Московский государственный технический

университет им. Н.Э. Баумана

Факультет Радиотехнический

Кафедра РТ5

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №3

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

Выполнил:

студент группы РТ5-61Б:

Бабасанова Н. С.

Руководитель:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Москва, 2025г.

Задание

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

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

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

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

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

6. Сравните метрики качества исходной и оптимальной моделей

Текст программы

```
import sklearn
import numpy as np
import pandas as pd

from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
```

Подготовка данных

```
data_df = pd.DataFrame(data = data.data,
                       columns = data.feature_names)
data_df
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	25.380
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	24.990
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	23.570
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	14.910
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	22.540
...	...	...	...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	25.450
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	23.690
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	18.980
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	25.740
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	9.456

569 rows × 30 columns

```
data_df["target"] = data.target
data_df
```

Стандартизация

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

scaler.fit(data_df.drop('target', axis=1))
scaled_features = scaler.transform(data_df.drop('target',
                                                axis=1))

df_feat = pd.DataFrame(scaled_features,
                       columns=data_df.columns[:-1])
df_feat.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.532475	2.217515	2.255747
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.548144	0.001392	-0.868652
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.037231	0.939685	-0.398008
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897	1.451707	2.867383	4.910919
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011	1.428493	-0.009560	-0.562450

5 rows × 30 columns

Прогноз с произвольным k

```
from sklearn.metrics import classification_report,\
    confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.model_selection import train_test_split
X_train, X_test,\
    y_train, y_test = train_test_split(scaled_features,
                                       data_df['target'],
                                       test_size=0.50, random_state=1)

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
pred = knn.predict(X_test)

print("Confusion матрица:\n", confusion_matrix(y_test, pred)) # в scikit learn
истинные - строки, предсказанные - столбцы

report = classification_report(y_test, pred, output_dict=True)
# print("\n", classification_report(y_test, pred))
for label in sorted(report.keys()):
    if label not in ["accuracy", "macro avg", "weighted avg"]:
        print(f"Класс {label}: {report[label]}")
```

```
Confusion матрица:
 [[ 93  10]
 [  7 175]]
Класс 0: {'precision': 0.93, 'recall': 0.9029126213592233, 'f1-score': 0.916256157635468, 'support': 103.0}
Класс 1: {'precision': 0.9459459459459459, 'recall': 0.9615384615384616, 'f1-score': 0.9536784741144414, 'support': 182.0}
```

Подбор k

GridSearchCV

```
from sklearn import svm
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_neighbors': list(range(1, 50)),
}

knn_cv = GridSearchCV(
            knn,
            param_grid,
            cv=5,
            scoring='accuracy'
        )
knn_cv.fit(X_train, y_train)
print(f"Лучший параметр: {knn_cv.best_params_}")
print(f"Лучшая кросс-валидационная accuracy: {knn_cv.best_score_:.2f}")
```

```
Лучший параметр: {'n_neighbors': 3}
Лучшая кросс-валидационная accuracy: 0.99
```

RandomSearchCV

```
from sklearn.model_selection import RandomizedSearchCV


knn = KNeighborsClassifier()


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


random_search = RandomizedSearchCV(
    knn,
    param_grid,
    n_iter=10,  # Limit the number of iterations (10 combinations)
    cv=5,  # 5-fold cross-validation
    scoring='accuracy',  # You can use 'precision', 'recall', or 'f1' here for
different use cases
    random_state=42
)

random_search.fit(X_train, y_train)

# Find the best parameters and score
print(f"Лучший параметр: {random_search.best_params_}")
print(f"Best cross-validated accuracy: {random_search.best_score_:.2f}")

# Evaluate the best model on the test set
best_knn = random_search.best_estimator_
test_accuracy = best_knn.score(X_test, y_test)

print(f"Лучший accuracy тестового сета: {test_accuracy:.2f}")
```

```
Лучший параметр: {'n_neighbors': 14}
Best cross-validated accuracy: 0.96
Лучший accuracy тестового сета: 0.93
```

Кросс-валидация

K-Fold

```
from sklearn.model_selection import StratifiedKFold, RepeatedStratifiedKFold,
ShuffleSplit, cross_val_score
```

```
X = df_feat
y = data_df['target']
kf = StratifiedKFold(n_splits=3)
for train, test in kf.split(X, y):
    print("%s %s" % (train, test))
kf = StratifiedKFold(n_splits=5)
scores = cross_val_score(KNeighborsClassifier(n_neighbors=5),
                         X, y, scoring='accuracy',
                         cv=kf)
scores
```

```
array([0.96491228, 0.95614035, 0.98245614, 0.95614035, 0.96460177])
```

```
y = data_df['target']
kf = StratifiedKFold(n_splits=5)
scores = cross_val_score(KNeighborsClassifier(n_neighbors=3),
                         X, y, scoring='accuracy',
                         cv=kf)
scores
```

```
array([0.97368421, 0.95614035, 0.98245614, 0.94736842, 0.92920354])
```

```
y = data_df['target']
kf = StratifiedKFold(n_splits=5)
scores = cross_val_score(KNeighborsClassifier(n_neighbors=14),
                         X, y, scoring='accuracy',
                         cv=kf)
scores
```

```
array([0.96491228, 0.93859649, 0.99122807, 0.95614035, 0.98230088])
```

Repeated K-Fold

```
kf = RepeatedStratifiedKFold(n_splits=3, n_repeats=2)
for train, test in kf.split(X, y):
    print("%s %s" % (train, test))
scores = cross_val_score(KNeighborsClassifier(n_neighbors=5),
                         X, y, scoring='accuracy',
                         cv=kf)
scores
```

```
array([0.96842105, 0.96315789, 0.96825397, 0.96842105, 0.96315789,
       0.96825397])
```

```
scores = cross_val_score(KNeighborsClassifier(n_neighbors=3),
                         X, y, scoring='accuracy',
                         cv=kf)
scores
```

```
array([0.96315789, 0.95789474, 0.97354497, 0.97894737, 0.96315789,
       0.96296296])
```

```
scores = cross_val_score(KNeighborsClassifier(n_neighbors=14),
                         X, y, scoring='accuracy',
                         cv=kf)
scores
```

```
array([0.96315789, 0.95789474, 0.96296296, 0.97894737, 0.93684211,
       0.96825397])
```

Shuffle and Split

```
kf = ShuffleSplit(n_splits=5, test_size=0.25)
for train, test in kf.split(X):
    print("%s %s" % (train, test))
scores = cross_val_score(KNeighborsClassifier(n_neighbors=5),
                         X, y, scoring='accuracy',
                         cv=kf)
scores
```

```
array([0.96842105, 0.95789474, 0.94708995, 0.95789474, 0.95789474,
       0.97354497])
```

```
scores = cross_val_score(KNeighborsClassifier(n_neighbors=3),
                         X, y, scoring='accuracy',
                         cv=kf)
scores
```

```
array([0.97368421, 0.97894737, 0.95767196, 0.96842105, 0.95789474,
       0.96825397])
```

```
scores = cross_val_score(KNeighborsClassifier(n_neighbors=14),
                         X, y, scoring='accuracy',
                         cv=kf)
scores
```

```
array([0.97368421, 0.93157895, 0.96825397, 0.95789474, 0.95789474,
       0.96296296])
```