer
        from sklearn.ensemble import RandomForestClassifier

        X = load_breast_cancer().data
        y = load_breast_cancer().target

        X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state =
        acc2=[]
        for i in np.arange(2,10,2):
            acc=[]
            for j in np.arange(50,500,50):
                model = RandomForestClassifier(n_estimators=j,max_features=i)
                model.fit(X_train,y_train)
                y_pred = model.predict(X_test)
                score = make_scorer(accuracy_score)
                cv = cross_val_score(estimator=model,X=X,y=y,cv=5,scoring=score)
                m_acc = np.mean(cv)
                acc.append(m_acc)
            acc1 = np.max(acc)
            nt = np.argmax(acc)
            ntree = np.arange(50,500,50)[nt]
            acc2.append(acc1)
            accuracy = np.max(acc2)
            mt = np.argmax(acc2)
            mtry = np.arange(2,10,2)[mt]
        ac=[]
        for j in np.arange(50,500,50):
            model1 = RandomForestClassifier(n_estimators=j)
            model1.fit(X_train,y_train)
            y_pred1 = model1.predict(X_test)
            score1 = make_scorer(accuracy_score)
            cv1 = cross_val_score(estimator=model1,X=X,y=y,cv=5,scoring=score1)
            m_acc1 = np.mean(cv1)
            ac.append(m_acc1)
        ac1 = np.max(ac)
        nt1 = np.argmax(ac)
        ntree1 = np.arange(50,500,50)[nt1]

        ac2=[]
        for i in np.arange(2,10,2):
            model2 = RandomForestClassifier(max_features=i)
            model2.fit(X_train,y_train)
            y_pred2 = model2.predict(X_test)
            score2 = make_scorer(accuracy_score)
            cv2 = cross_val_score(estimator=model2,X=X,y=y,cv=5,scoring=score2)
            m_acc2 = np.mean(cv2)
            ac2.append(m_acc2)
        ac3 = np.max(ac2)
        mt1 = np.argmax(ac2)
        mtry1 = np.arange(2,10,2)[mt1]
        print("For default Mtry and Varying Ntree Best Accuracy is %f with Best Ntree is %d"
        print("For default Ntree and Varying Mtry Best Accuracy is %f with Best Mtry is %d"%
        print("Best Mtry is %d, Best Ntree is %d with Best Accuracy %f"%(mtry,ntree,accuracy
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-1-a2884249642d> in <module>
     18         y_pred = model.predict(X_test)
     19         score = make_scorer(accuracy_score)
---> 20         cv = cross_val_score(estimator=model,X=X,y=y,cv=5,scoring=score)
```

```
     21         m_acc = np.mean(cv)
     22         acc.append(m_acc)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*a
rgs, **kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74 

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py in
cross_val_score(estimator, X, y, groups, scoring, cv, n_jobs, verbose, fit_params, p
re_dispatch, error_score)
    399     scorer = check_scoring(estimator, scoring=scoring)
    400 
--> 401     cv_results = cross_validate(estimator=estimator, X=X, y=y, groups=group
s,
    402                                 scoring={'score': scorer}, cv=cv,
    403                                 n_jobs=n_jobs, verbose=verbose,

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*a
rgs, **kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74 

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py in
cross_validate(estimator, X, y, groups, scoring, cv, n_jobs, verbose, fit_params, pr
e_dispatch, return_train_score, return_estimator, error_score)
    240     parallel = Parallel(n_jobs=n_jobs, verbose=verbose,
    241                         pre_dispatch=pre_dispatch)
--> 242     scores = parallel(
    243         delayed(_fit_and_score)(
    244             clone(estimator), X, y, scorers, train, test, verbose, None,

C:\ProgramData\Anaconda3\lib\site-packages\joblib\parallel.py in __call__(self, iter
able)
   1049                 self._iterating = self._original_iterator is not None
   1050 
-> 1051             while self.dispatch_one_batch(iterator):
   1052                 pass
   1053 

C:\ProgramData\Anaconda3\lib\site-packages\joblib\parallel.py in dispatch_one_batch
(self, iterator)
    864                 return False
    865             else:
--> 866                 self._dispatch(tasks)
    867                 return True
    868 

C:\ProgramData\Anaconda3\lib\site-packages\joblib\parallel.py in _dispatch(self, bat
ch)
    782         with self._lock:
    783             job_idx = len(self._jobs)
--> 784             job = self._backend.apply_async(batch, callback=cb)
    785             # A job can complete so quickly than its callback is
    786             # called before we get here, causing self._jobs to

C:\ProgramData\Anaconda3\lib\site-packages\joblib\_parallel_backends.py in apply_asy
nc(self, func, callback)
    206     def apply_async(self, func, callback=None):
    207         """Schedule a func to be run"""
--> 208         result = ImmediateResult(func)
    209         if callback:
    210             callback(result)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\joblib\_parallel_backends.py in __init__
(self, batch)
    570         # Don't delay the application, to avoid keeping the input
    571         # arguments in memory
--> 572         self.results = batch()
    573 
    574     def get(self):

C:\ProgramData\Anaconda3\lib\site-packages\joblib\parallel.py in __call__(self)
    260         # change the default number of processes to -1
    261         with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 262             return [func(*args, **kwargs)
    263                     for func, args, kwargs in self.items]
    264 

C:\ProgramData\Anaconda3\lib\site-packages\joblib\parallel.py in <listcomp>(.0)
    260         # change the default number of processes to -1
    261         with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 262             return [func(*args, **kwargs)
    263                     for func, args, kwargs in self.items]
    264 

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py in
_fit_and_score(estimator, X, y, scorer, train, test, verbose, parameters, fit_param
s, return_train_score, return_parameters, return_n_test_samples, return_times, retur
n_estimator, error_score)
    529             estimator.fit(X_train, **fit_params)
    530         else:
--> 531             estimator.fit(X_train, y_train, **fit_params)
    532 
    533     except Exception as e:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in fit(self,
 X, y, sample_weight)
    374                 random_state.randint(MAX_INT, size=len(self.estimators_))
    375 
--> 376             trees = [self._make_estimator(append=False,
    377                                           random_state=random_state)
    378                      for i in range(n_more_estimators)]

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in <listcomp>
(.0)
    374                 random_state.randint(MAX_INT, size=len(self.estimators_))
    375 
--> 376             trees = [self._make_estimator(append=False,
    377                                           random_state=random_state)
    378                      for i in range(n_more_estimators)]

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_base.py in _make_estima
tor(self, append, random_state)
    154 
    155         if random_state is not None:
--> 156             _set_random_states(estimator, random_state)
    157 
    158         if append:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_base.py in _set_random_
states(estimator, random_state)
     74     for key in sorted(estimator.get_params(deep=True)):
     75         if key == 'random_state' or key.endswith('__random_state'):
---> 76             to_set[key] = random_state.randint(np.iinfo(np.int32).max)
     77 
     78     if to_set:

C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\getlimits.py in __init__(self,
int_type)
    501         except TypeError:
    502             self.dtype = numeric.dtype(type(int_type))
```

```
--> 503         self.kind = self.dtype.kind
    504         self.bits = self.dtype.itemsize * 8
    505         self.key = "%s%d" % (self.kind, self.bits)

KeyboardInterrupt:
```

```
In [ ]:
```