이 집의 가격은??

Ybigta *summer project*

11기 김현우, 윤정수, 한솔

(feat. 10기 박승리, 채혜진)

Index

EDA

탐색적 자료분석

Data Preprocessing

데이터 전처리

Modeling

모델 만들기

Analysis & Conclusion

결과분석 & 결론

EDA 데이터 확인

80

EDA 데이터 확인

80

= index를 제외한 column의 개수

EDA 데이터 확인

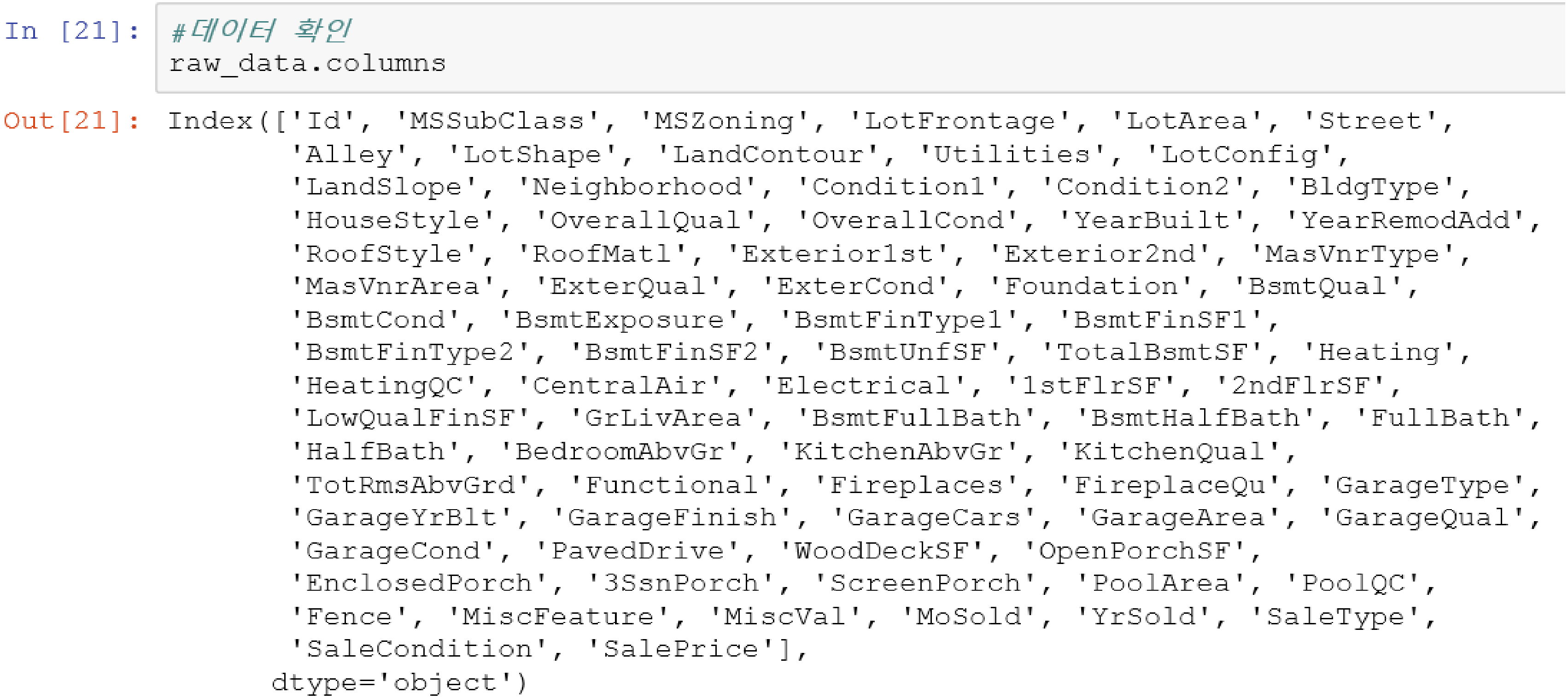
1459

EDA *데이터 확인*

1459

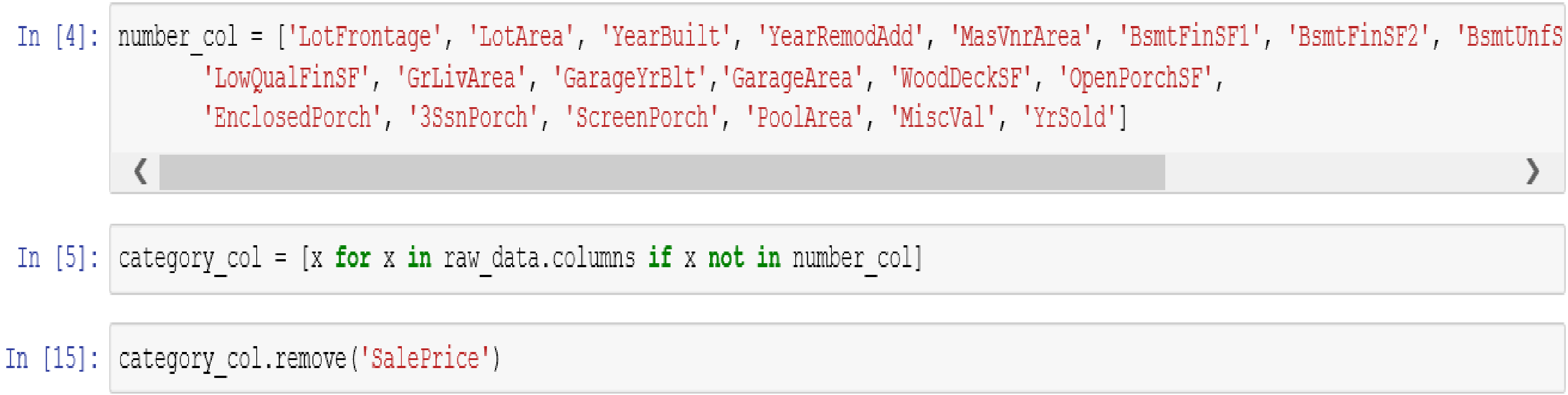
= 데이터의 개수

EDA 데이터 확인



```
In [21]: #데이터 확인
         raw_data.columns

Out[21]: Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
                'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
                'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
                'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
                'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
                'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
                'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
                'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
                'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
                'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
                'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
                'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
                'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
                'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
                'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
                'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
                'SaleCondition', 'SalePrice'],
               dtype='object')
```

EDA 명목형 변수와 연속형 변수 나누기

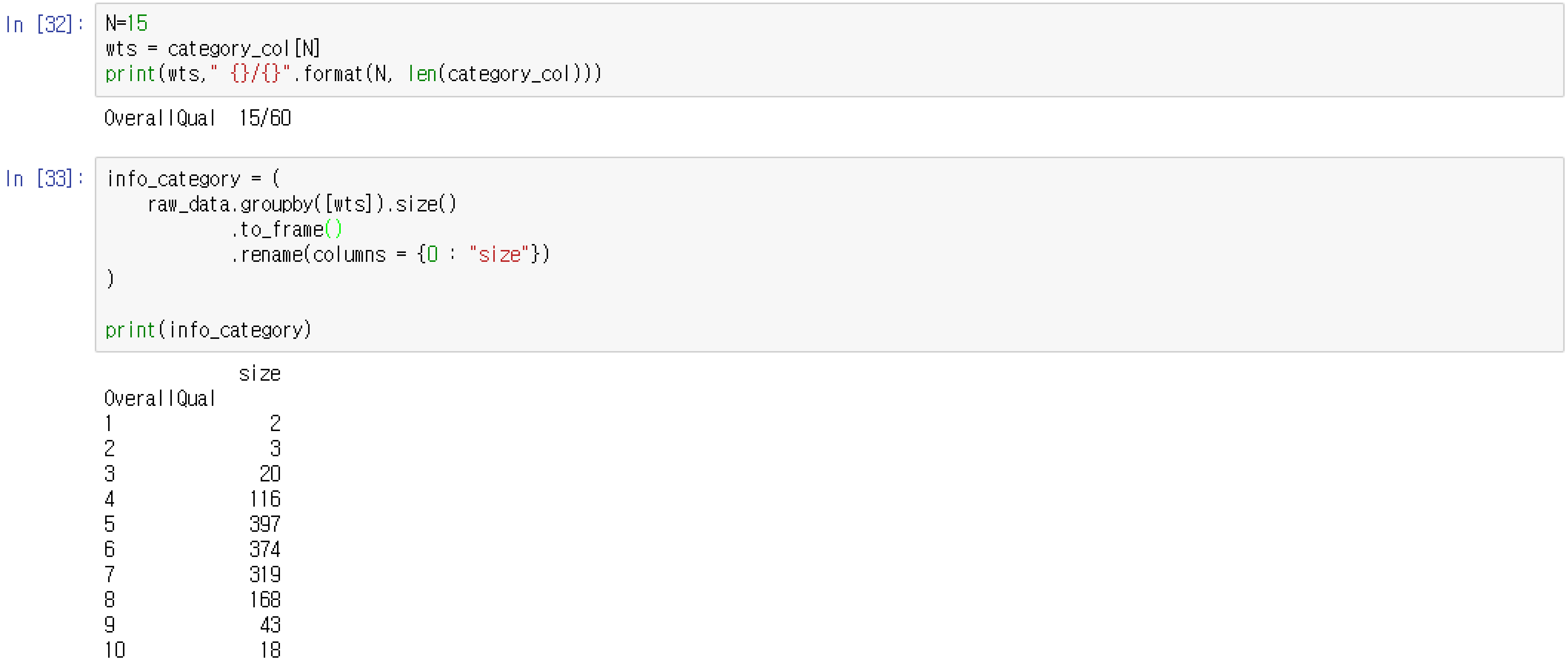


```
In [4]: number_col = ['LotFrontage', 'LotArea', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfS
               'LowQualFinSF', 'GrLivArea', 'GarageYrBlt','GarageArea', 'WoodDeckSF', 'OpenPorchSF',
               'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'YrSold']
```

```
In [5]: category_col = [x for x in raw_data.columns if x not in number_col]
```

```
In [15]: category_col.remove('SalePrice')
```

EDA 변수 탐색



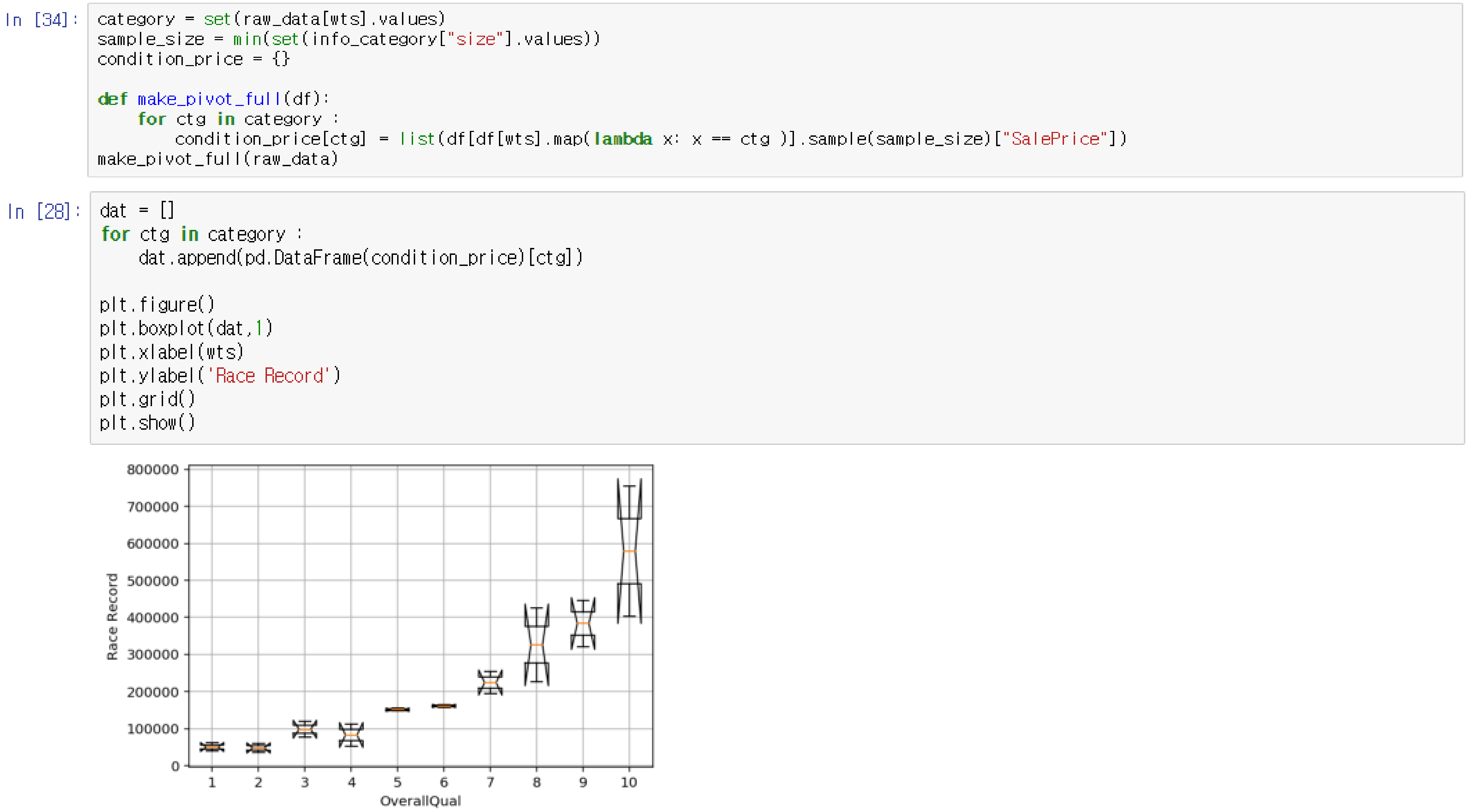
In [32]:
```
N=15
wts = category_col[N]
print(wts," {}/{}".format(N, len(category_col)))
```

```
OverallQual  15/60
```

In [33]:
```
info_category = (
    raw_data.groupby([wts]).size()
            .to_frame()
            .rename(columns = {0 : "size"})
)

print(info_category)
```

```
             size
OverallQual
1               2
2               3
3              20
4             116
5             397
6             374
7             319
8             168
9              43
10             18
```

EDA 변수 탐색



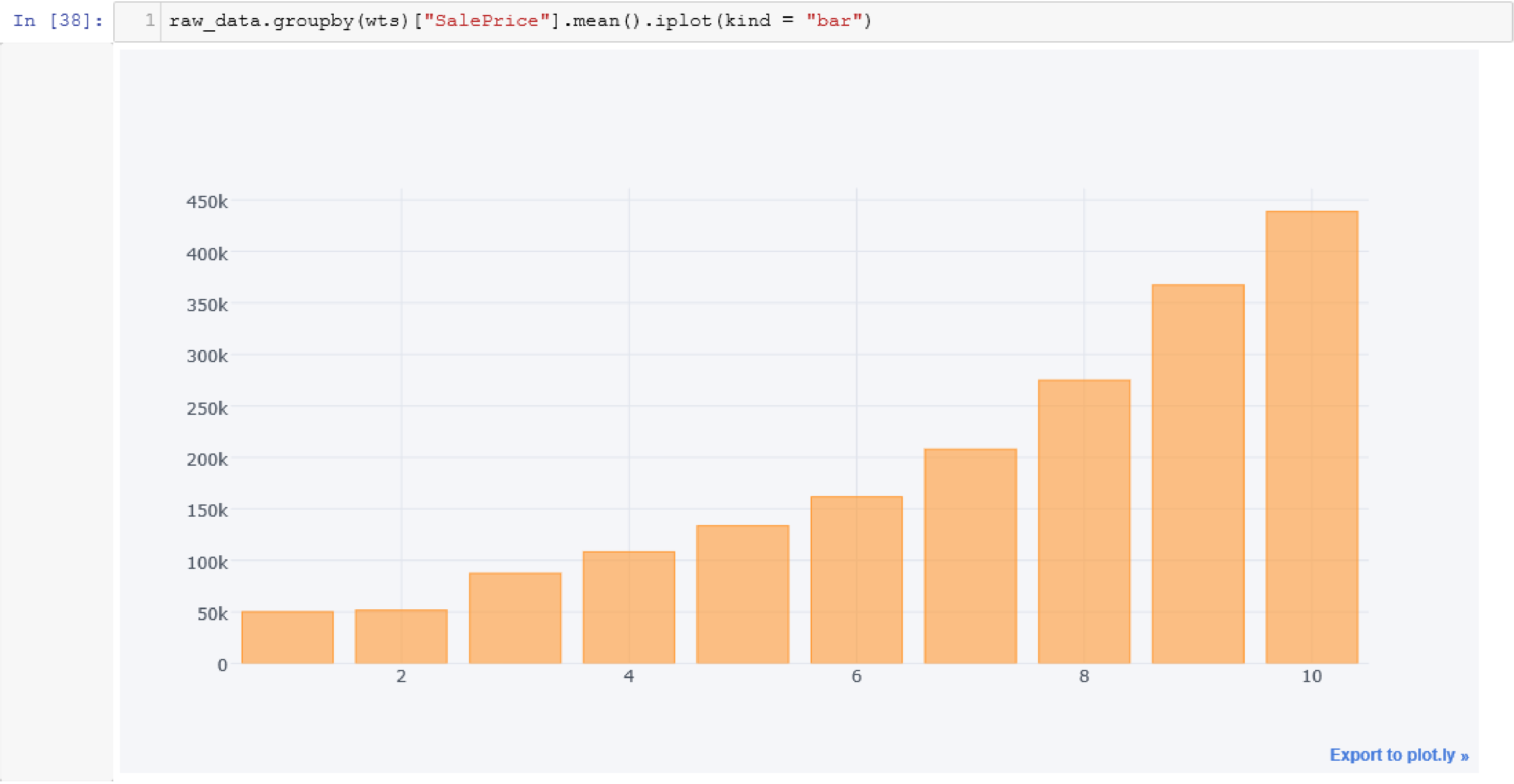
In [34]:

```
category = set(raw_data[wts].values)
sample_size = min(set(info_category["size"].values))
condition_price = {}

def make_pivot_full(df):
    for ctg in category :
        condition_price[ctg] = list(df[df[wts].map( lambda x: x == ctg )].sample(sample_size)["SalePrice"])
make_pivot_full(raw_data)
```

In [28]:

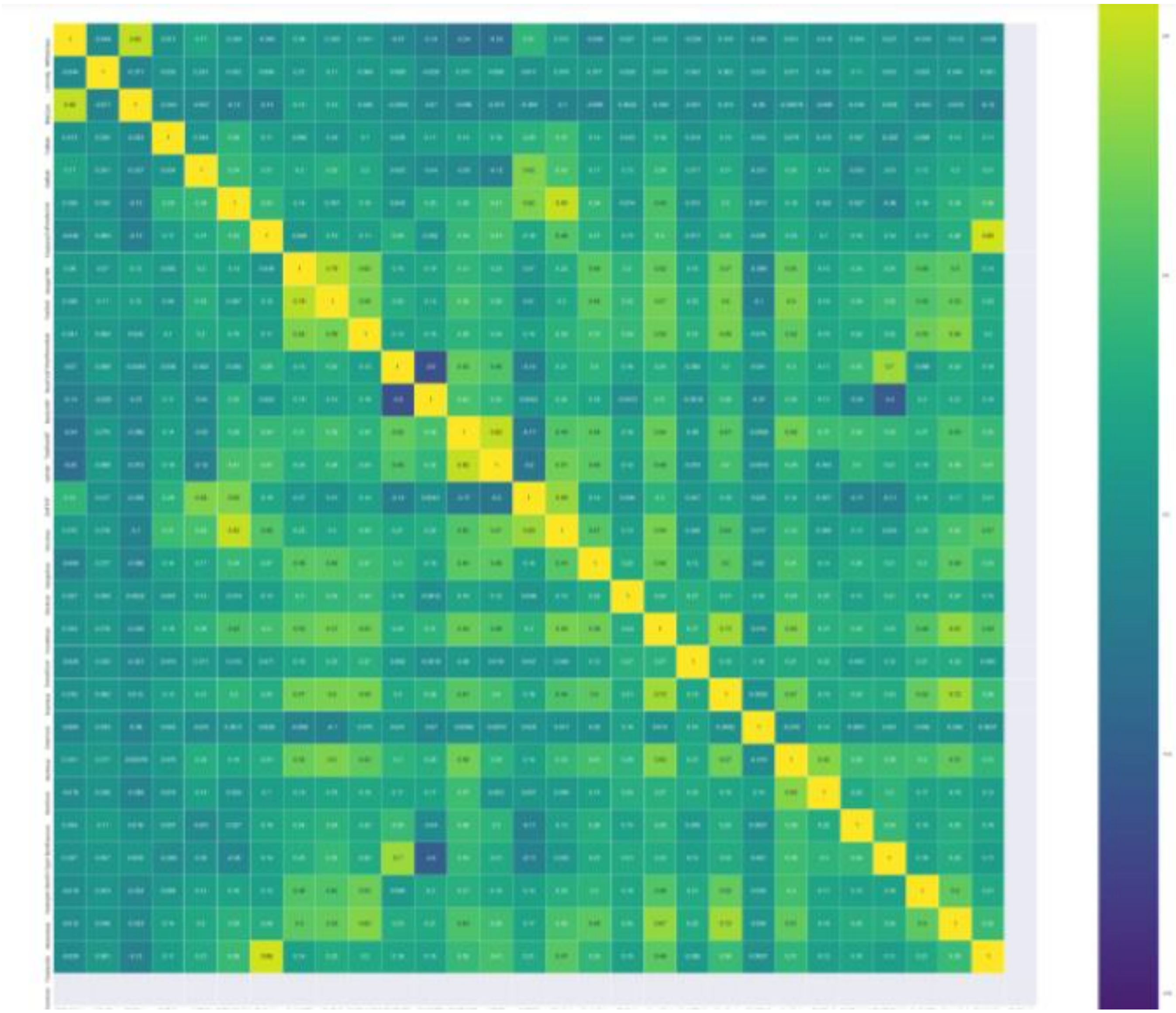
```
dat = []
for ctg in category :
    dat.append(pd.DataFrame(condition_price)[ctg])

plt.figure()
plt.boxplot(dat,1)
plt.xlabel(wts)
plt.ylabel('Race Record')
plt.grid()
plt.show()
```

EDA 변수 탐색

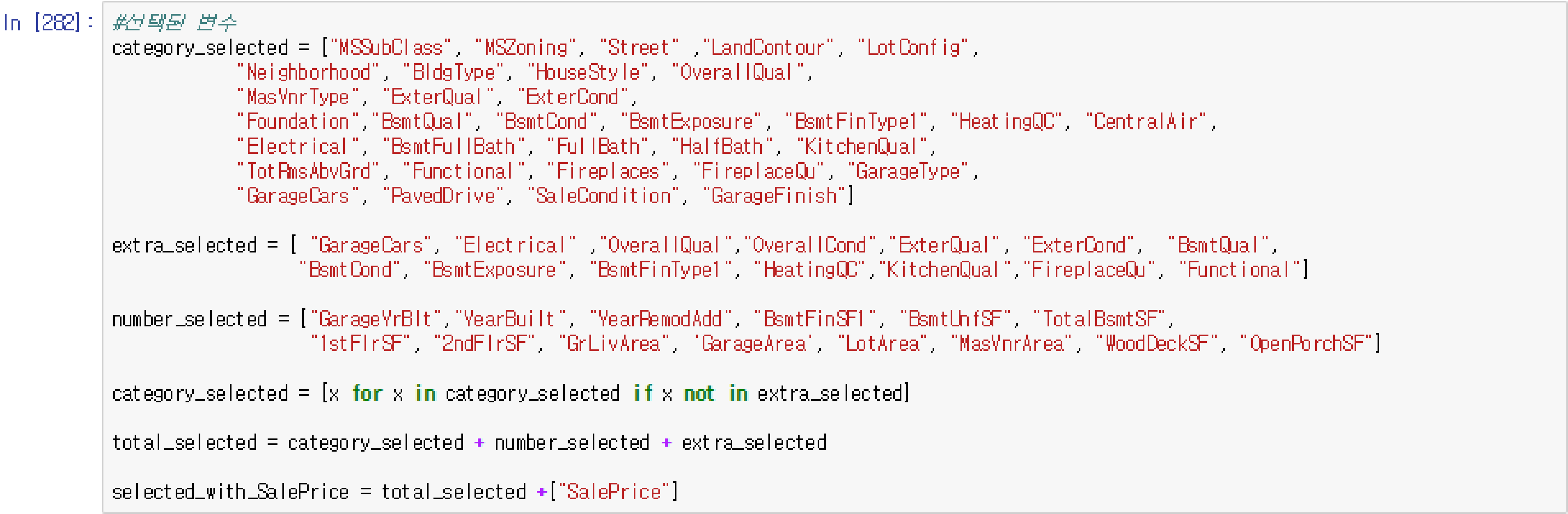


```
In [38]:  raw_data.groupby(wts)["SalePrice"].mean().iplot(kind = "bar")
```

EDA 변수간의 관계

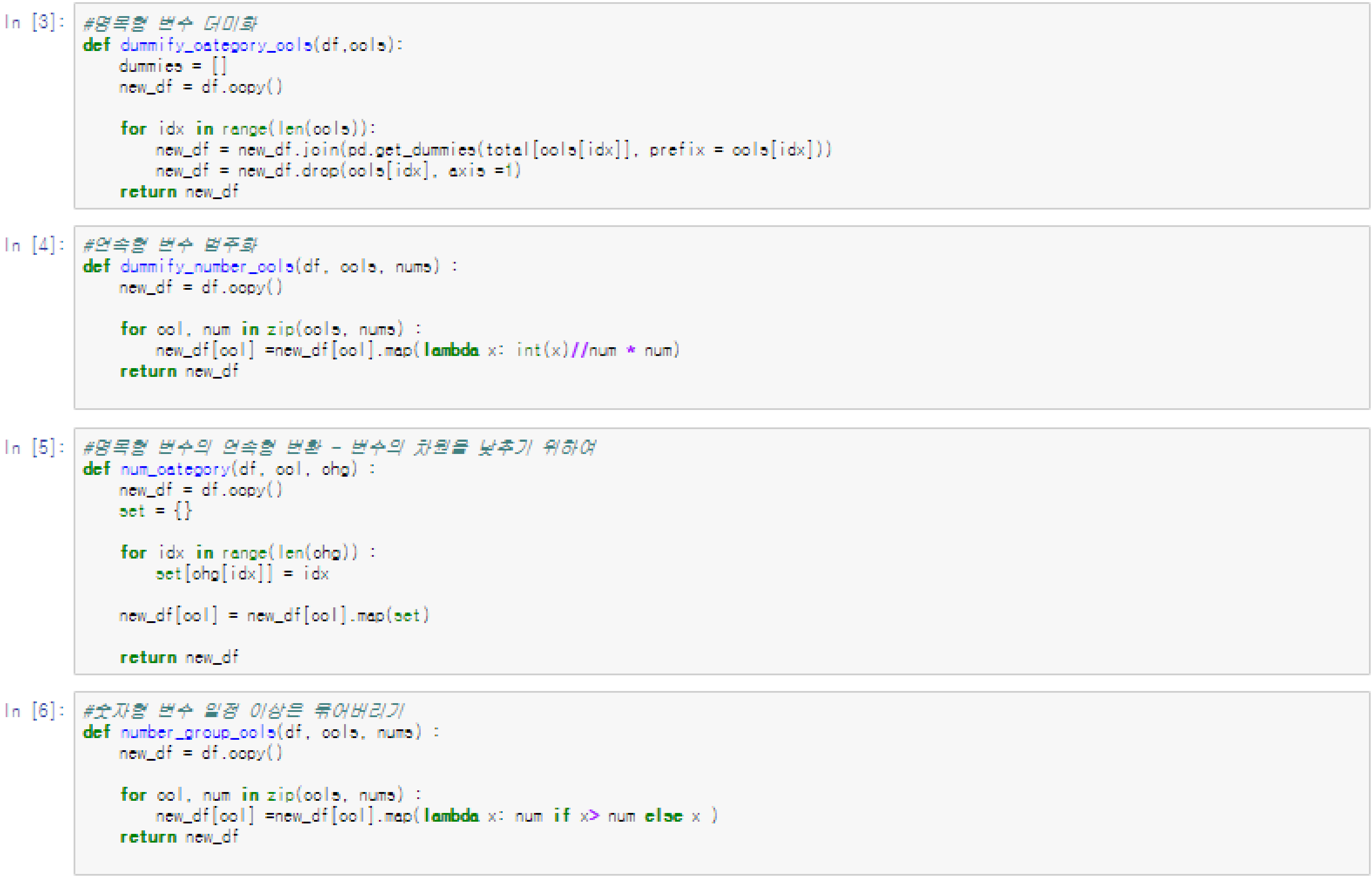


EDA 변수선택



```
In [282]:
#선택된 변수
category_selected = ["MSSubClass", "MSZoning", "Street" ,"LandContour", "LotConfig",
            "Neighborhood", "BldgType", "HouseStyle", "OverallQual",
            "MasVnrType", "ExterQual", "ExterCond",
            "Foundation","BsmtQual", "BsmtCond", "BsmtExposure", "BsmtFinType1", "HeatingQC", "CentralAir",
            "Electrical", "BsmtFullBath", "FullBath", "HalfBath", "KitchenQual",
            "TotRmsAbvGrd", "Functional", "Fireplaces", "FireplaceQu", "GarageType",
            "GarageCars", "PavedDrive", "SaleCondition", "GarageFinish"]

extra_selected = [ "GarageCars", "Electrical" ,"OverallQual","OverallCond","ExterQual", "ExterCond",  "BsmtQual",
                  "BsmtCond", "BsmtExposure", "BsmtFinType1", "HeatingQC","KitchenQual","FireplaceQu", "Functional"]

number_selected = ["GarageYrBlt","YearBuilt", "YearRemodAdd", "BsmtFinSF1", "BsmtUnfSF", "TotalBsmtSF",
                   "1stFlrSF", "2ndFlrSF", "GrLivArea", 'GarageArea', "LotArea", "MasVnrArea", "WoodDeckSF", "OpenPorchSF"]

category_selected = [x for x in category_selected if x not in extra_selected]

total_selected = category_selected + number_selected + extra_selected

selected_with_SalePrice = total_selected +["SalePrice"]
```

PreProcessing 함수 만들기



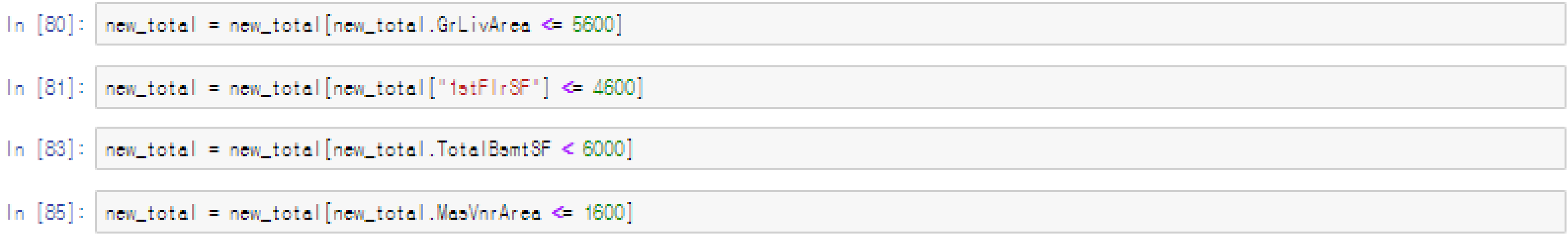
```
In [3]: #명목형 변수 더미화
        def dummify_category_cols(df,cols):
            dummies = []
            new_df = df.copy()

            for idx in range(len(cols)):
                new_df = new_df.join(pd.get_dummies(total[cols[idx]], prefix = cols[idx]))
                new_df = new_df.drop(cols[idx], axis =1)
            return new_df
```

```
In [4]: #연속형 변수 범주화
        def dummify_number_cols(df, cols, nums) :
            new_df = df.copy()

            for col, num in zip(cols, nums) :
                new_df[col] =new_df[col].map( lambda x: int(x)//num * num)
            return new_df
```

```
In [5]: #명목형 변수의 연속형 변환 - 변수의 차원을 낮추기 위하여
        def num_category(df, col, chg) :
            new_df = df.copy()
            set = {}

            for idx in range(len(chg)) :
                set[chg[idx]] = idx

            new_df[col] = new_df[col].map(set)

            return new_df
```

```
In [6]: #숫자형 변수 일정 이상은 묶어버리기
        def number_group_cols(df, cols, nums) :
            new_df = df.copy()

            for col, num in zip(cols, nums) :
                new_df[col] =new_df[col].map( lambda x: num if x> num else x )
            return new_df
```

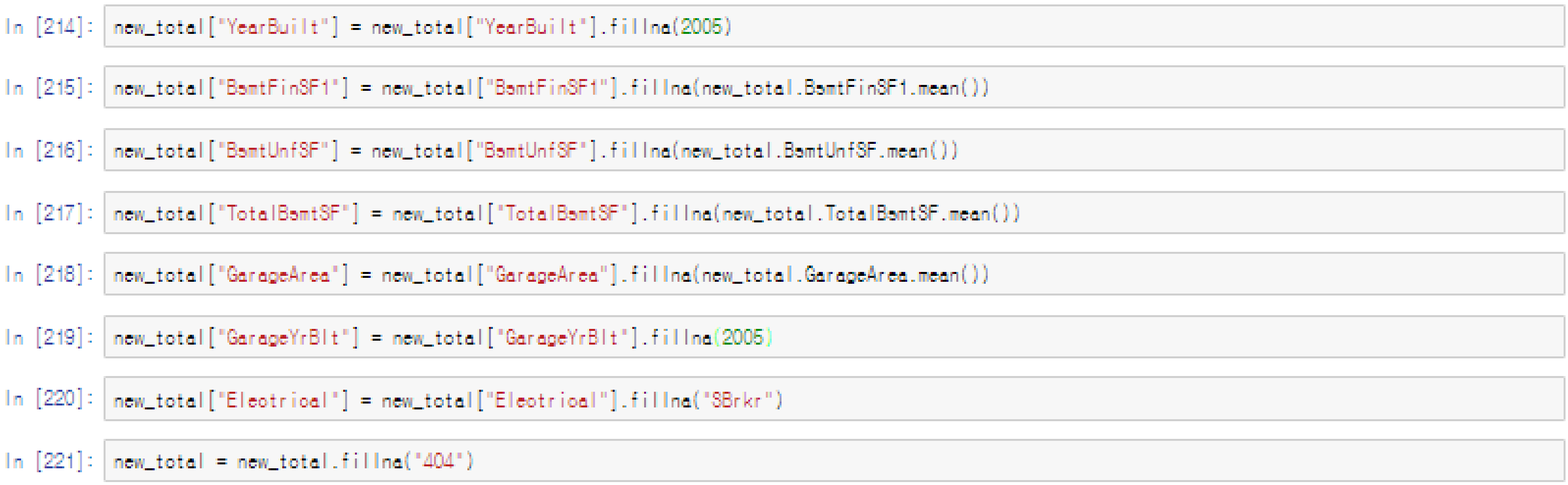
PreProcessing *특이값 삭제*



```
In [80]: new_total = new_total[new_total.GrLivArea <= 5600]

In [81]: new_total = new_total[new_total["1stFlrSF"] <= 4600]

In [83]: new_total = new_total[new_total.TotalBsmtSF < 6000]

In [85]: new_total = new_total[new_total.MasVnrArea <= 1600]
```

(버리지 않는 것이 더 좋은 결과가 나왔습니다…ㅜㅜ)

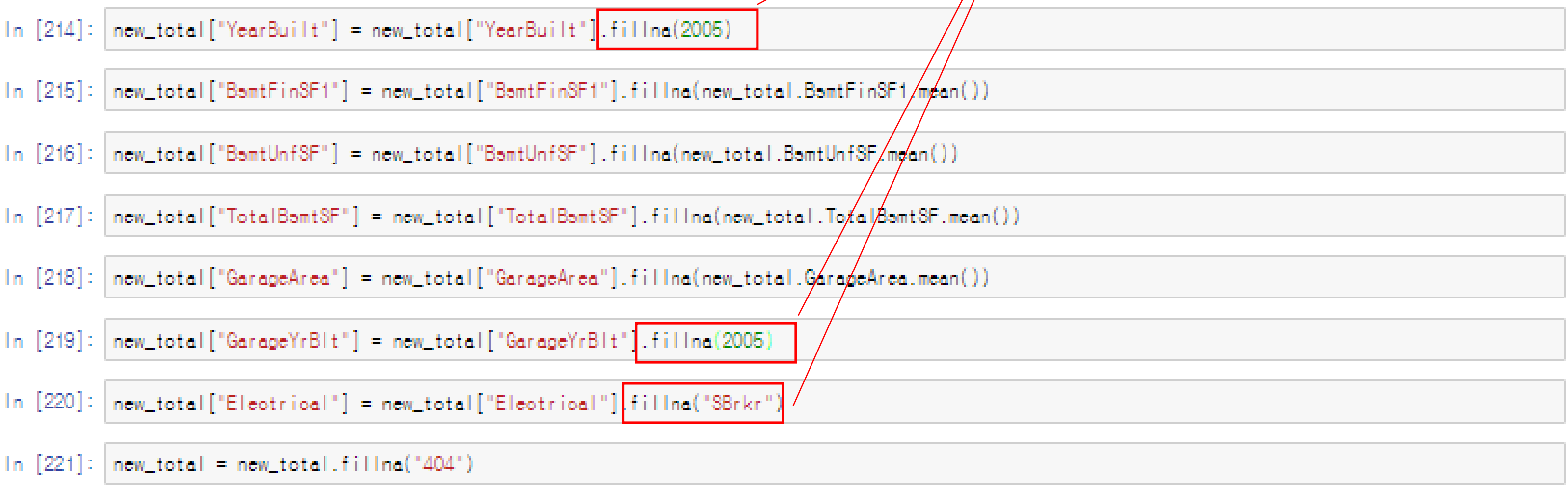
PreProcessing *Null값 채워주기*



```
In [214]: new_total["YearBuilt"] = new_total["YearBuilt"].fillna(2005)

In [215]: new_total["BsmtFinSF1"] = new_total["BsmtFinSF1"].fillna(new_total.BsmtFinSF1.mean())

In [216]: new_total["BsmtUnfSF"] = new_total["BsmtUnfSF"].fillna(new_total.BsmtUnfSF.mean())

In [217]: new_total["TotalBsmtSF"] = new_total["TotalBsmtSF"].fillna(new_total.TotalBsmtSF.mean())

In [218]: new_total["GarageArea"] = new_total["GarageArea"].fillna(new_total.GarageArea.mean())

In [219]: new_total["GarageYrBlt"] = new_total["GarageYrBlt"].fillna(2005)

In [220]: new_total["Electrical"] = new_total["Electrical"].fillna("SBrkr")

In [221]: new_total = new_total.fillna("404")
```

PreProcessing *Null값 채워주기*

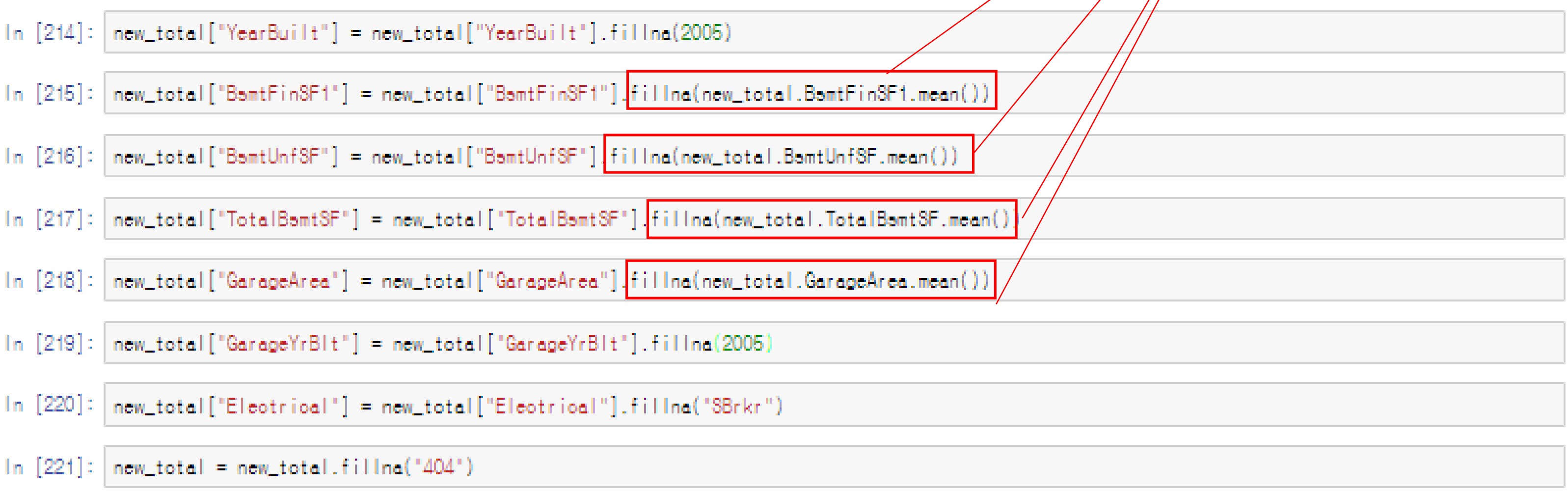
최빈값으로 채우기



```
In [214]: new_total["YearBuilt"] = new_total["YearBuilt"].fillna(2005)

In [215]: new_total["BsmtFinSF1"] = new_total["BsmtFinSF1"].fillna(new_total.BsmtFinSF1.mean())

In [216]: new_total["BsmtUnfSF"] = new_total["BsmtUnfSF"].fillna(new_total.BsmtUnfSF.mean())

In [217]: new_total["TotalBsmtSF"] = new_total["TotalBsmtSF"].fillna(new_total.TotalBsmtSF.mean())

In [218]: new_total["GarageArea"] = new_total["GarageArea"].fillna(new_total.GarageArea.mean())

In [219]: new_total["GarageYrBlt"] = new_total["GarageYrBlt"].fillna(2005)

In [220]: new_total["Electrical"] = new_total["Electrical"].fillna("SBrkr")

In [221]: new_total = new_total.fillna("404")
```

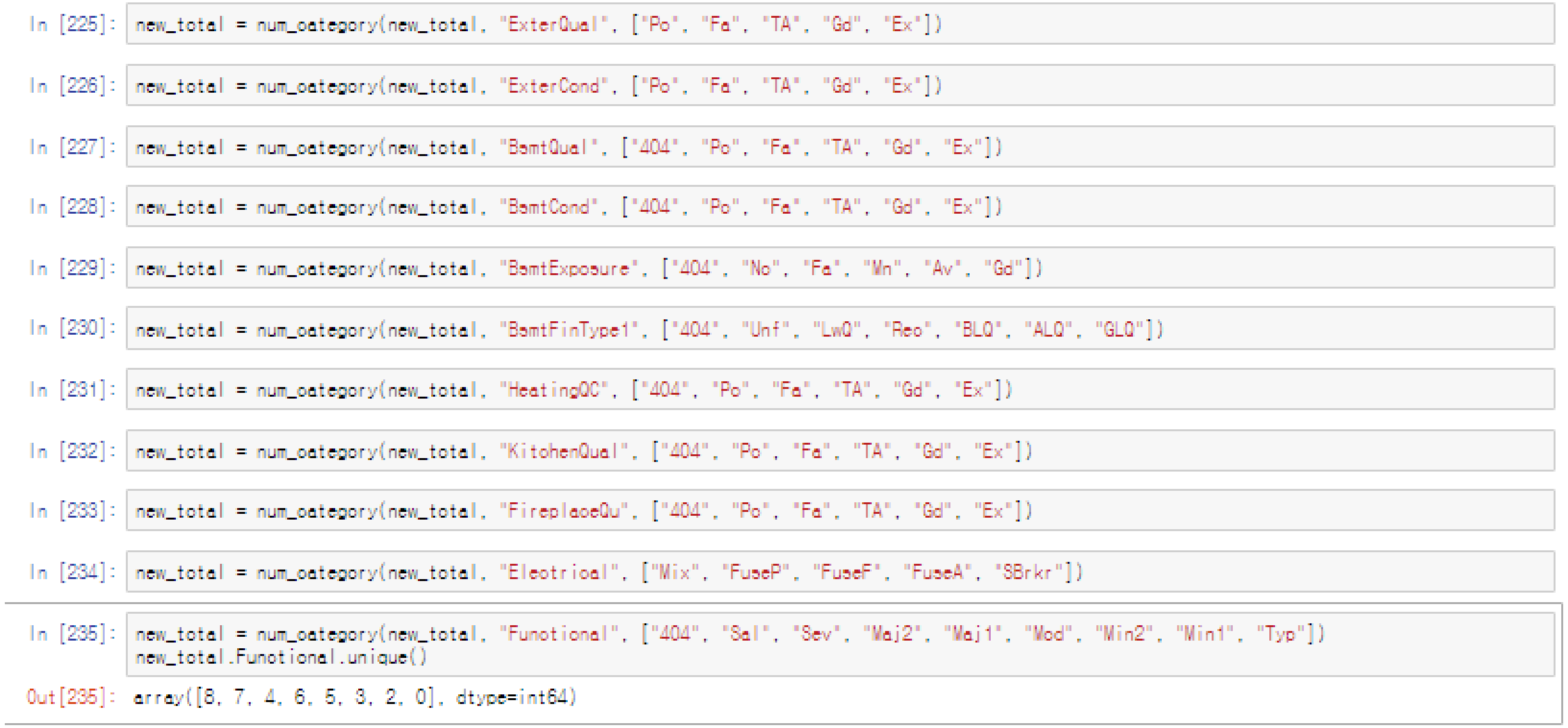
PreProcessing *Null값 채워주기*

평균으로 채우기



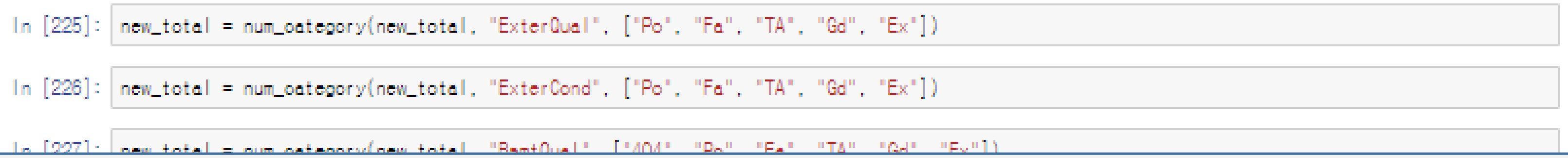
```
In [214]: new_total["YearBuilt"] = new_total["YearBuilt"].fillna(2005)

In [215]: new_total["BsmtFinSF1"] = new_total["BsmtFinSF1"].fillna(new_total.BsmtFinSF1.mean())

In [216]: new_total["BsmtUnfSF"] = new_total["BsmtUnfSF"].fillna(new_total.BsmtUnfSF.mean())

In [217]: new_total["TotalBsmtSF"] = new_total["TotalBsmtSF"].fillna(new_total.TotalBsmtSF.mean())

In [218]: new_total["GarageArea"] = new_total["GarageArea"].fillna(new_total.GarageArea.mean())

In [219]: new_total["GarageYrBlt"] = new_total["GarageYrBlt"].fillna(2005)

In [220]: new_total["Electrical"] = new_total["Electrical"].fillna("SBrkr")

In [221]: new_total = new_total.fillna("404")
```

PreProcessing 명목형 변수를 연속형 변수로 변환



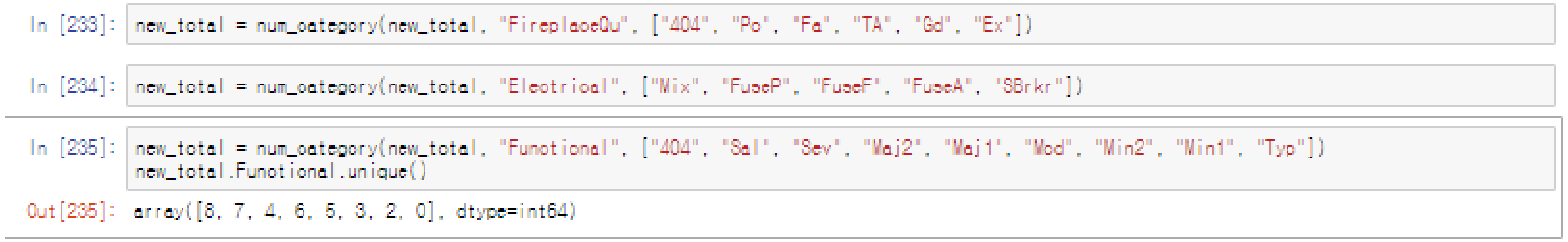
```
In [225]: new_total = num_category(new_total, "ExterQual", ["Po", "Fa", "TA", "Gd", "Ex"])

In [226]: new_total = num_category(new_total, "ExterCond", ["Po", "Fa", "TA", "Gd", "Ex"])

In [227]: new_total = num_category(new_total, "BsmtQual", ["404", "Po", "Fa", "TA", "Gd", "Ex"])

In [228]: new_total = num_category(new_total, "BsmtCond", ["404", "Po", "Fa", "TA", "Gd", "Ex"])

In [229]: new_total = num_category(new_total, "BsmtExposure", ["404", "No", "Fa", "Mn", "Av", "Gd"])

In [230]: new_total = num_category(new_total, "BsmtFinType1", ["404", "Unf", "LwQ", "Rec", "BLQ", "ALQ", "GLQ"])

In [231]: new_total = num_category(new_total, "HeatingQC", ["404", "Po", "Fa", "TA", "Gd", "Ex"])

In [232]: new_total = num_category(new_total, "KitchenQual", ["404", "Po", "Fa", "TA", "Gd", "Ex"])

In [233]: new_total = num_category(new_total, "FireplaceQu", ["404", "Po", "Fa", "TA", "Gd", "Ex"])

In [234]: new_total = num_category(new_total, "Electrical", ["Mix", "FuseP", "FuseF", "FuseA", "SBrkr"])

In [235]: new_total = num_category(new_total, "Functional", ["404", "Sal", "Sev", "Maj2", "Maj1", "Mod", "Min2", "Min1", "Typ"])
          new_total.Functional.unique()

Out[235]: array([8, 7, 4, 6, 5, 3, 2, 0], dtype=int64)
```

PreProcessing *명목형 변수를 연속형 변수로 변환*



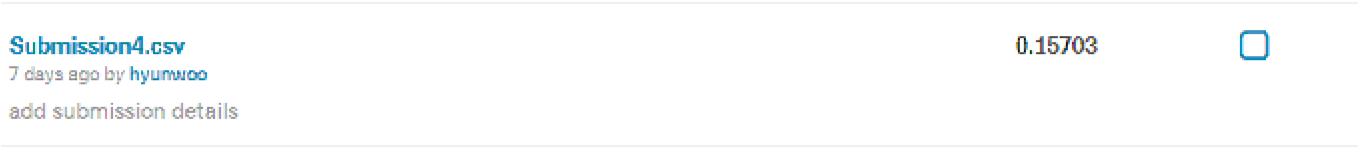
```
In [225]: new_total = num_category(new_total, "ExterQual", ["Po", "Fa", "TA", "Gd", "Ex"])

In [226]: new_total = num_category(new_total, "ExterCond", ["Po", "Fa", "TA", "Gd", "Ex"])

In [227]: new_total = num_category(new_total, "BsmtQual", ["404", "Po", "Fa", "TA", "Gd", "Ex"])
```

더미화할 양의 감소



```
In [233]: new_total = num_category(new_total, "FireplaceQu", ["404", "Po", "Fa", "TA", "Gd", "Ex"])

In [234]: new_total = num_category(new_total, "Electrical", ["Mix", "FuseP", "FuseF", "FuseA", "SBrkr"])

In [235]: new_total = num_category(new_total, "Functional", ["404", "Sal", "Sev", "Maj2", "Maj1", "Mod", "Min2", "Min1", "Typ"])
          new_total.Functional.unique()
Out[235]: array([8, 7, 4, 6, 5, 3, 2, 0], dtype=int64)
```

PreProcessing *결과-첫 제출*



Submission4.csv

7 days ago by hyunwoo

add submission details

0.15703

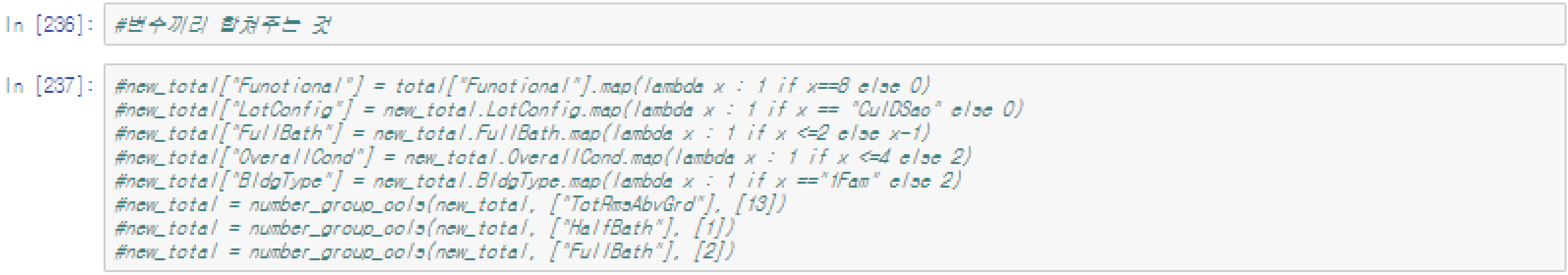
□

PreProcessing 결과

1100등/1600팀

PreProcessing 연속형 변수 전처리

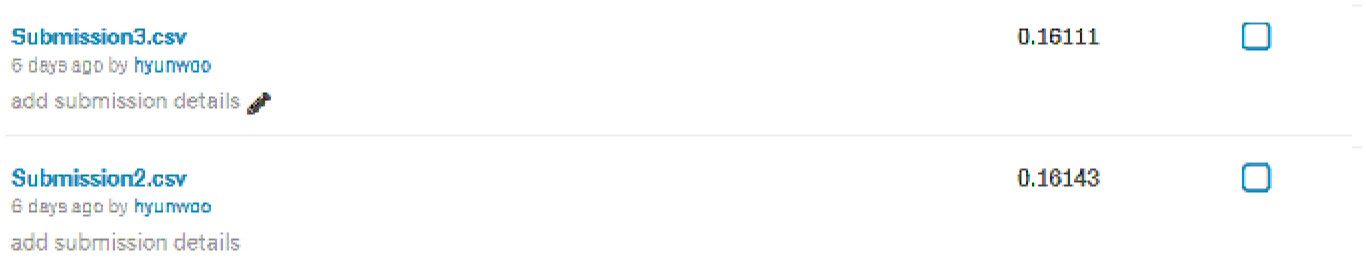
기적의 전처리



```
In [236]: #변수끼리 합쳐주는 것
```

```
In [237]: #new_total["Functional"] = total["Functional"].map(lambda x : 1 if x==8 else 0)
          #new_total["LotConfig"] = new_total.LotConfig.map(lambda x : 1 if x == "CulDSac" else 0)
          #new_total["FullBath"] = new_total.FullBath.map(lambda x : 1 if x <=2 else x-1)
          #new_total["OverallCond"] = new_total.OverallCond.map(lambda x : 1 if x <=4 else 2)
          #new_total["BldgType"] = new_total.BldgType.map(lambda x : 1 if x =="1Fam" else 2)
          #new_total = number_group_cols(new_total, ["TotRmsAbvGrd"], [13])
          #new_total = number_group_cols(new_total, ["HalfBath"], [1])
          #new_total = number_group_cols(new_total, ["FullBath"], [2])
```

PreProcessing 결과-두 번째 제출



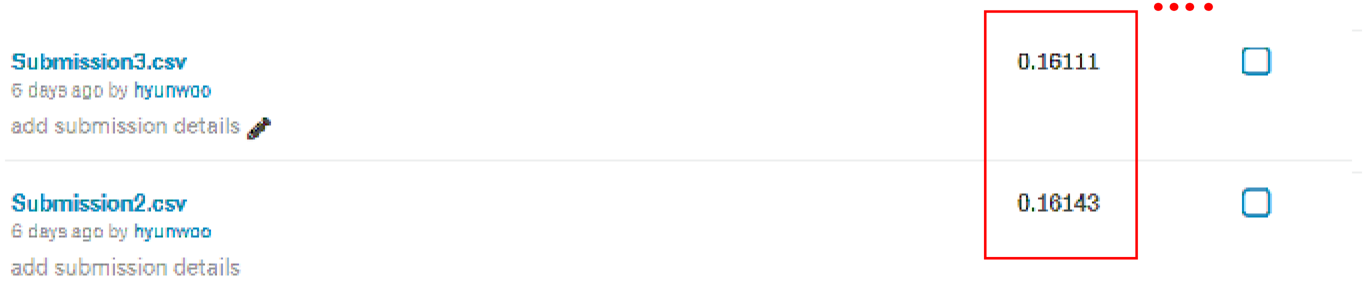
Submission3.csv
6 days ago by hyunwoo
add submission details

0.16111 □

Submission2.csv
6 days ago by hyunwoo
add submission details

0.16143 □

PreProcessing 결과



...?

Submission3.csv
6 days ago by hyunwoo
add submission details

0.16111 □

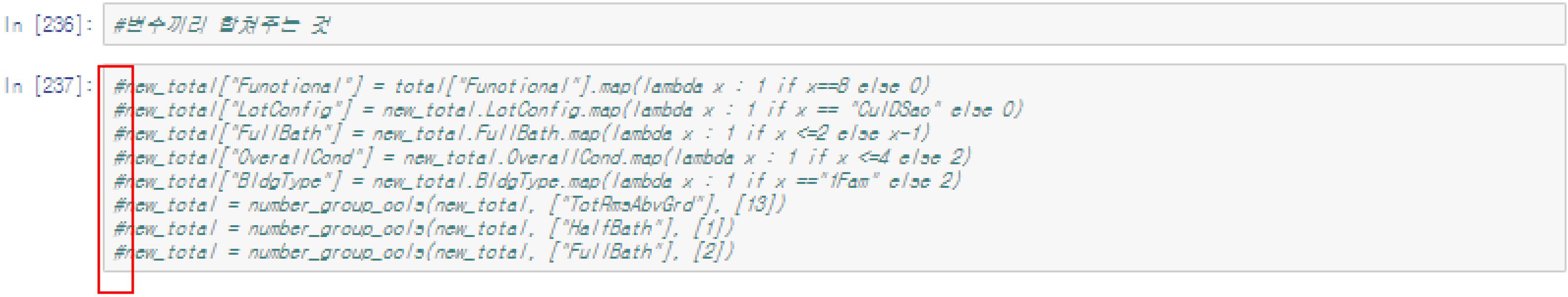
Submission2.csv
6 days ago by hyunwoo
add submission details

0.16143 □

PreProcessing 결과

1400등/1600팀 ...

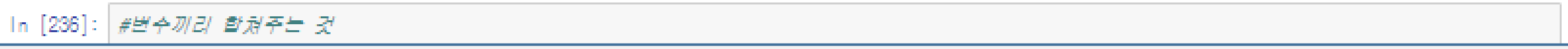
기적의 전처리에 각주를 한 이유



```
In [236]: #변수끼리 합쳐주는 것

In [237]: #new_total["Functional"] = total["Functional"].map(lambda x : 1 if x==8 else 0)
          #new_total["LotConfig"] = new_total.LotConfig.map(lambda x : 1 if x == "CulDSac" else 0)
          #new_total["FullBath"] = new_total.FullBath.map(lambda x : 1 if x <=2 else x-1)
          #new_total["OverallCond"] = new_total.OverallCond.map(lambda x : 1 if x <=4 else 2)
          #new_total["BldgType"] = new_total.BldgType.map(lambda x : 1 if x =="1Fam" else 2)
          #new_total = number_group_cols(new_total, ["TotRmsAbvGrd"], [13])
          #new_total = number_group_cols(new_total, ["HalfBath"], [1])
          #new_total = number_group_cols(new_total, ["FullBath"], [2])
```

PreProcessing *연속형 변수 전처리*

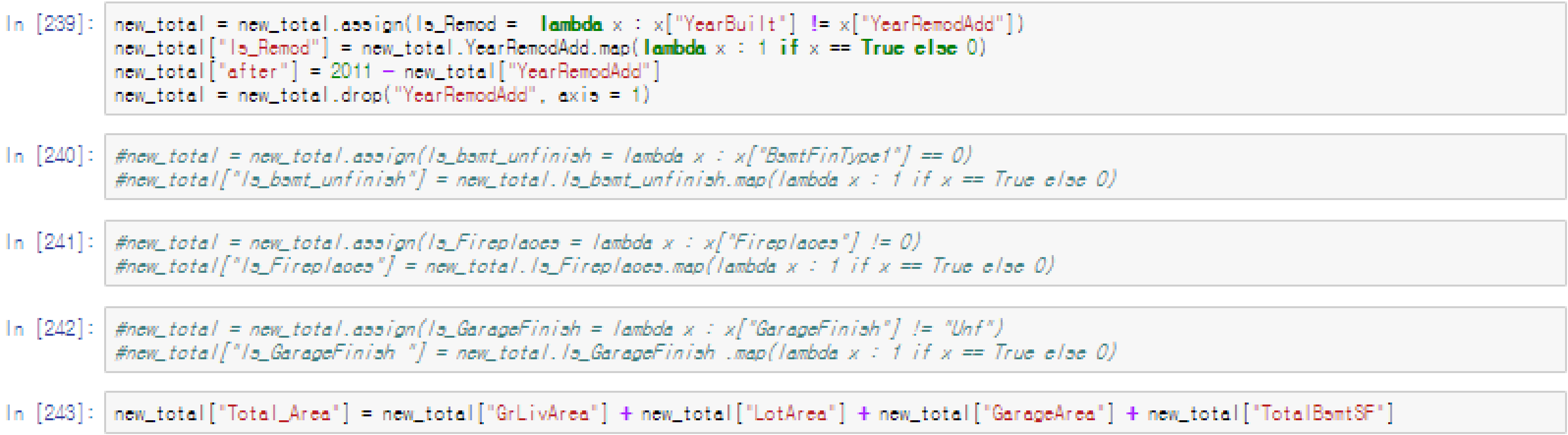
기적의 전처리에 각주를 한 이유



```
In [236]: #변수끼리 합쳐주는 것
```

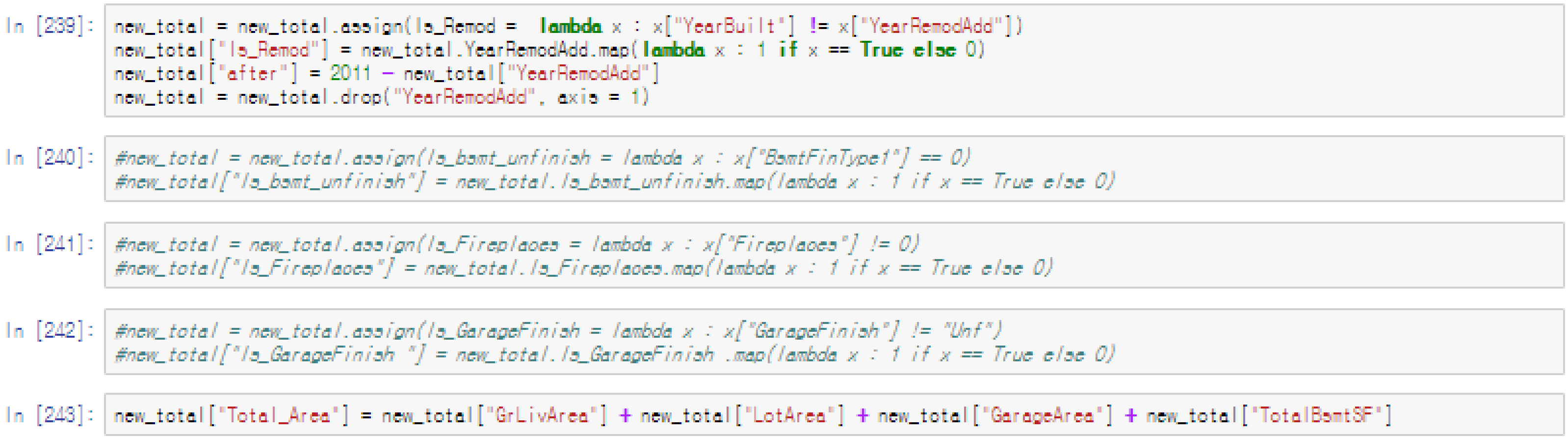
교훈 : 데이터에 손이 닿으면 결과가 나빠진다.

Modeling 파생변수 만들기

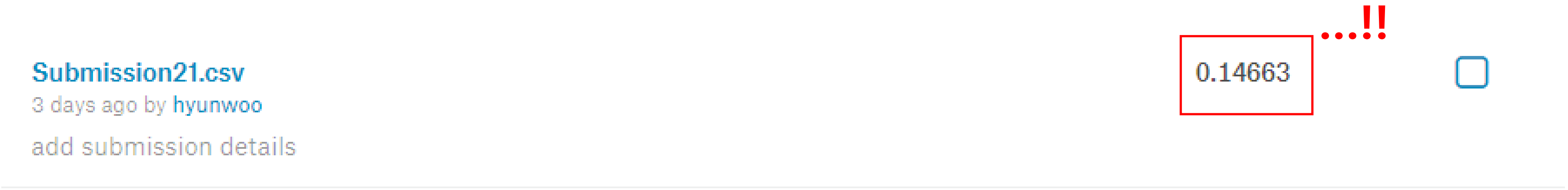


```
In [239]: new_total = new_total.assign(Is_Remod =  lambda x : x["YearBuilt"] != x["YearRemodAdd"])
          new_total["Is_Remod"] = new_total.YearRemodAdd.map( lambda x : 1 if x == True else 0)
          new_total["after"] = 2011 - new_total["YearRemodAdd"]
          new_total = new_total.drop("YearRemodAdd", axis = 1)

In [240]: #new_total = new_total.assign(Is_bsmt_unfinish = lambda x : x["BsmtFinType1"] == 0)
          #new_total["Is_bsmt_unfinish"] = new_total.Is_bsmt_unfinish.map(lambda x : 1 if x == True else 0)

In [241]: #new_total = new_total.assign(Is_Fireplaces = lambda x : x["Fireplaces"] != 0)
          #new_total["Is_Fireplaces"] = new_total.Is_Fireplaces.map(lambda x : 1 if x == True else 0)

In [242]: #new_total = new_total.assign(Is_GarageFinish = lambda x : x["GarageFinish"] != "Unf")
          #new_total["Is_GarageFinish "] = new_total.Is_GarageFinish .map(lambda x : 1 if x == True else 0)

In [243]: new_total["Total_Area"] = new_total["GrLivArea"] + new_total["LotArea"] + new_total["GarageArea"] + new_total["TotalBsmtSF"]
```

Modeling 파생변수 만들기



```
In [239]: new_total = new_total.assign(Is_Remod =  lambda x : x["YearBuilt"] != x["YearRemodAdd"])
          new_total["Is_Remod"] = new_total.YearRemodAdd.map(lambda x : 1 if x == True else 0)
          new_total["after"] = 2011 - new_total["YearRemodAdd"]
          new_total = new_total.drop("YearRemodAdd", axis = 1)

In [240]: #new_total = new_total.assign(Is_bsmt_unfinish = lambda x : x["BsmtFinType1"] == 0)
          #new_total["Is_bsmt_unfinish"] = new_total.Is_bsmt_unfinish.map(lambda x : 1 if x == True else 0)

In [241]: #new_total = new_total.assign(Is_Fireplaces = lambda x : x["Fireplaces"] != 0)
          #new_total["Is_Fireplaces"] = new_total.Is_Fireplaces.map(lambda x : 1 if x == True else 0)

In [242]: #new_total = new_total.assign(Is_GarageFinish = lambda x : x["GarageFinish"] != "Unf")
          #new_total["Is_GarageFinish "] = new_total.Is_GarageFinish .map(lambda x : 1 if x == True else 0)

In [243]: new_total["Total_Area"] = new_total["GrLivArea"] + new_total["LotArea"] + new_total["GarageArea"] + new_total["TotalBsmtSF"]
```



Submission21.csv

3 days ago by hyunwoo

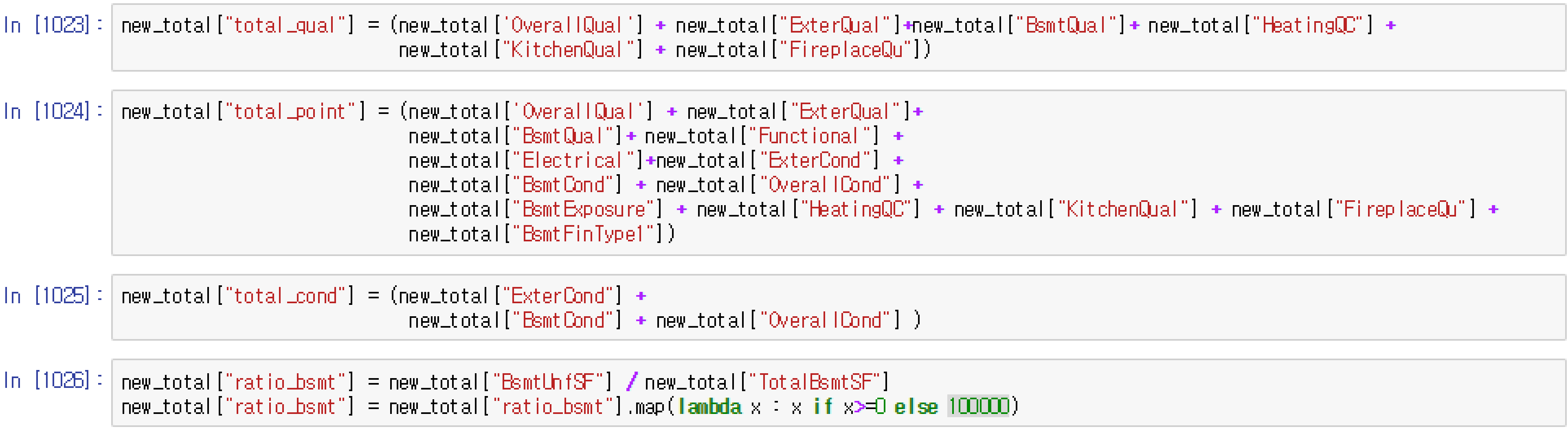
add submission details

0.14663

...!!

□

Modeling *파생변수 만들기*



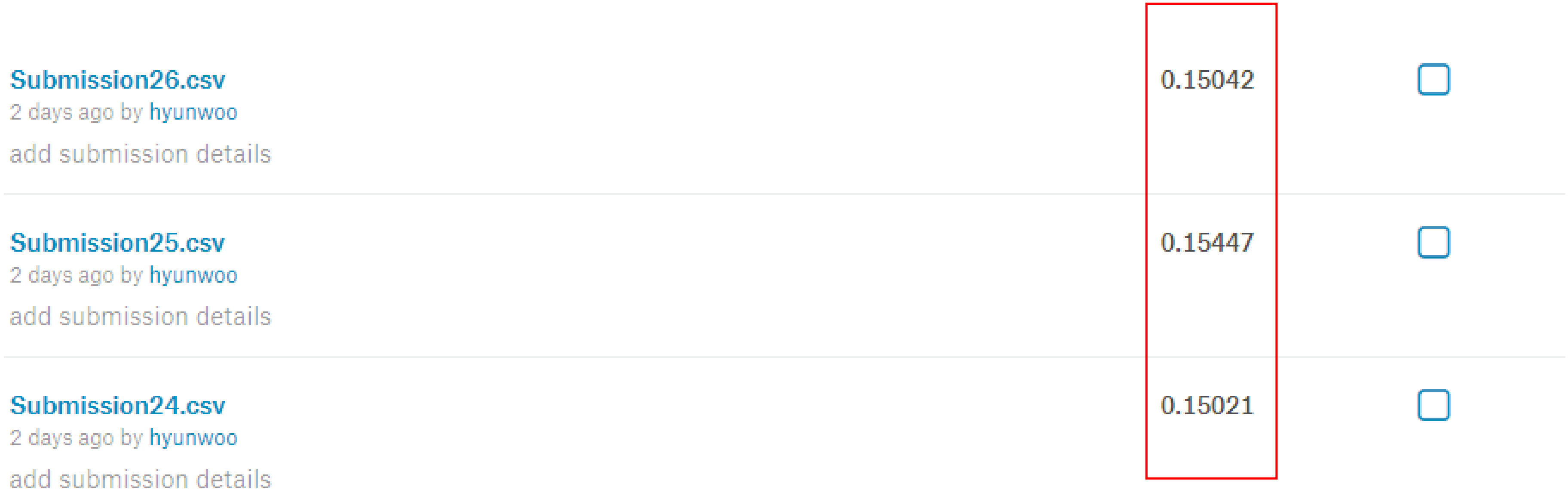
```
In [1023]: new_total["total_qual"] = (new_total['OverallQual'] + new_total["ExterQual"]+new_total["BsmtQual"]+ new_total["HeatingQC"] +
                                      new_total["KitchenQual"] + new_total["FireplaceQu"])
```

```
In [1024]: new_total["total_point"] = (new_total['OverallQual'] + new_total["ExterQual"]+
                                       new_total["BsmtQual"]+ new_total["Functional"] +
                                       new_total["Electrical"]+new_total["ExterCond"] +
                                       new_total["BsmtCond"] + new_total["OverallCond"] +
                                       new_total["BsmtExposure"] + new_total["HeatingQC"] + new_total["KitchenQual"] + new_total["FireplaceQu"] +
                                       new_total["BsmtFinType1"])
```

```
In [1025]: new_total["total_cond"] = (new_total["ExterCond"] +
                                       new_total["BsmtCond"] + new_total["OverallCond"] )
```

```
In [1026]: new_total["ratio_bsmt"] = new_total["BsmtUnfSF"] / new_total["TotalBsmtSF"]
           new_total["ratio_bsmt"] = new_total["ratio_bsmt"].map(lambda x : x if x>=0 else 100000)
```

Modeling *파생변수 만들기*

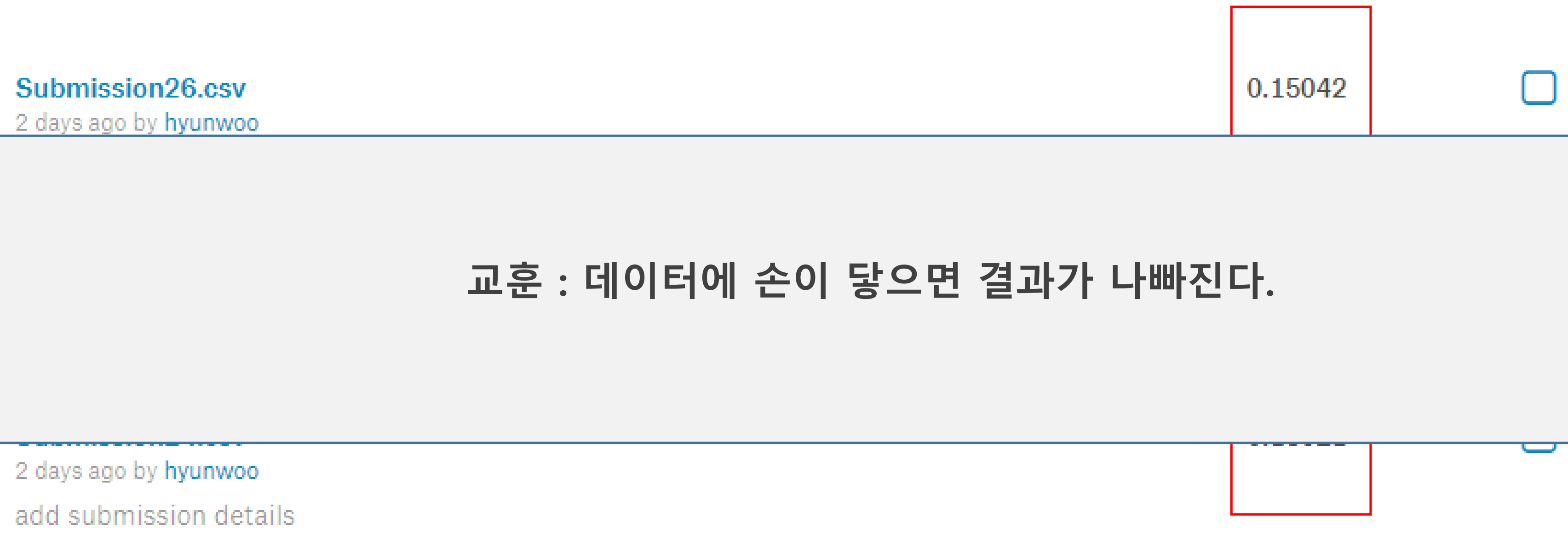


Submission26.csv
2 days ago by hyunwoo
add submission details
0.15042 □

Submission25.csv
2 days ago by hyunwoo
add submission details
0.15447 □

Submission24.csv
2 days ago by hyunwoo
add submission details
0.15021 □

Modeling *파생변수 만들기*



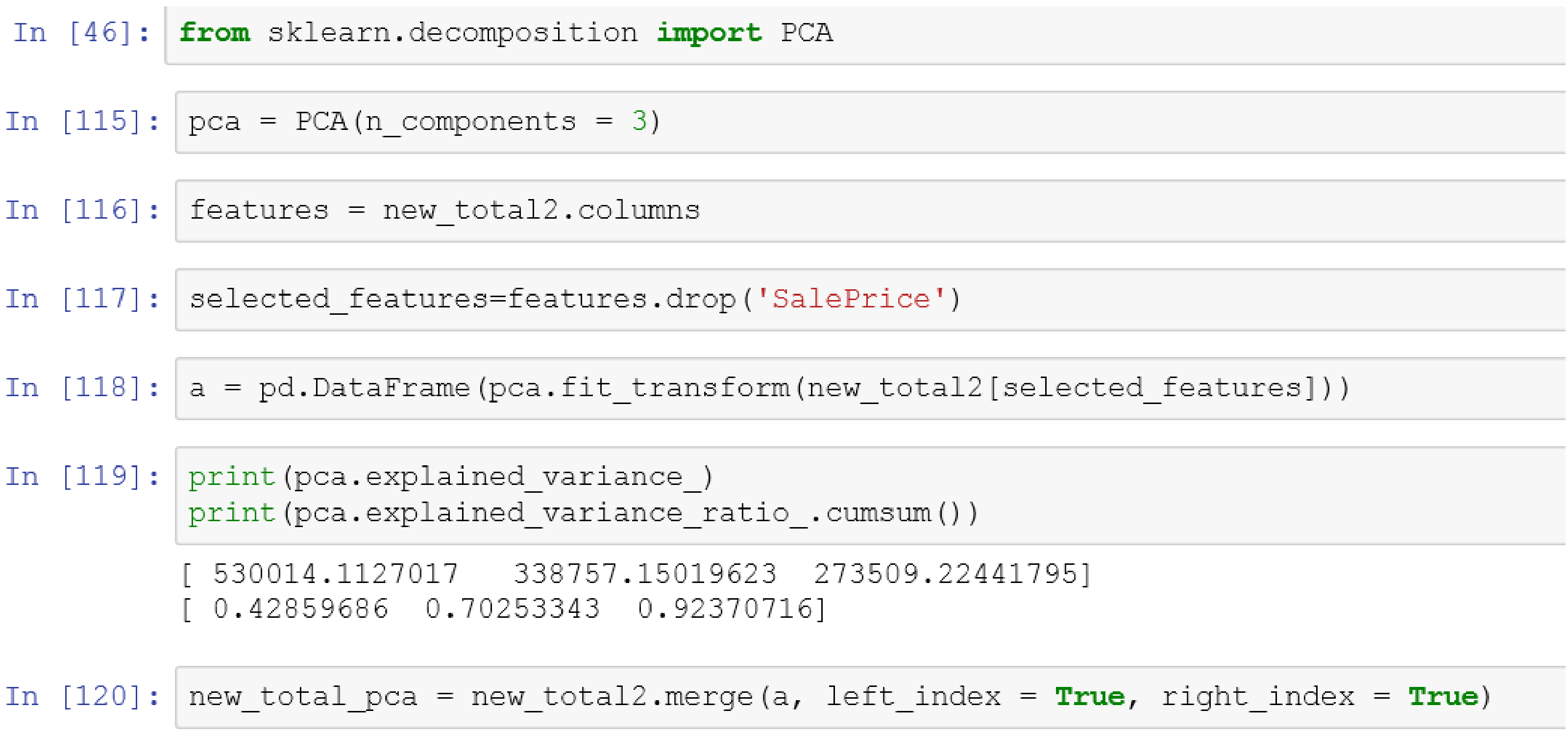
Submission26.csv
2 days ago by hyunwoo

0.15042 □

2 days ago by hyunwoo
add submission details

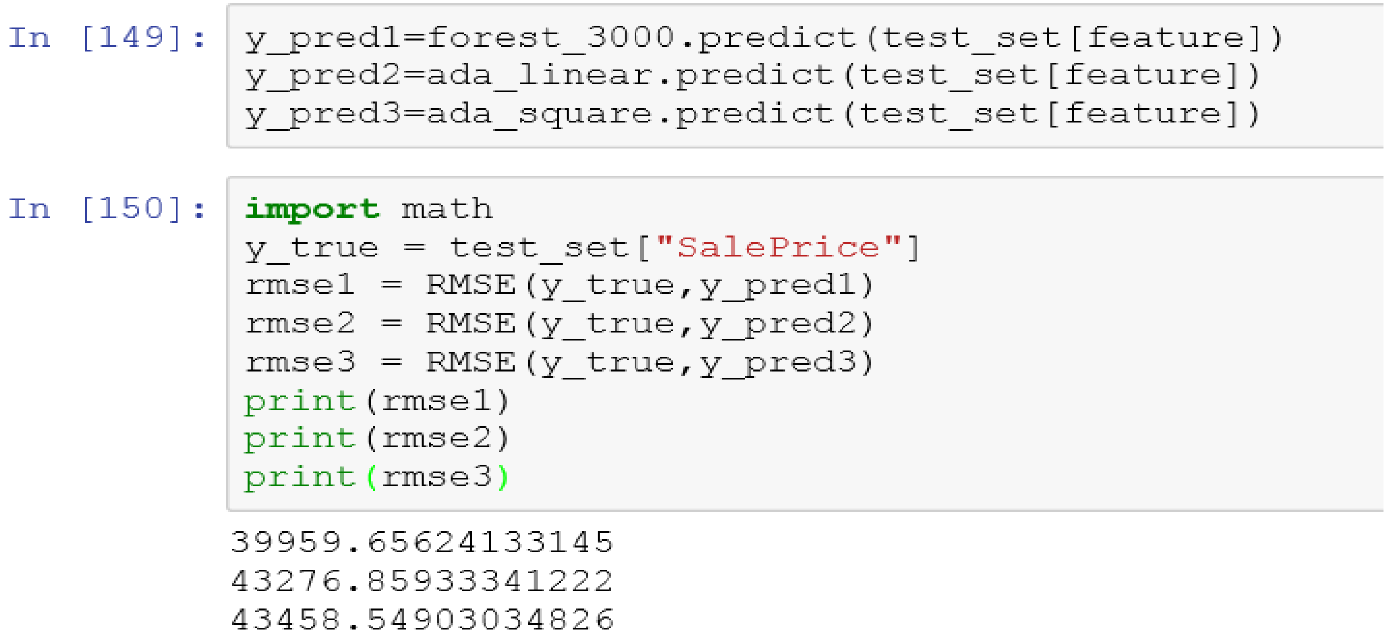
교훈 : 데이터에 손이 닿으면 결과가 나빠진다.

Modeling *PCA*



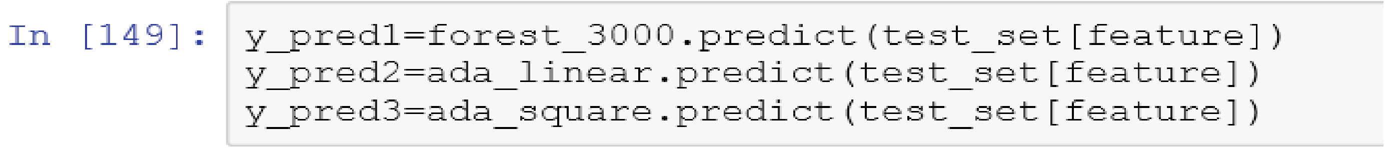
```
In [46]: from sklearn.decomposition import PCA

In [115]: pca = PCA(n_components = 3)

In [116]: features = new_total2.columns

In [117]: selected_features=features.drop('SalePrice')

In [118]: a = pd.DataFrame(pca.fit_transform(new_total2[selected_features]))

In [119]: print(pca.explained_variance_)
          print(pca.explained_variance_ratio_.cumsum())

          [ 530014.1127017   338757.15019623  273509.22441795]
          [ 0.42859686  0.70253343  0.92370716]

In [120]: new_total_pca = new_total2.merge(a, left_index = True, right_index = True)
```

Modeling *PCA*



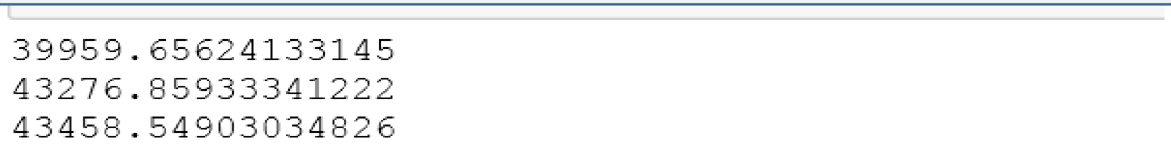
```
In [149]: y_pred1=forest_3000.predict(test_set[feature])
          y_pred2=ada_linear.predict(test_set[feature])
          y_pred3=ada_square.predict(test_set[feature])

In [150]: import math
          y_true = test_set["SalePrice"]
          rmse1 = RMSE(y_true,y_pred1)
          rmse2 = RMSE(y_true,y_pred2)
          rmse3 = RMSE(y_true,y_pred3)
          print(rmse1)
          print(rmse2)
          print(rmse3)

          39959.65624133145
          43276.85933341222
          43458.54903034826
```

Modeling *PCA*



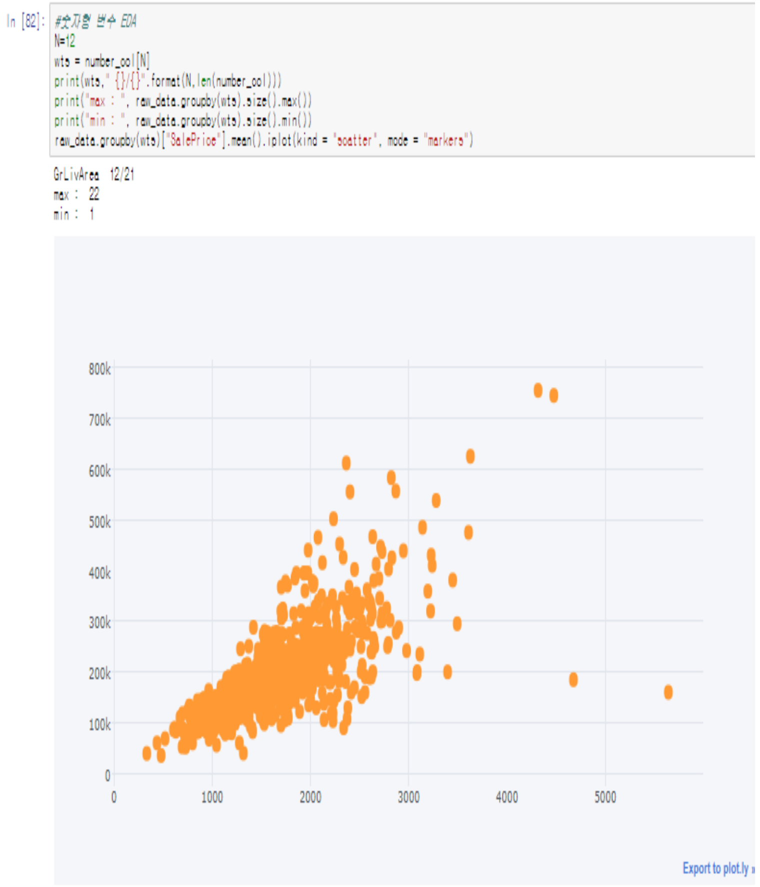
```
In [149]: y_pred1=forest_3000.predict(test_set[feature])
          y_pred2=ada_linear.predict(test_set[feature])
          y_pred3=ada_square.predict(test_set[feature])
```

교훈 : 데이터에 손이 닿으면 결과가 나빠진다.



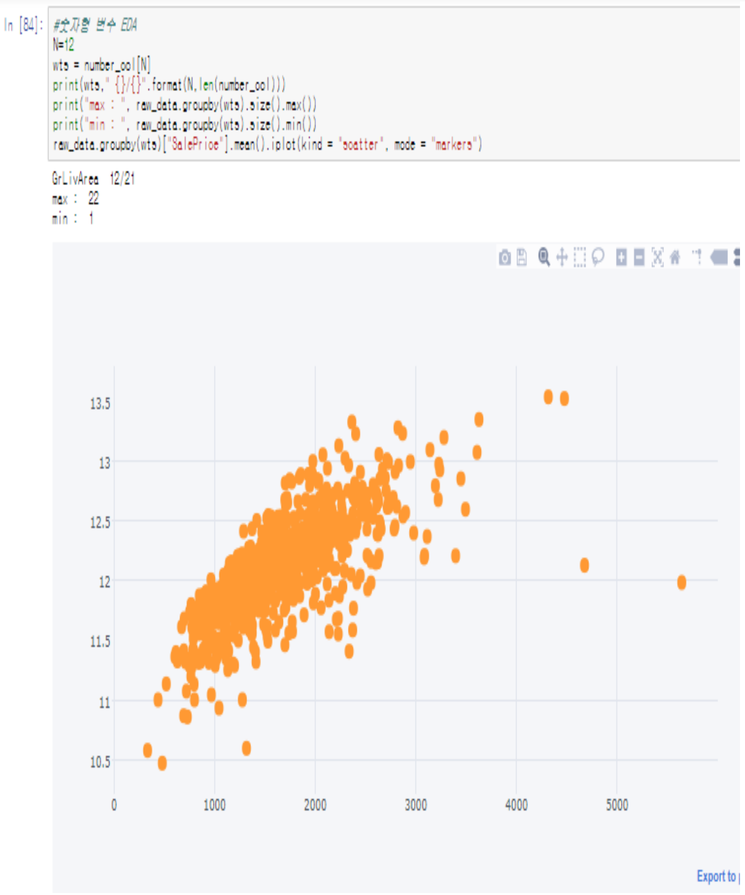
```
39959.65624133145
43276.85933341222
43458.54903034826
```

Modeling 로그 씌우기



```
In [82]:
#숫자형 변수 EDA
N=12
wts = number_col[N]
print(wts," {}/{}".format(N,len(number_col)))
print("max : ", raw_data.groupby(wts).size().max())
print("min : ", raw_data.groupby(wts).size().min())
raw_data.groupby(wts)["SalePrice"].mean().iplot(kind = 'scatter', mode = "markers")
```

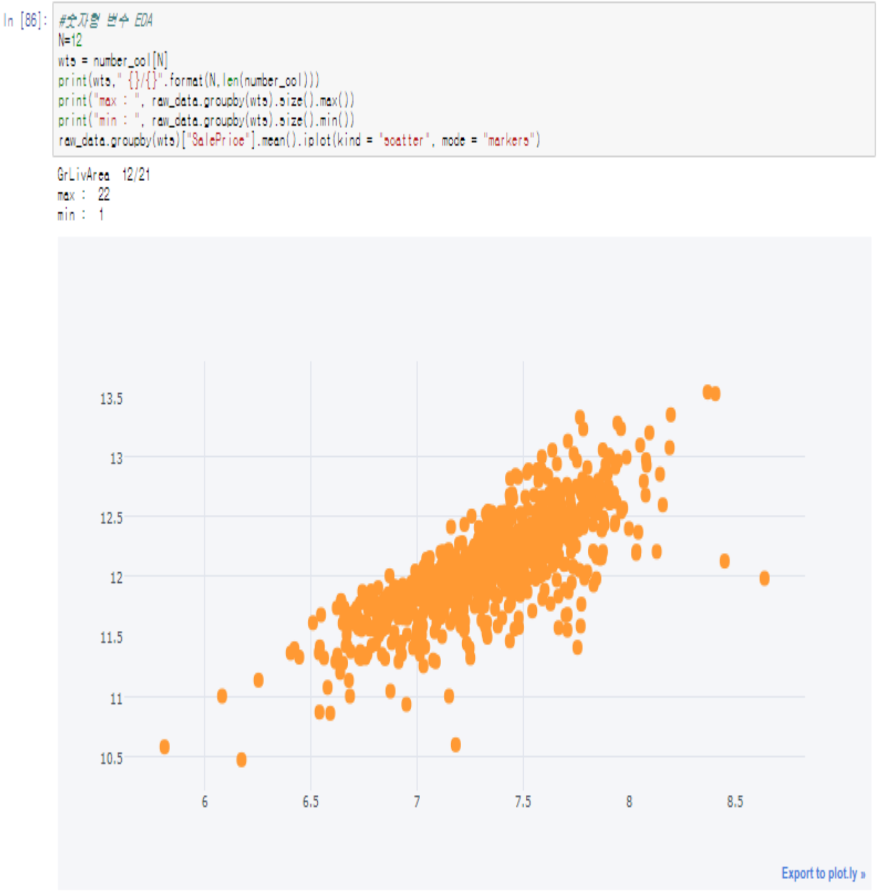
```
GrLivArea  12/21
max :  22
min :  1
```



```
In [84]:
#숫자형 변수 EDA
N=12
wts = number_col[N]
print(wts," {}/{}".format(N,len(number_col)))
print("max : ", raw_data.groupby(wts).size().max())
print("min : ", raw_data.groupby(wts).size().min())
raw_data.groupby(wts)["SalePrice"].mean().iplot(kind = 'scatter', mode = "markers")
```

```
GrLivArea  12/21
max :  22
min :  1
```

Log(SalePrice)



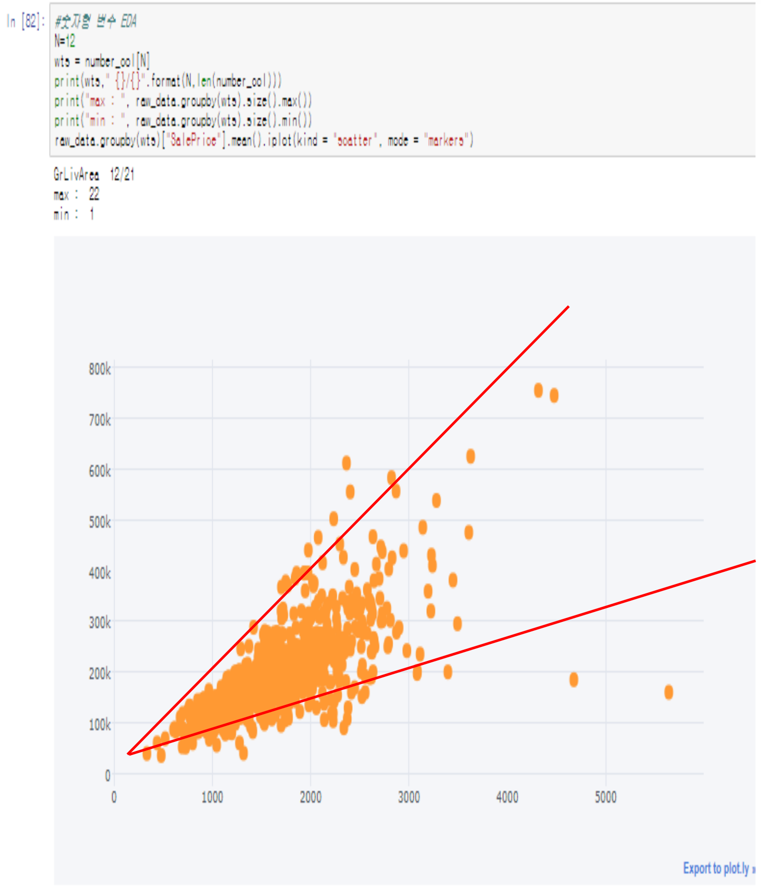
```
In [86]:
#숫자형 변수 EDA
N=12
wts = number_col[N]
print(wts," {}/{}".format(N,len(number_col)))
print("max : ", raw_data.groupby(wts).size().max())
print("min : ", raw_data.groupby(wts).size().min())
raw_data.groupby(wts)["SalePrice"].mean().iplot(kind = 'scatter', mode = "markers")
```

```
GrLivArea  12/21
max :  22
min :  1
```

Log(SalePrice)

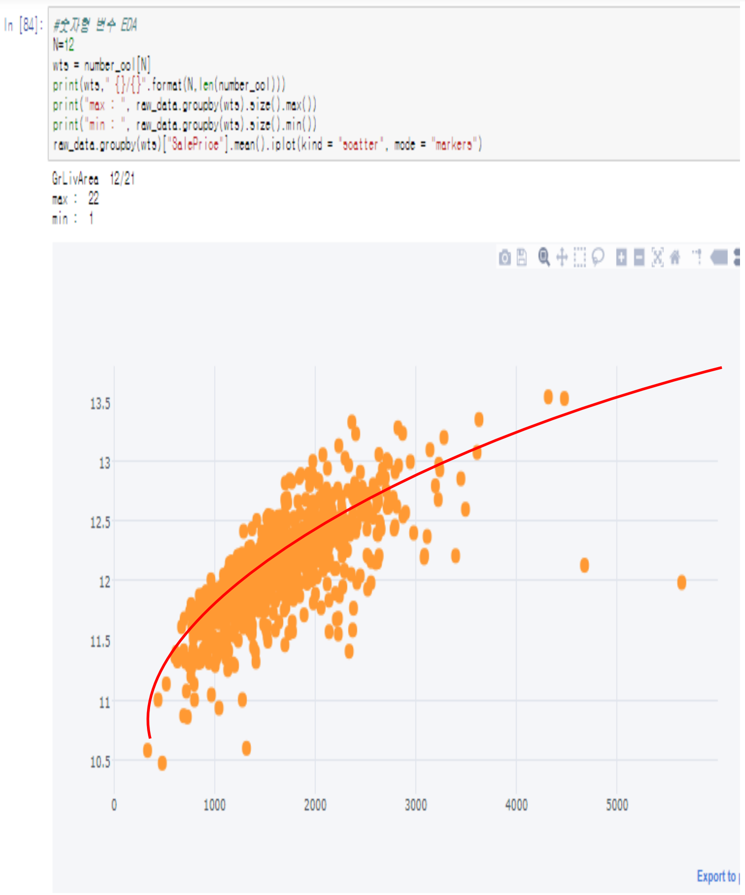
Log(GrLivArea)

Modeling 로그 씌우기



```
In [82]: #숫자형 변수 EDA
N=12
wts = number_col[N]
print(wts," {}/{}".format(N,len(number_col)))
print("max : ", raw_data.groupby(wts).size().max())
print("min : ", raw_data.groupby(wts).size().min())
raw_data.groupby(wts)["SalePrice"].mean().iplot(kind = 'scatter', mode = "markers")
```

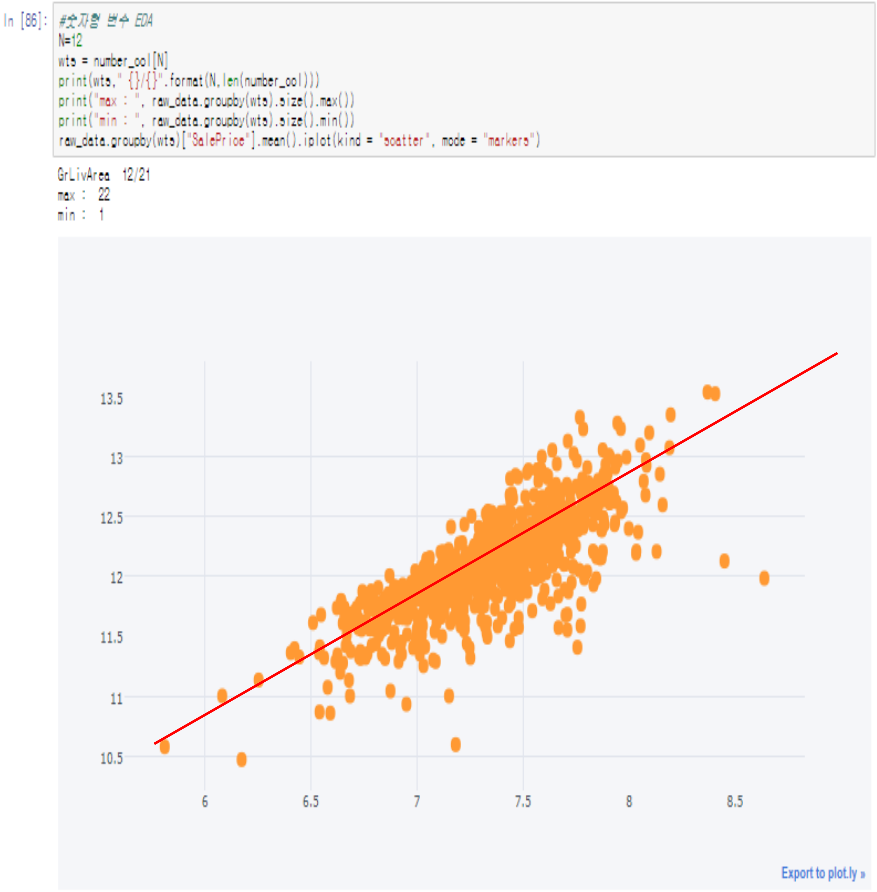
```
GrLivArea  12/21
max :  22
min :  1
```



```
In [84]: #숫자형 변수 EDA
N=12
wts = number_col[N]
print(wts," {}/{}".format(N,len(number_col)))
print("max : ", raw_data.groupby(wts).size().max())
print("min : ", raw_data.groupby(wts).size().min())
raw_data.groupby(wts)["SalePrice"].mean().iplot(kind = 'scatter', mode = "markers")
```

```
GrLivArea  12/21
max :  22
min :  1
```

Log(SalePrice)



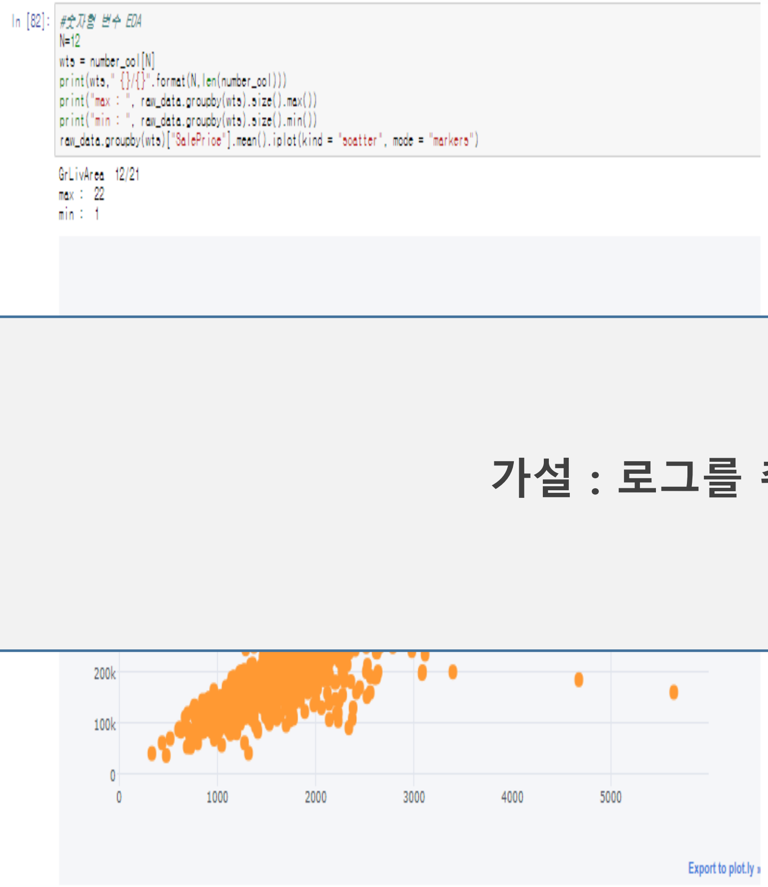
```
In [86]: #숫자형 변수 EDA
N=12
wts = number_col[N]
print(wts," {}/{}".format(N,len(number_col)))
print("max : ", raw_data.groupby(wts).size().max())
print("min : ", raw_data.groupby(wts).size().min())
raw_data.groupby(wts)["SalePrice"].mean().iplot(kind = 'scatter', mode = "markers")
```

```
GrLivArea  12/21
max :  22
min :  1
```

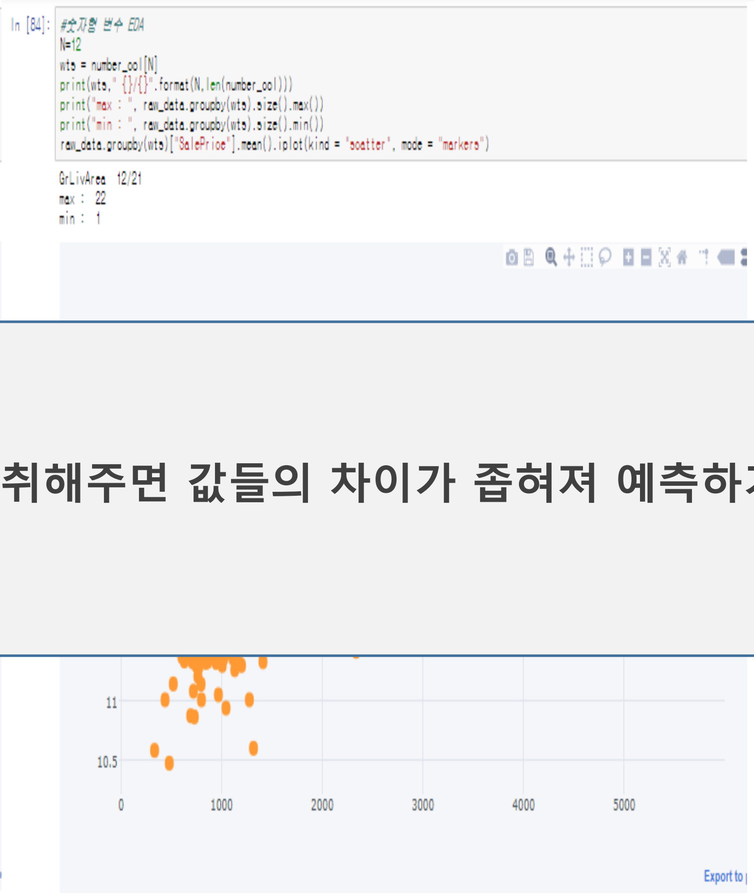
Log(SalePrice)

Log(GrLivArea)

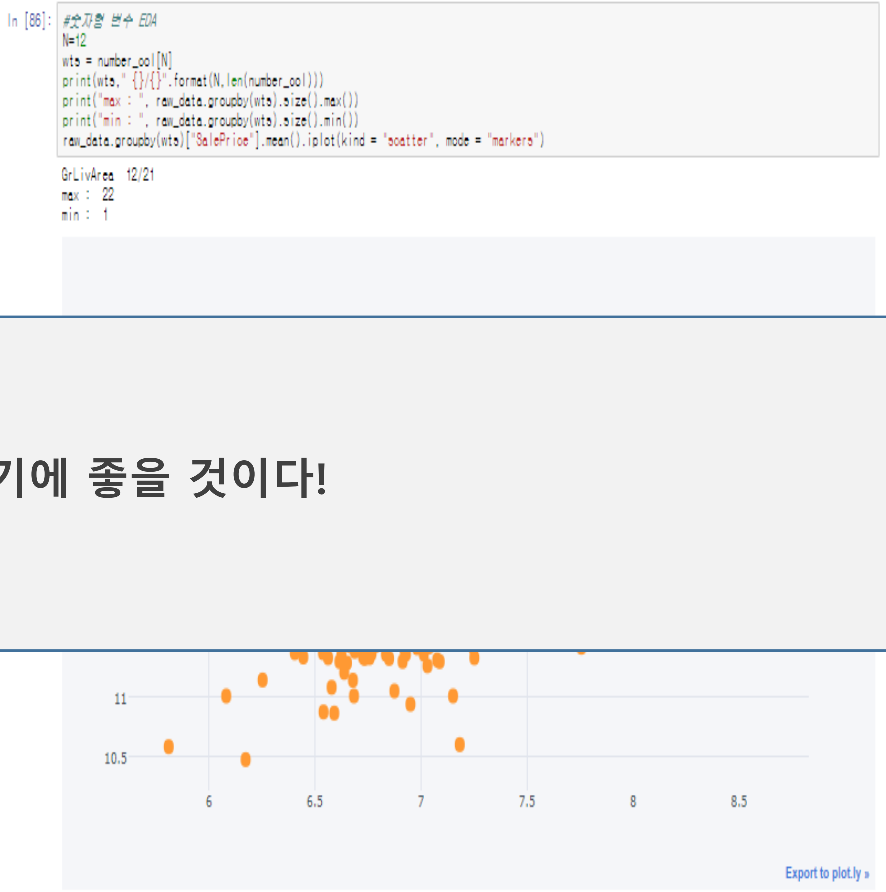
Modeling *로그 씌우기*



```
In [82]: #숫자형 변수 EDA
N=12
wts = number_col[N]
print(wts," {}/{}".format(N,len(number_col)))
print("max : ", raw_data.groupby(wts).size().max())
print("min : ", raw_data.groupby(wts).size().min())
raw_data.groupby(wts)["SalePrice"].mean().iplot(kind = 'scatter', mode = "markers")

GrLivArea  12/21
max :  22
min :  1
```



```
In [84]: #숫자형 변수 EDA
N=12
wts = number_col[N]
print(wts," {}/{}".format(N,len(number_col)))
print("max : ", raw_data.groupby(wts).size().max())
print("min : ", raw_data.groupby(wts).size().min())
raw_data.groupby(wts)["SalePrice"].mean().iplot(kind = 'scatter', mode = "markers")

GrLivArea  12/21
max :  22
min :  1
```



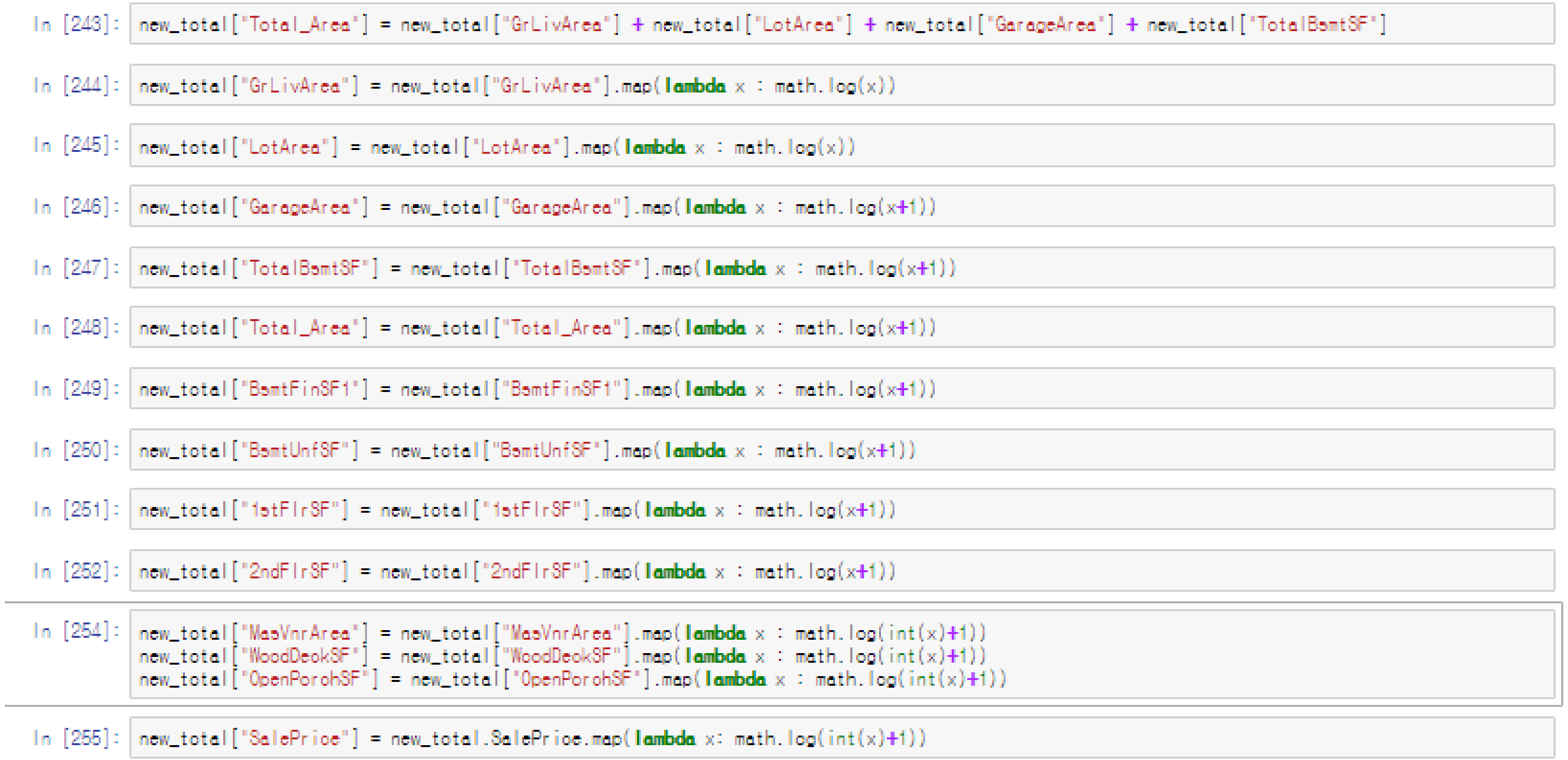
```
In [86]: #숫자형 변수 EDA
N=12
wts = number_col[N]
print(wts," {}/{}".format(N,len(number_col)))
print("max : ", raw_data.groupby(wts).size().max())
print("min : ", raw_data.groupby(wts).size().min())
raw_data.groupby(wts)["SalePrice"].mean().iplot(kind = 'scatter', mode = "markers")

GrLivArea  12/21
max :  22
min :  1
```

가설 : 로그를 취해주면 값들의 차이가 좁혀져 예측하기에 좋을 것이다!

Log(SalePrice)

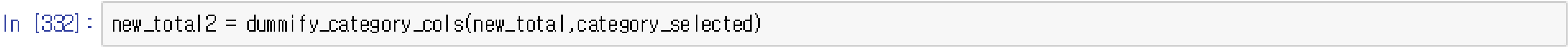
Log(SalePrice)
Log(GrLivArea)

Modeling *로그 씌우기*



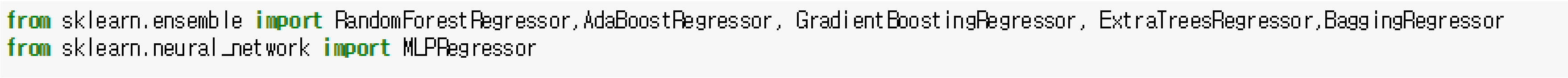
```
In [243]: new_total["Total_Area"] = new_total["GrLivArea"] + new_total["LotArea"] + new_total["GarageArea"] + new_total["TotalBsmtSF"]

In [244]: new_total["GrLivArea"] = new_total["GrLivArea"].map(lambda x : math.log(x))

In [245]: new_total["LotArea"] = new_total["LotArea"].map(lambda x : math.log(x))

In [246]: new_total["GarageArea"] = new_total["GarageArea"].map(lambda x : math.log(x+1))

In [247]: new_total["TotalBsmtSF"] = new_total["TotalBsmtSF"].map(lambda x : math.log(x+1))

In [248]: new_total["Total_Area"] = new_total["Total_Area"].map(lambda x : math.log(x+1))

In [249]: new_total["BsmtFinSF1"] = new_total["BsmtFinSF1"].map(lambda x : math.log(x+1))

In [250]: new_total["BsmtUnfSF"] = new_total["BsmtUnfSF"].map(lambda x : math.log(x+1))

In [251]: new_total["1stFlrSF"] = new_total["1stFlrSF"].map(lambda x : math.log(x+1))

In [252]: new_total["2ndFlrSF"] = new_total["2ndFlrSF"].map(lambda x : math.log(x+1))

In [254]: new_total["MasVnrArea"] = new_total["MasVnrArea"].map(lambda x : math.log(int(x)+1))
          new_total["WoodDeckSF"] = new_total["WoodDeckSF"].map(lambda x : math.log(int(x)+1))
          new_total["OpenPorchSF"] = new_total["OpenPorchSF"].map(lambda x : math.log(int(x)+1))

In [255]: new_total["SalePrice"] = new_total.SalePrice.map(lambda x: math.log(int(x)+1))
```

Modeling 더미화

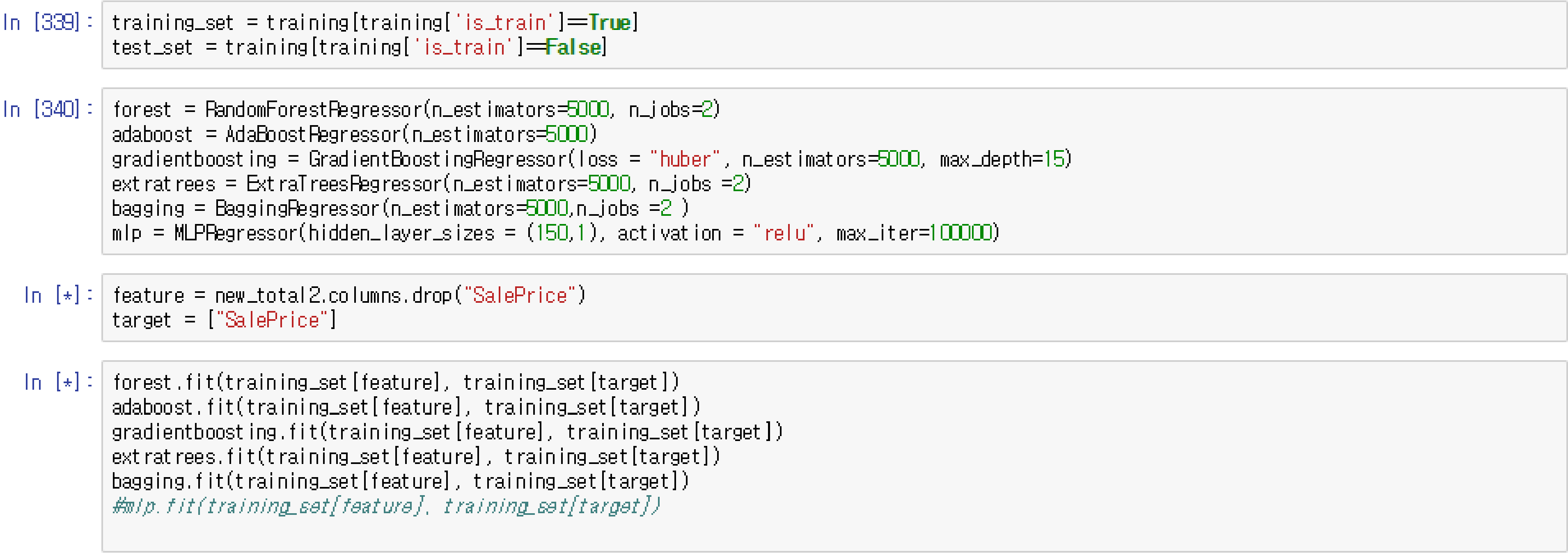


```
In [332]: new_total2 = dummify_category_cols(new_total,category_selected)
```

Modeling 모델 만들기

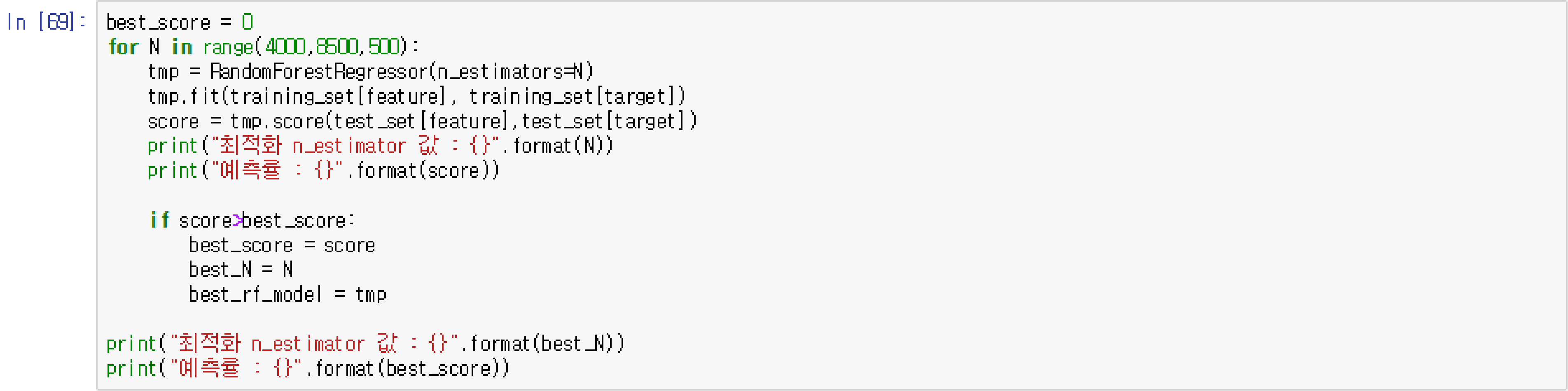


```
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor, GradientBoostingRegressor, ExtraTreesRegressor,BaggingRegressor
from sklearn.neural_network import MLPRegressor
```

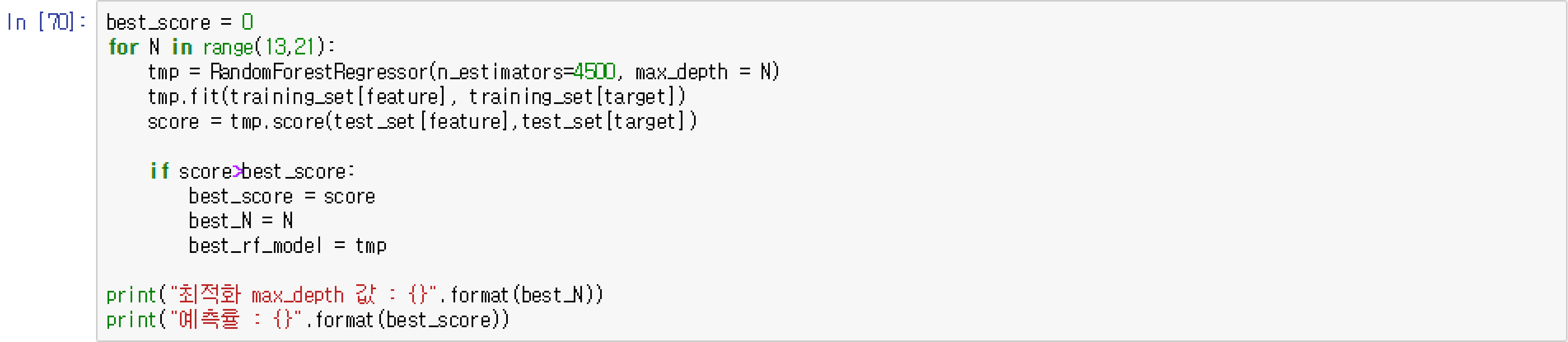


```
In [339]: training_set = training[training['is_train']==True]
          test_set = training[training['is_train']==False]

In [340]: forest = RandomForestRegressor(n_estimators=5000, n_jobs=2)
          adaboost = AdaBoostRegressor(n_estimators=5000)
          gradientboosting = GradientBoostingRegressor(loss = "huber", n_estimators=5000, max_depth=15)
          extratrees = ExtraTreesRegressor(n_estimators=5000, n_jobs =2)
          bagging = BaggingRegressor(n_estimators=5000,n_jobs =2 )
          mlp = MLPRegressor(hidden_layer_sizes = (150,1), activation = "relu", max_iter=100000)

In [*]:   feature = new_total2.columns.drop("SalePrice")
          target = ["SalePrice"]

In [*]:   forest.fit(training_set[feature], training_set[target])
          adaboost.fit(training_set[feature], training_set[target])
          gradientboosting.fit(training_set[feature], training_set[target])
          extratrees.fit(training_set[feature], training_set[target])
          bagging.fit(training_set[feature], training_set[target])
          #mlp.fit(training_set[feature], training_set[target])
```

Modeling 모델 최적화

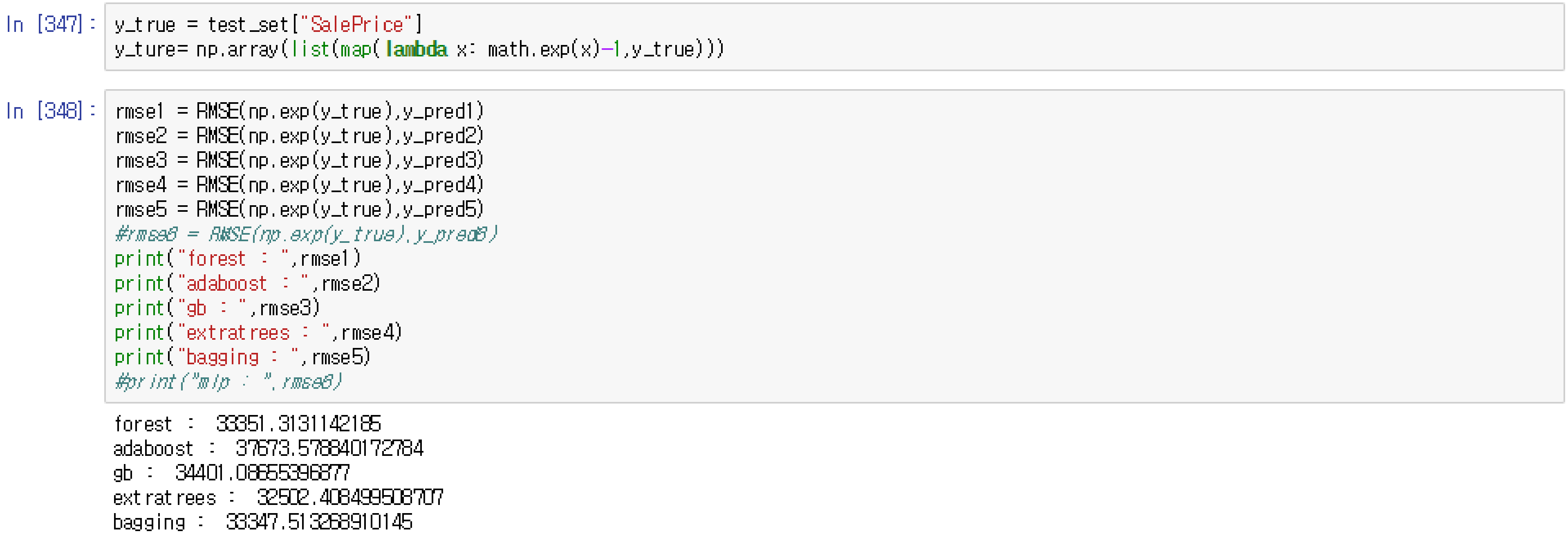


```
In [69]: best_score = 0
         for N in range(4000,8500,500):
             tmp = RandomForestRegressor(n_estimators=N)
             tmp.fit(training_set[feature], training_set[target])
             score = tmp.score(test_set[feature],test_set[target])
             print("최적화 n_estimator 값 : {}".format(N))
             print("예측률 : {}".format(score))

             if score>best_score:
                 best_score = score
                 best_N = N
                 best_rf_model = tmp

         print("최적화 n_estimator 값 : {}".format(best_N))
         print("예측률 : {}".format(best_score))
```



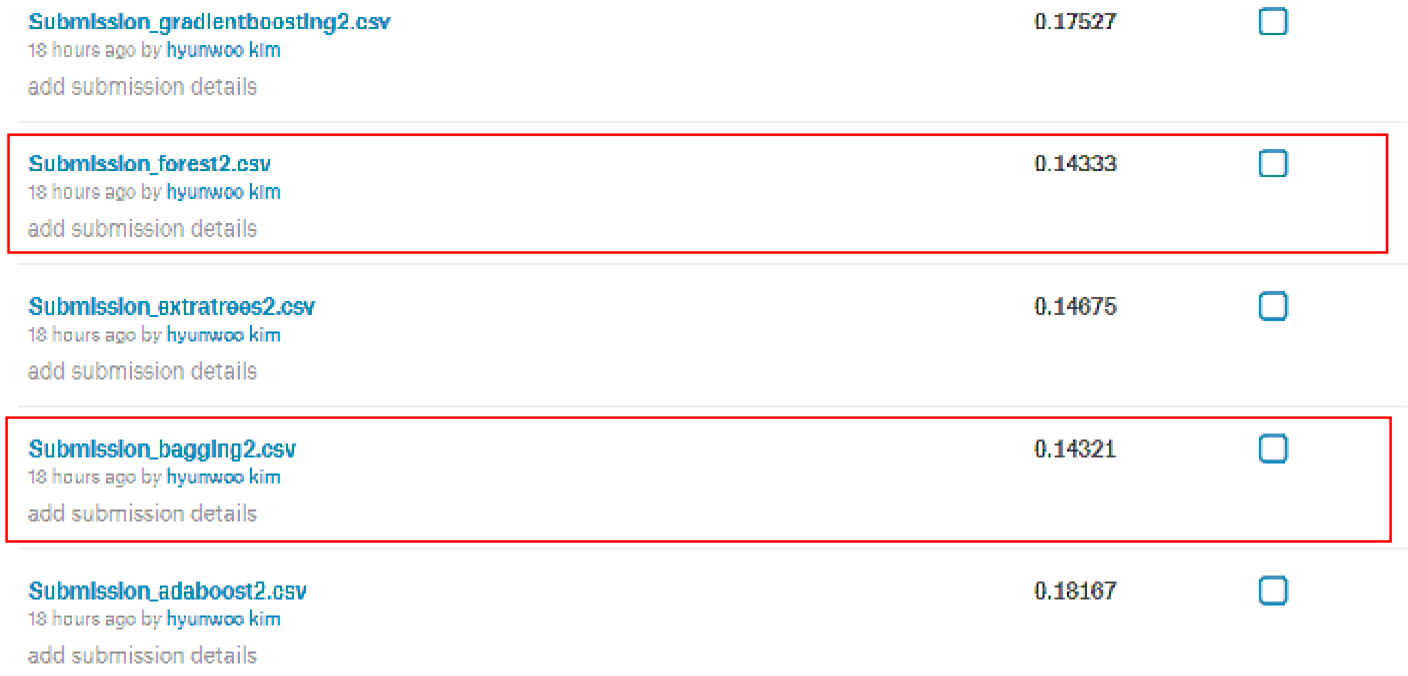
```
In [70]: best_score = 0
         for N in range(13,21):
             tmp = RandomForestRegressor(n_estimators=4500, max_depth = N)
             tmp.fit(training_set[feature], training_set[target])
             score = tmp.score(test_set[feature],test_set[target])

             if score>best_score:
                 best_score = score
                 best_N = N
                 best_rf_model = tmp

         print("최적화 max_depth 값 : {}".format(best_N))
         print("예측률 : {}".format(best_score))
```

Modeling *RMSE*



```
In [347]: y_true = test_set["SalePrice"]
          y_ture= np.array(list(map( lambda x: math.exp(x)-1,y_true)))

In [348]: rmse1 = RMSE(np.exp(y_true),y_pred1)
          rmse2 = RMSE(np.exp(y_true),y_pred2)
          rmse3 = RMSE(np.exp(y_true),y_pred3)
          rmse4 = RMSE(np.exp(y_true),y_pred4)
          rmse5 = RMSE(np.exp(y_true),y_pred5)
          #rmse6 = RMSE(np.exp(y_true),y_pred6)
          print("forest : ",rmse1)
          print("adaboost : ",rmse2)
          print("gb : ",rmse3)
          print("extratrees : ",rmse4)
          print("bagging : ",rmse5)
          #print("mlp : ",rmse6)

          forest :  33351.3131142185
          adaboost :  37673.578840172784
          gb :  34401.08655396877
          extratrees :  32502.408499508707
          bagging :  33347.513268910145
```

Modeling *Kaggle 제출*



Submission_gradientboosting2.csv
18 hours ago by hyunwoo kim
add submission details
0.17527 □

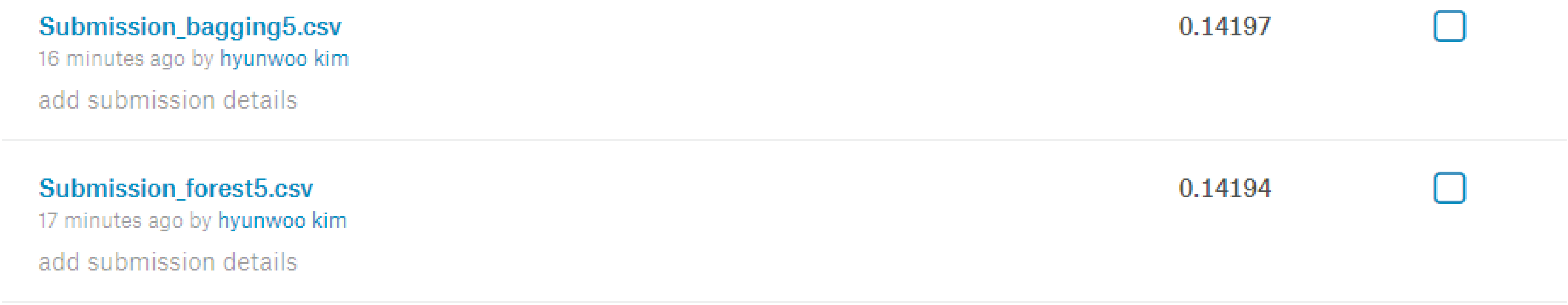
Submission_forest2.csv
18 hours ago by hyunwoo kim
add submission details
0.14333 □

Submission_extratrees2.csv
18 hours ago by hyunwoo kim
add submission details
0.14675 □

Submission_bagging2.csv
18 hours ago by hyunwoo kim
add submission details
0.14321 □

Submission_adaboost2.csv
18 hours ago by hyunwoo kim
add submission details
0.18167 □

Modeling *Model – Random Forest, Bagging*



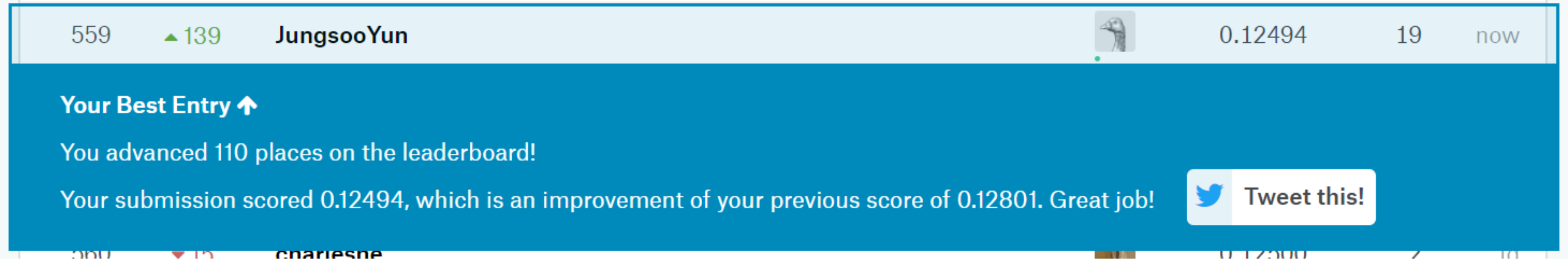
Submission_bagging5.csv
16 minutes ago by hyunwoo kim
add submission details

0.14197 □

Submission_forest5.csv
17 minutes ago by hyunwoo kim
add submission details

0.14194 □

Modeling *Best Model-Neural Network*



559 ▲139 JungsooYun 0.12494 19 now

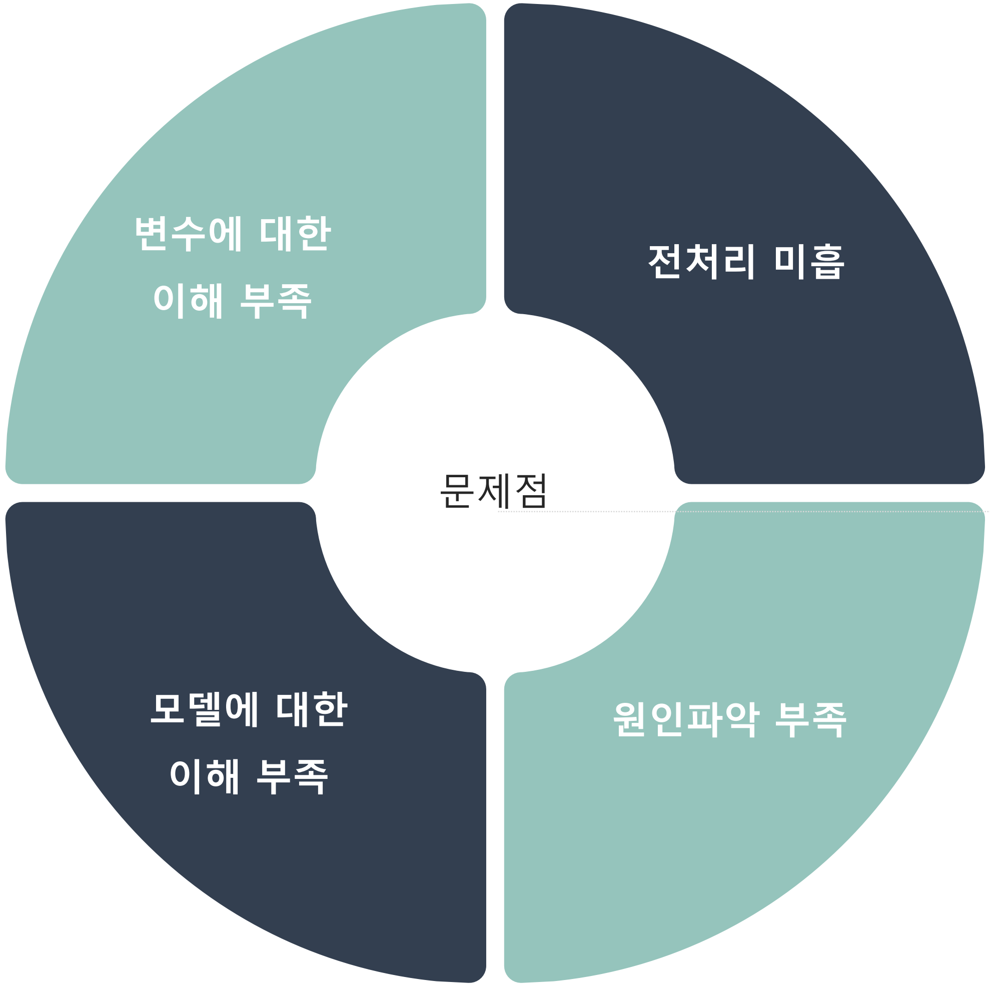
Your Best Entry

You advanced 110 places on the leaderboard!

Your submission scored 0.12494, which is an improvement of your previous score of 0.12801. Great job!

560 ▼15 charleshe 0.12500 2 1d

Analysis



문제점

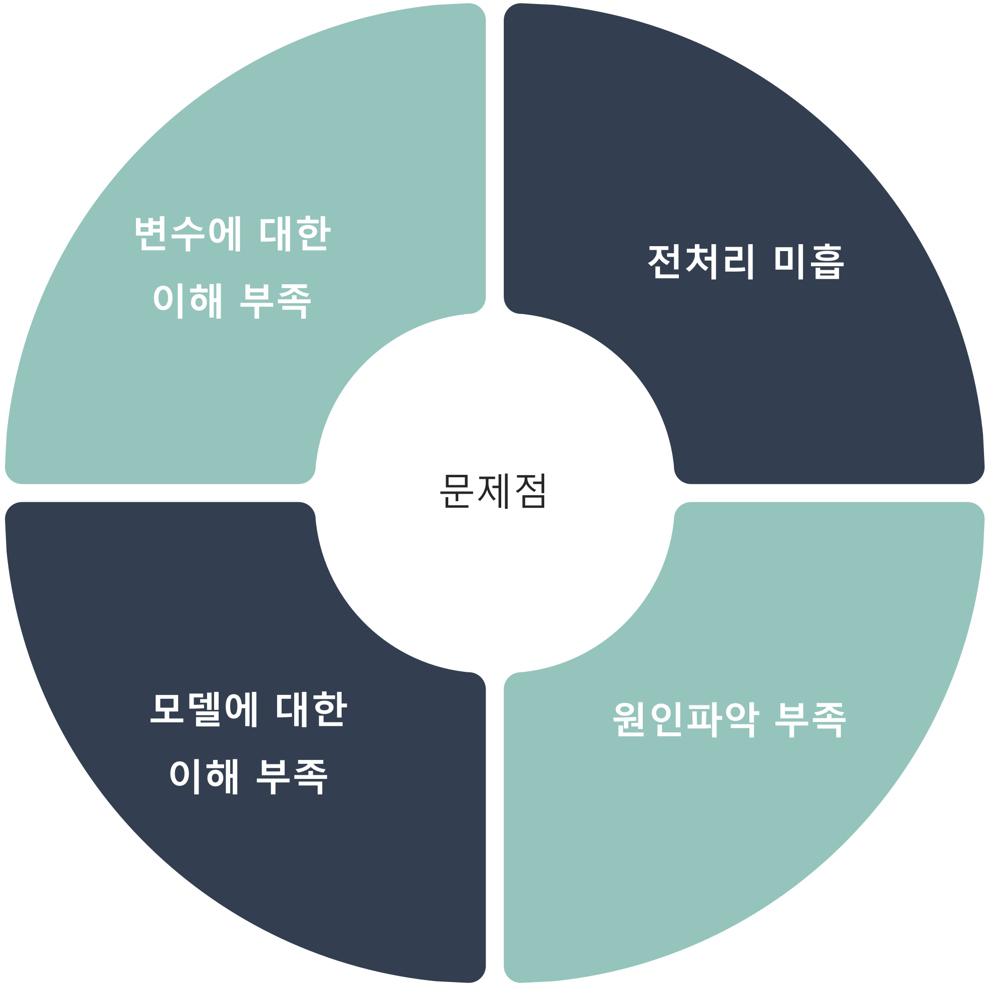
변수에 대한
이해 부족

전처리 미흡

모델에 대한
이해 부족

원인파악 부족

Analysis



1. 과도한 변수로 인한 overfitting

2. 전처리 과정 中, null값 채우기

   ex)정규분포나 추정을 통한 채우기

3. 모델에 대한 이해 부족

   ex) parameter

4. 다양한 모델을 사용못함

5. 결과에 대한 분석 미흡

Conclusion

1. Shift + enter는 생각보다 어렵다.

Conclusion

2. 열심히 공부하자.

Conclusion

감사합니다!!

Q&A

Q&A