Лаб 3

РТ5-61Б Слкуни Герман

Задание: Выберите набор данных (датасет) для решения задачи классификации или регрессии. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков. С использованием метода train_test_split разделите выборку на обучающую и тестовую. Обучите модель ближайших соседей для произвольно заданного гиперпараметра K. Оцените качество модели с помощью подходящих для задачи метрик. Произведите подбор гиперпараметра K с использованием GridSearchCV и RandomizedSearchCV и кросс-валидации, оцените качество оптимальной модели. Используйте не менее двух стратегий кросс-валидации. Сравните метрики качества исходной и оптимальной моделей.

Подключение библиотек

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV, Randomiz
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusior
```

Загрузка датасета

In [2]:
```
data = pd.read_csv('Cancer_Data.csv', sep=",")
data = data.dropna(axis=1)

X = data.drop('diagnosis', axis=1)
y = data['diagnosis']

X.head()
```

Out[2]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoot
0	842302	17.99	10.38	122.80	1001.0	
1	842517	20.57	17.77	132.90	1326.0	
2	84300903	19.69	21.25	130.00	1203.0	
3	84348301	11.42	20.38	77.58	386.1	
4	84358402	20.29	14.34	135.10	1297.0	

5 rows × 31 columns

In [9]:
```
print("Уникальные классы:", np.unique(y))
```

```
Уникальные классы: ['B' 'M']
```

Разделение на обучающую и тестовую выборки

In [4]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rar
```

Обучение KNN с произвольным K и оценка

In [5]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

# Оценка качества модели
print("Точность:", accuracy_score(y_test, y_pred))
print("\nКлассификационный отчёт:\n", classification_report(y_test, y_pred))
```

```
Точность: 0.695906432748538

Классификационный отчёт:
               precision    recall  f1-score   support

           B       0.69      0.93      0.79       107
           M       0.71      0.31      0.43        64

    accuracy                           0.70       171
   macro avg       0.70      0.62      0.61       171
weighted avg       0.70      0.70      0.66       171
```

Подбор гиперпараметров: GridSearchCV

In [6]:
```
param_grid = {'n_neighbors': np.arange(1, 31)}
cv1 = KFold(n_splits=5, shuffle=True, random_state=42)
grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=cv1, scori
grid_search.fit(X_train, y_train)
```

```
print("Лучший параметр K (GridSearchCV):", grid_search.best_params_)
print("Лучшая точность (GridSearchCV):", grid_search.best_score_)
```

```
Лучший параметр K (GridSearchCV): {'n_neighbors': np.int64(1)}
Лучшая точность (GridSearchCV): 0.7814240506329113
```

Подбор гиперпараметров: RandomizedSearchCV

In [7]:
```
cv2 = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
random_search = RandomizedSearchCV(KNeighborsClassifier(), param_distributic
random_search.fit(X_train, y_train)

print("Лучший параметр K (RandomizedSearchCV):", random_search.best_params_)
print("Лучшая точность (RandomizedSearchCV):", random_search.best_score_)
```

```
Лучший параметр K (RandomizedSearchCV): {'n_neighbors': np.int64(1)}
Лучшая точность (RandomizedSearchCV): 0.7990822784810125
```

Финальная модель с оптимальным K

In [8]:
```
best_k = grid_search.best_params_['n_neighbors']
final_model = KNeighborsClassifier(n_neighbors=best_k)
final_model.fit(X_train, y_train)
final_preds = final_model.predict(X_test)

print("\n=== Оценка оптимальной модели ===")
print("Точность (accuracy):", accuracy_score(y_test, final_preds))
print("\nКлассификационный отчёт:\n", classification_report(y_test, final_pr

# Сравнение точности
base_acc = accuracy_score(y_test, y_pred)
final_acc = accuracy_score(y_test, final_preds)

print(f"\nТочность модели с K=5: {base_acc:.4f}")
print(f"Точность модели с оптимальным K={best_k}: {final_acc:.4f}")
```

```
=== Оценка оптимальной модели ===
Точность (accuracy): 0.8011695906432749

Классификационный отчёт:
               precision    recall  f1-score   support

           B       0.81      0.90      0.85       107
           M       0.79      0.64      0.71        64

    accuracy                           0.80       171
   macro avg       0.80      0.77      0.78       171
weighted avg       0.80      0.80      0.80       171


Точность модели с K=5: 0.6959
Точность модели с оптимальным K=1: 0.8012
```

In [ ]:
```
```

In [ ]:

This notebook was converted with convert.ploomber.io