2.0-Handling Imbalance Dataset

December 29, 2023

0.1 Handling Imbalanced Dataset

1. Up Sampling
2. Down Sampling

[1]:
```
import numpy as np
import pandas as pd

# Set the random seed for reproducibility
np.random.seed(123)

# Create a dataframe with two classes
n_samples = 1000
class_0_ratio = 0.9
n_class_0 = int(n_samples * class_0_ratio)
n_class_1 = n_samples - n_class_0
```

[2]:
```
n_class_0,n_class_1
```

[2]:
```
(900, 100)
```

[3]:
```
## CREATE MY DATAFRAME WITH IMBALANCED DATASET
class_0 = pd.DataFrame({
    'feature_1': np.random.normal(loc=0, scale=1, size=n_class_0),
    'feature_2': np.random.normal(loc=0, scale=1, size=n_class_0),
    'target': [0] * n_class_0
})

class_1 = pd.DataFrame({
    'feature_1': np.random.normal(loc=2, scale=1, size=n_class_1),
    'feature_2': np.random.normal(loc=2, scale=1, size=n_class_1),
    'target': [1] * n_class_1
})
```

[5]:
```
df=pd.concat([class_0,class_1]).reset_index(drop=True)
```

[6]:
```
df.tail()
```

[6]:
```
     feature_1  feature_2  target
995   1.376371   2.845701       1
996   2.239810   0.880077       1
997   1.131760   1.640703       1
998   2.902006   0.390305       1
999   2.697490   2.013570       1
```

[7]:
```
df['target'].value_counts()
```

[7]:
```
0    900
1    100
Name: target, dtype: int64
```

[9]:
```
## upsampling
df_minority=df[df['target']==1]
df_majority=df[df['target']==0]
```

[10]:
```
from sklearn.utils import resample
df_minority_upsampled=resample(df_minority,replace=True, #Sample With␣
 ↪replacement
        n_samples=len(df_majority),
        random_state=42
       )
```

[11]:
```
df_minority_upsampled.shape
```

[11]:
```
(900, 3)
```

[12]:
```
df_minority_upsampled.head()
```

[12]:
```
     feature_1  feature_2  target
951   1.125854   1.843917       1
992   2.196570   1.397425       1
914   1.932170   2.998053       1
971   2.272825   3.034197       1
960   2.870056   1.550485       1
```

[13]:
```
df_upsampled=pd.concat([df_majority,df_minority_upsampled])
```

[14]:
```
df_upsampled['target'].value_counts()
```

[14]:
```
0    900
1    900
Name: target, dtype: int64
```

0.2 Down Sampling

[45]:
```
import pandas as pd

# Set the random seed for reproducibility
np.random.seed(123)

# Create a dataframe with two classes
n_samples = 1000
class_0_ratio = 0.9
n_class_0 = int(n_samples * class_0_ratio)
n_class_1 = n_samples - n_class_0

class_0 = pd.DataFrame({
    'feature_1': np.random.normal(loc=0, scale=1, size=n_class_0),
    'feature_2': np.random.normal(loc=0, scale=1, size=n_class_0),
    'target': [0] * n_class_0
})

class_1 = pd.DataFrame({
    'feature_1': np.random.normal(loc=2, scale=1, size=n_class_1),
    'feature_2': np.random.normal(loc=2, scale=1, size=n_class_1),
    'target': [1] * n_class_1
})

df = pd.concat([class_0, class_1]).reset_index(drop=True)

# Check the class distribution
print(df['target'].value_counts())
```

```
0    900
1    100
Name: target, dtype: int64
```

[46]:
```
## downsampling
df_minority=df[df['target']==1]
df_majority=df[df['target']==0]
```

[52]:
```
from sklearn.utils import resample
df_majority_downsampled=resample(df_majority,replace=False, #Sample With␣
 ↪replacement
        n_samples=len(df_minority),
        random_state=42
       )
```

[53]:
```
df_majority_downsampled.shape
```

[53]:
```
(100, 3)
```

[54]:
```
df_downsampled=pd.concat([df_minority,df_majority_downsampled])
```

[55]:
```
df_downsampled['target'].value_counts()
```

[55]:
```
1    100
0    100
Name: target, dtype: int64
```

[ ]:
```

```