[A3] Classification with Reject Option

2021312882 이준서

## 1. 데이터셋 설명 (data point 개수, feature 개수, label 분포)

(1) 차량의 2D 실루엣을 이용해 차종을 4가지로 레이블로 구분하는 데 사용할 수 있는 데이터셋입니다.

(2) data point의 개수는 845개이며, feature는 18개, label은 ‘opel’,‘saab’,‘bus’,‘van’ 4가지가 각각 212개/217개/217개/199개 존재하고 있습니다.

## 2. 인공신경망 학습을 위한 최적의 데이터 전처리 방법 및 하이퍼파라미터 설정 제시

(1) 데이터 전처리 - (a) 레이블이 ‘204’로 잘못 기입되어 있는 데이터 포인트 하나 존재, 해당 값 삭제,

- (b) feature값들이 모두 정수형이므로, StandardScalar를 이용해 스케일링,
- (c) 레이블을 원핫 인코딩하여 변환했습니다.

(2) 최적의 하이퍼파라미터 설정: **solver= ‘adam’,‘hidden_layer_sizes’=(32,), alpha=0.1, test_accuracy =0.980769, test_accuracy = 0.863905 (max_iter=1500)**

(a) 데이터 포인트의 개수가 1000개 이하로 적었기 때문에, solver가 lbfgs인 경우와 비교했습니다.

(a)-2 lbfgs의 경우, 오버피팅 경향이 강해 alpha의 범위를 [0.01, 0.1, 1.0]로 설정했음에도 오버피팅의 경향이 강했습니다.

| | hidden_layer_sizes | alpha | train_accuracy | test_accuracy |
|---|---|---|---|---|
| 0 | (32,) | 0.001 | 0.989645 | 0.863905 |
| 1 | (32,) | 0.010 | 0.989645 | 0.834320 |
| 2 | (32,) | 0.100 | 0.980769 | 0.863905 |
| 3 | (64,) | 0.001 | 1.000000 | 0.840237 |
| 4 | (64,) | 0.010 | 0.998521 | 0.834320 |
| 5 | (64,) | 0.100 | 0.989645 | 0.816568 |
| 6 | (32, 16) | 0.001 | 0.998521 | 0.804734 |
| 7 | (32, 16) | 0.010 | 1.000000 | 0.804734 |
| 8 | (32, 16) | 0.100 | 0.997041 | 0.804734 |
| 9 | (64, 32) | 0.001 | 1.000000 | 0.822485 |
| 10 | (64, 32) | 0.010 | 1.000000 | 0.810651 |
| 11 | (64, 32) | 0.100 | 1.000000 | 0.804734 |

그림 1 ‘adam’의 하이퍼파라미터 설정 결과값

| | hidden_layer_sizes | alpha | train_accuracy | test_accuracy |
|---|---|---|---|---|
| 0 | (16,) | 0.01 | 1.000000 | 0.816568 |
| 1 | (16,) | 0.10 | 1.000000 | 0.834320 |
| 2 | (16,) | 1.00 | 0.983728 | 0.816568 |
| 3 | (32,) | 0.01 | 1.000000 | 0.804734 |
| 4 | (32,) | 0.10 | 1.000000 | 0.822485 |
| 5 | (32,) | 1.00 | 0.997041 | 0.804734 |
| 6 | (32, 16) | 0.01 | 1.000000 | 0.816568 |
| 7 | (32, 16) | 0.10 | 1.000000 | 0.822485 |
| 8 | (32, 16) | 1.00 | 1.000000 | 0.840237 |
| 9 | (64, 32) | 0.01 | 1.000000 | 0.828402 |
| 10 | (64, 32) | 0.10 | 1.000000 | 0.822485 |
| 11 | (64, 32) | 1.00 | 1.000000 | 0.857988 |

그림 2 ‘lbfgs’의 하이퍼파라미터 설정 결과값

## 3. 3개 이상의 불확실성 정량화(Uncertainty quantification) 방법 비교 :

(1) Confidece, Margin, Entropy에 대해 상위,하위 5개의 데이터 포인트 비교

Top 5 UQ Samples

| | confidence_idx | confidence_val | margin_idx | margin_val | entropy_idx | entropy_val |
|---|---|---|---|---|---|---|
| 0 | 23 | 1.000000 | 23 | 1.000000 | 23 | 0.000002 |
| 1 | 73 | 1.000000 | 73 | 0.999999 | 73 | 0.000005 |
| 2 | 58 | 1.000000 | 58 | 0.999999 | 58 | 0.000006 |
| 3 | 36 | 1.000000 | 36 | 0.999999 | 36 | 0.000007 |
| 4 | 29 | 0.999999 | 29 | 0.999999 | 29 | 0.000008 |

그림 3 상위 5개 데이터 포인트 인덱스

Bottom 5 UQ Samples

| | Toggle output scrolling | nce_val | margin_idx | margin_val | entropy_idx | entropy_val |
|---|---|---|---|---|---|---|
| 0 | 162 | 0.375504 | 56 | 0.002228 | 162 | 1.186192 |
| 1 | 14 | 0.421686 | 14 | 0.014892 | 14 | 1.078986 |
| 2 | 56 | 0.500895 | 162 | 0.070127 | 76 | 0.892552 |
| 3 | 76 | 0.511600 | 135 | 0.080683 | 83 | 0.867202 |
| 4 | 135 | 0.525394 | 76 | 0.092605 | 92 | 0.827697 |

그림 4 하위 5개 데이터 포인트 인덱스

## 4. Test set에서 Rejection rate(예측 불확실성이 높은 data point의 제외)에 따른 성능 개선 분석:

(1) Rejection rate에 따른 성능 비교 (solver= ‘adam’,‘hidden_layer_sizes’=(32,), alpha=0.1)

| | Rejection Rate | Confidence | Margin | Entropy |
|---|---|---|---|---|
| 0 | 0.0 | 0.863905 | 0.863905 | 0.863905 |
| 1 | 0.1 | 0.895425 | 0.895425 | 0.888889 |
| 2 | 0.2 | 0.919118 | 0.911765 | 0.919118 |
| 3 | 0.3 | 0.949580 | 0.949580 | 0.941176 |
| 4 | 0.4 | 1.000000 | 0.990196 | 1.000000 |
| 5 | 0.5 | 1.000000 | 1.000000 | 1.000000 |

## 5. 추가 성능 개선을 위한 방안:

(1) ‘sgd’ solver를 사용해볼 수 있습니다. (2) ‘lbfgs’는 스케일링에 민감하기 때문에, 스케일링을 더 정교하게 해볼 수 있습니다. (3) hidden_layer_sizes나 alpha의 다른 값들로 튜닝할 수 있습니다.

<코드 첨부>

1. DATA LOAD

```
pip install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in c:\anacon\lib\site-packages (0.0.7)
Requirement already satisfied: pandas>=1.0.0 in c:\anacon\lib\site-packages (from ucimlrepo) (2.1.4)
Requirement already satisfied: certifi>=2020.12.5 in c:\anacon\lib\site-packages (from ucimlrepo) (2024.2.2)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\anacon\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\anacon\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\anacon\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\anacon\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2023.3)
Requirement already satisfied: six>=1.5 in c:\anacon\lib\site-packages (from python-dateutil>=2.8.2->pandas>=1.0.0->ucimlrepo) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
statlog_vehicle_silhouettes = fetch_ucirepo(id=149)

# data (as pandas dataframes)
X = statlog_vehicle_silhouettes.data.features
y = statlog_vehicle_silhouettes.data.targets

# metadata
print(statlog_vehicle_silhouettes.metadata)

# variable information
print(statlog_vehicle_silhouettes.variables)
```


```
{'uci_id': 149, 'name': 'Statlog (Vehicle Silhouettes)', 'repository_url': 'https://archive.ics.uci.edu/dataset/149/statlog+vehicle+silhouettes', 'data_
url': 'https://archive.ics.uci.edu/static/public/149/data.csv', 'abstract': '3D objects within a 2D image by application of an ensemble of shape feature
extractors to the 2D silhouettes of the objects.', 'area': 'Other', 'tasks': ['Classification'], 'characteristics': ['Multivariate'], 'num_instances': 9
46, 'num_features': 18, 'feature_types': ['Integer'], 'demographics': [], 'target_col': ['class'], 'index_col': None, 'has_missing_values': 'no', 'missi
ng_values_symbol': None, 'year_of_dataset_creation': None, 'last_updated': 'Fri Feb 16 2024', 'dataset_doi': '10.24432/C5HG6N', 'creators': ['Pete Mowfo
rth', 'Barry Shepherd'], 'intro_paper': {'ID': 393, 'type': 'NATIVE', 'title': 'Vehicle Recognition Using Rule Based Methods', 'authors': 'J. Siebert',
'venue': 'Turing Institute', 'year': 1987, 'journal': None, 'DOI': None, 'URL': 'https://www.semanticscholar.org/paper/Vehicle-Recognition-Using-Rule-Ba
sed-Methods-Siebert/79f050f5890760223f2298a6c512af63a2bc251b', 'sha': None, 'corpus': None, 'arxiv': None, 'mag': None, 'acl': None, 'pmid': None, 'pmci
d': None}, 'additional_info': {'summary': 'The purpose is to classify a given silhouette as one of four types of vehicle, using  a set of features extra
cted from the silhouette. The vehicle may be viewed from one of many different angles.  \r\n\r\nHISTORY:\r\n\r\nThis data was originally gathered at the
TI in 1986-87 by JP Siebert. It was partially financed by Barr and Stroud Ltd. The original purpose was to find a method of distinguishing 3D objects wi
thin a 2D image by application of an ensemble of shape feature extractors to the 2D silhouettes of the objects. Measures of shape features extracted fro
m example silhouettes of objects to be discriminated were used to generate a classification rule tree by means of computer induction.\r\n\r\nThis object
recognition strategy was successfully used to discriminate between silhouettes of model cars, vans and buses viewed from constrained elevation but all a
ngles of rotation.\r\n\r\nThe rule tree classification performance compared favourably to MDC (Minimum Distance Classifier) and k-NN (k-Nearest Neighbou
r) statistical classifiers in terms of both error rate and computational efficiency. An investigation of these rule trees generated by example indicated
that the tree structure was heavily influenced by the orientation of the objects, and grouped similar object views into single decisions.\r\n\r\n\r\nDES
CRIPTION:\r\n\r\nThe features were extracted from the silhouettes by the HIPS (Hierarchical Image Processing System) extension BINATTS, which extracts a
combination of scale independent features utilising both classical moments based measures such as scaled variance, skewness and kurtosis about the majo
r/minor axes and heuristic measures such as hollows, circularity, rectangularity and compactness.\r\n\r\nFour "Corgie" model vehicles were used for the
experiment: a double decker bus, Cheverolet van, Saab 9000 and an Opel Manta 400. This particular combination of vehicles was chosen with the expectatio
n that the bus, van and either one of the cars would be readily distinguishable, but it would be more difficult to distinguish between the cars.\r\n\r\n
The images were acquired by a camera looking downwards at the model vehicle from a fixed angle of elevation (34.2 degrees to the horizontal). The vehicl
es were placed on a diffuse backlit surface (lightbox). The vehicles were painted matte black to minimise highlights. The images were captured using a C
RS4000 framestore connected to a vax 750. All images were captured with a spatial resolution of 128x128 pixels quantised to 64 greylevels. These images
were thresholded to produce binary vehicle silhouettes, negated (to comply with the processing requirements of BINATTS) and thereafter subjected to shri
nk-expand-expand-shrink HIPS modules to remove "salt and pepper" image noise.\r\n\r\nThe vehicles were rotated and their angle of orientation was measur
ed using a radial graticule beneath the vehicle. 0 and 180 degrees corresponded to "head on" and "rear" views respectively while 90 and 270 corresponded
to profiles in opposite directions. Two sets of 60 images, each set covering a full 360 degree rotation, were captured for each vehicle. The vehicle was
rotated by a fixed angle between images. These datasets are known as e2 and e3 respectively.\r\n\r\n A further two sets of images, e4 and e5, were captu
red with the camera at elevations of 37.5 degs and 30.8 degs respectively. These sets also contain 60 images per vehicle apart from e4.van which contain
s only 46 owing to the difficulty of containing the van in the image at some orientations.\r\n', 'purpose': None, 'funded_by': None, 'instances_represen
t': None, 'recommended_data_splits': None, 'sensitive_data': None, 'preprocessing_description': None, 'variable_info': 'ATTRIBUTES\r\n\t\r\n\tCOMPACTNES
S\t(average perim)**2/area\r\n\t\r\n\tCIRCULARITY\t(average radius)**2/area\r\n\t\r\n\tDISTANCE CIRCULARITY\tarea/(av.distance from border)**2\r\n\t\r\n
\tRADIUS RATIO\t(max.rad-min.rad)/av.radius\r\n\t\r\n\tPR.AXIS ASPECT RATIO\t(minor axis)/(major axis)\r\n\t\r\n\tMAX.LENGTH ASPECT RATIO\t(length perp.
max length)/(max length)\r\n\t\r\n\tSCATTER RATIO\t(inertia about minor axis)/(inertia about major axis)\r\n\t\r\n\tELONGATEDNESS\t\tarea/(shrink width)
**2\r\n\t\r\n\tPR.AXIS RECTANGULARITY\tarea/(pr.axis length*pr.axis width)\r\n\t\r\n\tMAX.LENGTH RECTANGULARITY area/(max.length*length perp. to this)\r
\n\t\r\n\tSCALED VARIANCE \t(2nd order moment about minor axis)/area\r\n\tALONG MAJOR AXIS\r\n\t\r\n\tSCALED VARIANCE \t(2nd order moment about major ax
is)/area\r\n\tALONG MINOR AXIS \r\n\t\r\n\tSCALED RADIUS OF GYRATION\t(mavar+mivar)/area\r\n\t\r\n\tSKEWNESS ABOUT \t(3rd order moment about major axi
s)/sigma_min**3\r\n\tMAJOR AXIS\r\n\t\r\n\tSKEWNESS ABOUT \t(3rd order moment about minor axis)/sigma_maj**3\r\n\tMINOR AXIS\r\n\t  \t\r\n\tKURTOSIS ABO
UT \t(4th order moment about major axis)/sigma_min**4\r\n\tMINOR AXIS  \r\n\t  \t\r\n\tKURTOSIS ABOUT \t(4th order moment about minor axis)/sigma_maj**4
\r\n\tMAJOR AXIS\r\n\t\r\n\tHOLLOWS RATIO\t(area of hollows)/(area of bounding polygon)\r\n\t\r\n\t Where sigma_maj**2 is the variance along the major a
xis and sigma_min**2 is the variance along the minor axis, and\r\n\t\r\n\tarea of hollows= area of bounding poly-area of object \r\n\t\r\n\t The area of
the bounding polygon is found as a side result of the computation to find the maximum length. Each individual length computation yields a pair of calipe
rs to the object orientated at every 5 degrees. The object is propagated into an image containing the union of these calipers to obtain an image of the
bounding polygon. \r\n\t\r\nNUMBER OF CLASSES\r\n\r\n\t4\tOPEL, SAAB, BUS, VAN\r\n', 'citation': None}}
```

```
                               name    role         type demographic  \
0                       COMPACTNESS  Feature      Integer        None
1                       CIRCULARITY  Feature      Integer        None
2              DISTANCE CIRCULARITY  Feature      Integer        None
3                      RADIUS RATIO  Feature      Integer        None
4              PR.AXIS ASPECT RATIO  Feature      Integer        None
5           MAX.LENGTH ASPECT RATIO  Feature      Integer        None
6                     SCATTER RATIO  Feature      Integer        None
7                     ELONGATEDNESS  Feature      Integer        None
8           PR.AXIS RECTANGULARITY  Feature      Integer        None
9         MAX.LENGTH RECTANGULARITY  Feature      Integer        None
10   SCALED VARIANCE ALONG MAJOR AXIS  Feature      Integer        None
11   SCALED VARIANCE ALONG MINOR AXIS  Feature      Integer        None
12        SCALED RADIUS OF GYRATION  Feature      Integer        None
13         SKEWNESS ABOUT MAJOR AXIS  Feature      Integer        None
14         SKEWNESS ABOUT MINOR AXIS  Feature      Integer        None
15         KURTOSIS ABOUT MINOR AXIS  Feature      Integer        None
16         KURTOSIS ABOUT MAJOR AXIS  Feature      Integer        None
17                    HOLLOWS RATIO  Feature      Integer        None
18                            class   Target  Categorical        None

   description units missing_values
0         None  None             no
1         None  None             no
2         None  None             no
3         None  None             no
4         None  None             no
5         None  None             no
6         None  None             no
7         None  None             no
8         None  None             no
9         None  None             no
10        None  None             no
11        None  None             no
12        None  None             no
13        None  None             no
14        None  None             no
15        None  None             no
16        None  None             no
17        None  None             no
18        None  None             no
```

[4]:
```
y
```

[4]:

| | class |
|---|---|
| 0 | van |
| 1 | van |
| 2 | saab |
| 3 | van |
| 4 | bus |
| ... | ... |
| 841 | saab |
| 842 | van |
| 843 | saab |
| 844 | saab |
| 845 | van |

846 rows × 1 columns

```
x
```

| | COMPACTNESS | CIRCULARITY | DISTANCE CIRCULARITY | RADIUS RATIO | PR.AXIS ASPECT RATIO | MAX.LENGTH ASPECT RATIO | SCATTER RATIO | ELONGATEDNESS | PR.AXIS RECTANGULARITY | MAX.LENGTH RECTANGULARITY | SCALED VARIANCE ALONG MAJOR AXIS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 95.0 | 48 | 83 | 178 | 72 | 10 | 162 | 42 | 20 | 159 | 176 |
| **1** | 91.0 | 41 | 84 | 141 | 57 | 9 | 149 | 45 | 19 | 143 | 170 |
| **2** | 104.0 | 50 | 106 | 209 | 66 | 10 | 207 | 32 | 23 | 158 | 223 |
| **3** | 93.0 | 41 | 82 | 159 | 63 | 9 | 144 | 46 | 19 | 143 | 160 |
| **4** | 85.0 | 44 | 70 | 205 | 103 | 52 | 149 | 45 | 19 | 144 | 241 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **841** | 93.0 | 39 | 87 | 183 | 64 | 8 | 169 | 40 | 20 | 134 | 200 |
| **842** | 89.0 | 46 | 84 | 163 | 66 | 11 | 159 | 43 | 20 | 159 | 173 |
| **843** | 106.0 | 54 | 101 | 222 | 67 | 12 | 222 | 30 | 25 | 173 | 228 |
| **844** | 86.0 | 36 | 78 | 146 | 58 | 7 | 135 | 50 | 18 | 124 | 155 |
| **845** | 85.0 | 36 | 66 | 123 | 55 | 5 | 120 | 56 | 17 | 128 | 140 |

846 rows × 18 columns

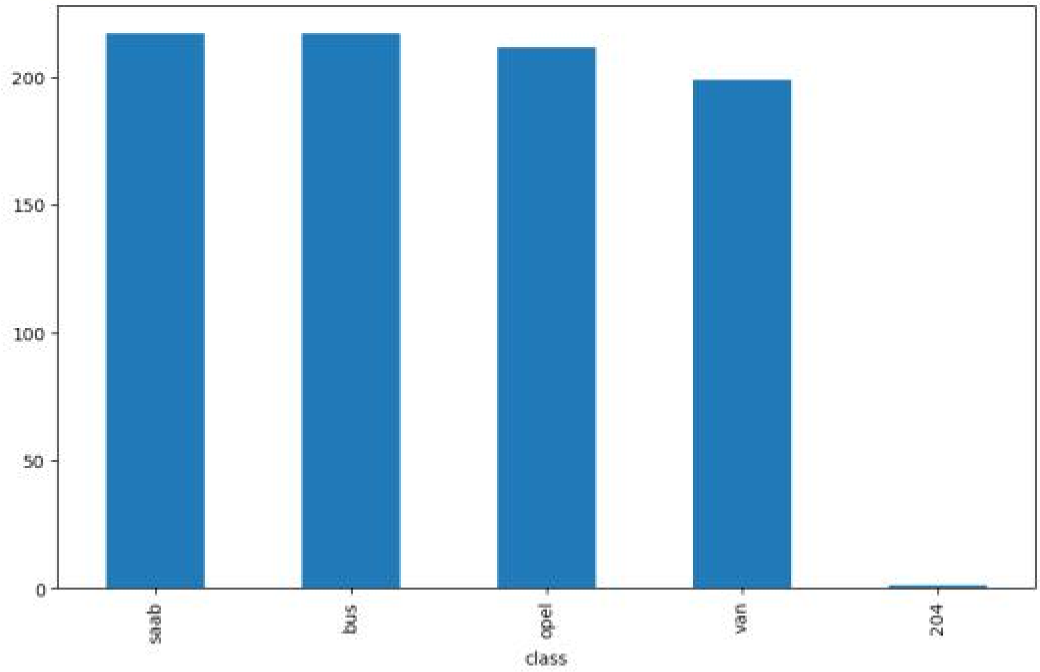
```python
y_series = y['class'].copy()
print(y_series.value_counts())  # 각 클래스 몇 개인지 보기

idx_204 = y_series[y_series == '204'].index
print(idx_204)
```

```
class
saab    217
bus     217
opel    212
van     199
204       1
Name: count, dtype: int64
Index([752], dtype='int64')
```

```python
print(y_series)
```

```
0       van
1       van
2      saab
3       van
4       bus
       ...
841    saab
842     van
843    saab
844    saab
845     van
Name: class, Length: 846, dtype: object
```

```python
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
y_series.value_counts().plot(kind='bar')
plt.show()
```




[9]:

```python
X_cleaned, y_series_cleaned = X[y_series != '204'], y_series[y_series != '204']
```

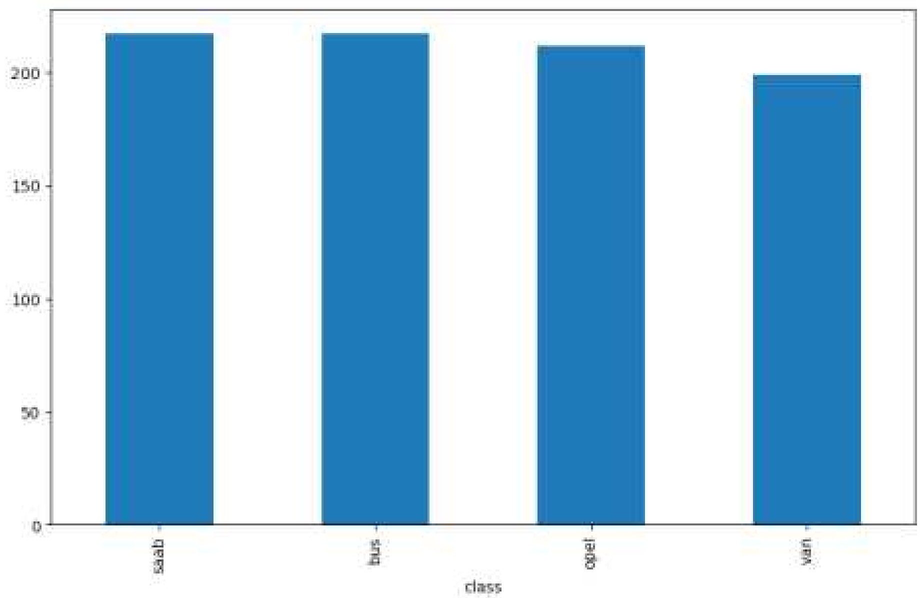
[10]:

```python
print(y_series_cleaned.value_counts())
```

```
class
saab    217
bus     217
opel    212
van     199
Name: count, dtype: int64
```

[11]:

```python
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
y_series_cleaned.value_counts().plot(kind='bar')
plt.show()
```



2. DATA Split

[12]:

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split

encoder = OneHotEncoder(sparse_output=False)
y_onehot = encoder.fit_transform(y_series_cleaned.values.reshape(-1, 1))

X_train, X_test, y_train, y_test = train_test_split(X_cleaned,y_series_cleaned,test_size=0.2,random_state=42)
```

3. DATA Preprocessing

[13]:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_train)

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

print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (676, 18)
X_test shape: (169, 18)
y_train shape: (676,)
y_test shape: (169,)
```

4. Training

[14]:

```
# solver가 adam인 경우
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
import pandas as pd

results=[]
hidden_options = [(32,), (64,), (32, 16), (64, 32)]
alpha_options = [0.001, 0.01, 0.1]

for h in hidden_options:
    for a in alpha_options:
        clf = MLPClassifier(hidden_layer_sizes=h,alpha=a,max_iter=1500,solver='adam', random_state=42)
        clf.fit(X_train_scaled, y_train)
        y_train_hat = clf.predict(X_train_scaled)
        y_test_hat = clf.predict(X_test_scaled)

        train_acc = accuracy_score(y_train, y_train_hat)
        test_acc = accuracy_score(y_test, y_test_hat)

        results.append({
                    'hidden_layer_sizes': h,
                    'alpha': a,
                    'train_accuracy': train_acc,
                    'test_accuracy': test_acc
            })


results_df = pd.DataFrame(results)
print(results_df)
```

```
C:\anacon\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1500)
ched and the optimization hasn't converged yet.
  warnings.warn(
C:\anacon\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1500)
ched and the optimization hasn't converged yet.
  warnings.warn(
C:\anacon\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1500)
ched and the optimization hasn't converged yet.
  warnings.warn(
C:\anacon\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1500)
ched and the optimization hasn't converged yet.
  warnings.warn(
   hidden_layer_sizes  alpha  train_accuracy  test_accuracy
0               (32,)  0.001        0.989645       0.863905
1               (32,)  0.010        0.989645       0.834320
2               (32,)  0.100        0.988769       0.863905
3               (64,)  0.001        1.000000       0.840237
4               (64,)  0.010        0.998521       0.834320
5               (64,)  0.100        0.989645       0.816568
6            (32, 16)  0.001        0.998521       0.804734
7            (32, 16)  0.010        1.000000       0.804734
8            (32, 16)  0.100        0.997041       0.804734
9            (64, 32)  0.001        1.000000       0.822485
10           (64, 32)  0.010        1.000000       0.810651
11           (64, 32)  0.100        1.000000       0.804734
```

[18]:

```python
### solver가 lbfgs 인 경우

from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
import pandas as pd

results=[]
hidden_options = [(16,), (32,), (32, 16), (64, 32)]
alpha_options = [0.01, 0.1, 1.0]

for h in hidden_options:
    for a in alpha_options:
        clf = MLPClassifier(hidden_layer_sizes=h,alpha=a,max_iter=1500,solver='lbfgs', random_state=42)
        clf.fit(X_train_scaled, y_train)
        y_train_hat = clf.predict(X_train_scaled)
        y_test_hat = clf.predict(X_test_scaled)

        train_acc = accuracy_score(y_train, y_train_hat)
        test_acc = accuracy_score(y_test, y_test_hat)

        results.append({
                    'hidden_layer_sizes': h,
                    'alpha': a,
                    'train_accuracy': train_acc,
                    'test_accuracy': test_acc
            })

results_df = pd.DataFrame(results)
print(results_df)
```

```
C:\anacon\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
C:\anacon\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
C:\anacon\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
C:\anacon\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
C:\anacon\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

| | hidden_layer_sizes | alpha | train_accuracy | test_accuracy |
|---|---|---|---|---|
| 0 | (16,) | 0.01 | 1.000000 | 0.816568 |
| 1 | (16,) | 0.10 | 1.000000 | 0.834320 |
| 2 | (16,) | 1.00 | 0.983728 | 0.816568 |
| 3 | (32,) | 0.01 | 1.000000 | 0.804734 |
| 4 | (32,) | 0.10 | 1.000000 | 0.822485 |
| 5 | (32,) | 1.00 | 0.997041 | 0.804734 |
| 6 | (32, 16) | 0.01 | 1.000000 | 0.816568 |
| 7 | (32, 16) | 0.10 | 1.000000 | 0.822485 |
| 8 | (32, 16) | 1.00 | 1.000000 | 0.840237 |
| 9 | (64, 32) | 0.01 | 1.000000 | 0.828402 |
| 10 | (64, 32) | 0.10 | 1.000000 | 0.822485 |
| 11 | (64, 32) | 1.00 | 1.000000 | 0.857988 |

5. Uncertainty quantification

```python
from scipy.stats import entropy
import numpy as np
import pandas as pd

clf = MLPClassifier(hidden_layer_sizes=(32,),solver='adam', max_iter=1500, random_state=42, alpha=0.1)
clf.fit(X_train_scaled, y_train)

y_proba = clf.predict_proba(X_test_scaled)

confidence = np.max(y_proba, axis=1)

sorted_probs = -np.sort(-y_proba, axis=1)  # 내림차순 정렬
margin = sorted_probs[:, 0] - sorted_probs[:, 1]

entropy_val = entropy(y_proba.T)  # 각 sample에 대해 엔트로피 계산

uq_df = pd.DataFrame({
    'confidence': confidence,
    'margin': margin,
    'entropy': entropy_val
})

print(uq_df.head())
```

| | confidence | margin | entropy |
|---|---|---|---|
| 0 | 0.958780 | 0.917712 | 0.172903 |
| 1 | 0.677793 | 0.358791 | 0.646979 |
| 2 | 0.983447 | 0.966898 | 0.084328 |
| 3 | 0.986852 | 0.973704 | 0.070011 |
| 4 | 0.997862 | 0.996662 | 0.017311 |

```python
import pandas as pd

# Top 5 인덱스 및 값 추출
confidence_top5 = uq_df.sort_values(by='confidence', ascending=False).head(5)[['confidence']]
margin_top5 = uq_df.sort_values(by='margin', ascending=False).head(5)[['margin']]
entropy_top5 = uq_df.sort_values(by='entropy', ascending=True).head(5)[['entropy']]  # entropy는 낮을수록 확신 높음

# 인덱스와 값 정리
top5_df = pd.DataFrame({
    'confidence_idx': confidence_top5.index,
    'confidence_val': confidence_top5['confidence'].values,
    'margin_idx': margin_top5.index,
    'margin_val': margin_top5['margin'].values,
    'entropy_idx': entropy_top5.index,
    'entropy_val': entropy_top5['entropy'].values
})

# Bottom 5 인덱스 및 값 추출
confidence_bottom5 = uq_df.sort_values(by='confidence', ascending=True).head(5)[['confidence']]
margin_bottom5 = uq_df.sort_values(by='margin', ascending=True).head(5)[['margin']]
entropy_bottom5 = uq_df.sort_values(by='entropy', ascending=False).head(5)[['entropy']]  # entropy는 클수록 불확신 큼

# 인덱스와 값 정리
bottom5_df = pd.DataFrame({
    'confidence_idx': confidence_bottom5.index,
    'confidence_val': confidence_bottom5['confidence'].values,
    'margin_idx': margin_bottom5.index,
    'margin_val': margin_bottom5['margin'].values,
    'entropy_idx': entropy_bottom5.index,
    'entropy_val': entropy_bottom5['entropy'].values
})

# 출력
print("🔹 Top 5 UQ Samples")
display(top5_df)

print("\n🔻 Bottom 5 UQ Samples")
display(bottom5_df)
```

🔹 Top 5 UQ Samples

| | confidence_idx | confidence_val | margin_idx | margin_val | entropy_idx | entropy_val |
|---|---|---|---|---|---|---|
| **0** | 23 | 1.000000 | 23 | 1.000000 | 23 | 0.000002 |
| **1** | 73 | 1.000000 | 73 | 0.999999 | 73 | 0.000005 |
| **2** | 58 | 1.000000 | 58 | 0.999999 | 58 | 0.000006 |
| **3** | 36 | 1.000000 | 36 | 0.999999 | 36 | 0.000007 |
| **4** | 29 | 0.999999 | 29 | 0.999999 | 29 | 0.000008 |

🔻 Bottom 5 UQ Samples

| | confidence_idx | confidence_val | margin_idx | margin_val | entropy_idx | entropy_val |
|---|---|---|---|---|---|---|
| **0** | 162 | 0.375504 | 56 | 0.002228 | 162 | 1.186192 |
| **1** | 14 | 0.421686 | 14 | 0.014892 | 14 | 1.078986 |
| **2** | 56 | 0.500895 | 162 | 0.070127 | 76 | 0.892552 |
| **3** | 76 | 0.511600 | 135 | 0.080683 | 83 | 0.867202 |
| **4** | 135 | 0.525394 | 76 | 0.092605 | 92 | 0.827697 |

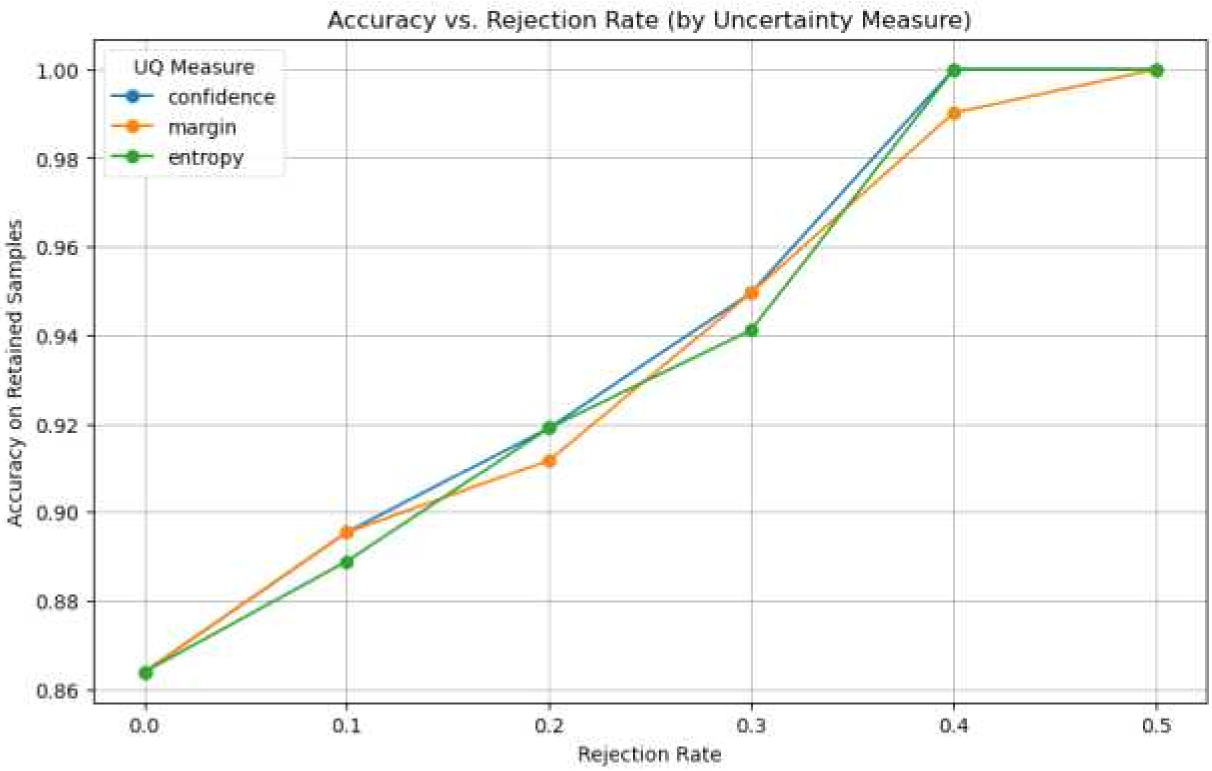
6. Evaluation (without reject option)

[27]:
```python
y_test_hat = clf.predict(X_test_scaled)
y_test = pd.Series(y_test).reset_index(drop=True)
y_test_hat = pd.Series(y_test_hat).reset_index(drop=True)
print(f"Accuracy (without reject option): {accuracy_score(y_test, y_test_hat):.5f}")
```

```
Accuracy (without reject option): 0.86391
```

7. Evaluation (with reject option)

```
[28]: import pandas as pd
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# 예측값
y_test_hat = clf.predict(X_test_scaled)
y_test = pd.Series(y_test).reset_index(drop=True)
y_test_hat = pd.Series(y_test_hat).reset_index(drop=True)

# Rejection rate 리스트
rejection_rates = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
results = {'Rejection Rate': rejection_rates}

plt.figure(figsize=(10, 6))

# 정확도 계산 및 시각화
for measure in ['confidence', 'margin', 'entropy']:
    ascending = True if measure in ['confidence', 'margin'] else False
    sorted_idx = uq_df[measure].sort_values(ascending=ascending).index

    accuracies = []
    for r in rejection_rates:
        n_reject = int(len(y_test) * r)
        keep_idx = sorted_idx[n_reject:]

        acc = accuracy_score(y_test.iloc[keep_idx], y_test_hat.iloc[keep_idx])
        accuracies.append(acc)

    results[measure.capitalize()] = accuracies  # 대문자로 칼럼명 저장
    plt.plot(rejection_rates, accuracies, marker='o', label=measure)

# 그래프 시각화
plt.title("Accuracy vs. Rejection Rate (by Uncertainty Measure)")
plt.xlabel("Rejection Rate")
plt.ylabel("Accuracy on Retained Samples")
plt.grid(True)
plt.legend(title="UQ Measure")
plt.show()

# 정확도 결과를 표로 출력
result_df = pd.DataFrame(results)
print(result_df)
```




| | Rejection Rate | Confidence | Margin | Entropy |
|---|---|---|---|---|
| 0 | 0.0 | 0.863905 | 0.863905 | 0.863905 |
| 1 | 0.1 | 0.895425 | 0.895425 | 0.888889 |
| 2 | 0.2 | 0.919118 | 0.911765 | 0.919118 |
| 3 | 0.3 | 0.949580 | 0.949580 | 0.941176 |
| 4 | 0.4 | 1.000000 | 0.990196 | 1.000000 |
| 5 | 0.5 | 1.000000 | 1.000000 | 1.000000 |