In [17]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import tree, ensemble
from sklearn.model_selection import cross_val_score
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import KFold
from collections import defaultdict
import pprint
```

In [2]:
```
# Assignment Constants
RANDOM_STATE = 10
FIGSIZE = (12,8)
#### Use the following line before plt.plot(....) to increase the p
lot size ####
# plt.figure(figsize=FIGSIZE)
```

Question 1 ¶

Use the breast cancer data set from Homework 0 to create a training set. Recall that the label is 0 if the patient’s data indicates a malignant cancer and 1 otherwise. Compute the base rate of malignant cancer occurrence over the entire data set. In other words, what would be your best guess for the probability of malignant cancer of a single example using only the labels in the training set? This question is very simple, so try not to overthink it.

In [22]:
```
cancer = load_breast_cancer()
prob = np.sum(1 - cancer['target']) / len(cancer['target'])

print(f"Probability of maligant cancer is {prob:.4f}")
```

```
Probability of maligant cancer is 0.3726
```

2

The goal is to build a decision tree that, based on the other features in the set, predicts whether or not a patient has malignant cancer. So this is a classification problem. Using `tree.DecisionTreeClassifier` and other functions in the scikit-learn library, one can build a decision tree and calculate both its training accuracy when fitted to the entire data set as well as its accuracy using 10-fold cross validation (which gives a better idea of true accuracy). In this question you will need to complete two sub-components:

(a)

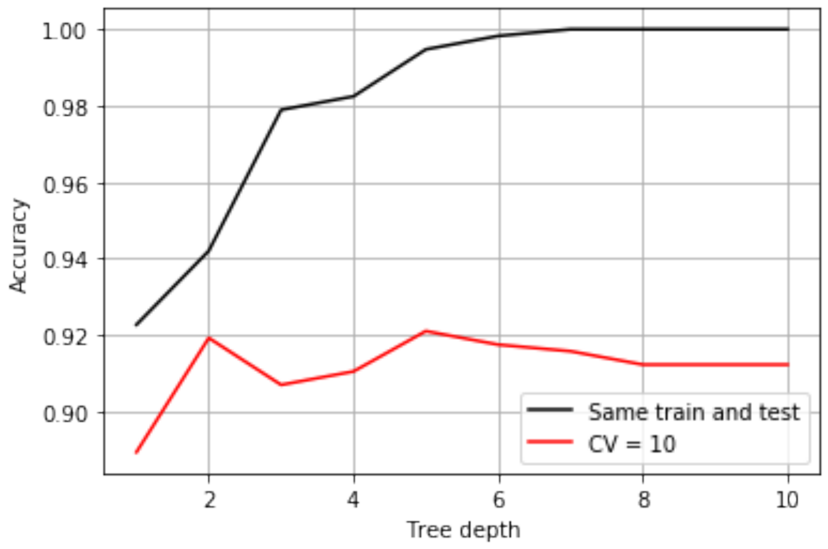
(a) Make a plot visualizing the performance of a `tree.DecisionTreeClassifier` as you search for an optimal `max_depth` parameter. Vary the depth of your decision tree using max depth = 1,2,. . . ,10 and record the results from the following evaluation procedures for each setting:

- The accuracy when training and testing on the full dataset.
- 10-fold cross-validated accuracy.

Plot the results of both evaluation procedures on the same plot with evaluation scores on the y-axis and max depth values on the x-axis. Use 10 as your random seed/state for the decision tree and the cross-validation. Use a legend to label both evaluation procedures.

In [37]:
```
depths = list(range(1, 11))
dt_scores, dt_cv_scores = [], []
for depth in depths:
    decision_tree = tree.DecisionTreeClassifier(max_depth=depth, ra
ndom_state=RANDOM_STATE)
    decision_tree.fit(cancer.data, cancer.target)
    dt_scores.append(decision_tree.score(cancer.data, cancer.target
))
    dt_cv_scores.append(cross_val_score(decision_tree, cancer.data,
cancer.target, cv=10).mean())

plt.plot(depths, dt_scores, c='k', label="Same train and test");
plt.plot(depths, dt_cv_scores, c="r", label="CV = 10")
plt.legend();
plt.xlabel("Tree depth")
plt.ylabel("Accuracy");
plt.grid(b=True)
plt.show();
```



(b)

Answer the questions below based on the results of 2a. Write your answers in the corresponding field in the markdown cell that is present in the HW1 template notebook. Do this by double clicking the markdown cell and writing your answer directly in the cell. Pressing enter will re-render the markdown.

(i.)

What setting of `max_depth` gave the best accuracy w.r.t. the **full-dataset** accuracy? If more than one setting equaled the best accuracy, list each of the best settings.

Student answer here: [7, 8, 9, 10]

(ii.)

What setting of `max_depth` gave the best accuracy w.r.t. the **cross- validated** accuracy? If more than one setting equaled the best accuracy, list each of the best settings.

Student answer here: 5

3.

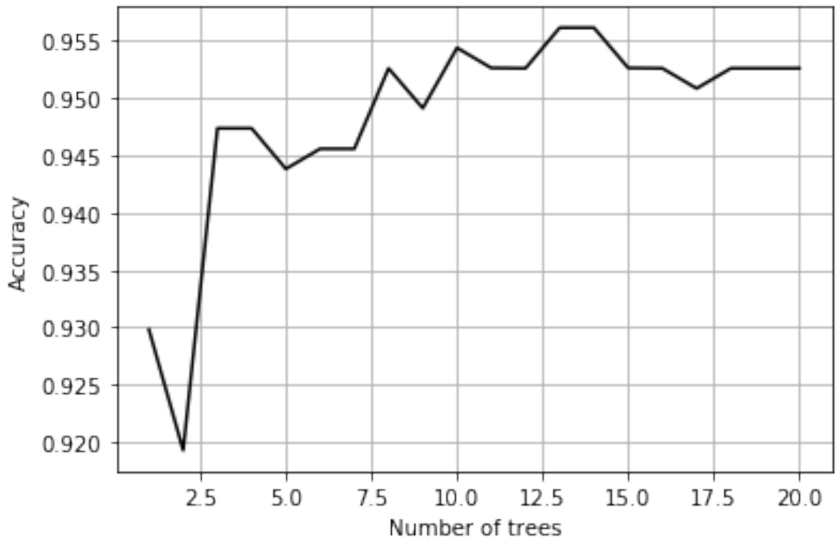
This question explores random forest classifiers by using scikit-learn's `ensemble.RandomForestClassifier` . You will make two plots and answer questions about them.

(a)

For the first plot, use a `ensemble.RandomForestClassifier` and the best depth you found 2(b)ii as `max_depth` . We will now find the optimal setting of a second parameter, n estimators. Vary the number of trees in the forest via the parameter `n_estimators` and plot its 10-fold cross-validated accuracy (use `n_estimators` = 1, 2, . . . , 20). Again, use 10 as your random seed for your classifier and cross-validation.

In [40]:
```
max_depth = 5
n_estimators_list = list(range(1, 21))
scores = []
for n_estimators in n_estimators_list:
    forest = ensemble.RandomForestClassifier(n_estimators=n_estimat
ors, random_state=RANDOM_STATE, max_depth=max_depth)
    scores.append(cross_val_score(forest, cancer.data, cancer.targe
t, cv=10).mean())

plt.plot(n_estimators_list, scores, c="k", label="Random forest");
plt.grid(b=True)
plt.xlabel("Number of trees");
plt.ylabel("Accuracy");
plt.show();
```



(b)

Do you see an improvement using random forests versus using a single tree? (Note: use the `n_estimators` =1 result as the result for a single tree.)

Student answer here: Yes, there is about 2% accuracy improvement in going from 1 tree to 20 trees

(c)

What setting of `n_estimators` gave the best accuracy w.r.t. the cross-validated ac- curacy?

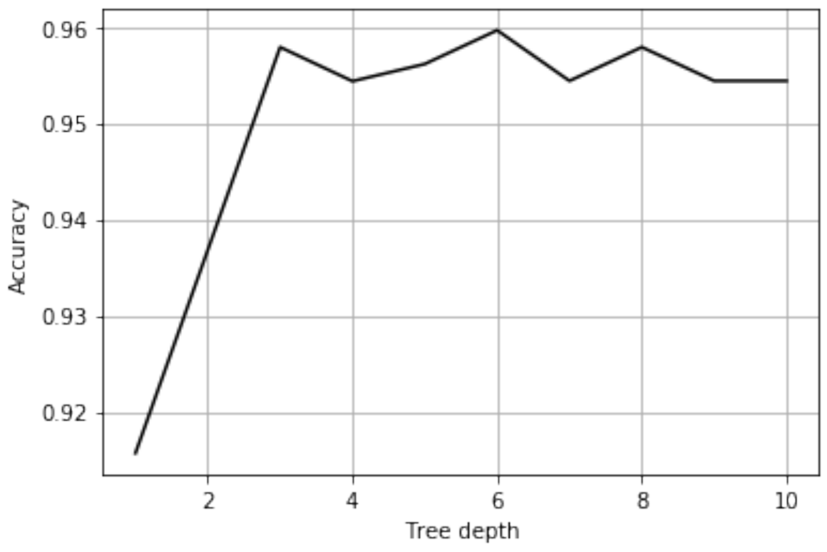
Student answer here: 13

(d)

For the second plot, again use a `ensemble.RandomForestClassifier` , but this time you will fix the `n_estimators` parameter and again attempt to find the optimal setting of a `max_depth` . Use your answer to 3c as the setting for `n_estimators` and follow the procedure from 2a to find the best setting for max depth. This time, only plot the results from cross-validation and not the full set, but the plot should be the same structure as in 2a otherwise (use `max_depth` = 1,2,. . . ,10). Again, use 10 as your random seed.

In [41]:
```
n_estimators = 13
depths = list(range(1, 11))
scores = []
for depth in depths:
    forest = ensemble.RandomForestClassifier(n_estimators=n_estimat
ors, max_depth=depth, random_state=RANDOM_STATE)
    scores.append(cross_val_score(forest, cancer.data, cancer.targe
t, cv=10).mean())

plt.plot(depths, scores, c="k", label=f"Random forest, n_trees = {n
_estimators}");
plt.grid(b=True)
plt.xlabel("Tree depth");
plt.ylabel("Accuracy");
plt.show();
```



(e)

In the plot in 3d, is the optimal setting of `max_depth` the same as in 2(b)ii? If not, what is the new optimal setting of `max_depth` ?

Student answer here: New `max_depth` is 6. Close but not the same

4.

For this last question, we will explore the dependability of our estimates.

(a)

Make a plot using the following procedure:

i.

Using random state values from 0, 1, $\cdots$, 99 calculate the 10-fold cross-validation accuracy of different `tree.DecisionTreeClassifiers` with max depth settings from 1, 2, $\cdots$, 10. As before, you should use the same random state value for your classifier and cross-validation.

ii.

Then record the best max depth settings for each random state. Be sure to check whether multiple settings achieve the best accuracy.

Plot the counts for the best max depth settings as a bar chart with the max depth settings on the x-axis and the 'best parameter counts' on the y-axis (number of times that parameter was selected as the best max depth setting).

Note: this calculation might take some time. For debugging, try a smaller range of random states.

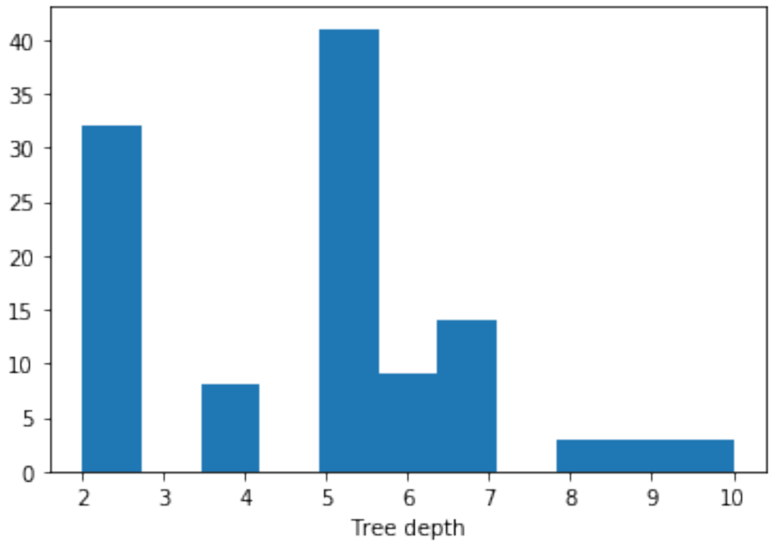
In [42]:
```
from tqdm import tqdm
```

In [51]:
```
random_states = np.array(range(1, 100))
depths = np.array(range(1, 11))
state2best_depth = []
for random_state in tqdm(random_states):
    scores = []
    for depth in depths:
        decision_tree = tree.DecisionTreeClassifier(max_depth=depth
, random_state=random_state)
        scores.append(cross_val_score(decision_tree, cancer.data, c
ancer.target, cv=10).mean())

    best_score = np.max(scores)
    best_score_indx = np.where(scores == best_score)[0]
    for d in depths[best_score_indx]:
        state2best_depth.append((random_state, d))
```

```
100%|██████████| 99/99 [00:52<00:00,  1.88it/s]
```

In [57]:
```
plt.hist([el[-1] for el in state2best_depth], bins=11);
plt.xlabel("Tree depth")
plt.show()
```



(b)

What are the top two most frequent parameter settings?

Student answer here: 5 and 2