Ensembles

4

Zukhra Abdiakhmetova

In this lecture

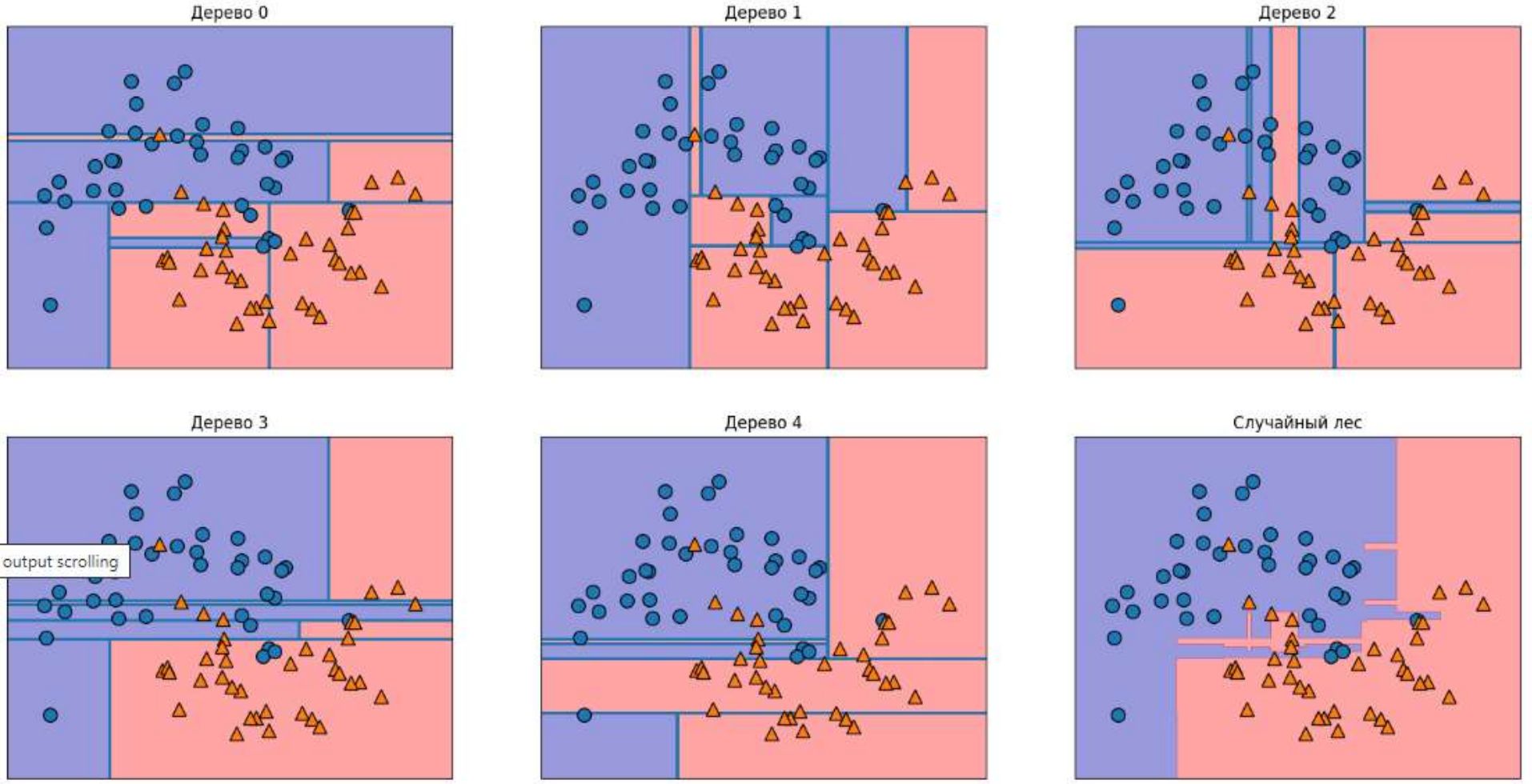
- Random Forest
- Gradient Boosting of regression trees

- Ensemble methods are techniques that create multiple models and then combine them to produce improved results

Random Forest

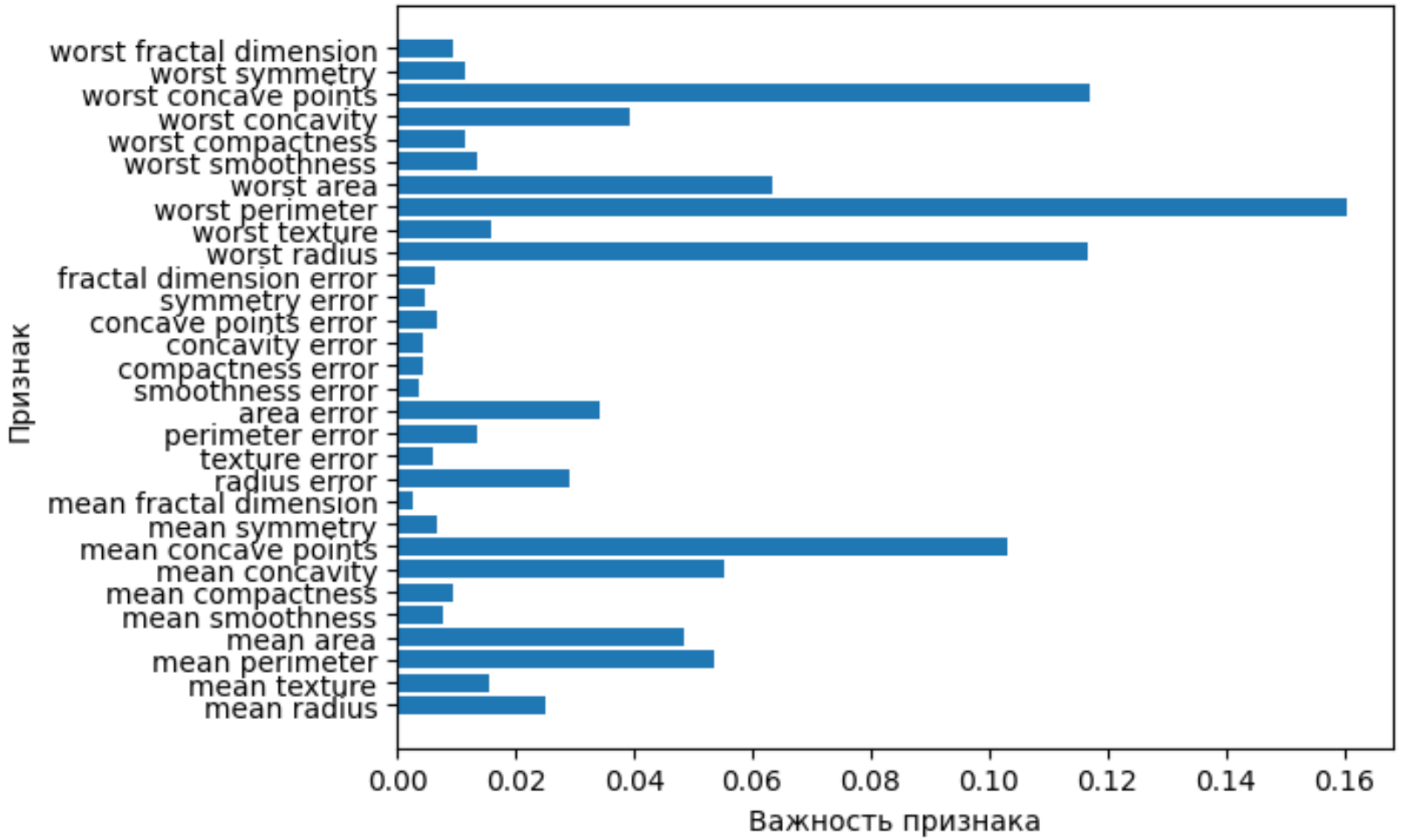
- Decision trees – overtraining problem

```
from sklearn.ensemble
import RandomForestClassifierfrom sklearn.datasets
import make_moonsX, y = make_moons(n_samples=100, noise=0.25, random_state=3)

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42)

forest = RandomForestClassifier(n_estimators=5, random_state=2)forest.fit(X_train, y_train)
fig, axes = plt.subplots(2, 3, figsize=(20, 10))
for i, (ax, tree) in enumerate(zip(axes.ravel(), forest.estimators_)): ax.set_title("Дерево {}".format(i))
mglearn.plots.plot_tree_partition(X_train, y_train, tree, ax=ax)
mglearn.plots.plot_2d_separator(forest, X_train, fill=True, ax=axes[-1, -1], alpha=.4)axes[-1, -1].set_title("Случайный лес")
mglearn.discrete_scatter(X_train[:, 0], X_train[:, 1], y_train)
```


Дерево 0
Дерево 1
Дерево 2
Дерево 3
Дерево 4
Случайный лес
output scrolling

```
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, random_state=0)
forest = RandomForestClassifier(n_estimators=100, random_state=0)
forest.fit(X_train, y_train)
print("Правильность на обучающем наборе: {:.3f}".format(forest.score(X_train, y_train)))
print("Правильность на тестовом наборе: {:.3f}".format(forest.score(X_test, y_test)))
```

```
def plot_feature_importances_cancer(model): n_features = cancer.data.shape[1]
plt.barh(range(n_features), model.feature_importances_, align='center')
plt.yticks(np.arange(n_features), cancer.feature_names)
plt.xlabel("Важность признака")
plt.ylabel("Признак")plot_feature_importances_cancer(forest)
```



+/-

- widely used
- High prediction
- n_jobs for parallel programming
- N_jobs=-1 for use all computer cores
- Bad work on high measuring data as text
- max_features=sqrt(n_features) for classification and max_features=n_features for regression

Gradient boosting of regression trees

Unlike random forest, gradient boosting builds a sequence of trees in which each tree tries to correct the mistakes of the previous one.

```
from sklearn.ensemble import GradientBoostingClassifier
X_train, X_test, y_train, y_test = train_test_split(
 cancer.data, cancer.target, random_state=0)
gbrt = GradientBoostingClassifier(random_state=0)
gbrt.fit(X_train, y_train)

print("Правильность на обучающем наборе: {:.3f}".format(gbrt.score(X_train, y_train)))
print("Правильность на тестовом наборе: {:.3f}".format(gbrt.score(X_test, y_test)))
```

```
Правильность на обучающем наборе: 1.000
Правильность на тестовом наборе: 0.965
```

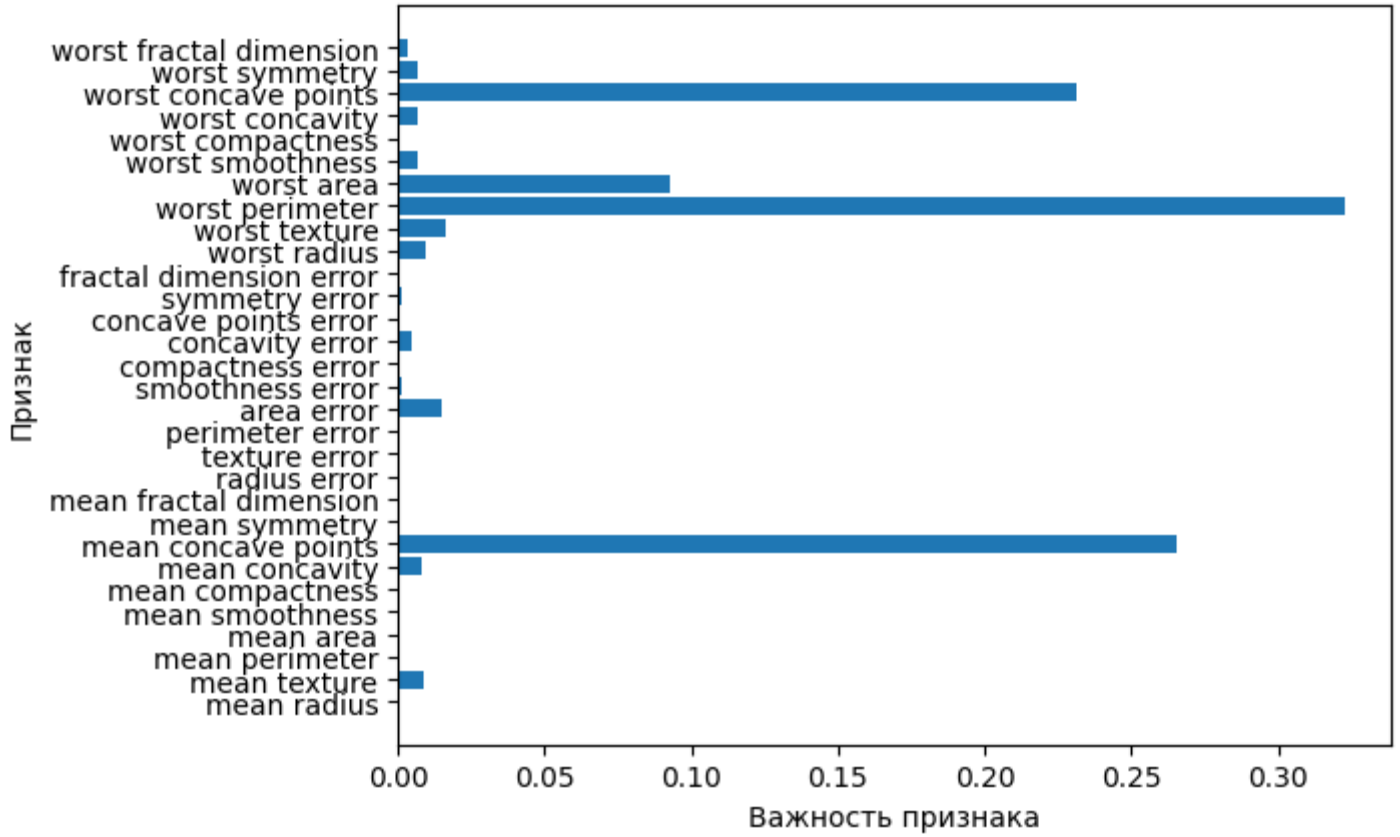
```
gbrt = GradientBoostingClassifier(random_state=0, max_depth=1)
gbrt.fit(X_train, y_train)

print("Правильность на обучающем наборе: {:.3f}".format(gbrt.score(X_train, y_train)))
print("Правильность на тестовом наборе: {:.3f}".format(gbrt.score(X_test, y_test)))
```

```
Правильность на обучающем наборе: 0.991
Правильность на тестовом наборе: 0.972
```

```
gbrt = GradientBoostingClassifier(random_state=0, learning_rate=0.01)
gbrt.fit(X_train, y_train)

print("Правильность на обучающем наборе: {:.3f}".format(gbrt.score(X_train, y_train)))
print("Правильность на тестовом наборе: {:.3f}".format(gbrt.score(X_test, y_test)))
```

```
Правильность на обучающем наборе: 0.988
Правильность на тестовом наборе: 0.958
```

```
gbrt = GradientBoostingClassifier(random_state=0, max_depth=1)
gbrt.fit(X_train, y_train)

def plot_feature_importances_cancer(model):
 n_features = cancer.data.shape[1]
 plt.barh(range(n_features), model.feature_importances_, align='center')
 plt.yticks(np.arange(n_features), cancer.feature_names)
 plt.xlabel("Важность признака")
 plt.ylabel("Признак")
plot_feature_importances_cancer(gbrt)
```



+/-

- Общепринятое мнение что он лучше случайного леса, в экономии времени на обучении – да
- Имеет параметры для котроля процессом, перечислить