Over_Under_Fitting

April 1, 2023

Overfitting When we provide too many feature inputs in our training data then the model turns out to be a overfitted model. Let's take an example. We have a dataset where shapes of different objects are provided. The features are state as below -

- Shape
- Radius
- Weight
- Usage

Looks fine , suppose the object in discussion is a ball and the balls those are present in the data are consisiting of below specifications

- Shape - Round
- Radius - 5.6 cm
- Weight - 150 gm to 250 gm
- Usage - Play

Now let's say we feed our model with all of these data and train the model , now the model knows that a ball must be round , weighted between 150-250 gm but must contain a radius of 5.6 cm and is used for playing.

Now let's say a new data point comes in where it is a ball , the feture of that data is as below -

- Shape - Round
- Radius - 7.6 cm
- Weight - 225 gm
- Usage - Play

We can see that apart from radius all other data points are matching properly for the new data here. But as our model only know that a ball's radius can only be 5.6 cm hence it will refuse the identify this new data as a ball despite the fact that it is a ball.

[1]:
```
# Code example
# Importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

[2]:
```
# dataset creation
df1 = pd.DataFrame(data = [['round',5.6,175,'play','ball'],
                           ['round',5.6,180,'play','ball'],
                           ['triangle',8.6,170,'play','boomerang'],
                           ['round',5.6,200,'play','ball'],
                           ['triangle',8.7,186,'play','boomerang'],
                           ['round',5.6,185,'play','ball'],
                           ['round',5.6,170,'play','ball'],
                           ['triangle',8.9,176,'play','boomerang'],
                           ['round',5.6,220,'play','ball'],
                           ['round',5.6,193,'play','ball'],
                           ['round',5.6,192,'play','ball'],
                           ['round',5.6,183,'play','ball']],columns =␣
 ↪['Shape','Radius','Weight','Usage','Label'])
```

[3]:
```
df1.head()
```

[3]:

	Shape	Radius	Weight	Usage	Label
0	round	5.6	175	play	ball
1	round	5.6	180	play	ball
2	triangle	8.6	170	play	boomerang
3	round	5.6	200	play	ball
4	triangle	8.7	186	play	boomerang

[4]:
```
# data encoding for categorical column
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
# Finding object type columns and storing it in a list
objlst = df1.select_dtypes(include='object').columns
for i in objlst:
    df1[i] = le.fit_transform(df1[i])
```

[5]:
```
df1.head()
```

[5]:

	Shape	Radius	Weight	Usage	Label
0	0	5.6	175	0	0
1	0	5.6	180	0	0
2	1	8.6	170	0	1
3	0	5.6	200	0	0
4	1	8.7	186	0	1

[ ]:
```

```

[6]:
```
# x y split
x = df1.iloc[:, 0:-1]
y = df1.iloc[:, -1]
```

[7]:
```
x
```

[7]:

	Shape	Radius	Weight	Usage
0	0	5.6	175	0
1	0	5.6	180	0
2	1	8.6	170	0
3	0	5.6	200	0
4	1	8.7	186	0
5	0	5.6	185	0
6	0	5.6	170	0
7	1	8.9	176	0
8	0	5.6	220	0
9	0	5.6	193	0
10	0	5.6	192	0
11	0	5.6	183	0

[8]:
```
y
```

[8]:
```
0     0
1     0
2     1
3     0
4     1
5     0
6     0
7     1
8     0
9     0
10    0
11    0
Name: Label, dtype: int32
```

[9]:
```
# now let's scale our x data
scaler = StandardScaler()
scaler.fit_transform(x)
```

[9]:
```
array([[-0.57735027, -0.57674139, -0.79726099,  0.        ],
       [-0.57735027, -0.57674139, -0.42929438,  0.        ],
       [ 1.73205081,  1.63205542, -1.16522761,  0.        ],
       [-0.57735027, -0.57674139,  1.04257207,  0.        ],
       [ 1.73205081,  1.70568198,  0.01226555,  0.        ],
       [-0.57735027, -0.57674139, -0.06132777,  0.        ],
       [-0.57735027, -0.57674139, -1.16522761,  0.        ],
       [ 1.73205081,  1.8529351 , -0.72366767,  0.        ],
       [-0.57735027, -0.57674139,  2.51443852,  0.        ],
       [-0.57735027, -0.57674139,  0.52741881,  0.        ],
       [-0.57735027, -0.57674139,  0.45382549,  0.        ],
       [-0.57735027, -0.57674139, -0.20851441,  0.        ]])
```

[10]:
```
# let's do the train test split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.2 , random_state
 ↪= 42)
```

[11]:
```
xtrain.shape
```

[11]: (9, 4)

[13]:
```
xtest.shape
```

[13]: (3, 4)

[14]:
```
ytrain.shape
```

[14]: (9,)

[15]:
```
ytest.shape
```

[15]: (3,)

[16]:
```
lr = LogisticRegression()
```

[17]:
```
lr.fit(xtrain,ytrain)
```

[17]: LogisticRegression()

[19]:
```
lr.score(xtest,ytest)
```

[19]: 1.0

[22]:
```
ypred = lr.predict(xtest)
ypred
```

[22]: array([0, 0, 0])

[23]:
```
xtest
```

[23]:

	Shape	Radius	Weight	Usage
10	0	5.6	192	0
9	0	5.6	193	0
0	0	5.6	175	0

As we can see that we are getting a score of 1 in our test set , which is not good in this case , let's try to predict an outside data now and understand the reason here, we know that round shape is defined as 0 , usage play is defined as 0 and also label "ball" is defined as 0 here. let's take a new data using a different radius now

[24]:
```
nd = pd.DataFrame(data = [[0,8.6,195,0],
[0,8.8,197,0]], columns = ['Shape','Radius','Weight','Usage'])
```

[25]:
```
# let's predict
npred = lr.predict(nd)
npred
```

[25]:
```
array([1, 1])
```

We can clearly see that our model is providing a wrong result despite the new data containing every required details for a ball except the radius.