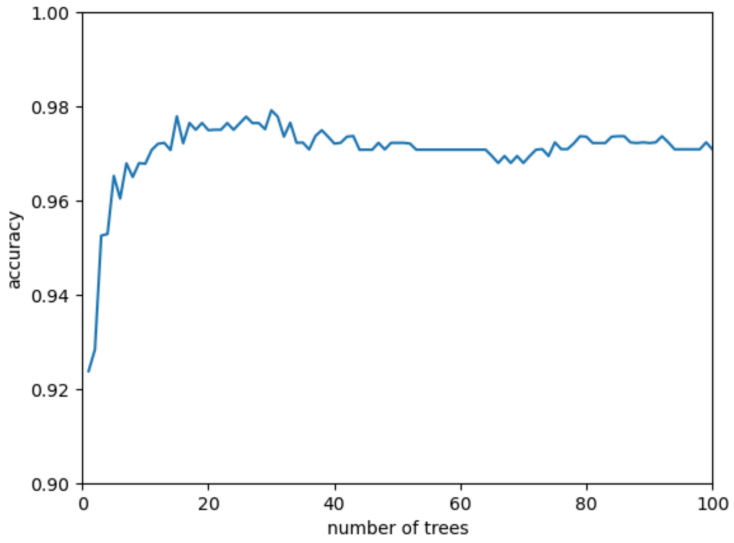
```python
# This Python program looks at tuning a Random Forest model for
machine learning.
# It uses Grid Search to compare the performance
# of the random forest with different numbers of trees. It
displays the number of trees
# for the best performance, and uses an Elbow Graph to find the
sweet spot between
# performance and complexity.

#import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# load data set
cancer_data = load_breast_cancer()
df = pd.DataFrame(cancer_data['data'],
columns=cancer_data['feature_names'])
df['target'] = cancer_data['target']

# create features and target data set for training
X = df[cancer_data.feature_names].values
y = df['target'].values

# select 100 trees to compare
n_estimators = list(range(1, 101))
param_grid = {
    'n_estimators': n_estimators,
}

rf = RandomForestClassifier(random_state=123)
gs = GridSearchCV(rf, param_grid, scoring='f1', cv=5)
gs.fit(X, y)

# display the best number of trees
print("best number of trees:", gs.best_params_)

# plot a Elbow Graph
scores = gs.cv_results_['mean_test_score']
plt.plot(n_estimators, scores)
plt.xlabel("number of trees")
plt.ylabel("accuracy")
plt.xlim(0, 100)
```

```
plt.ylim(0.9, 1)
plt.show()
```

```
best number of trees: {'n_estimators': 30}
```



Based on the graph, the accuracy levels out around 10 tress. This number gives best performance vs complexity (number of trees).