Lesson_6B_Ensemble_learning_and_random_forests

August 30, 2020

Chapter 7 – Ensemble Learning and Random Forests

Often each model that you estimate grasps some feature of the data, while missing the other. The aggregation of the predictions may result in a better prediction than even the best model by itself. A group of predictors is called an ensemble; thus, this technique is called Ensemble Learning, and an Ensemble Learning algorithm is called an Ensemble method. If you estimate a group of decision trees to make predictions, you just obtain the predictions of all individual trees, then predict the class that gets the most votes. Such an ensemble of Decision Trees is called a Random Forest, and despite its simplicity, this is one of the most powerful Machine Learning algorithms available today. Most of the winners of ML competitions use some kinds of Ensemble learning often involving hundreds of very different models.

If you start with Logistic, SVM, Random Forest and K-Nearest Neighbors classifier. The simples way to aggregate classifiers by the majority voting: this is called *hard voting.* Even an average of weak diverse classifiers can produce better prediction that a single "optimal" classifier.

This notebook contains all the sample code and solutions to the exercises in chapter 7.

1 Setup

First, let's make sure this notebook works well in both python 2 and 3, import a few common modules, ensure MatplotLib plots figures inline and prepare a function to save the figures:

[58]:
```
# Python 3.5 is required
import sys
assert sys.version_info >= (3, 5)

# Scikit-Learn 0.20 is required
import sklearn
assert sklearn.__version__ >= "0.20"

# Common imports
import numpy as np
import os

# to make this notebook's output stable across runs
np.random.seed(42)

# To plot pretty figures
```

```
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

# Where to save the figures
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "ensembles"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID)
os.makedirs(IMAGES_PATH, exist_ok=True)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

2 Voting classifiers

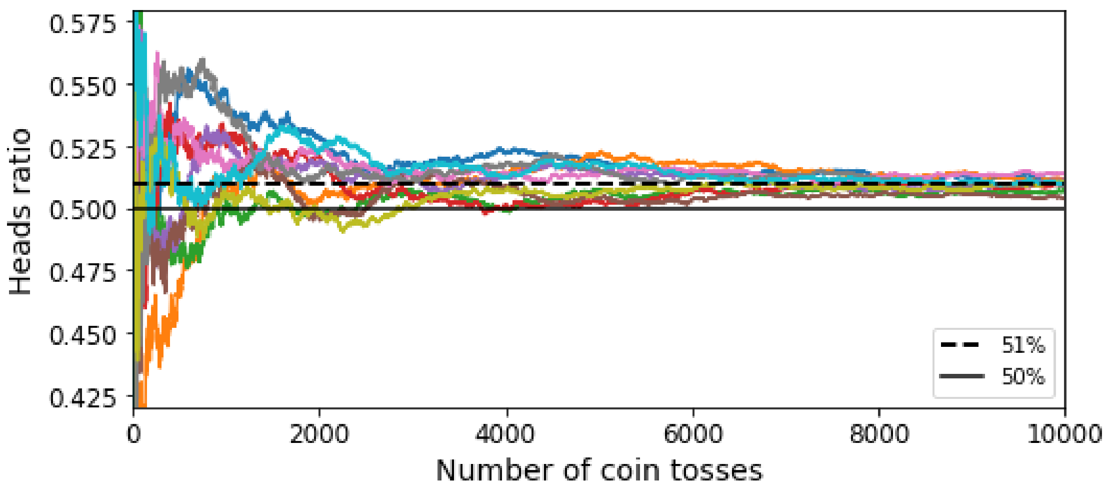
Suppose you are have a biased coin with the probability of heads of 51%. On average you obtain the number of heads equal to $N * p_h$, where N is the number of tosses and p_h is the probability of heads. By the law of large numbers the more tosses you make the more likely you observe that there is a different number of heads and tails produced by a biased coin. Binomial distribution tells us the probability of k successes in a n trials with PDF: $\binom{n}{x}p^k(1-p)^{n-k}$. Using CDF we can calculate probability of observing less than 50% of successes.

$$P(X \geq 0.5) = 1 - (X < 0.5) = 1 - \sum_{i=0}^{k} \binom{n}{x} p^k (1-p)^{n-k}$$

where $k = \frac{n}{2} - 1$. This probability of more heads and tails is 0.51 for 1 toss, 0.75 for 1,000 tosses, and 0.97 for 10,000 tosses.

[2]:
```
heads_proba = 0.51
coin_tosses = (np.random.rand(10000, 10) < heads_proba).astype(np.int32)
cumulative_heads_ratio = np.cumsum(coin_tosses, axis=0) / np.arange(1, 10001).
↪reshape(-1, 1)
```

[59]:
```
plt.figure(figsize=(8,3.5))
plt.plot(cumulative_heads_ratio)
plt.plot([0, 10000], [0.51, 0.51], "k--", linewidth=2, label="51%")
plt.plot([0, 10000], [0.5, 0.5], "k-", label="50%")
plt.xlabel("Number of coin tosses")
plt.ylabel("Heads ratio")
plt.legend(loc="lower right")
plt.axis([0, 10000, 0.42, 0.58])
plt.show()
```



Using the same logic if you build 1,000 classifiers that are 51% correct and you use the majority voting class you will get 75% accuracy. However, this is only true if all classifiers are perfectly independent, making uncorrelated errors, which is clearly not the case since they are trained on the same data. They are likely to make the same types of errors, so there will be many majority votes for the wrong class, reducing the ensemble's accuracy. Diversity of classifiers is the key to success of ensemble learning.

[4]:
```
# Let's start with a data of 500 points from the moon function
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_moons
X, y = make_moons(n_samples=500, noise=0.30, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

We start with simple voting classifier that trains Logistic, Random Forest and SVM classifiers.

[5]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

log_clf = LogisticRegression(random_state=42)
rnd_clf = RandomForestClassifier(random_state=42)
svm_clf = SVC(random_state=42)
# voting classifier has syntaxis akin to pipeline
voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='hard')
```

```
# hard voting: each classifier has one vote.
```

[6]:
```
# Fit all three classifiers in ensemble learning
voting_clf.fit(X_train, y_train)
```

[6]:
```
VotingClassifier(estimators=[('lr',
                              LogisticRegression(C=1.0, class_weight=None,
                                                 dual=False, fit_intercept=True,
                                                 intercept_scaling=1,
                                                 l1_ratio=None, max_iter=100,
                                                 multi_class='auto',
                                                 n_jobs=None, penalty='l2',
                                                 random_state=42,
                                                 solver='lbfgs', tol=0.0001,
                                                 verbose=0, warm_start=False)),
                             ('rf',
                              RandomForestClassifier(bootstrap=True,
                                                     ccp_alpha=0.0,
                                                     class_weight=None,
                                                     crit…
                                                     oob_score=False,
                                                     random_state=42, verbose=0,
                                                     warm_start=False)),
                             ('svc',
                              SVC(C=1.0, break_ties=False, cache_size=200,
                                  class_weight=None, coef0=0.0,
                                  decision_function_shape='ovr', degree=3,
                                  gamma='scale', kernel='rbf', max_iter=-1,
                                  probability=False, random_state=42,
                                  shrinking=True, tol=0.001, verbose=False))],
                 flatten_transform=True, n_jobs=None, voting='hard',
                 weights=None)
```

Let's compare the accuracy of three individual classifiers and the ensemble learning. The latter has the highest accuracy.

[7]:
```
from sklearn.metrics import accuracy_score

for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(clf.__class__.__name__, accuracy_score(y_test, y_pred))
```

```
LogisticRegression 0.864
RandomForestClassifier 0.896
SVC 0.896
VotingClassifier 0.912
```

In the example above we use hard-voting: if the majority of classifiers predict a particular class, this class is predicted by the ensemble. A more powerful technique is the soft-voting, where instead of class, each classifier predicts are probability. This gives more weight to the classifiers that are more confident in their predictions. All you need to do is replace voting="hard" with voting="soft" and ensure that all classifiers can estimate class probabilities. This is not the case of the SVC class by default, so you need to set its probability hyperparameter to True (this will make the SVC class use cross-validation to estimate class probabilities, slowing down training, and it will add a predict_proba() method).

[8]:
```
log_clf = LogisticRegression(random_state=42)
rnd_clf = RandomForestClassifier(random_state=42)
# set probability to True
svm_clf = SVC(probability=True, random_state=42)

voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    # soft voting
    voting='soft')
voting_clf.fit(X_train, y_train)
```

[8]:
```
VotingClassifier(estimators=[('lr',
                              LogisticRegression(C=1.0, class_weight=None,
                                                 dual=False, fit_intercept=True,
                                                 intercept_scaling=1,
                                                 l1_ratio=None, max_iter=100,
                                                 multi_class='auto',
                                                 n_jobs=None, penalty='l2',
                                                 random_state=42,
                                                 solver='lbfgs', tol=0.0001,
                                                 verbose=0, warm_start=False)),
                             ('rf',
                              RandomForestClassifier(bootstrap=True,
                                                     ccp_alpha=0.0,
                                                     class_weight=None,
                                                     crit…
                                                     oob_score=False,
                                                     random_state=42, verbose=0,
                                                     warm_start=False)),
                             ('svc',
                              SVC(C=1.0, break_ties=False, cache_size=200,
                                  class_weight=None, coef0=0.0,
                                  decision_function_shape='ovr', degree=3,
                                  gamma='scale', kernel='rbf', max_iter=-1,
                                  probability=True, random_state=42,
                                  shrinking=True, tol=0.001, verbose=False))],
                 flatten_transform=True, n_jobs=None, voting='soft',
                 weights=None)
```

[9]:
```
from sklearn.metrics import accuracy_score

for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(clf.__class__.__name__, accuracy_score(y_test, y_pred))
```

```
LogisticRegression 0.864
RandomForestClassifier 0.896
SVC 0.896
VotingClassifier 0.92
```

Using hard voting ensemble learning accuracy was 0.896, with soft voting it is 0.912.

3 Bagging ensembles

Another approach to ensemble learning is to training the classifiers on different data. We will sample some of the training data for each classifier. If the sampling is done with replacement it is called *bagging* if without replacement it is called *pasting*.

Once we trained all classifiers we aggregate predictions using statistical mode (most frequent prediction like hard voting. Each individual predictor has a higher bias than if it were trained on the original training set, but aggregation reduces both bias and variance. Generally this results in a similar bias, but lower variance that a single predictor trained on a original training dataset. We can train out data on parallel CPU cores or servers. Bagging and pasting scale very well.

In SkLearn we will use BaggingRegressor. We will train 500 decision Tree classifiers with 100 instances in each. The instances will be randomly sampled with replacement. If you want *pasting* set (bootstrap=False).The n_jobs parameter tells Scikit-Learn the number of CPU cores to use for training and predictions (–1 tells Scikit-Learn to use all available cores).

The BaggingClassifier automatically performs soft voting instead of hard voting if the base classifier can estimate class probabilities (i.e., if it has a predict_proba() method), which is the case with Decision Trees classifiers.

[10]:
```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

bag_clf = BaggingClassifier(
    DecisionTreeClassifier(random_state=42), n_estimators=500,
    max_samples=100, bootstrap=True, n_jobs=-1, random_state=42)
bag_clf.fit(X_train, y_train)
y_pred = bag_clf.predict(X_test)
```

Accuracy of bagging classifier is 90.4% and the accuracy of an individual tree is 85.6%

[11]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
```

```
0.904
```

```
[12]: tree_clf = DecisionTreeClassifier(random_state=42)
      tree_clf.fit(X_train, y_train)
      y_pred_tree = tree_clf.predict(X_test)
      print(accuracy_score(y_test, y_pred_tree))
```

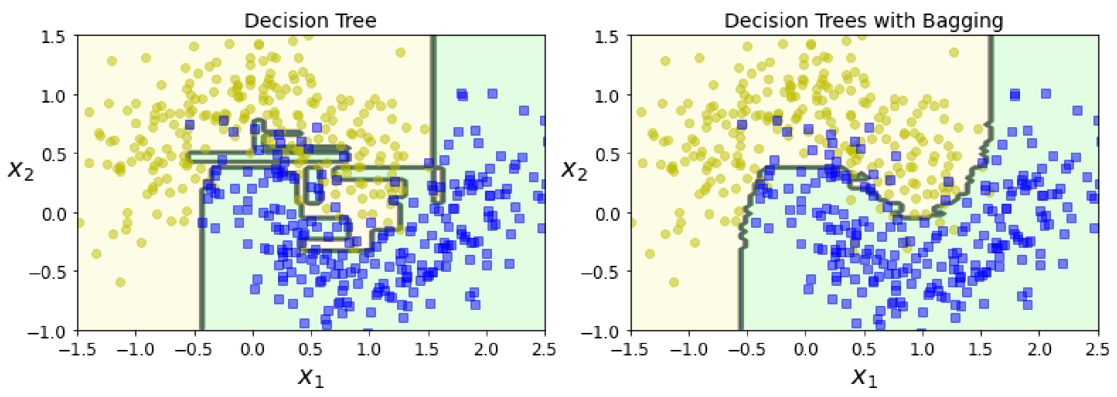
```
0.856
```

Let's plot decision boundaries.

```
[68]: from matplotlib.colors import ListedColormap

      def plot_decision_boundary(clf, X, y, axes=[-1.5, 2.5, -1, 1.5], alpha=0.5,␣
       ↪contour=True):
          x1s = np.linspace(axes[0], axes[1], 100)
          x2s = np.linspace(axes[2], axes[3], 100)
          x1, x2 = np.meshgrid(x1s, x2s)
          X_new = np.c_[x1.ravel(), x2.ravel()]
          y_pred = clf.predict(X_new).reshape(x1.shape)
          custom_cmap = ListedColormap(['#fafab0','#9898ff','#a0faa0'])
          plt.contourf(x1, x2, y_pred, alpha=0.3, cmap=custom_cmap)
          if contour:
              custom_cmap2 = ListedColormap(['#7d7d58','#4c4c7f','#507d50'])
              plt.contour(x1, x2, y_pred, cmap=custom_cmap2, alpha=0.8)
          plt.plot(X[:, 0][y==0], X[:, 1][y==0], "yo", alpha=alpha)
          plt.plot(X[:, 0][y==1], X[:, 1][y==1], "bs", alpha=alpha)
          plt.axis(axes)
          plt.xlabel(r"$x_1$", fontsize=18)
          plt.ylabel(r"$x_2$", fontsize=18, rotation=0)
```

```
[14]: plt.figure(figsize=(11,4))
      plt.subplot(121)
      plot_decision_boundary(tree_clf, X, y)
      plt.title("Decision Tree", fontsize=14)
      plt.subplot(122)
      plot_decision_boundary(bag_clf, X, y)
      plt.title("Decision Trees with Bagging", fontsize=14)
      save_fig("decision_tree_without_and_with_bagging_plot")
      plt.show()
```

```
Saving figure decision_tree_without_and_with_bagging_plot
```



The ensemble's predictions will likely generalize much better than the single Decision Tree's predictions: the ensemble has a comparable bias but a smaller variance (it makes roughly the same number of errors on the training set, but the decision boundary is less irregular).

Bootstrapping (pasting) introduces a bit more diversity in the subsets that each predictor is trained on, so bagging ends up with a slightly higher bias than pasting, but this also means that predictors end up being less correlated so the ensemble's variance is reduced. Overall, bagging often results in better models, which explains why it is generally preferred. However, if you have spare time and CPU power you can use crossvalidation to evaluate both bagging and pasting and select the one that works best.

```
[15]: pas_clf = BaggingClassifier(
          # We can only estimate 5 estimators withour replacement.
          DecisionTreeClassifier(random_state=42), n_estimators=5,
          max_samples=100, bootstrap=False, n_jobs=-1, random_state=42)
      pas_clf.fit(X_train, y_train)
      y_pred = pas_clf.predict(X_test)
      print(accuracy_score(y_test, y_pred))
```

```
0.88
```

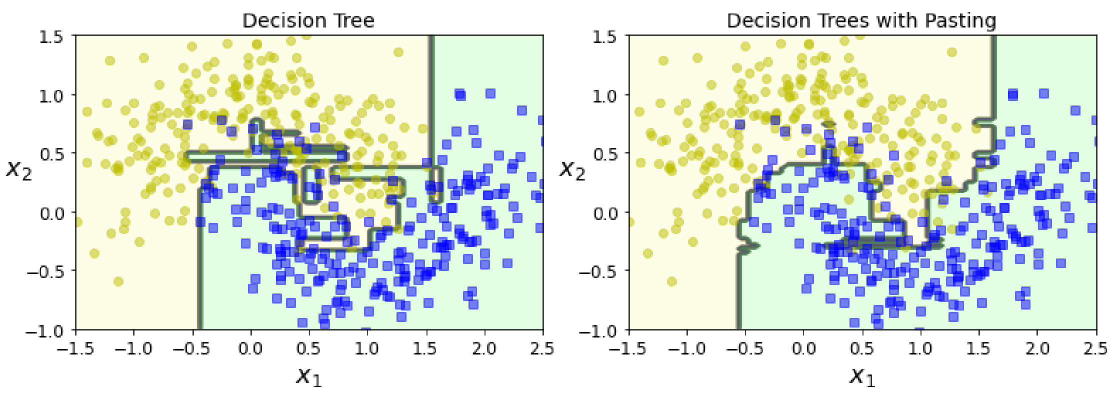
```
[17]: tree_clf = DecisionTreeClassifier(random_state=42)
      tree_clf.fit(X_train, y_train)
      y_pred_tree = tree_clf.predict(X_test)
      print(accuracy_score(y_test, y_pred_tree))
```

```
0.856
```

```
[18]: plt.figure(figsize=(11,4))
      plt.subplot(121)
      plot_decision_boundary(tree_clf, X, y)
      plt.title("Decision Tree", fontsize=14)
      plt.subplot(122)
      plot_decision_boundary(pas_clf, X, y)
```

```
plt.title("Decision Trees with Pasting", fontsize=14)
save_fig("decision_tree_without_and_with_pasting_plot")
plt.show()
```

```
Saving figure decision_tree_without_and_with_pasting_plot
```



Pasting classifier seems much weaker in this example because we only have 500 observations. So we can only draw 5 samples without replacement. A better comparison is with Bagging Classifier that would use only 5 trees.

[31]:
```
bag_clf = BaggingClassifier(
    DecisionTreeClassifier(splitter="random", max_leaf_nodes=16,␣
 ↪random_state=42),
    n_estimators=5, max_samples=1.0, bootstrap=True, random_state=42)
bag5_clf.fit(X_train, y_train)
y_pred = bag5_clf.predict(X_test)
print(accuracy_score(y_test, y_pred))
```

```
0.912
```

[32]:
```
from sklearn.ensemble import RandomForestClassifier
rnd_clf = RandomForestClassifier(n_estimators=5, max_leaf_nodes=16,␣
 ↪random_state=42)
rnd_clf.fit(X_train, y_train)
y_pred_rf = rnd_clf.predict(X_test)
print(accuracy_score(y_test, y_pred_rf))
```

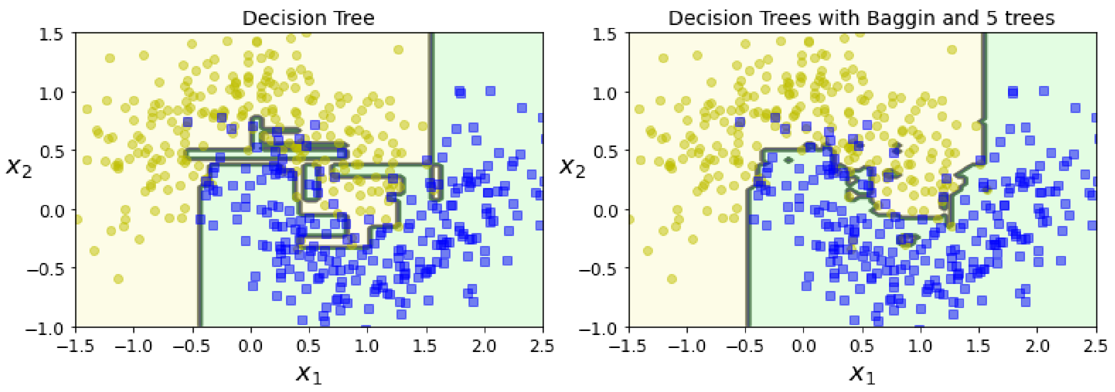
```
0.848
```

[33]:
```
np.sum(y_pred == y_pred_rf) / len(y_pred)  # almost identical predictions
```

[33]:
```
0.904
```

```
[34]: plt.figure(figsize=(11,4))
plt.subplot(121)
plot_decision_boundary(tree_clf, X, y)
plt.title("Decision Tree", fontsize=14)
plt.subplot(122)
plot_decision_boundary(bag5_clf, X, y)
plt.title("Decision Trees with Baggin and 5 trees", fontsize=14)
save_fig("decision_tree_without_and_with_pasting_plot")
plt.show()
```

```
Saving figure decision_tree_without_and_with_pasting_plot
```



The accuracy is much better using bootrap. The variance probably higher.

- When dealing with high-dimensional inputs, such as images. You may try both sampling of features using max_features and bootstrap_features, this is called Random Spaces. Each predictor will be trained on a random subset of the input features. (bootstrap=False and max_samples=1.0)
- If you have both large number of features and instance, you can sample both: (i.e., bootstrap_features=True and/or max_fea tures smaller than 1.0). This is called Random Patches.
- Sampling features results in even more predictor diversity, trading a bit more bias for a lower variance.

3.1 Out-of-Bag evaluation

With bagging, some instances may be sampled several times for any given predictor, while others may not be sampled at all. By default a BaggingClassifier samples m training instances with replacement (bootstrap=True), where m is the size of the training set. This means that only about 63% of the training instances are sampled on average for each predictor assuming we sample the data equal to the number of instances.

The remaining 37% of the training instances that are not sampled are called out-of-bag (oob) instances. Note that they are not the same 37% for all predictors. We can average predictions for each estimator using oob instances, without separate validation set or cross-validaion.

To do that we set: oob_score=True.

```
[35]: bag_clf = BaggingClassifier(
          DecisionTreeClassifier(random_state=42), n_estimators=500,
          bootstrap=True, n_jobs=-1, oob_score=True, random_state=40)
      bag_clf.fit(X_train, y_train)
      bag_clf.oob_score_
```

```
[35]: 0.9013333333333333
```

Even though validation produced slightly lower accuracy, the overall out-of-sample fit maybe better because of superior validation.

```
[36]: # We can extract estimated probabilities from the oob data.
      bag_clf.oob_decision_function_
```

```
[36]: array([[0.31746032, 0.68253968],
             [0.34117647, 0.65882353],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.08379888, 0.91620112],
             [0.31693989, 0.68306011],
             [0.02923977, 0.97076023],
             [0.97687861, 0.02312139],
             [0.97765363, 0.02234637],
             [0.74404762, 0.25595238],
             [0.        , 1.        ],
             [0.71195652, 0.28804348],
             [0.83957219, 0.16042781],
             [0.97777778, 0.02222222],
             [0.0625    , 0.9375    ],
             [0.        , 1.        ],
             [0.97297297, 0.02702703],
             [0.95238095, 0.04761905],
             [1.        , 0.        ],
             [0.01704545, 0.98295455],
             [0.38947368, 0.61052632],
             [0.88700565, 0.11299435],
             [1.        , 0.        ],
             [0.96685083, 0.03314917],
             [0.        , 1.        ],
             [0.99428571, 0.00571429],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.64804469, 0.35195531],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.13402062, 0.86597938],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.36065574, 0.63934426],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.27093596, 0.72906404],
             [0.34146341, 0.65853659],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.00531915, 0.99468085],
             [0.98265896, 0.01734104],
             [0.91428571, 0.08571429],
             [0.97282609, 0.02717391],
             [0.97029703, 0.02970297],
             [0.        , 1.        ],
             [0.06134969, 0.93865031],
             [0.98019802, 0.01980198],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.97790055, 0.02209945],
             [0.79473684, 0.20526316],
             [0.41919192, 0.58080808],
             [0.99473684, 0.00526316],
             [0.        , 1.        ],
             [0.67613636, 0.32386364],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [0.87356322, 0.12643678],
             [1.        , 0.        ],
             [0.56140351, 0.43859649],
             [0.16304348, 0.83695652],
             [0.67539267, 0.32460733],
             [0.90673575, 0.09326425],
             [0.        , 1.        ],
             [0.16201117, 0.83798883],
             [0.89005236, 0.10994764],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.995     , 0.005     ],
             [0.        , 1.        ],
             [0.07272727, 0.92727273],
             [0.05418719, 0.94581281],
             [0.29533679, 0.70466321],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.81871345, 0.18128655],
             [0.01092896, 0.98907104],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.22513089, 0.77486911],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.9368932 , 0.0631068 ],
             [0.76536313, 0.23463687],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.17127072, 0.82872928],
             [0.65306122, 0.34693878],
             [0.        , 1.        ],
             [0.03076923, 0.96923077],
             [0.49444444, 0.50555556],
             [1.        , 0.        ],
             [0.02673797, 0.97326203],
             [0.98870056, 0.01129944],
             [0.23121387, 0.76878613],
             [0.5       , 0.5       ],
             [0.9947644 , 0.0052356 ],
             [0.00555556, 0.99444444],
             [0.98963731, 0.01036269],
             [0.25641026, 0.74358974],
             [0.92972973, 0.07027027],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.80681818, 0.19318182],
             [1.        , 0.        ],
             [0.0106383 , 0.9893617 ],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [0.98181818, 0.01818182],
             [1.        , 0.        ],
             [0.01036269, 0.98963731],
             [0.97752809, 0.02247191],
             [0.99453552, 0.00546448],
             [0.01960784, 0.98039216],
             [0.18367347, 0.81632653],
             [0.98387097, 0.01612903],
             [0.29533679, 0.70466321],
             [0.98295455, 0.01704545],
             [0.        , 1.        ],
             [0.00561798, 0.99438202],
             [0.75138122, 0.24861878],
             [0.38624339, 0.61375661],
             [0.42708333, 0.57291667],
             [0.86315789, 0.13684211],
             [0.92964824, 0.07035176],
             [0.05699482, 0.94300518],
             [0.82802548, 0.17197452],
             [0.01546392, 0.98453608],
             [0.        , 1.        ],
             [0.02298851, 0.97701149],
             [0.96721311, 0.03278689],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [0.01041667, 0.98958333],
             [0.        , 1.        ],
             [0.0326087 , 0.9673913 ],
             [0.01020408, 0.98979592],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [0.93785311, 0.06214689],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [0.99462366, 0.00537634],
             [0.        , 1.        ],
             [0.38860104, 0.61139896],
             [0.32065217, 0.67934783],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.31182796, 0.68817204],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.00588235, 0.99411765],
             [0.        , 1.        ],
             [0.98387097, 0.01612903],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.62264151, 0.37735849],
             [0.92344498, 0.07655502],
             [0.        , 1.        ],
             [0.99526066, 0.00473934],
             [1.        , 0.        ],
             [0.98888889, 0.01111111],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.06451613, 0.93548387],
             [1.        , 0.        ],
             [0.05154639, 0.94845361],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.03278689, 0.96721311],
             [1.        , 0.        ],
             [0.95808383, 0.04191617],
             [0.79532164, 0.20467836],
             [0.55665025, 0.44334975],
             [0.        , 1.        ],
             [0.18604651, 0.81395349],
             [1.        , 0.        ],
             [0.93121693, 0.06878307],
             [0.97740113, 0.02259887],
             [1.        , 0.        ],
             [0.00531915, 0.99468085],
             [0.        , 1.        ],
             [0.44623656, 0.55376344],
             [0.86363636, 0.13636364],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.00558659, 0.99441341],
             [0.        , 1.        ],
             [0.96923077, 0.03076923],
             [0.        , 1.        ],
             [0.21649485, 0.78350515],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.98477157, 0.01522843],
             [0.8       , 0.2       ],
             [0.99441341, 0.00558659],
             [0.        , 1.        ],
             [0.08379888, 0.91620112],
             [0.98984772, 0.01015228],
             [0.01142857, 0.98857143],
             [0.        , 1.        ],
             [0.02747253, 0.97252747],
             [1.        , 0.        ],
             [0.79144385, 0.20855615],
             [0.        , 1.        ],
             [0.90804598, 0.09195402],
             [0.98387097, 0.01612903],
             [0.20634921, 0.79365079],
             [0.19767442, 0.80232558],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.20338983, 0.79661017],
             [0.98181818, 0.01818182],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.98969072, 0.01030928],
             [0.        , 1.        ],
             [0.48663102, 0.51336898],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.07821229, 0.92178771],
             [0.11176471, 0.88823529],
             [0.99415205, 0.00584795],
             [0.03015075, 0.96984925],
             [1.        , 0.        ],
             [0.40837696, 0.59162304],
             [0.04891304, 0.95108696],
             [0.51595745, 0.48404255],
             [0.51898734, 0.48101266],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.59903382, 0.40096618],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.24157303, 0.75842697],
             [0.81052632, 0.18947368],
             [0.08717949, 0.91282051],
             [0.99453552, 0.00546448],
             [0.82142857, 0.17857143],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [0.125     , 0.875     ],
             [0.04712042, 0.95287958],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.89150943, 0.10849057],
             [0.1978022 , 0.8021978 ],
             [0.95238095, 0.04761905],
             [0.00515464, 0.99484536],
             [0.609375  , 0.390625  ],
             [0.07692308, 0.92307692],
             [0.99484536, 0.00515464],
             [0.84210526, 0.15789474],
             [0.        , 1.        ],
             [0.99484536, 0.00515464],
             [0.95876289, 0.04123711],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.26903553, 0.73096447],
             [0.98461538, 0.01538462],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.00574713, 0.99425287],
             [0.85142857, 0.14857143],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.76506024, 0.23493976],
             [0.8969697 , 0.1030303 ],
             [1.        , 0.        ],
             [0.73333333, 0.26666667],
             [0.47727273, 0.52272727],
             [0.        , 1.        ],
             [0.92473118, 0.07526882],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.87709497, 0.12290503],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [0.74752475, 0.25247525],
             [0.09146341, 0.90853659],
             [0.44329897, 0.55670103],
             [0.22395833, 0.77604167],
             [0.        , 1.        ],
             [0.87046632, 0.12953368],
             [0.78212291, 0.21787709],
             [0.00507614, 0.99492386],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.02884615, 0.97115385],
             [0.96571429, 0.03428571],
             [0.93478261, 0.06521739],
             [1.        , 0.        ],
             [0.49756098, 0.50243902],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.01604278, 0.98395722],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.96987952, 0.03012048],
             [0.        , 1.        ],
             [0.05747126, 0.94252874],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.98989899, 0.01010101],
             [0.01675978, 0.98324022],
             [1.        , 0.        ],
             [0.13541667, 0.86458333],
             [0.        , 1.        ],
             [0.00546448, 0.99453552],
             [0.        , 1.        ],
             [0.41836735, 0.58163265],
             [0.11309524, 0.88690476],
             [0.22110553, 0.77889447],
             [1.        , 0.        ],
             [0.97647059, 0.02352941],
             [0.22826087, 0.77173913],
             [0.98882682, 0.01117318],
             [0.        , 1.        ],
             [0.        , 1.        ],
             [1.        , 0.        ],
             [0.96428571, 0.03571429],
             [0.33507853, 0.66492147],
             [0.98235294, 0.01764706],
             [1.        , 0.        ],
             [0.        , 1.        ],
             [0.99465241, 0.00534759],
             [0.        , 1.        ],
             [0.06043956, 0.93956044],
             [0.97619048, 0.02380952],
             [1.        , 0.        ],
             [0.03108808, 0.96891192],
             [0.57291667, 0.42708333]])
```

[37]:
```
from sklearn.metrics import accuracy_score
y_pred = bag_clf.predict(X_test)
accuracy_score(y_test, y_pred)
```

[37]:
```
0.912
```

4 Random Forests

Random Forest is the ensemble of Decision Trees, generally trained via the bagging method (or sometimes pasting), typically with max_samples set to the size of the training set. Instead of building Bagging classifier by hand, you can use RandomForestClassifier, which is more convenient and optimized for Decision Trees (similarly, there is a RandomForestRegressor class for regression tasks). The BaggingClassifier class remains useful if you want a bag of something other than Decision Trees.

The following code trains a Random Forest classifier with 500 trees (each limited to maximum 16 nodes).

With a few exceptions, a RandomForestClassifier has all the hyperparameters of a DecisionTreeClassifier (to control how trees are grown), plus all the hyperparameters of a BaggingClassifier to control the ensemble itself.

The Random Forest algorithm introduces extra randomness when growing trees; instead of searching for the very best feature when splitting a node, it searches for the best feature among a random subset of features. This results in a greater tree diversity, which (once again) trades a higher bias for a lower variance, generally yielding an overall better model.

[42]:
```
bag_clf = BaggingClassifier(
    DecisionTreeClassifier(splitter="random", max_leaf_nodes=16,␣
 ↪random_state=42),
    n_estimators=500, max_samples=1.0, bootstrap=True, n_jobs=-1,␣
 ↪random_state=42)
```

[43]:
```
bag_clf.fit(X_train, y_train)
y_pred = bag_clf.predict(X_test)
```

The following BaggingClassifier is roughly equivalent to the previous RandomForestClassifier:

[44]:
```
from sklearn.ensemble import RandomForestClassifier

rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16,␣
 ↪n_jobs=-1, random_state=42)
rnd_clf.fit(X_train, y_train)

y_pred_rf = rnd_clf.predict(X_test)
```

[45]:
```
np.sum(y_pred == y_pred_rf) / len(y_pred)  # Similar predictions predictions
```

[45]:
```
0.976
```

5 Feature Importance

if you look at a single Decision Tree, important features are likely to appear closer to the root of the tree, while unimportant features will often appear closer to the leaves (or not at all). We can estimate of a feature's importance by computing the average depth at which it appears across all trees in the forest.

Scikit-Learn computes this automatically for every feature after training. You can access the result using the feature_importances_ variable. For example, the following code trains a RandomForest-Classifier on the iris dataset and outputs each feature's importance.

The most important features are the petal length (44%) and width (42%), while sepal length and width are rather unimportant in comparison (11% and 2%, respectively):

[46]:
```
from sklearn.datasets import load_iris
iris = load_iris()
rnd_clf = RandomForestClassifier(n_estimators=500, n_jobs=-1, random_state=42)
rnd_clf.fit(iris["data"], iris["target"])
for name, score in zip(iris["feature_names"], rnd_clf.feature_importances_):
    print(name, score)
```

```
sepal length (cm) 0.11249225099876375
sepal width (cm) 0.02311928828251033
petal length (cm) 0.4410304643639577
petal width (cm) 0.4233579963547682
```

[47]:
```
rnd_clf.feature_importances_
```

[47]:
```
array([0.11249225, 0.02311929, 0.44103046, 0.423358  ])
```

Feature importance is a key to datasets when you have a lot of features, such as in graphics, when you need to compare the pixels that are the most important for identification. Random Forests are very handy in this task. Let's go back to our digit identification data.

```
[51]: from sklearn.datasets import fetch_openml
      mnist = fetch_openml('mnist_784', version=1)
      mnist.keys()
      #X, y = mnist["data"], mnist["target"]
```

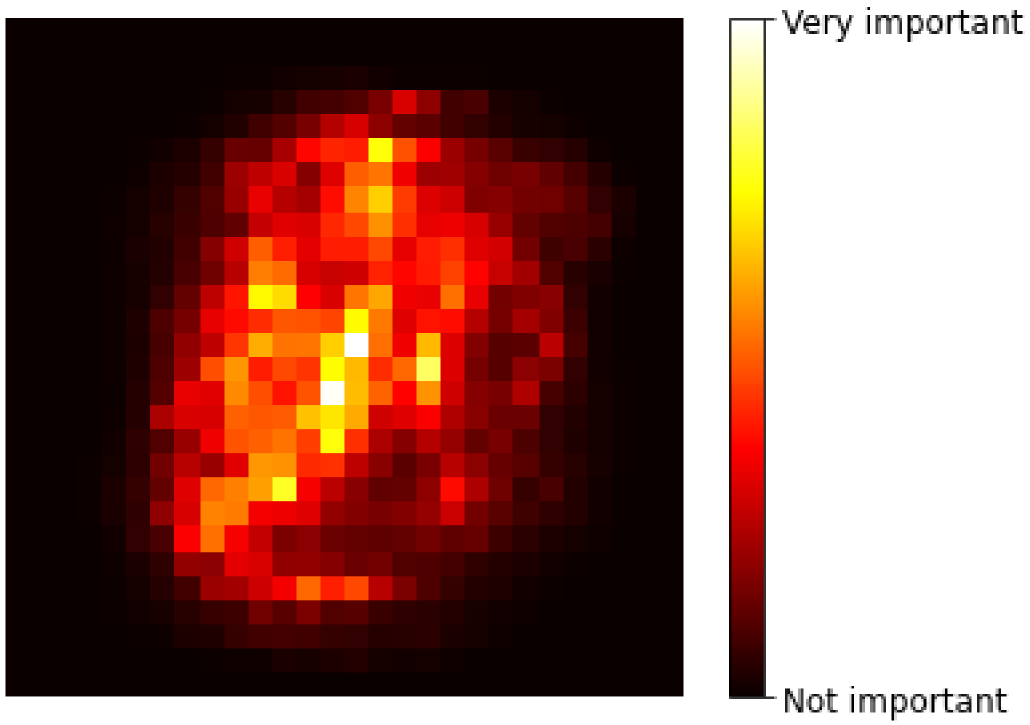
```
[52]: # Estimate unrestricted random forest classifier.
      rnd_clf = RandomForestClassifier(random_state=42)
      rnd_clf.fit(mnist["data"], mnist["target"])
```

```
[52]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                             criterion='gini', max_depth=None, max_features='auto',
                             max_leaf_nodes=None, max_samples=None,
                             min_impurity_decrease=0.0, min_impurity_split=None,
                             min_samples_leaf=1, min_samples_split=2,
                             min_weight_fraction_leaf=0.0, n_estimators=100,
                             n_jobs=None, oob_score=False, random_state=42, verbose=0,
                             warm_start=False)
```

```
[63]: # Reshape image vector into matrix
      def plot_digit(data):
          image = data.reshape(28, 28)
          plt.imshow(image, cmap = plt.cm.hot,
                     interpolation="nearest")
          plt.axis("off")
```

```
[64]: # Plot feature importance
      plot_digit(rnd_clf.feature_importances_)
      cbar = plt.colorbar(ticks=[rnd_clf.feature_importances_.min(), rnd_clf.
       ↪feature_importances_.max()])
      cbar.ax.set_yticklabels(['Not important', 'Very important'])
      save_fig("mnist_feature_importance_plot")
      plt.show()
```

```
Saving figure mnist_feature_importance_plot
```



6 Boosting

Boosting (originally called hypothesis boosting) refers to any Ensemble method that can combine several weak learners into a strong learner. The general idea of most boosting methods is to train predictors sequentially, each trying to correct its predecessor.

7 AdaBoost

Adaptive Boosting (AdaBoost) focuses on the training instances that the predecessor under-fitted. After each classifier we increase the weight of misclassified training instances. (picture of changing of importance of instances). AdaBoost adds predictors to the ensemble,gradually making it better.

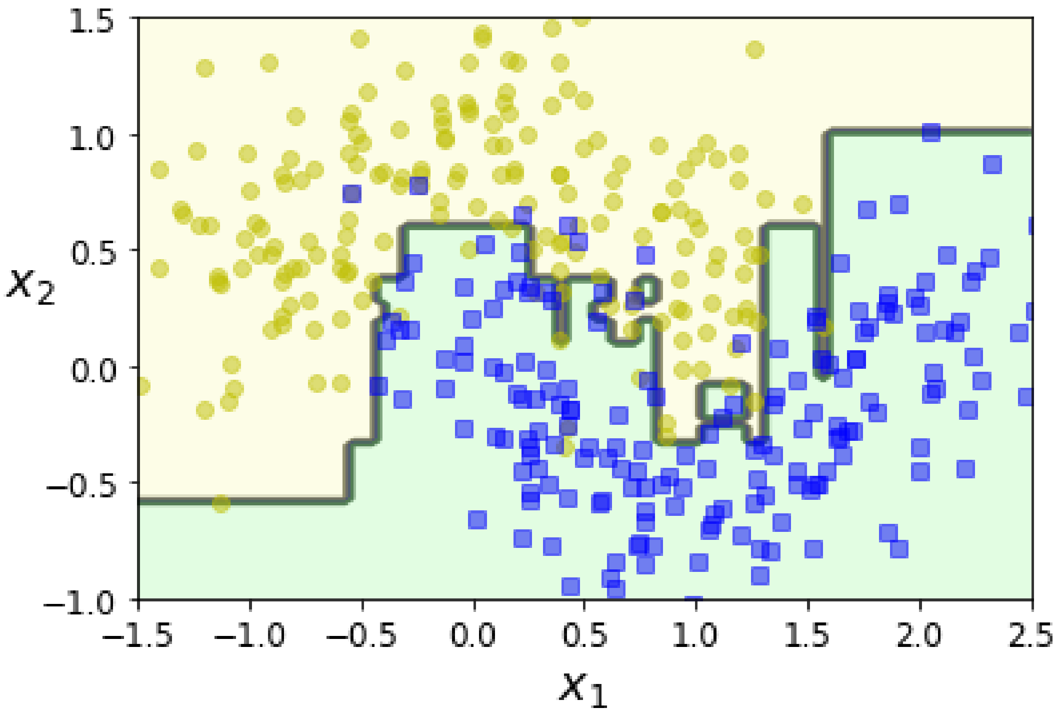
Let's try it on training data.

[67]:
```
from sklearn.ensemble import AdaBoostClassifier

ada_clf = AdaBoostClassifier(
    DecisionTreeClassifier(max_depth=1), n_estimators=200,
    algorithm="SAMME.R", learning_rate=0.5, random_state=42)
ada_clf.fit(X_train, y_train)
```

[67]:
```
AdaBoostClassifier(algorithm='SAMME.R',
                   base_estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                                                          class_weight=None,
                                                                          criterion='gini',
                                                                          max_depth=1,
                                                                          max_features=None,
                                                                          max_leaf_nodes=None,
min_impurity_decrease=0.0,
min_impurity_split=None,
                                                                          min_samples_leaf=1,
                                                                          min_samples_split=2,
min_weight_fraction_leaf=0.0,
                                                                          presort='deprecated',
                                                                          random_state=None,
                                                                          splitter='best'),
                    learning_rate=0.5, n_estimators=200, random_state=42)
```

SAMME.R uses the probability estimates to update the additive model, while SAMME uses the classifications only. As the example illustrates, the SAMME.R algorithm typically converges faster than SAMME, achieving a lower test error with fewer boosting iterations.

[70]:
```
plot_decision_boundary(ada_clf, X_train, y_train)
```



[ ]:
```

```

A drawback to this sequential learning technique: it cannot be parallelized (or only partially), since each predictor can only be trained after the previous predictor has been trained and evaluated. As a result, it does not scale as well as bagging or pasting.

Initially each instance has a weight $w^{(i)} = \frac{1}{m}$. After the first classifier is trained we compute the weighted error rate:

$$r_j = \frac{\sum_{\substack{i=1 \\ \hat{y}_j^{(i)} \neq y^{(i)}}}^{m} w^{(i)}}{\sum_{i=1}^{m} w^{(i)}}$$

where $y_j^{(i)}$ is the j^{th} predictor's prediction for the i^{th} instance.

Each predictor is assigned a weight α_j, which is computed using equation below, where η is the learning rate hyperparameter (defaults to 1). The more accurate the predictor is, the higher its weight will be. If it is just guessing randomly, then its weight will be close to zero. However, if it is most often wrong (i.e., less accurate than random guessing), then its weight will be negative.

$$\alpha_j = \eta \log \left(\frac{1 - r_j}{r_j} \right)$$

After each estimation we update weights of each observation:

$$w^{(i)} = \begin{cases} w^{(i)} & if \quad \hat{y}_j^{(i)} = y^{(i)} \\ w^{(i)} \exp(\alpha_j) & if \quad \hat{y}_j^{(i)} \neq y^{(i)} \end{cases}$$

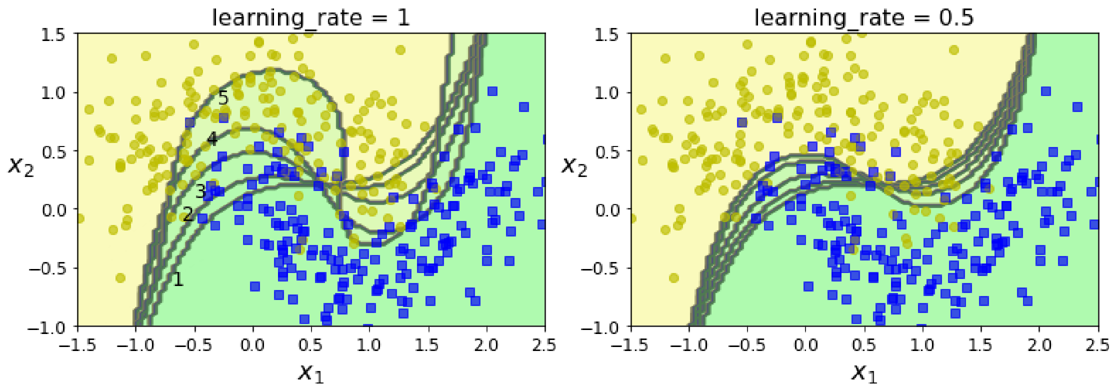
This updating increases the weight of the incorrectly predicted instance. The increase in weight is a function of the reliabilitiy of the classifier. If an accurate (high α_j) classifier misclassifies an instance, the weight of this instance increases more than if it missclassified by the inaccurate (low α_j) classifier. The all weights are normalized, i.e., divided by $\sum_{i=1}^{m} w^{(i)}$). This process is repeated until are run out of classifiers or we reached a perfect classifier. To make predictions the AdaBoost calculates a weighted average of predictions from all classifiers using α_j as a weight.

Scikit-Learn uses a multiclass version of AdaBoost called SAMME (Stagewise Additive Modeling using a Multiclass Exponential loss function). When there are just two classes, SAMME is equivalent to AdaBoost. Moreover, if the predictors can estimate class probabilities, Scikit-Learn can use a variant of SAMME called SAMME.R (the R stands for "Real"), which relies on class probabilities rather than predictions and generally performs better.

Let's try to see this training using SVM classifier just updating sample weights.

[72]:
```
m = len(X_train)

plt.figure(figsize=(11, 4))
for subplot, learning_rate in ((121, 1), (122, 0.5)):
    sample_weights = np.ones(m)
    plt.subplot(subplot)
    for i in range(5):
        svm_clf = SVC(kernel="rbf", C=0.05, random_state=42)
        svm_clf.fit(X_train, y_train, sample_weight=sample_weights)
        y_pred = svm_clf.predict(X_train)
        sample_weights[y_pred != y_train] *= (1 + learning_rate)
        plot_decision_boundary(svm_clf, X_train, y_train, alpha=0.2)
        plt.title("learning_rate = {}".format(learning_rate), fontsize=16)
    if subplot == 121:
        plt.text(-0.7, -0.65, "1", fontsize=14)
        plt.text(-0.6, -0.10, "2", fontsize=14)
        plt.text(-0.5,  0.10, "3", fontsize=14)
        plt.text(-0.4,  0.55, "4", fontsize=14)
        plt.text(-0.3,  0.90, "5", fontsize=14)

save_fig("boosting_plot")
plt.show()
```

```
Saving figure boosting_plot
```



Lerning rate 1, may be too fast.

8 Gradient Boosting

Another Boosting algorithm is Gradient Boosting. Just like AdaBoost, Gradient Boosting works by sequentially adding predictors to an ensemble, each one correcting its predecessor. However, instead of tweaking the instance weights at every iteration like AdaBoost does, this method tries to fit the new predictor to the residual errors made by the previous predictor. Let's try it on Tree Regression: $X = u - 0.5$, $y = 3X^2 + 0.05 * u$.

[73]:
```
np.random.seed(42)
X = np.random.rand(100, 1) - 0.5
y = 3*X[:, 0]**2 + 0.05 * np.random.randn(100)
```

Set Decision Tree regression with maximum depth of 2

[74]:
```
from sklearn.tree import DecisionTreeRegressor

tree_reg1 = DecisionTreeRegressor(max_depth=2, random_state=42)
tree_reg1.fit(X, y)
```

```
[74]: DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=2,
                            max_features=None, max_leaf_nodes=None,
                            min_impurity_decrease=0.0, min_impurity_split=None,
                            min_samples_leaf=1, min_samples_split=2,
                            min_weight_fraction_leaf=0.0, presort='deprecated',
                            random_state=42, splitter='best')
```

Now train a second DecisionTreeRegressor on the residual errors made by the first predictor:

```
[77]: # Residuals: y - hat(y)
      y2 = y - tree_reg1.predict(X)
      tree_reg2 = DecisionTreeRegressor(max_depth=2, random_state=42)
      tree_reg2.fit(X, y2)
```

```
[77]: DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=2,
                            max_features=None, max_leaf_nodes=None,
                            min_impurity_decrease=0.0, min_impurity_split=None,
                            min_samples_leaf=1, min_samples_split=2,
                            min_weight_fraction_leaf=0.0, presort='deprecated',
                            random_state=42, splitter='best')
```

Then we train a third regressor on the residual errors made by the second predictor:

```
[78]: y3 = y2 - tree_reg2.predict(X)
      tree_reg3 = DecisionTreeRegressor(max_depth=2, random_state=42)
      tree_reg3.fit(X, y3)
```

```
[78]: DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=2,
                            max_features=None, max_leaf_nodes=None,
                            min_impurity_decrease=0.0, min_impurity_split=None,
                            min_samples_leaf=1, min_samples_split=2,
                            min_weight_fraction_leaf=0.0, presort='deprecated',
                            random_state=42, splitter='best')
```

```
[79]: X_new = np.array([[0.8]])
```

Now we have an ensemble containing three trees. It can make predictions on a new instance simply by adding up the predictions of all the trees:

```
[80]: y_pred = sum(tree.predict(X_new) for tree in (tree_reg1, tree_reg2, tree_reg3))
```

```
[81]: y_pred
```

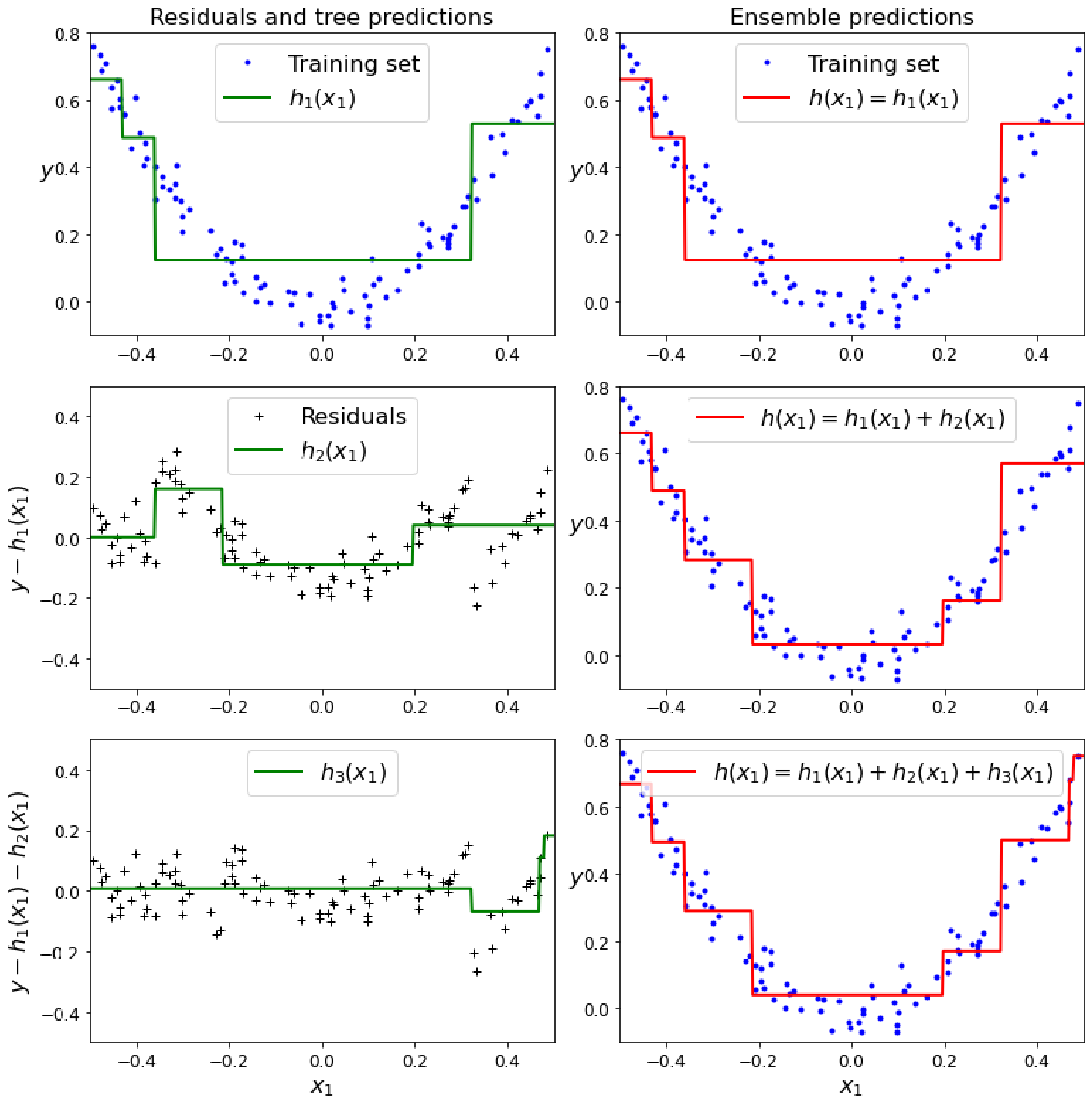
```
[81]: array([0.75026781])
```

Next we plot the predictions of these three trees in the left column, and the ensemble's predictions in the right column.

- In the first row, the ensemble has just one tree, so its predictions are exactly the same as the first tree's predictions.
- In the second row, a new tree is trained on the residual errors of the first tree. On the right you can see that the ensemble's predictions are equal to the sum of the predictions of the first two trees.
- In the third row another tree is trained on the residual errors of the second tree. You can see that the ensemble's predictions gradually get better as trees are added to the ensemble.

[82]:
```
def plot_predictions(regressors, X, y, axes, label=None, style="r-",␣
 ↪data_style="b.", data_label=None):
    x1 = np.linspace(axes[0], axes[1], 500)
    y_pred = sum(regressor.predict(x1.reshape(-1, 1)) for regressor in␣
 ↪regressors)
    plt.plot(X[:, 0], y, data_style, label=data_label)
    plt.plot(x1, y_pred, style, linewidth=2, label=label)
    if label or data_label:
        plt.legend(loc="upper center", fontsize=16)
    plt.axis(axes)

plt.figure(figsize=(11,11))

plt.subplot(321)
plot_predictions([tree_reg1], X, y, axes=[-0.5, 0.5, -0.1, 0.8],␣
 ↪label="$h_1(x_1)$", style="g-", data_label="Training set")
plt.ylabel("$y$", fontsize=16, rotation=0)
plt.title("Residuals and tree predictions", fontsize=16)

plt.subplot(322)
plot_predictions([tree_reg1], X, y, axes=[-0.5, 0.5, -0.1, 0.8], label="$h(x_1)␣
 ↪= h_1(x_1)$", data_label="Training set")
plt.ylabel("$y$", fontsize=16, rotation=0)
plt.title("Ensemble predictions", fontsize=16)

plt.subplot(323)
plot_predictions([tree_reg2], X, y2, axes=[-0.5, 0.5, -0.5, 0.5],␣
 ↪label="$h_2(x_1)$", style="g-", data_style="k+", data_label="Residuals")
plt.ylabel("$y - h_1(x_1)$", fontsize=16)

plt.subplot(324)
plot_predictions([tree_reg1, tree_reg2], X, y, axes=[-0.5, 0.5, -0.1, 0.8],␣
 ↪label="$h(x_1) = h_1(x_1) + h_2(x_1)$")
plt.ylabel("$y$", fontsize=16, rotation=0)

plt.subplot(325)
plot_predictions([tree_reg3], X, y3, axes=[-0.5, 0.5, -0.5, 0.5],␣
 ↪label="$h_3(x_1)$", style="g-", data_style="k+")
plt.ylabel("$y - h_1(x_1) - h_2(x_1)$", fontsize=16)
plt.xlabel("$x_1$", fontsize=16)

plt.subplot(326)
plot_predictions([tree_reg1, tree_reg2, tree_reg3], X, y, axes=[-0.5, 0.5, -0.
↪1, 0.8], label="$h(x_1) = h_1(x_1) + h_2(x_1) + h_3(x_1)$")
plt.xlabel("$x_1$", fontsize=16)
plt.ylabel("$y$", fontsize=16, rotation=0)

save_fig("gradient_boosting_plot")
plt.show()
```

```
Saving figure gradient_boosting_plot
```



A simpler way to train GBRT ensembles is to use Scikit-Learn's GradientBoostingRegressor class. Much like the RandomForestRegressor class, it has hyperparameters to control the growth of Decision Trees (e.g., max_depth, min_samples_leaf, and so on), as well as hyperparameters to control the ensemble training, such as the number of trees (n_estimators).

[83]:
```
from sklearn.ensemble import GradientBoostingRegressor

gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=5, learning_rate=1.
 ↪0, random_state=42)
gbrt.fit(X, y)
```

[83]:
```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=1.0, loss='ls', max_depth=2,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=5,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=42, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

[84]:
```
gbrt_slow = GradientBoostingRegressor(max_depth=2, n_estimators=5,␣
 ↪learning_rate=0.1, random_state=42)
gbrt_slow.fit(X, y)
```

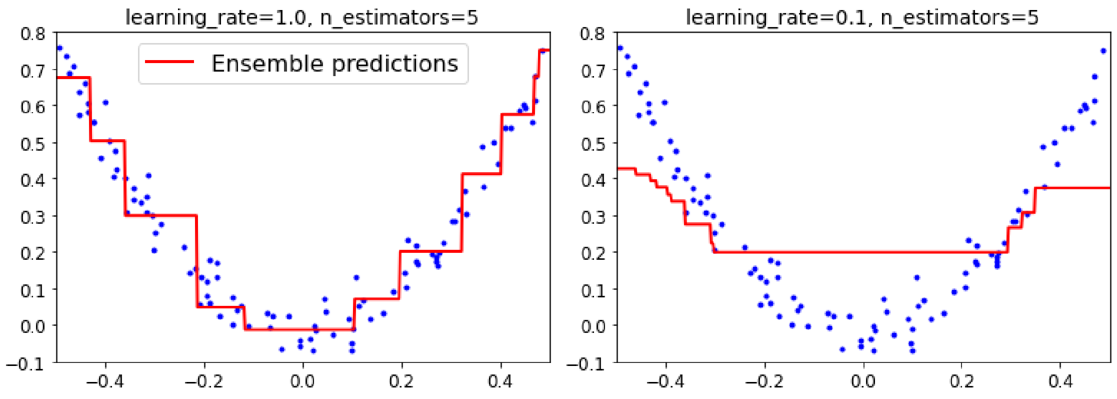
[84]:
```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='ls', max_depth=2,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=5,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=42, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

The learning_rate hyperparameter scales the contribution of each tree. If you set it to a low value, such as 0.1, you will need more trees in the ensemble to fit the training set, but the predictions will usually generalize better. This is a regularization technique called shrinkage.

[85]:
```
plt.figure(figsize=(11,4))

plt.subplot(121)
plot_predictions([gbrt], X, y, axes=[-0.5, 0.5, -0.1, 0.8], label="Ensemble␣
 ↪predictions")
plt.title("learning_rate={}, n_estimators={}".format(gbrt.learning_rate, gbrt.
 ↪n_estimators), fontsize=14)
```

```
plt.subplot(122)
plot_predictions([gbrt_slow], X, y, axes=[-0.5, 0.5, -0.1, 0.8])
plt.title("learning_rate={}, n_estimators={}".format(gbrt_slow.learning_rate,␣
 ↪gbrt_slow.n_estimators), fontsize=14)

save_fig("gbrt_learning_rate_plot")
plt.show()
```

```
Saving figure gbrt_learning_rate_plot
```

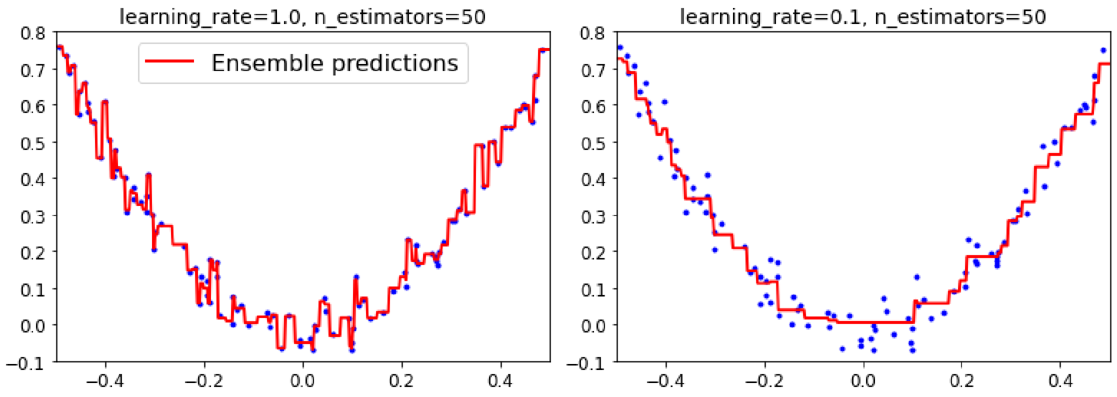


Above we show two GBRT ensembles trained with a low learning rate: the one on the left have enough trees to fit the training set, while the one on the right has too few trees and under-fits the training set. With more (50) trees the performance of slow-learning tree is optimal, while fast-learning is over-fittig.

[86]:
```
gbrt_slow = GradientBoostingRegressor(max_depth=2, n_estimators=50,␣
 ↪learning_rate=0.1, random_state=42)
gbrt_slow.fit(X, y)
gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=50, learning_rate=1.
 ↪0, random_state=42)
gbrt.fit(X, y)
plt.figure(figsize=(11,4))

plt.subplot(121)
plot_predictions([gbrt], X, y, axes=[-0.5, 0.5, -0.1, 0.8], label="Ensemble␣
 ↪predictions")
plt.title("learning_rate={}, n_estimators={}".format(gbrt.learning_rate, gbrt.
 ↪n_estimators), fontsize=14)

plt.subplot(122)
plot_predictions([gbrt_slow], X, y, axes=[-0.5, 0.5, -0.1, 0.8])
plt.title("learning_rate={}, n_estimators={}".format(gbrt_slow.learning_rate,␣
 ↪gbrt_slow.n_estimators), fontsize=14)
```

```
save_fig("gbrt_learning_rate_plot")
plt.show()
```

```
Saving figure gbrt_learning_rate_plot
```



8.1 Gradient Boosting with Early stopping

In order to find the optimal number of trees, you can use early stopping. A simple way to implement this is to use the staged_predict() method: it returns an iterator over the predictions made by the ensemble at each stage of training (with one tree, two trees, etc.). The following code trains a GBRT ensemble with 120 trees, then measures the validation error at each stage of training to find the optimal number of trees, and finally trains another GBRT ensemble using the optimal number of trees:

[87]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X_train, X_val, y_train, y_val = train_test_split(X, y, random_state=49)
# 120 trees
gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=120, random_state=42)
gbrt.fit(X_train, y_train)
# Calculate errors: staged.predict has a matrix of predictions for adding␣
↪estimators from 1 to 120.
errors = [mean_squared_error(y_val, y_pred)
          for y_pred in gbrt.staged_predict(X_val)]
#best number of estimators produces the smallest error.
bst_n_estimators = np.argmin(errors)

gbrt_best =␣
↪GradientBoostingRegressor(max_depth=2,n_estimators=bst_n_estimators,␣
↪random_state=42)
```

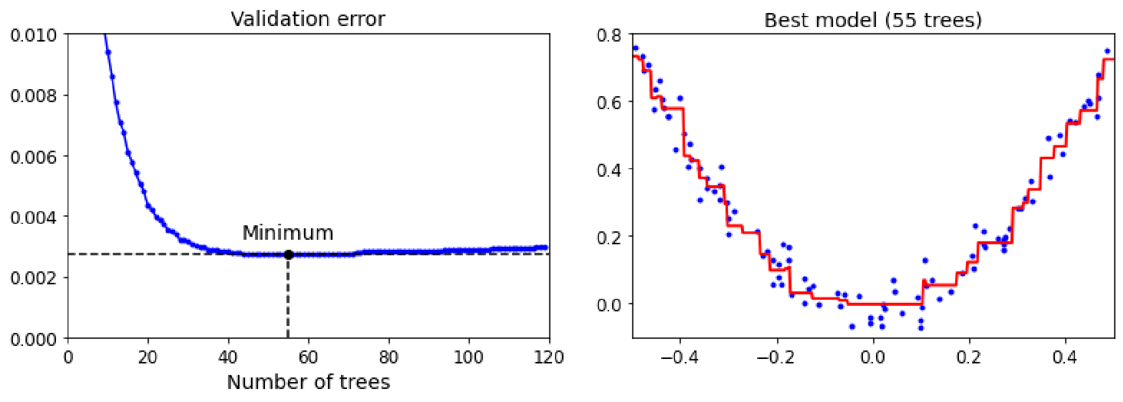
```
gbrt_best.fit(X_train, y_train)
```

```
[87]: GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                                init=None, learning_rate=0.1, loss='ls', max_depth=2,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=55,
                                n_iter_no_change=None, presort='deprecated',
                                random_state=42, subsample=1.0, tol=0.0001,
                                validation_fraction=0.1, verbose=0, warm_start=False)
```

```
[88]: min_error = np.min(errors)
```

```
[89]: plt.figure(figsize=(11, 4))

      plt.subplot(121)
      plt.plot(errors, "b.-")
      plt.plot([bst_n_estimators, bst_n_estimators], [0, min_error], "k--")
      plt.plot([0, 120], [min_error, min_error], "k--")
      plt.plot(bst_n_estimators, min_error, "ko")
      plt.text(bst_n_estimators, min_error*1.2, "Minimum", ha="center", fontsize=14)
      plt.axis([0, 120, 0, 0.01])
      plt.xlabel("Number of trees")
      plt.title("Validation error", fontsize=14)

      plt.subplot(122)
      plot_predictions([gbrt_best], X, y, axes=[-0.5, 0.5, -0.1, 0.8])
      plt.title("Best model (%d trees)" % bst_n_estimators, fontsize=14)

      save_fig("early_stopping_gbrt_plot")
      plt.show()
```

```
Saving figure early_stopping_gbrt_plot
```



There are few ways to speed thing up: * When you move from estimating n trees to estimating $n+1$ trees, we can keep the trees we already estimated and just add a one tree by setting warm_start=True allowing incremental training. * You can stop learning if the validation error does not improve. The following code stops training when the validation error does not improve for five iterations in a row:

[91]:
```
gbrt = GradientBoostingRegressor(max_depth=2, warm_start=True, random_state=42)
# Start with large error
min_val_error = float("inf")
error_going_up = 0
for n_estimators in range(1, 120):
    # Set the number of estimators
    gbrt.n_estimators = n_estimators
    gbrt.fit(X_train, y_train)
    y_pred = gbrt.predict(X_val)
    # calculate validation error
    val_error = mean_squared_error(y_val, y_pred)
    # if errors are decreasing nothing happens
    if val_error < min_val_error:
        min_val_error = val_error
        error_going_up = 0
        #if they are increasing start counting, if five steps in a row increase␣
 ↪errors then stop
    else:
        error_going_up += 1
        if error_going_up == 5:
            break  # early stopping
```

We know that the smallest validation error is achieved with 55 trees. Out algorithm stopped at $56 + 5 = 61$ trees:

[92]:
```
print(gbrt.n_estimators)
```

```
61
```

[93]:
```
print("Minimum validation MSE:", min_val_error)
```

```
Minimum validation MSE: 0.002712853325235463
```

The GradientBoostingRegressor class supports a subsample hyperparameter, which specifies the fraction of training instances to be used for training each tree. For example, if subsample=0.25, then each tree is trained on 25% of the training instances, selected randomly. This trades a higher bias for a lower variance. It also speeds up training considerably. This technique is called Stochastic Gradient Boosting.

8.2 Using XGBoost

Use for fast parallel boosting. Probably will not cover in class.

[94]:
```
try:
    import xgboost
except ImportError as ex:
    print("Error: the xgboost library is not installed.")
    xgboost = None
```

[95]:
```
if xgboost is not None:  # not shown in the book
    xgb_reg = xgboost.XGBRegressor(random_state=42)
    xgb_reg.fit(X_train, y_train)
    y_pred = xgb_reg.predict(X_val)
    val_error = mean_squared_error(y_val, y_pred)
    print("Validation MSE:", val_error)
```

```
[21:06:43] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
Validation MSE: 0.00285125597265563943
```

[96]:
```
if xgboost is not None:  # not shown in the book
    xgb_reg.fit(X_train, y_train,
                eval_set=[(X_val, y_val)], early_stopping_rounds=2)
    y_pred = xgb_reg.predict(X_val)
    val_error = mean_squared_error(y_val, y_pred)
    print("Validation MSE:", val_error)
```

```
[21:06:44] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[0]     validation_0-rmse:0.286719
Will train until validation_0-rmse hasn't improved in 2 rounds.
[1]     validation_0-rmse:0.258221
[2]     validation_0-rmse:0.232634
[3]     validation_0-rmse:0.210526
[4]     validation_0-rmse:0.190232
[5]     validation_0-rmse:0.172196
[6]     validation_0-rmse:0.156394
[7]     validation_0-rmse:0.142241
[8]     validation_0-rmse:0.129789
[9]     validation_0-rmse:0.118752
[10]    validation_0-rmse:0.108388
[11]    validation_0-rmse:0.100155
[12]    validation_0-rmse:0.09208
[13]    validation_0-rmse:0.084791
[14]    validation_0-rmse:0.078699
[15]    validation_0-rmse:0.073248
[16]    validation_0-rmse:0.069391
[17]    validation_0-rmse:0.066277
[18]    validation_0-rmse:0.063458
[19]    validation_0-rmse:0.060326
[20]    validation_0-rmse:0.0578
```

```
[21]	validation_0-rmse:0.055643
[22]	validation_0-rmse:0.053943
[23]	validation_0-rmse:0.053138
[24]	validation_0-rmse:0.052415
[25]	validation_0-rmse:0.051821
[26]	validation_0-rmse:0.051226
[27]	validation_0-rmse:0.051135
[28]	validation_0-rmse:0.05091
[29]	validation_0-rmse:0.050893
[30]	validation_0-rmse:0.050725
[31]	validation_0-rmse:0.050471
[32]	validation_0-rmse:0.050285
[33]	validation_0-rmse:0.050492
[34]	validation_0-rmse:0.050348
Stopping. Best iteration:
[32]	validation_0-rmse:0.050285

Validation MSE: 0.002528626115371327
```

[97]:
```
%timeit xgboost.XGBRegressor().fit(X_train, y_train) if xgboost is not None␣
↪else None
```

```
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

```
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
[21:06:50] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
100 loops, best of 3: 10.2 ms per loop
```

[98]:
```
%timeit GradientBoostingRegressor().fit(X_train, y_train)
```

```
10 loops, best of 3: 21.6 ms per loop
```