HW2

DATA MINING

Aisan Aghazade

AUT

Computer and Information Technology

Engineering Department

مرحله‌ی اول

پیش‌پردازش

- در مرحله اول باید نام هر ستون را اضافه کنیم که با دستور زیر انجام می‌دهیم:

```
train.columns = [a,b,…]
```

- در داده‌های داده شده به جای missing valueها علامت سؤال گذاشته شده است. که در نتیجه نمی‌توان با استفاده از count() تعداد داده‌های گم شده را پیدا برای این کار نیز باید علامت سؤال را حذف کنیم:

```
train.replace({"?":None},inplace=True)
```

- پس از اینکه داده‌ها تصحیح شد، تعداد missing valueهای هر ویژگی بررسی می‌شود با توجه به این موضوع که در برخی ستون‌ها تعداد داده‌هایی که مقدار دارند بسیار کمتر از تعداد missingهاست، پر کردن آن‌ها با مقدارهای موجود خطاهای بسیاری تولید می‌کند درنتیجه با توجه به اینکه پرکردن داده‌ها به این روش تنها باعث ایجاد خطا می‌شود، تصمیم به حذف ستون شد:

```
for i in train:
    if(train.count()[i])<100:
        del train[i]
        del test[i]
```

ستون‌های زیر با استفاده از این دستورها حذف شد:

```
non-aging                 95
surface-finish             8
bc                         1
bt                        62
m                          0
chrom                     23
phose                      7
cbond                     68
marvi                      0
exptl                      2
ferro                     26
corr                       0
blue/brigh/varn/clean      5
lustre                    45
jurofm                     0
s                          0
p                          0
oil                       58
packing                    9
```

- در برخی ستون‌های باقی مانده تمام داده‌های یکسان است در نتیجه تأثیری در نتیجه نهایی ندارد. با توجه به این موضوع برای کاهش حجم محاسبات این داده‌ها را نیز حذف می‌کنیم.

```
del train['product-type']
del test['product-type']
del train['temper-rolling']
del test['temper-rolling']
del train['bf']
del test['bf']
del train['bl']
del test['bl']
```

- داده‌هایی که مقدار پیوسته ندارند را با استفاده از احتمال(احتمال را براساس نسبت مقدارهای موجود وارد می‌کنیم) پر می‌کنیم، نمونه‌ای از روش پر کردن داده به شرح زیر است:

```
train['family'] = train['family'].fillna(pd.Series(np.random.choice(['TN', 'ZS'],
p=[60/111, 51/111], size=len(train))))
```

- داده‌هایی که مقادیر گسسته دارند را تبدیل به مقادیر باینری می‌کنیم به این صورت که هر مقدار را یک ستون می‌گیریم. یک نمونه از این تبدیل در زیر آمده است.

```
family = pd.get_dummies(train["family"])
for key in family:
    train["family:"+key] = family[key]
del (train["family"])
```

مرحله دوم

ساخت درخت تصمیم‌گیری

ستون class را در output قرار می‌دهیم و باقی ستون‌ها را در input.

با استفاده از کتابخانه scikit learn.tree و همچنین تابع DecisionTreeClassifier() درخت را می‌سازیم و با داده‌های input و output fit می‌کنیم.

مرحله سوم

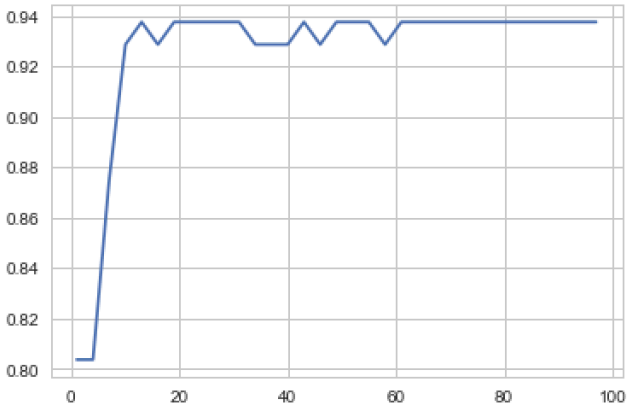
بهبود درخت تصمیم‌گیری

حال پارامترها را با مقدارهای مختلف بررسی می‌کنیم و نمودار آن را رسم می‌کنیم. به عنوان نمونه به صورت زیر عمل می‌کنیم:

```
max_depth={}
for i in range(1,100,3):
    clf.set_params(max_depth = i)
    clf.fit(input[0:683],output[0:683])
    max_depth[i]=clf.score(input[684:796],output[684:796])
print(max_depth)
max_depth={}
for i in range(1,100,3):
    clf.set_params(max_depth = i)
    clf.fit(input[0:683],output[0:683])
    max_depth[i]=clf.score(input[684:796],output[684:796])
print(max_depth)
```

نمودارهای تولید شده به شکل زیر است:

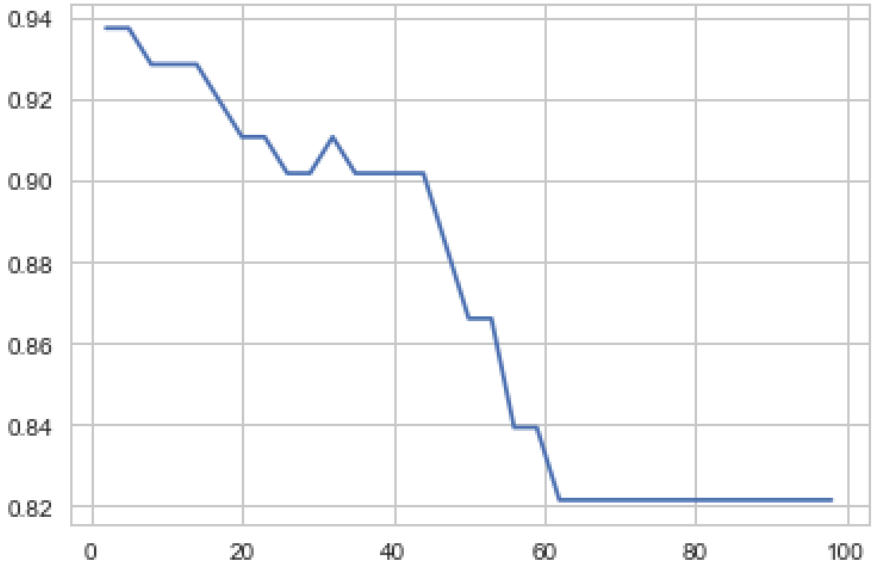
```
Out[4]: [<matplotlib.lines.Line2D at 0x11bba9518>]
```



max_depth

تا یه جایی افزایش می‌یابد و از یه مقداری به بعد تأثیر بسیاری روی score ندارد و تنها بار محاسباتی را زیاد می‌کند.

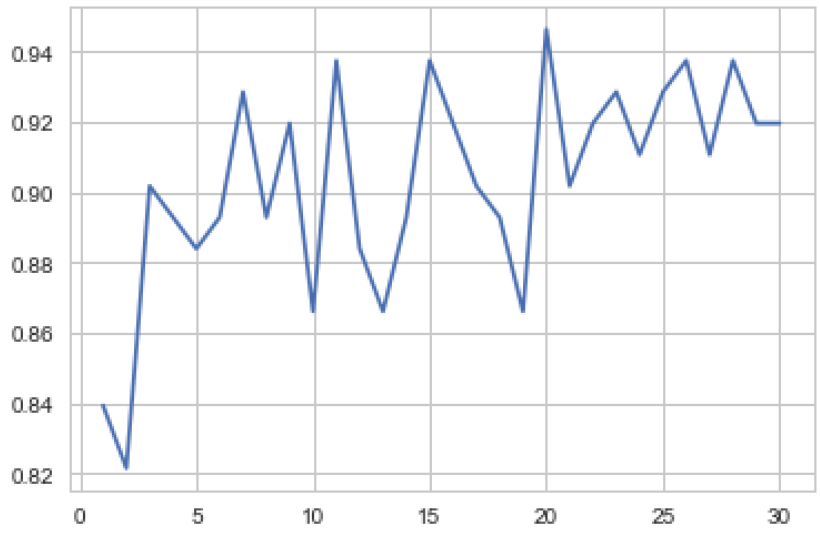
```
Out[6]: [<matplotlib.lines.Line2D at 0x11bcb76a0>]
```



min_samples_split

همان‌طور که مشخص است با افزایش آن score کاهش می‌یابد.

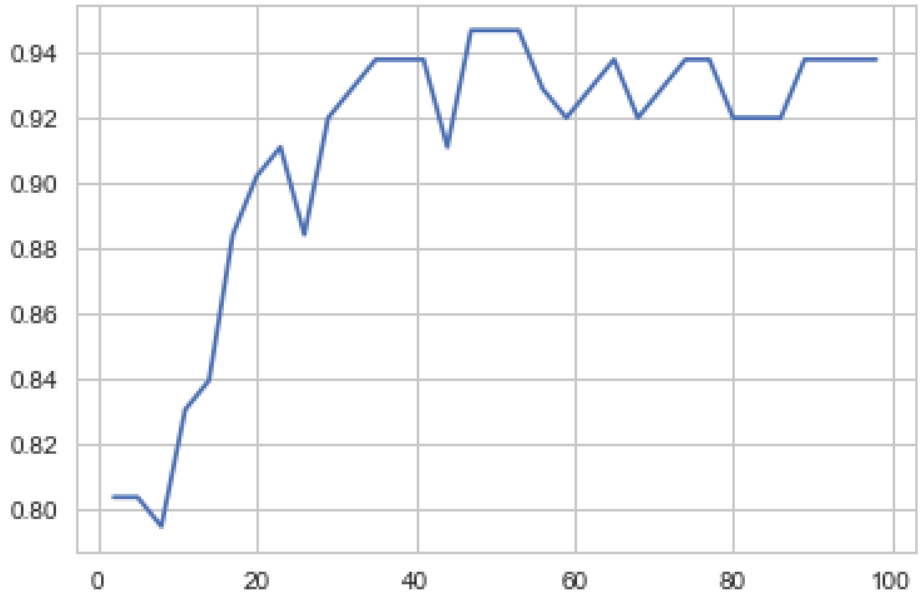
Out[10]: `[<matplotlib.lines.Line2D at 0x11c12a4a8>]`



max_features

به صورت ثابت افزایش یا کاهش نمی‌یابد ولی به طور کلی افزایش می‌یابد.
به همین روش می‌توان باقی پارامترها را هم بررسی کرد.

Out[12]: `[<matplotlib.lines.Line2D at 0x11c052860>]`



max_leaf_nodes

splitter: {'random': 0.9107142857142857, 'best': 0.9464285714285714}

criterion: {'gini': 0.9285714285714286, 'entropy': 0.9464285714285714}

حال برای پیدا کردن بهترین مقادیر از grid_search استفاده می‌کنیم: برای کاهش محاسبات از نمودارها جهت وارد کردن ورودی‌های grid_search استفاده می‌کنیم.
{'max_depth': 50, 'max_features': 20, 'max_leaf_nodes': 50, 'min_samples_split': 5}

با استفاده از پارامترها مقدار score بدست آمده 0.901 می‌شود که باز هم حالت بهینه نیست و همان‌طور که در نمودارها آمده است در بسیاری از موارد score به 0.95 هم رسیده است.

حال با تغییر پارامتر class_weight می‌توان میزان تأثیر هر ویژگی را تغییر داد اما قبل از آن بهتر است تشخیص دهیم بهترین حالت پیدا شده چه مقادیری دارد که به شرح زیر خواهد بود:

```
'}carbon': 0.061459088024186904, 'hardness': 0.18580694656005101, 'strength':
0.021595644588152489, 'thick': 0.23615899990039405, 'width': 0.10220203410340042,
'len': 0.009620477653732952, 'bore': 0.0, 'family:TN': 0.0, 'family:ZS':
0.060859602923602249, 'steel:A': 0.059520663176287195, 'steel:K': 0.0, 'steel:M':
0.0084648384315350201, 'steel:R': 0.013624358999327795, 'steel:S':
0.022092298104280966, 'steel:V': 0.012729648610688512, 'steel:W':
0.024349660461374451, 'condition:A': 0.0, 'condition:S': 0.0050789030589210131,
'formability:1': 0.0065637523264641768, 'formability:2': 0.05919031607237283,
'formability:3': 0.0072555757984585885, 'formability:5': 0.0, 'surface-quality:D':
0.0097190272305461827, 'surface-quality:E': 0.027956561707265799, 'surface-
quality:F': 0.0032320292193133704, 'surface-quality:G': 0.008551214333897628,
'bw/me:B': 0.01246784873939717, 'bw/me:M': 0.0, 'shape:COIL': 0.030012514962123031,
'shape:SHEET': 0.011487995014226099}
```

همان‌طور که مشخص است در این حالت thick با میزان تأثیر 0.2361589999 بیشترین تأثیر را بر روی نتیجه دارد.
با تغییر دادن این میزان تأثیرگذاری‌ها می‌توان به score مناسب دست یافت.
با اضافه کردن میزان تأثیرگذاری formability:3 به thick، score نهایی افزایش یافت.