Лабораторное занятие 10. Реализация алгоритма RandomForest на языке python с помощью рассмотрения точных задач.

10-зертханалық сабақ. Нақты есеп қарастыру арқылы RandomForest алгоритмін python тілінде жүзеге асыру

class sklearn.ensemble.RandomForestRegressor(

n_estimators — число деревьев в "лесу" (по дефолту – 10)

criterion — функция, которая измеряет качество разбиения ветки дерева (по дефолту — "mse" , так же можно выбрать "mae")

max_features — число признаков, по которым ищется разбиение. Вы можете указать конкретное число или процент признаков, либо выбрать из доступных значений: "auto" (все признаки), "sqrt", "log2". По дефолту стоит "auto".

max_depth — максимальная глубина дерева (по дефолту глубина не ограничена)

min_samples_split — минимальное количество объектов, необходимое для разделения внутреннего узла. Можно задать числом или процентом от общего числа объектов (по дефолту — 2)

min_samples_leaf — минимальное число объектов в листе. Можно задать числом или процентом от общего числа объектов (по дефолту — 1)

min_weight_fraction_leaf — минимальная взвешенная доля от общей суммы весов (всех входных объектов) должна быть в листе (по дефолту имеют одинаковый вес)

max_leaf_nodes — максимальное количество листьев (по дефолту нет ограничения)

min_impurity_split — порог для остановки наращивания дерева (по дефолту 1e-7)

bootstrap — применять ли бустрэп для построения дерева (по дефолту True)

oob_score — использовать ли out-of-bag объекты для оценки R^2 (по дефолту False)

n_jobs — количество ядер для построения модели и предсказаний (по дефолту 1, если поставить -1, то будут использоваться все ядра)

random_state — начальное значение для генерации случайных чисел (по дефолту его нет, если хотите воспроизводимые результаты, то нужно указать любое число типа int

verbose — вывод логов по построению деревьев (по дефолту 0)

warm_start — использует уже натренированую модель и добавляет деревьев в ансамбль (по дефолту False)

)

Пример.

Для задачи классификации все почти то же самое, мы приведем только те параметры, которыми RandomForestClassifier отличается от RandomForestRegressor

class sklearn.ensemble.RandomForestClassifier(

criterion — поскольку у нас теперь задача классификации, то по дефолту выбран критерий "gini" (можно выбрать "entropy")

class_weight — вес каждого класса (по дефолту все веса равны 1, но можно передать словарь с весами, либо явно указать "balanced", тогда веса классов будут равны их исходным частям в генеральной совокупности; также можно указать "balanced_subsample", тогда веса на каждой подвыборке будут меняться в зависимости от распределения классов на этой подвыборке.

)

Далее рассмотрим несколько параметров, на которые в первую очередь стоит обратить внимание при построении модели:

- n_estimators — число деревьев в "лесу"
- criterion — критерий для разбиения выборки в вершине
- max_features — число признаков, по которым ищется разбиение
- min_samples_leaf — минимальное число объектов в листе
- max_depth — максимальная глубина дерева

Рассмотрим применение случайного леса в реальной задаче

Для этого будем использовать пример с задачей оттока клиентов. Это задача классификации, поэтому будем использовать метрику accuracy для оценки качества модели. Для начала построим самый простой классификатор, который будет нашим бейслайном. Возьмем только числовые признаки для упрощения.

```
import pandas as pd

from sklearn.model_selection import cross_val_score, StratifiedKFold, GridSearchCV

from sklearn.metrics import accuracy_score

# Загружаем данные

df = pd.read_csv("../../data/telecom_churn.csv")

# Выбираем сначала только колонки с числовым типом данных

cols = []
```

```
for i in df.columns:
    if (df[i].dtype == "float64") or (df[i].dtype == 'int64'):
        cols.append(i)
# Разделяем на признаки и объекты
X, y = df[cols].copy(), np.asarray(df["Churn"],dtype='int8')
# Инициализируем страифицированную разбивку нашего датасета для валидации
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
# Инициализируем наш классификатор с дефолтными параметрами
rfc = RandomForestClassifier(random_state=42, n_jobs=-1, oob_score=True)
# Обучаем на тренировочном датасете
results = cross_val_score(rfc, X, y, cv=skf)
# Оцениваем долю верных ответов на тестовом датасете
print("CV accuracy score: {:.2f}%".format(results.mean()*100))
```

Улучшим результат

Код для построения кривых валидации по подбору количества деревьев:

```
# Инициализируем валидацию
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
# Создаем списки для сохранения точности на тренировочном и тестовом датасете
train_acc = []
test_acc = []
temp_train_acc = []
temp_test_acc = []
trees_grid = [5, 10, 15, 20, 30, 50, 75, 100]
# Обучаем на тренировочном датасете
for ntrees in trees_grid:
    rfc = RandomForestClassifier(n_estimators=ntrees, random_state=42, n_jobs=-1,
oob_score=True)
    temp_train_acc = []
    temp_test_acc = []
```

```
    for train_index, test_index in skf.split(X, y):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y[train_index], y[test_index]
        rfc.fit(X_train, y_train)
        temp_train_acc.append(rfc.score(X_train, y_train))
        temp_test_acc.append(rfc.score(X_test, y_test))
    train_acc.append(temp_train_acc)
    test_acc.append(temp_test_acc)
train_acc, test_acc = np.asarray(train_acc), np.asarray(test_acc)
print("Best accuracy on CV is {:.2f}% with {} trees".format(max(test_acc.mean(axis=1))*100,
                                                            trees_grid[np.argmax(test_acc.mean(axis=1))]))
```

Код для построения графика кривых валидации:

```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
%matplotlib inline
fig, ax = plt.subplots(figsize=(8, 4))
ax.plot(trees_grid, train_acc.mean(axis=1), alpha=0.5, color='blue', label='train')
ax.plot(trees_grid, test_acc.mean(axis=1), alpha=0.5, color='red', label='cv')
ax.fill_between(trees_grid, test_acc.mean(axis=1) - test_acc.std(axis=1), test_acc.mean(axis=1) +
test_acc.std(axis=1), color='#888888', alpha=0.4)
ax.fill_between(trees_grid, test_acc.mean(axis=1) - 2*test_acc.std(axis=1), test_acc.mean(axis=1) +
2*test_acc.std(axis=1), color='#888888', alpha=0.2)
ax.legend(loc='best')
ax.set_ylim([0.88,1.02])
ax.set_ylabel("Accuracy")
ax.set_xlabel("N_estimators")
```

Зафикцируем максимальную глубину дерева – max_depth. =100

```
# Создаем списки для сохранения точности на тренировочном и тестовом датасете
train_acc = []
test_acc = []
temp_train_acc = []
```

```
temp_test_acc = []

max_depth_grid = [3, 5, 7, 9, 11, 13, 15, 17, 20, 22, 24]

# Обучаем на тренировочном датасете

for max_depth in max_depth_grid:

    rfc = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1, oob_score=True,
max_depth=max_depth)

    temp_train_acc = []

    temp_test_acc = []

    for train_index, test_index in skf.split(X, y):

        X_train, X_test = X.iloc[train_index], X.iloc[test_index]

        y_train, y_test = y[train_index], y[test_index]

        rfc.fit(X_train, y_train)

        temp_train_acc.append(rfc.score(X_train, y_train))

        temp_test_acc.append(rfc.score(X_test, y_test))

    train_acc.append(temp_train_acc)

    test_acc.append(temp_test_acc)

train_acc, test_acc = np.asarray(train_acc), np.asarray(test_acc)

print("Best accuracy on CV is {:.2f}% with {}
max_depth".format(max(test_acc.mean(axis=1))*100,

                                        max_depth_grid[np.argmax(test_acc.mean(axis=1))]))

# Почтроим график кривых

fig, ax = plt.subplots(figsize=(8, 4))

ax.plot(max_depth_grid, train_acc.mean(axis=1), alpha=0.5, color='blue', label='train')

ax.plot(max_depth_grid, test_acc.mean(axis=1), alpha=0.5, color='red', label='cv')

ax.fill_between(max_depth_grid, test_acc.mean(axis=1) - test_acc.std(axis=1),
test_acc.mean(axis=1) + test_acc.std(axis=1), color='#888888', alpha=0.4)

ax.fill_between(max_depth_grid, test_acc.mean(axis=1) - 2*test_acc.std(axis=1),
test_acc.mean(axis=1) + 2*test_acc.std(axis=1), color='#888888', alpha=0.2)

ax.legend(loc='best')

ax.set_ylim([0.88,1.02])

ax.set_ylabel("Accuracy")
```

```
ax.set_xlabel("Max_depth")
```

Задание №1

Поменяйте еще две важные параметры min_samples_leaf и max_features и постройте график кривых валидации. Сделайте анализ.