作业四：离群点分析与异常检测

白思萌 3120190975

数据集 1：skin_benchmarks

一、数据分析

skin_benchmark 共有 1500 个 csv 文件，每个文件中共包含 6000 条数据项。此处以 skin_benchmark_0001 为例，进行数据分析。

首先，将数据中的标量数据进行五数概括处理。

```
df = pd.read_csv("skin_benchmark_0001.csv")
print(df.describe())
```

结果如下。

	original.label	diff.score	R	G	B
count	6000.000000	6000.000000	6000.000000	6000.000000	6000.000000
mean	1.797167	0.060019	-0.001530	-0.002480	0.007713
std	0.402143	0.125721	0.998430	1.005298	0.995998
min	1.000000	0.000024	-2.008901	-2.210622	-1.697540
25%	2.000000	0.000873	-0.916631	-0.759199	-0.650162
50%	2.000000	0.002967	0.223828	0.358563	0.066465
75%	2.000000	0.024870	0.802089	0.742272	0.562591
max	2.000000	0.697589	2.087113	2.043547	1.816688

其中 R、G、B 三个属性作为离群点检测开展对象。取前 5 行 R、G、B 数据进行展示。

	R	G	B
0	0.035294	0.023529	0.031373
1	0.584314	0.737255	0.980392
2	0.250980	0.262745	0.086275
3	0.215686	0.392157	0.592157
4	0.254902	0.266667	0.090196

二、离群点检测

离群点检测共采用 7 个模型进行检测，分别为：

"Angle-based Outlier Detector(ABOD)"

"Cluster-based Local Outiler Factor (CBLOF)"

"Feature Bagging"

"Histogram-base Outlier Detection(HBOS)"

"Isolation Forest"

"KNN"

"Average KNN"

```
# 定义7个后续会使用的离群点检测模型
classifiers = {
    "Angle-based Outlier Detector(ABOD)": ABOD(contamination=outliers_fraction),
    "Cluster-based Local Outiler Factor (CBLOF)": CBLOF(contamination=outliers_fraction,check_estimator=False,random_state=random_state),
    "Feature Bagging": FeatureBagging(LOF(n_neighbors=35),contamination=outliers_fraction,check_estimator=False,random_state=random_state),
    "Histogram-base Outlier Detection(HBOS)": HBOS(contamination=outliers_fraction),
    "Isolation Forest":IForest(contamination=outliers_fraction,random_state=random_state),
    "KNN": KNN(contamination=outliers_fraction),
    "Average KNN":KNN(method='mean',contamination=outliers_fraction)
}
```

首先，将 csv 文件进行读取，并计算数据集中离群点占所有点的比例（anomaly/anomaly +nominal），并将此比例作为模型输入。

若比例大于 0.5，模型不可正常运行，因此若大于 0.5 则舍弃此 benchmark，不作为后期参考。

```
t1=df['ground.truth'].value_counts(normalize=True)
t2=df['ground.truth'].value_counts(normalize=False)

x1 = df['R'].values.reshape(-1,1)
x2 = df['G'].values.reshape(-1,1)
x3 = df['B'].values.reshape(-1,1)
x = np.concatenate((x1,x2,x3),axis=1)
# 设置离群点数据
random_state = np.random.RandomState(42)
outliers_fraction = t1["anomaly"]
outliers = t2["anomaly"]
print("benchmark_",'{0:04}'.format(i),"的离群点共",outliers,"个，占比为",outliers_fraction,"%")
mytxt = open('out_skin.txt', mode='a', encoding='utf-8')
print("benchmark_",'{0:04}'.format(i),"的离群点共", outliers, "个，占比为", outliers_fraction, "%",file=mytxt)
mytxt.close()
if (outliers_fraction > 0.5):
    mytxt = open('out_skin.txt', mode='a', encoding='utf-8')
    print("离群点占比过大，放弃此benchmark", file=mytxt)
    print("\n", file=mytxt)
    mytxt.close()
    continue
```

之后将所有数据以及离群点比例输入每个模型中，训练后对所有数据进行预测是否为离群点，并将预测的离群点数量进行统计，并再次计算预测离群点占所有点的比例。

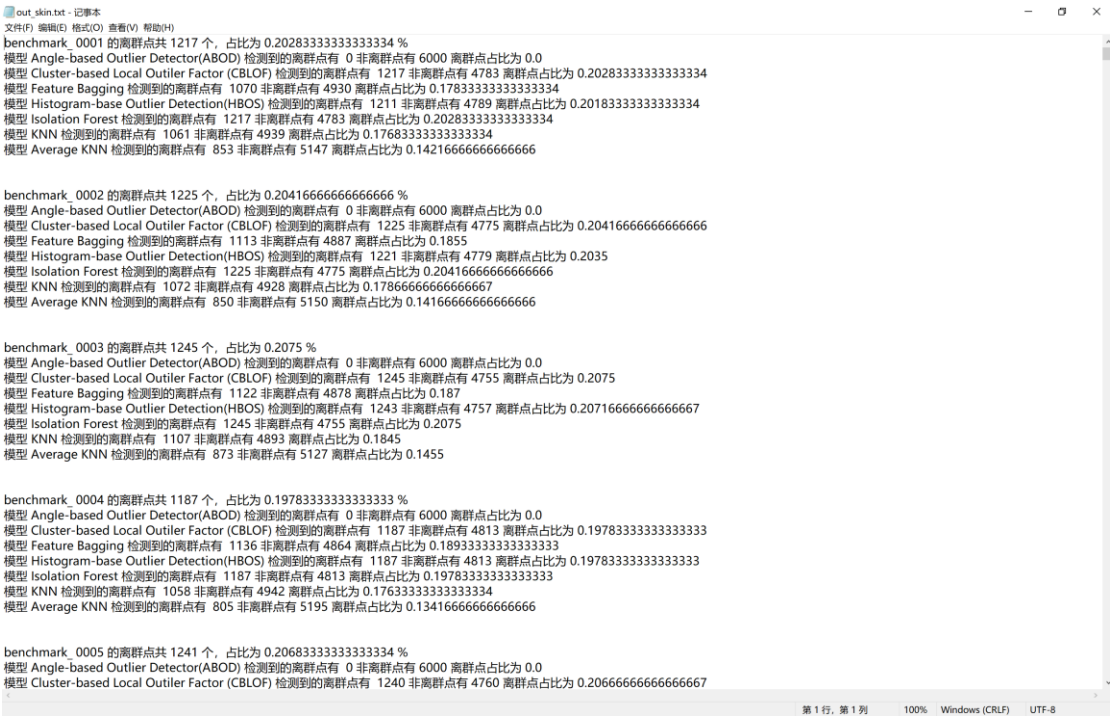
```
#逐一 比较模型
xx,yy,zz = np.meshgrid(np.linspace(0,1,200),np.linspace(0,1,200),np.linspace(0,1,200))
for i,(clf_name,clf) in enumerate(classifiers.items()):
    clf.fit(x)
    # 预测利群得分
    scores_pred = clf.decision_function(x)*-1
    # 预测数据点是否为 离群点
    y_pred = clf.predict(x)
    n_inliers = len(y_pred)-np.count_nonzero(y_pred)
    n_outliers = np.count_nonzero(y_pred==1)
    plt.figure(figsize=(10,10))
    percent = n_outliers / len(df.index)
    print("模型",clf_name,"检测到的离群点有 ",n_outliers,"非离群点有",n_inliers,"离群点占比为",percent)
    mytxt = open('out_skin.txt', mode='a', encoding='utf-8')
    print("模型",clf_name,"检测到的离群点有 ",n_outliers,"非离群点有",n_inliers,"离群点占比为",percent,file=mytxt)
    mytxt.close()
```

首先对 benchmark_0001 做尝试，结果如下。

benchmark_ 0001 的离群点共 1217 个，占比为 0.20283333333333334 %
模型 Angle-based Outlier Detector(ABOD) 检测到的离群点有 0 非离群点有 6000 离群点占比为 0.0
模型 Cluster-based Local Outiler Factor (CBLOF) 检测到的离群点有 1217 非离群点有 4783 离群点占比为 0.20283333333333334
模型 Feature Bagging 检测到的离群点有 1070 非离群点有 4930 离群点占比为 0.17833333333333334
模型 Histogram-base Outlier Detection(HBOS) 检测到的离群点有 1211 非离群点有 4789 离群点占比为 0.20183333333333334
模型 Isolation Forest 检测到的离群点有 1217 非离群点有 4783 离群点占比为 0.20283333333333334
模型 KNN 检测到的离群点有 1061 非离群点有 4939 离群点占比为 0.17683333333333334
模型 Average KNN 检测到的离群点有 853 非离群点有 5147 离群点占比为 0.14216666666666666

从结果中可以看出，benchmark_0001 中原本有 1217 个离群点。7 个模型预测的离群点数量均不同，其中 Cluster-based Local Outiler Factor (CBLOF)和 Isolation Forest 两个模型预测的结果较为准确，占比与原始数据集中离群点占比一致。

为得到更为准确的结果，循环遍历所有的 benchmark，最终结果输出在 out_skin.txt 中，如下图所示。



经对所有 csv 结果的输出的统计，可以认为在 skin_benchmarks 这个数据集上，Cluster-based Local Outiler Factor (CBLOF)和 Isolation Forest 两个模型预测的结果最为准确，Histogram-base Outlier Detection(HBOS)的预测效果也不错。

数据集 2：pageb_benchmarks

一、数据分析

pageb_benchmark 共有 940 个 csv 文件，每个文件中共包含 4423 条数据项。此处以 pageb_benchmark_0001 为例，进行数据分析。

首先，将数据中的标量数据进行五数概括处理。

```
df = pd.read_csv("pageb_benchmark_0001.csv")
print(df.describe())
```

结果如下。

	original.label	diff.score	V	V.1	V.2	V.3	V.4	V.5	V.6	V.7	V.8	V.9
count	4423.000000	4423.000000	4423.000000	4423.000000	4423.000000	4423.000000	4423.000000	4423.000000	4423.000000	4423.000000	4423.000000	4423.000000
mean	1.203256	0.046482	0.008358	0.006594	0.009978	0.009404	0.006491	0.000366	0.003668	0.007343	0.007729	0.006958
std	0.715259	0.109165	1.084678	1.014646	1.091699	1.014434	1.007222	1.005425	1.104857	1.060102	1.053958	1.038777
min	1.000000	0.000553	-0.499628	-0.772027	-0.245682	-0.447730	-1.781324	-4.236770	-0.075555	-0.282549	-0.390171	-0.631546
25%	1.000000	0.010451	-0.183182	-0.632559	-0.224030	-0.378748	-0.605560	-0.621422	-0.066869	-0.254997	-0.344463	-0.535914
50%	1.000000	0.021841	-0.130441	-0.423357	-0.181138	-0.281040	-0.172384	0.111023	-0.059921	-0.204616	-0.262082	-0.350627
75%	1.000000	0.046635	-0.024959	0.151948	-0.045038	0.000782	0.319862	0.831749	-0.046169	-0.062921	-0.014673	0.109601
max	5.000000	0.988245	41.851433	4.039615	29.445967	17.041770	3.551806	1.259497	71.639416	25.702762	24.125319	18.560561

其中 V、V.1、V.2、V.3、V.4、V.5、V.6、V.7、V.8、V.9 十个属性作为离群点检测开展对象。取前 5 行 V、V.1、V.2、V.3、V.4、V.5、V.6、V.7、V.8、V.9 数据进行展示。

	V	V.1	V.2	V.3	V.4	V.5	V.6	V.7	V.8	V.9
0	0.011208	0.010870	0.000438	0.001291	0.292194	0.985075	0.000155	0.000485	0.001344	0.003737
1	0.001245	0.038043	0.000257	0.020471	0.688819	0.855011	0.002927	0.000727	0.000672	0.000311
2	0.007472	0.081522	0.002188	0.012224	0.265823	0.960554	0.000319	0.002757	0.006569	0.011523
3	0.000000	0.028986	0.000069	0.031645	0.752110	1.000000	0.001110	0.000182	0.000217	0.000311
4	0.011208	0.179348	0.006897	0.018609	0.168776	0.682303	0.000091	0.006210	0.015067	0.045157

二、离群点检测

离群点检测共采用 7 个模型进行检测，分别为：

"Angle-based Outlier Detector(ABOD)"

"Cluster-based Local Outiler Factor (CBLOF)"

"Feature Bagging"

"Histogram-base Outlier Detection(HBOS)"

"Isolation Forest"

"KNN"

"Average KNN"

```
# 定义7个后续会使用的离群点检测模型
classifiers = {
    "Angle-based Outlier Detector(ABOD)": ABOD(contamination=outliers_fraction),
    "Cluster-based Local Outiler Factor (CBLOF)": CBLOF(contamination=outliers_fraction,check_estimator=False,random_state=random_state),
    "Feature Bagging": FeatureBagging(LOF(n_neighbors=35),contamination=outliers_fraction,check_estimator=False,random_state=random_state),
    "Histogram-base Outlier Detection(HBOS)": HBOS(contamination=outliers_fraction),
    "Isolation Forest":IForest(contamination=outliers_fraction,random_state=random_state),
    "KNN": KNN(contamination=outliers_fraction),
    "Average KNN":KNN(method='mean',contamination=outliers_fraction)
}
```

首先，将 csv 文件进行读取，并计算数据集中离群点占所有点的比例（anomaly/ anomaly +nominal），并将此比例作为模型输入。

若比例大于 0.5，模型不可正常运行，因此若大于 0.5 则舍弃此 benchmark，不作为后期参考。

```
t1=df['ground.truth'].value_counts(normalize=True)
t2=df['ground.truth'].value_counts(normalize=False)
x1 = df['V'].values.reshape(-1,1)
x2 = df['V.1'].values.reshape(-1,1)
x3 = df['V.2'].values.reshape(-1,1)
x4 = df['V.3'].values.reshape(-1, 1)
x5 = df['V.4'].values.reshape(-1, 1)
x6 = df['V.5'].values.reshape(-1, 1)
x7 = df['V.6'].values.reshape(-1, 1)
x8 = df['V.7'].values.reshape(-1, 1)
x9 = df['V.8'].values.reshape(-1, 1)
x10 = df['V.9'].values.reshape(-1, 1)
x = np.concatenate((x1,x2,x3,x4,x5,x6,x7,x8,x9,x10),axis=1)
# 设置离群点数据
random_state = np.random.RandomState(42)
outliers_fraction = t1["anomaly"]
outliers = t2["anomaly"]
print("benchmark_",'{0:04}'.format(i),"的离群点共",outliers,"个，占比为",outliers_fraction,"%")
mytxt = open('out_skin.txt', mode='a', encoding='utf-8')
print("benchmark_",'{0:04}'.format(i),"的离群点共", outliers, "个，占比为", outliers_fraction, "%",file=mytxt)
mytxt.close()
if (outliers_fraction > 0.5):
    mytxt = open('out_pageb.txt', mode='a', encoding='utf-8')
    print("离群点占比过大，放弃此benchmark", file=mytxt)
    print("\n", file=mytxt)
    mytxt.close()
    continue
```

之后将所有数据以及离群点比例输入每个模型中，训练后对所有数据进行预测是否为离群点，并将预测的离群点数量进行统计，并再次计算预测离群点占所有点的比例。

```
#逐一 比较模型
for i,(clf_name,clf) in enumerate(classifiers.items()):
    clf.fit(x)
    # 预测利群得分
    scores_pred = clf.decision_function(x)*-1
    # 预测数据点是否为 离群点
    y_pred = clf.predict(x)
    n_inliers = len(y_pred)-np.count_nonzero(y_pred)
    n_outliers = np.count_nonzero(y_pred==1)
    plt.figure(figsize=(10,10))
    percent = n_outliers / len(df.index)
    print("模型",clf_name,"检测到的离群点有 ",n_outliers,"非离群点有",n_inliers,"离群点占比为",percent)
    mytxt = open('out_pageb.txt', mode='a', encoding='utf-8')
    print("模型",clf_name,"检测到的离群点有 ",n_outliers,"非离群点有",n_inliers,"离群点占比为",percent,file=mytxt)
    mytxt.close()
```

首先对 benchmark_0001 做尝试，结果如下。

```
benchmark_ 0001 的离群点共 465 个，占比为 0.10513226316979425 %
模型 Angle-based Outlier Detector(ABOD) 检测到的离群点有  0 非离群点有 4423 离群点占比为 0.0
模型 Cluster-based Local Outiler Factor (CBLOF) 检测到的离群点有  465 非离群点有 3958 离群点占比为 0.10513226316979425
模型 Feature Bagging 检测到的离群点有  443 非离群点有 3980 离群点占比为 0.10015826362197604
模型 Histogram-base Outlier Detection(HBOS) 检测到的离群点有  465 非离群点有 3958 离群点占比为 0.10513226316979425
模型 Isolation Forest 检测到的离群点有  465 非离群点有 3958 离群点占比为 0.10513226316979425
模型 KNN 检测到的离群点有  418 非离群点有 4005 离群点占比为 0.09450599140854624
模型 Average KNN 检测到的离群点有  292 非离群点有 4131 离群点占比为 0.06601853945286006
```

从结果中可以看出，benchmark_0001 中原本有 465 个离群点。7 个模型预测的离群点数量均不同，其中 Cluster-based Local Outiler Factor (CBLOF)和 Isolation Forest 两个模型预测的结果较为准确，占比与原始数据集中离群点占比一致。

为得到更为准确的结果，循环遍历所有的 benchmark，最终结果输出在 out_skin.txt 中，如下图所示。

经对所有 csv 结果的输出的统计，可以认为在 pageb_benchmarks 这个数据集上，Cluster-based Local Outiler Factor (CBLOF)和 Isolation Forest 两个模型预测的结果最为准确，Histogram-base Outlier Detection(HBOS)的预测效果也不错。

模型对比分析

对上面所提到的 7 个模型进行对比分析。

```
classifiers = {
    'Angle-based Outlier Detector (ABOD)':
        ABOD(contamination=outliers_fraction),
    'Cluster-based Local Outlier Factor (CBLOF)':
        CBLOF(contamination=outliers_fraction,
              check_estimator=False, random_state=random_state),
    'Feature Bagging':
        FeatureBagging(LOF(n_neighbors=35),
                       contamination=outliers_fraction,
                       random_state=random_state),
    'Histogram-base Outlier Detection (HBOS)': HBOS(
        contamination=outliers_fraction),
    'Isolation Forest': IForest(contamination=outliers_fraction,
                                random_state=random_state),
    'K Nearest Neighbors (KNN)': KNN(
        contamination=outliers_fraction),
    'Average KNN': KNN(method='mean',
                       contamination=outliers_fraction),
}
```

首先，设置一组个数为 200 的样本作为数据，其中 25%为离群点。

```
n_samples = 200
outliers_fraction = 0.25
clusters_separation = [0]
```

分别采用 7 个模型进行预测并输出结果。

```
# Fit the models with the generated data and
# compare model performances
for i, offset in enumerate(clusters_separation):
    np.random.seed(42)
    # Data generation
    X1 = 0.3 * np.random.randn(n_inliers // 2, 2) - offset
    X2 = 0.3 * np.random.randn(n_inliers // 2, 2) + offset
    X = np.r_[X1, X2]
    # Add outliers
    X = np.r_[X, np.random.uniform(low=-6, high=6, size=(n_outliers, 2))]

    # Fit the model
    plt.figure(figsize=(15, 12))
    for i, (clf_name, clf) in enumerate(classifiers.items()):
        print()
        print(i + 1, 'fitting', clf_name)
        # fit the data and tag outliers
        clf.fit(X)
        scores_pred = clf.decision_function(X) * -1
        y_pred = clf.predict(X)
        threshold = percentile(scores_pred, 100 * outliers_fraction)
        n_errors = (y_pred != ground_truth).sum()
```

经绘制离群点图，结果如下。