Лабораторная работа

Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей.

Цель лабораторной работы: изучение способов подготовки выборки и подбора гиперпараметров на примере метода ближайших соседей

Задание:

- Выберите набор данных (датасет) для решения задачи классификации или регрессии.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, GridSearchCV,
RandomizedSearchCV, KFold, StratifiedKFold
import pandas as pd

iris = load_iris()

iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target

print(iris_df.head())
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)  \
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
4                5.0               3.6                1.4
0.2

   target
0       0
1       0
2       0
3       0
4       0
```

0 - ирис сетоса 1 - ирис версиколор 2 - ирис виргиника

- В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.

```
print(iris_df.isnull().sum())
```

```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
target               0
dtype: int64
```

- С использованием метода train_test_split разделите выборку на обучающую и тестовую.

```
X = iris_df.drop(columns=['target'])
y = iris_df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

print(f"Размер обучающей выборки: {X_train.shape}")
print(f"Размер тестовой выборки: {X_test.shape}")
```

```
Размер обучающей выборки: (120, 4)
Размер тестовой выборки: (30, 4)
```

- Обучите модель ближайших соседей для произвольно заданного гиперпараметра K. Оцените качество модели с помощью подходящих для задачи метрик.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

k = 50
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

# Оценка качества модели
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Точность модели: {accuracy}")
print("Отчет по классификации:")
print(report)
```

```
Точность модели: 0.9666666666666667
Отчет по классификации:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
```

```
           1       0.90      1.00      0.95         9
           2       1.00      0.91      0.95        11

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
```

- Произведите подбор гиперпараметра K с использованием GridSearchCV и RandomizedSearchCV и кросс-валидации, оцените качество оптимальной модели. Используйте не менее двух стратегий кросс-валидации.

```
knn = KNeighborsClassifier()
param_grid = {'n_neighbors': range(1, 50)}

kf = KFold(n_splits=5, shuffle=True, random_state=42)
grid_search = GridSearchCV(knn, param_grid, cv=kf, scoring='accuracy')
grid_search.fit(X_train, y_train)
print(f"Лучший параметр K (GridSearchCV с KFold):
{grid_search.best_params_}")
print(f"Лучшая точность (GridSearchCV с KFold):
{grid_search.best_score_}")
```

```
Лучший параметр K (GridSearchCV с KFold): {'n_neighbors': 12}
Лучшая точность (GridSearchCV с KFold): 0.9666666666666666
```

```
### KFold и GridSearchCV

**KFold** — это метод кросс-валидации, который делит данные на `k`
равных частей (фолдов). На каждой итерации одна из частей используется
как тестовая выборка, а остальные — как обучающая. Таким образом,
модель обучается и тестируется `k` раз, что позволяет получить более
надежную оценку качества модели. В данном случае используется `KFold`
с параметрами `n_splits=5`, `shuffle=True` и `random_state=42`, что
означает, что данные делятся на 5 фолдов с перемешиванием и
фиксированным состоянием генератора случайных чисел для
воспроизводимости.

**GridSearchCV** — это метод для подбора гиперпараметров модели. Он
перебирает все возможные комбинации значений гиперпараметров,
указанных в `param_grid`, и оценивает каждую комбинацию с
использованием кросс-валидации. В данном случае `GridSearchCV`
используется для подбора оптимального значения гиперпараметра
`n_neighbors` для модели `KNeighborsClassifier`. Кросс-валидация
выполняется с использованием `KFold`, а метрика оценки — `accuracy`.
Результатом работы `GridSearchCV` является лучшая модель с
оптимальными гиперпараметрами, а также оценка её качества.

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
random_search = RandomizedSearchCV(knn, param_grid, cv=skf,
scoring='accuracy', n_iter=10, random_state=42)
```

```
random_search.fit(X_train, y_train)

print(f"Лучший параметр K (RandomizedSearchCV с StratifiedKFold):
{random_search.best_params_}")
print(f"Лучшая точность (RandomizedSearchCV с StratifiedKFold):
{random_search.best_score_}")
```

```
Лучший параметр K (RandomizedSearchCV с StratifiedKFold):
{'n_neighbors': 14}
Лучшая точность (RandomizedSearchCV с StratifiedKFold):
0.9583333333333334
```

```
best_knn = grid_search.best_estimator_
y_pred = best_knn.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Точность оптимальной модели: {accuracy}")
print("Отчет по классификации:")
print(report)
```

```
Точность оптимальной модели: 1.0
Отчет по классификации:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```