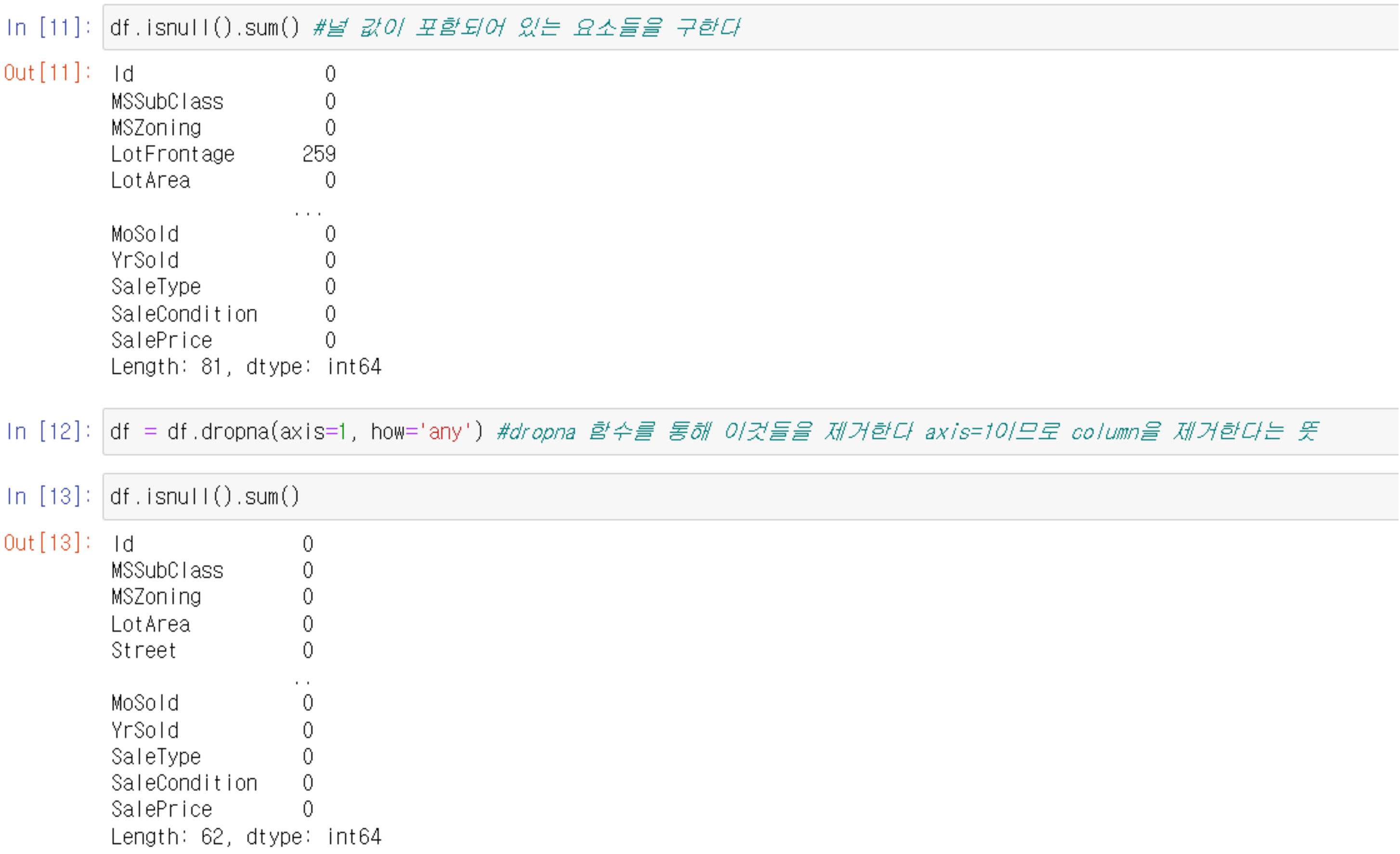
6주차 Basic 발표

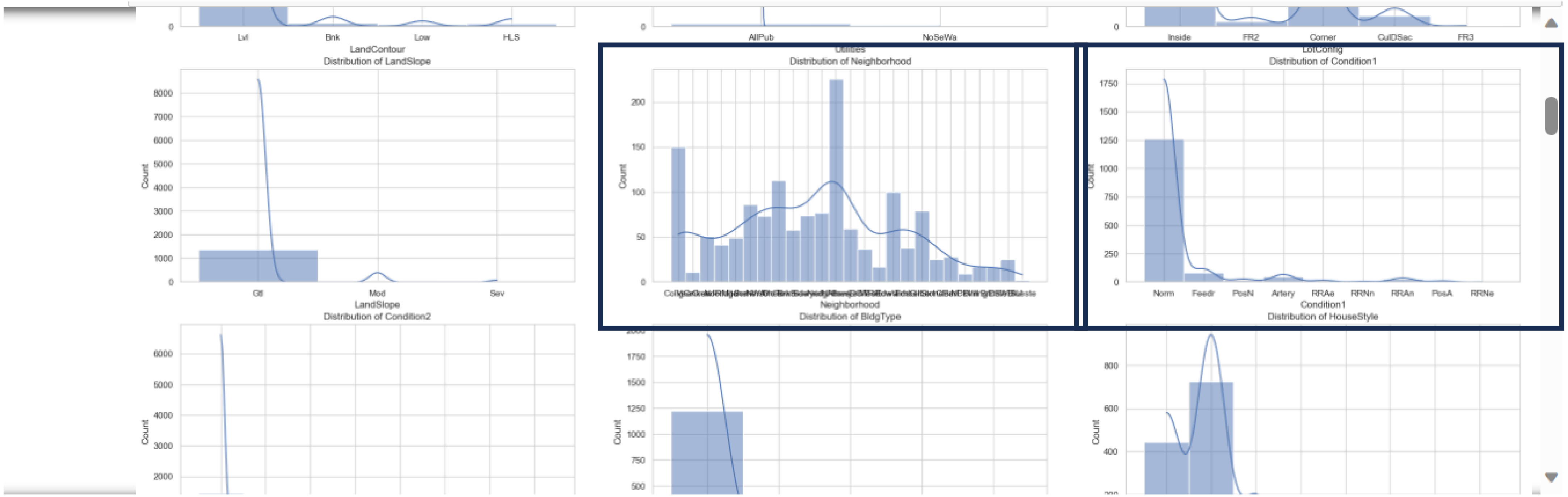
김찬원

df.isnull().sum()



```
In [11]: df.isnull().sum() #널 값이 포함되어 있는 요소들을 구한다

Out[11]: Id                 0
         MSSubClass         0
         MSZoning           0
         LotFrontage      259
         LotArea            0
                         ...
         MoSold             0
         YrSold             0
         SaleType           0
         SaleCondition      0
         SalePrice          0
         Length: 81, dtype: int64

In [12]: df = df.dropna(axis=1, how='any') #dropna 함수를 통해 이것들을 제거한다 axis=1이므로 column을 제거한다는 뜻

In [13]: df.isnull().sum()

Out[13]: Id               0
         MSSubClass       0
         MSZoning         0
         LotArea          0
         Street           0
                         ..
         MoSold           0
         YrSold           0
         SaleType         0
         SaleCondition    0
         SalePrice        0
         Length: 62, dtype: int64
```

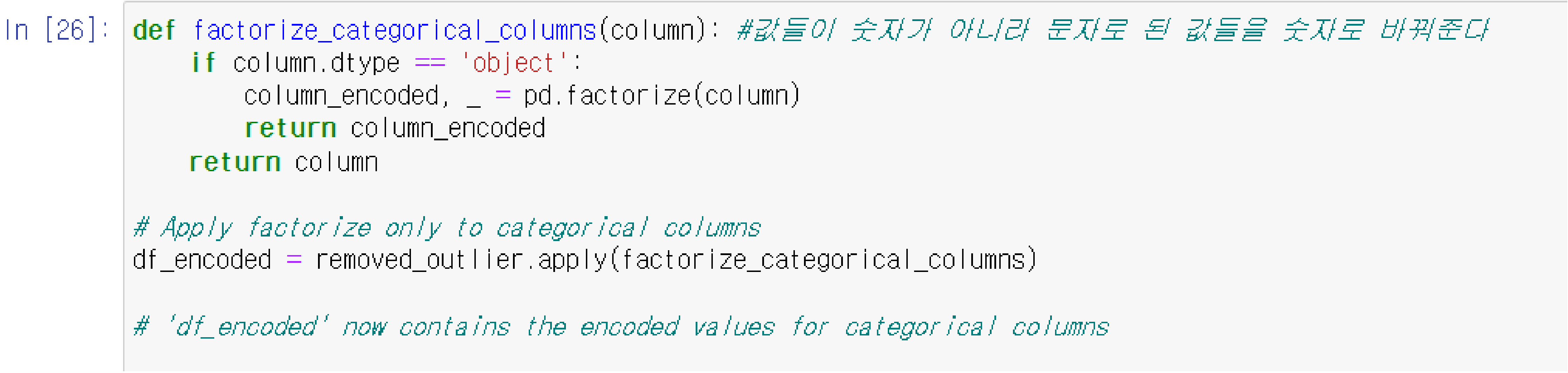
Training에 필요 없는 column 제거



we can clearly see that there are outlier present in our datset

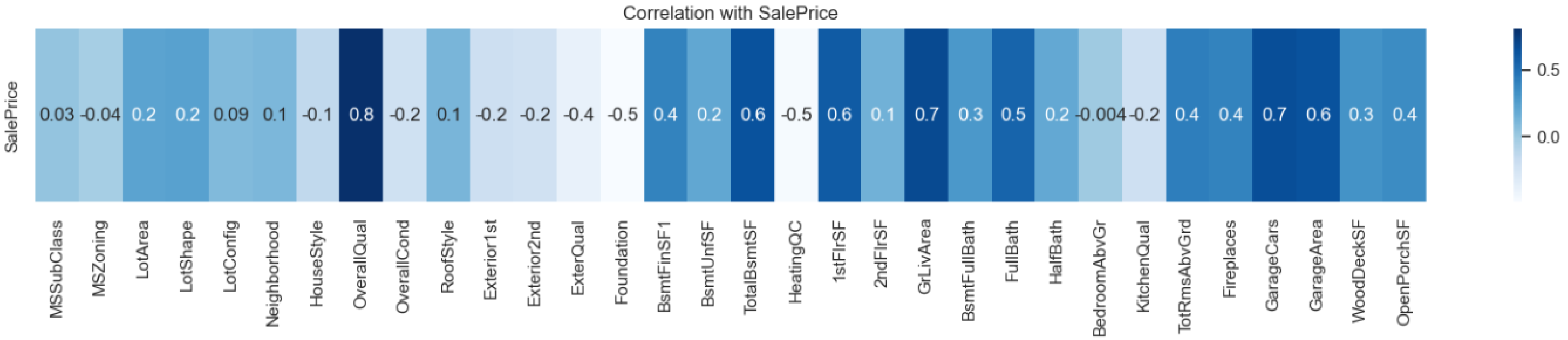
```
In [20]: #여기서 특정 항에만 너무 많은 값들이 모여 training에 도움이 되지 않는 column등을 구해 이들을 추후에 빼버린다.
         column_are_not_helpful_in_prediction=ptr_df[['Street','LandContour','Utilities','LandSlope','Condition1','Condition2','RoofMatl',
                                                     'ExterCond','BsmtFinSF2','Heating','CentralAir','LowQualFinSF','BsmtHalfBath',
                                                     'KitchenAbvGr','Functional','PavedDrive','EnclosedPorch','3SsnPorch','ScreenPorch',
                                                     'PoolArea','MiscVal','SaleType','SaleCondition','BldgType']]
         column_are_not_helpful_in_prediction
```

factorize_categorical_columns(column)

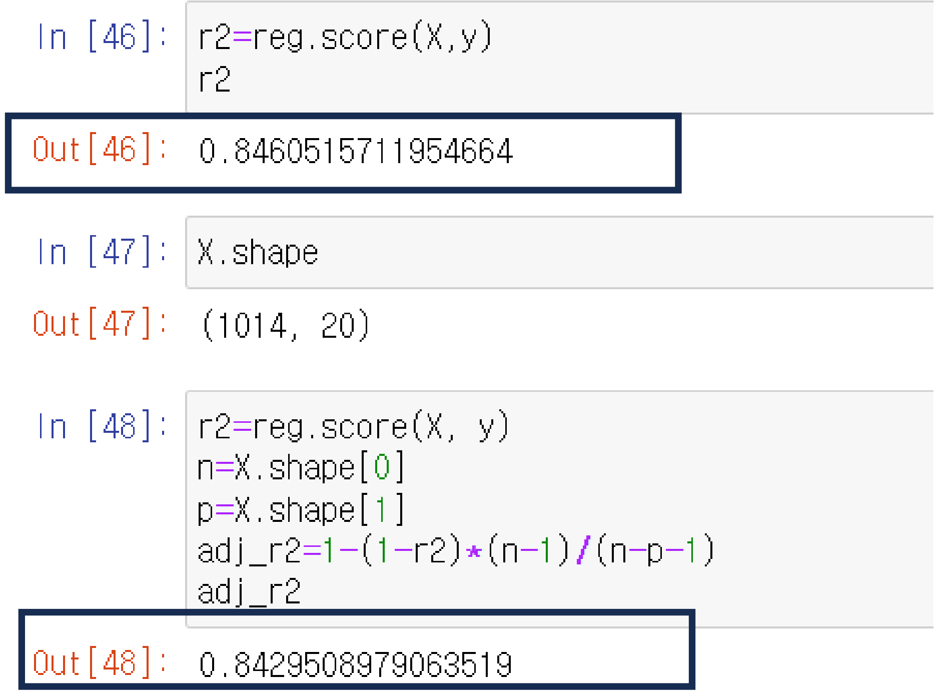


```
In [26]:
def factorize_categorical_columns(column): #값들이 숫자가 아니라 문자로 된 값들을 숫자로 바꿔준다
    if column.dtype == 'object':
        column_encoded, _ = pd.factorize(column)
        return column_encoded
    return column

# Apply factorize only to categorical columns
df_encoded = removed_outlier.apply(factorize_categorical_columns)

# 'df_encoded' now contains the encoded values for categorical columns
```

correlation



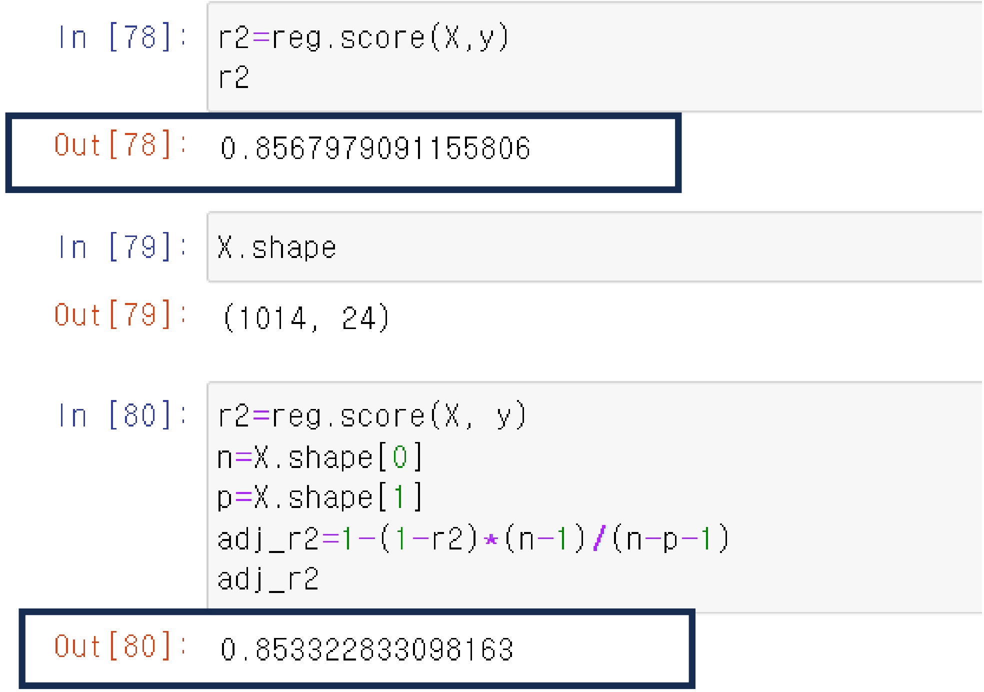
Baseline 코드 성능



```
In [46]: r2=reg.score(X,y)
         r2

Out[46]: 0.8460515711954664

In [47]: X.shape

Out[47]: (1014, 20)

In [48]: r2=reg.score(X, y)
         n=X.shape[0]
         p=X.shape[1]
         adj_r2=1-(1-r2)*(n-1)/(n-p-1)
         adj_r2

Out[48]: 0.8429508979063519
```

성능개선1.

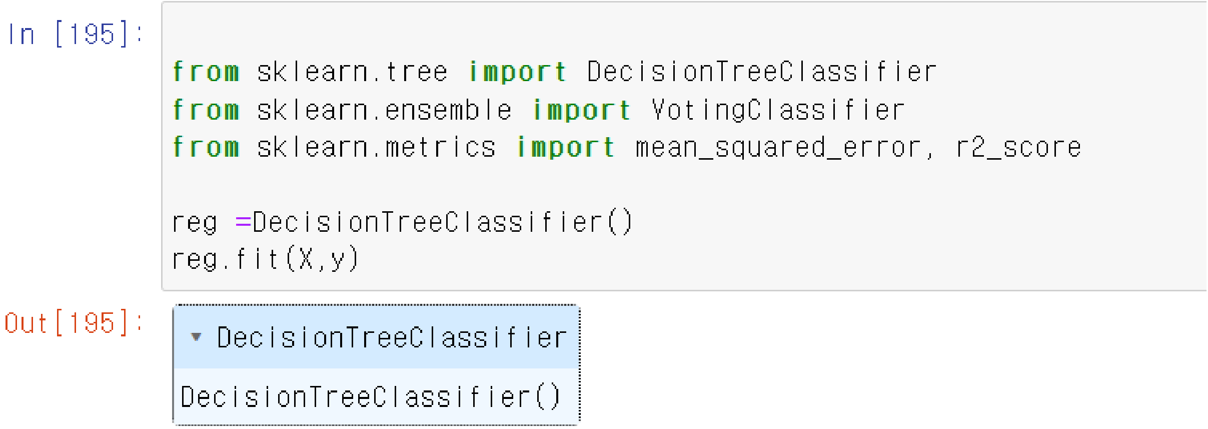
Training에 불필요하다가 생각하여 지운 column들 중 data분포가 너무 치우치지 않은 column 몇 개를 살림



```
In [78]: r2=reg.score(X,y)
         r2

Out[78]: 0.8567979091155806

In [79]: X.shape

Out[79]: (1014, 24)

In [80]: r2=reg.score(X, y)
         n=X.shape[0]
         p=X.shape[1]
         adj_r2=1-(1-r2)*(n-1)/(n-p-1)
         adj_r2

Out[80]: 0.853322833098163
```

성능개선2.

DescisionTreeClassifier 사용

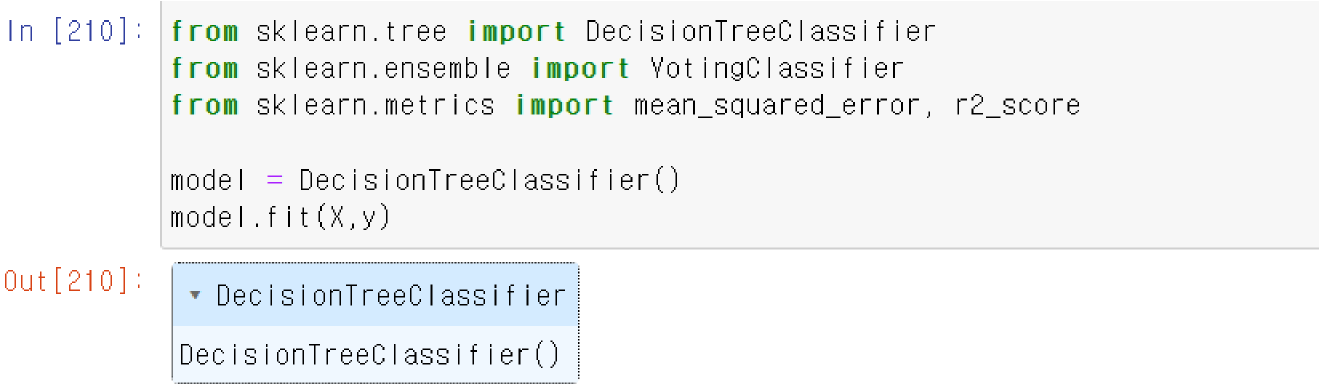


In [195]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import mean_squared_error, r2_score

reg =DecisionTreeClassifier()
reg.fit(X,y)
```

Out[195]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```



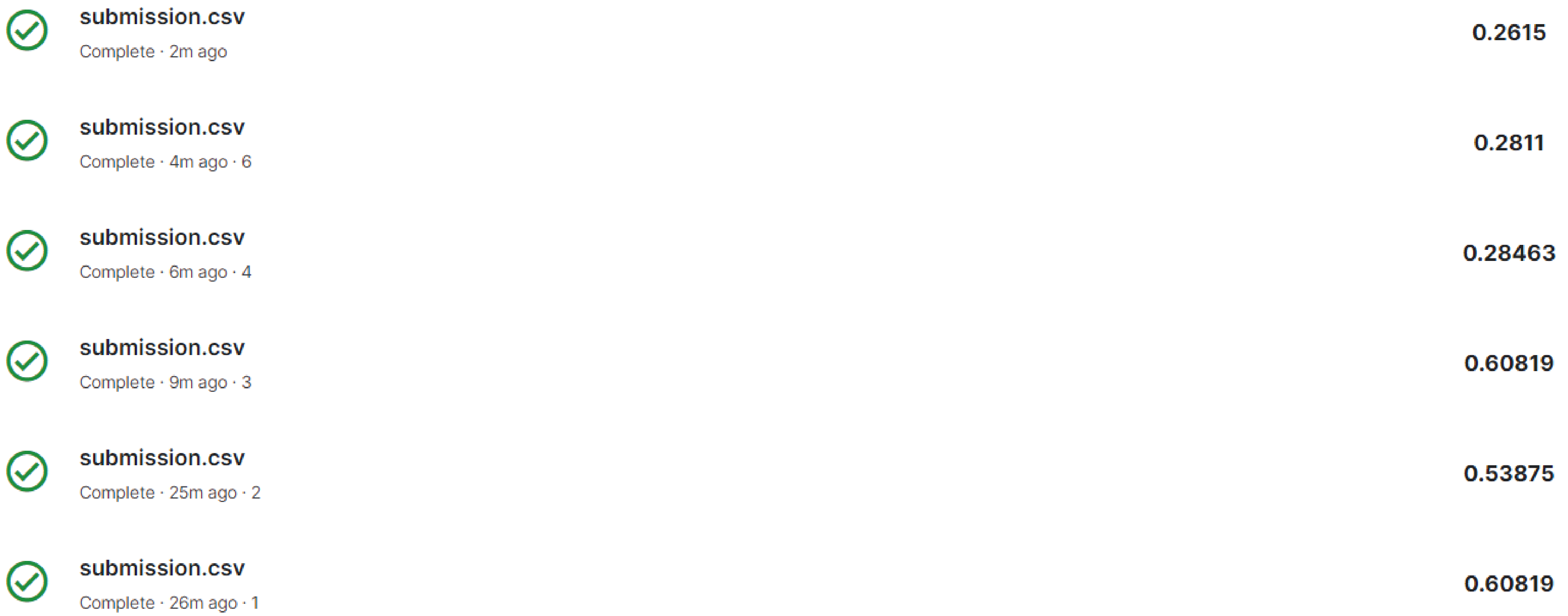
In [210]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import mean_squared_error, r2_score

model = DecisionTreeClassifier()
model.fit(X,y)
```

Out[210]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

최종결과



✓	submission.csv Complete · 2m ago	0.2615
✓	submission.csv Complete · 4m ago · 6	0.2811
✓	submission.csv Complete · 6m ago · 4	0.28463
✓	submission.csv Complete · 9m ago · 3	0.60819
✓	submission.csv Complete · 25m ago · 2	0.53875
✓	submission.csv Complete · 26m ago · 1	0.60819