k近邻实验2

班级： 21计科4班 姓名：陈昊天

一、实验原理

本次实验利用 K-最近邻算法，在 Pima 印第安人糖尿病数据集上实现一个糖尿病预测任务。

k-NN 的核心思想是“近朱者赤，近墨者黑”。对于一个未知类别的数据点，算法会计算它与训练数据集中所有样本点的距离。选取与该测试样本距离最近的 k 个训练样本。根据这 k 个邻居的类别标签，通过多数投票的方式，将出现次数最多的类别赋给该测试样本，作为其预测类别。

二、实验代码与运行结果

数据集

	A	B	C	D	E	F	G	H	I
1	Pregnancies	Glucose	BloodPressu	SkinThickn	Insulin	BMI	DiabetesPe	Age	Outcome
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1
7	5	116	74	0	0	25.6	0.201	30	0
8	3	78	50	32	88	31	0.248	26	1
9	10	115	0	0	0	35.3	0.134	29	0
10	2	197	70	45	543	30.5	0.158	53	1
11	8	125	96	0	0	0	0.232	54	1
12	4	110	92	0	0	37.6	0.191	30	0
13	10	168	74	0	0	38	0.537	34	1
14	10	139	80	0	0	27.1	1.441	57	0
15	1	189	60	23	846	30.1	0.398	59	1
16	5	166	72	19	175	25.8	0.587	51	1
17	7	100	0	0	0	30	0.484	32	1
18	0	118	84	47	230	45.8	0.551	31	1
19	7	107	74	0	0	29.6	0.254	31	1
20	1	103	30	38	83	43.3	0.183	33	0
21	1	115	70	30	96	34.6	0.529	32	1
22	3	126	88	41	235	39.3	0.704	27	0
23	8	99	84	0	0	35.4	0.388	50	0
24	7	196	90	0	0	39.8	0.451	41	1
25	9	119	80	35	0	29	0.263	29	1
26	11	143	94	33	146	36.6	0.254	51	1
27	10	125	70	26	115	31.1	0.205	41	1
28	7	147	76	0	0	39.4	0.257	43	1
29	1	97	66	15	140	23.2	0.487	22	0

实验代码

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.impute import SimpleImputer

diabetes_df = pd.read_csv("diabetes.csv")

print("\n数据信息:")
diabetes_df.info()
```

```
X = diabetes_df.drop("Outcome", axis=1)
y = diabetes_df["Outcome"]
feature_names = X.columns.tolist()
target_names = ["No Diabetes", "Diabetes"]

cols_with_zeros_as_missing = [
    "Glucose",
    "BloodPressure",
    "SkinThickness",
    "Insulin",
    "BMI",
]

X[cols_with_zeros_as_missing] =
X[cols_with_zeros_as_missing].replace(0, np.nan)

print("\n替换 0 值为 NaN 后，每列的 NaN 数量:")
print(X.isnull().sum())

imputer = SimpleImputer(strategy="median")
X_imputed = imputer.fit_transform(X)

X = pd.DataFrame(X_imputed, columns=feature_names)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

k = 11
knn = KNeighborsClassifier(n_neighbors=k, weights="distance")  # 尝试
加权
knn.fit(X_train_scaled, y_train)

y_pred = knn.predict(X_test_scaled)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred,
target_names=target_names)

print("\n")
```

```
print(f"准确率: {accuracy:.4f}")
print("\n混淆矩阵:")
print(conf_matrix)
print("\n分类报告:")
print(class_report)
```

实验结果

准确率: 0.7532

混淆矩阵:

[[128 22]

 [35 46]]

分类报告:

	precision	recall	f1-score	support
No Diabetes	0.79	0.85	0.82	150
Diabetes	0.68	0.57	0.62	81
accuracy			0.75	231
macro avg	0.73	0.71	0.72	231
weighted avg	0.75	0.75	0.75	231

```
(ML) (base) nanmener@Haotians-MacBook-Pro 作业3 % /Applic
anmener/Sync/zstu/机器学习项目实践/作业3/T2-submit.py

数据信息:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB

替换 0 值为 NaN 后，每列的 NaN 数量:
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
dtype: int64


准确率: 0.7532

混淆矩阵:
[[128  22]
 [ 35  46]]

分类报告:
              precision    recall  f1-score   support

 No Diabetes       0.79      0.85      0.82       150
    Diabetes       0.68      0.57      0.62        81

    accuracy                           0.75       231
   macro avg       0.73      0.71      0.72       231
weighted avg       0.75      0.75      0.75       231

(ML) (base) nanmener@Haotians-MacBook-Pro 作业3 %
```

三、结果分析与心得体会

结果分析

模型在测试集上的总体预测正确率约为 75%。

正确预测“无糖尿病”： 128 例

错误预测为“有糖尿病”但实际无: 22 例

错误预测为“无糖尿病”但实际有：35 例

正确预测“有糖尿病”：46 例

说明模型在识别“无糖尿病”的病例上表现较好，但在识别“有糖尿病”的病例上存在较多的漏诊。

无糖尿病: 精确率 0.79，召回率 0.85，F1 分数 0.82。模型识别此类别的能力较强，大部分无糖尿病患者被正确识别。

有糖尿病: 精确率 0.68，召回率 0.57，F1 分数 0.62。模型识别此类别的能力相对较弱，只有 57% 的糖尿病患者被模型成功检出，漏诊率较高。模型预测为糖尿病的病例中，有 68% 是准确的。

整体准确率尚可，但对少数类的识别能力有待提高。这可能与数据集本身的不平衡性以及当前模型参数k=11的选择有关。

如何提升分类效果?

为提升分类效果，可以进行:

（1）精细化超参数调优，找到最优的 K 值、距离度量和投票权重组合。在实验中我使用 K=11 和 weights='distance'，可以使用 scikit-learn 的 GridSearchCV或 RandomizedSearchCV结合交叉验证来系统地搜索最佳参数。

（2）处理类别不平衡：提高模型对少数类的识别能力。使用 imbalanced-learn`库中的 SMOTE，通过在少数类样本附近合成新的、相似的样本增加少数类的数量。

（3）特征工程：创建可能包含更多预测信息的新特征。

心得体会

通过本次实验，我对 kNN 算法及其在疾病诊断任务中的应用有更深入的理解。kNN 算法原理相对简单直观，易于理解和实现。借助 scikit-learn 库，可以方便地构建和评估模型。对于 kNN 这种依赖距离度量的算法，特征标准化是必不可少的步骤。Pima 数据集中特殊的0值代表缺失信息，必须进行处理，忽略这些值会严重影响模型结果的可靠性。

在类别不平衡的情况下仅看整体准确率是不够的，混淆矩阵和分类报告能提供更全面的视角，帮助我们理解模型在不同类别上的具体表现和存在的偏见。本次实验中对“Diabetes”类别较低的召回率就是一个很好的例子。kNN 的性能很大程度上取决于 K 值、距离度量和是否使用加权等超参数的选择。实验中初步选择的 K=11 未必是最优的，后续可以通过网格搜索等方法结合交叉验证来系统地寻找最佳参数组合，以提升模型性能。