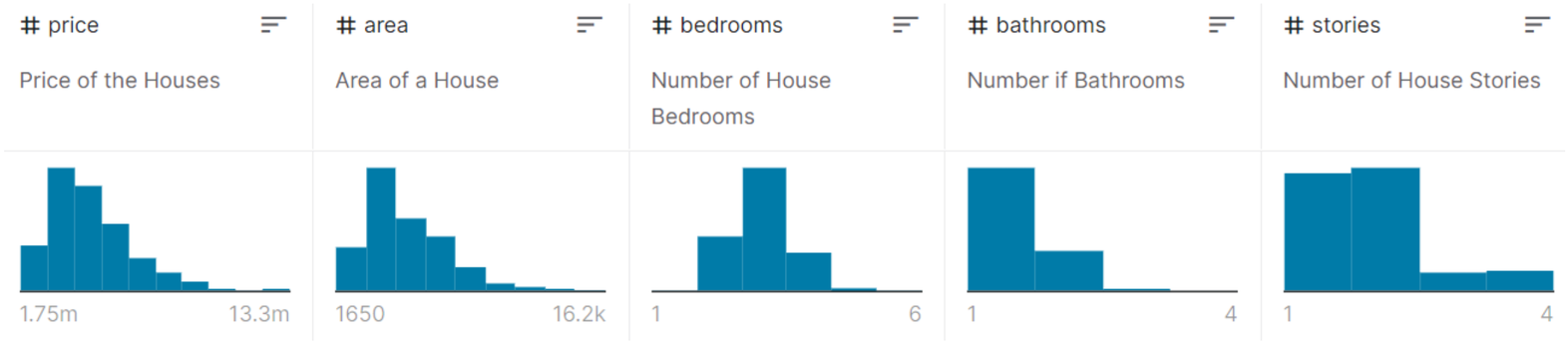
https://www.kaggle.com/yasserh/housing-price-prediction-best-ml-algorithms

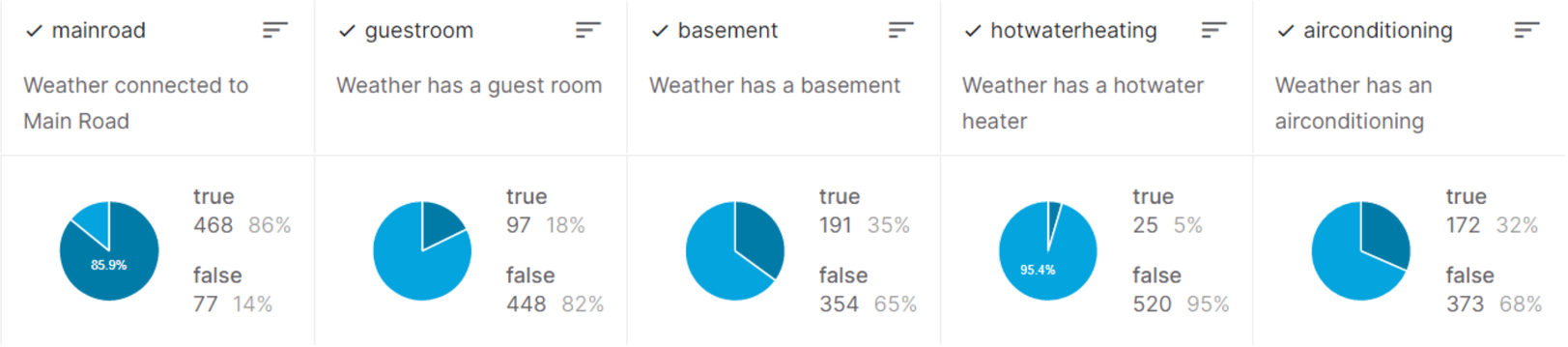
4주차. Housing Price Prediction – (Best ML Algorithms)

목차

1. 기본적인 정보

Housing 데이터를 이용해 예측
- 13개의 columns
- 545 observations





1. 기본적인 정보

Stractegic Plan of Action:

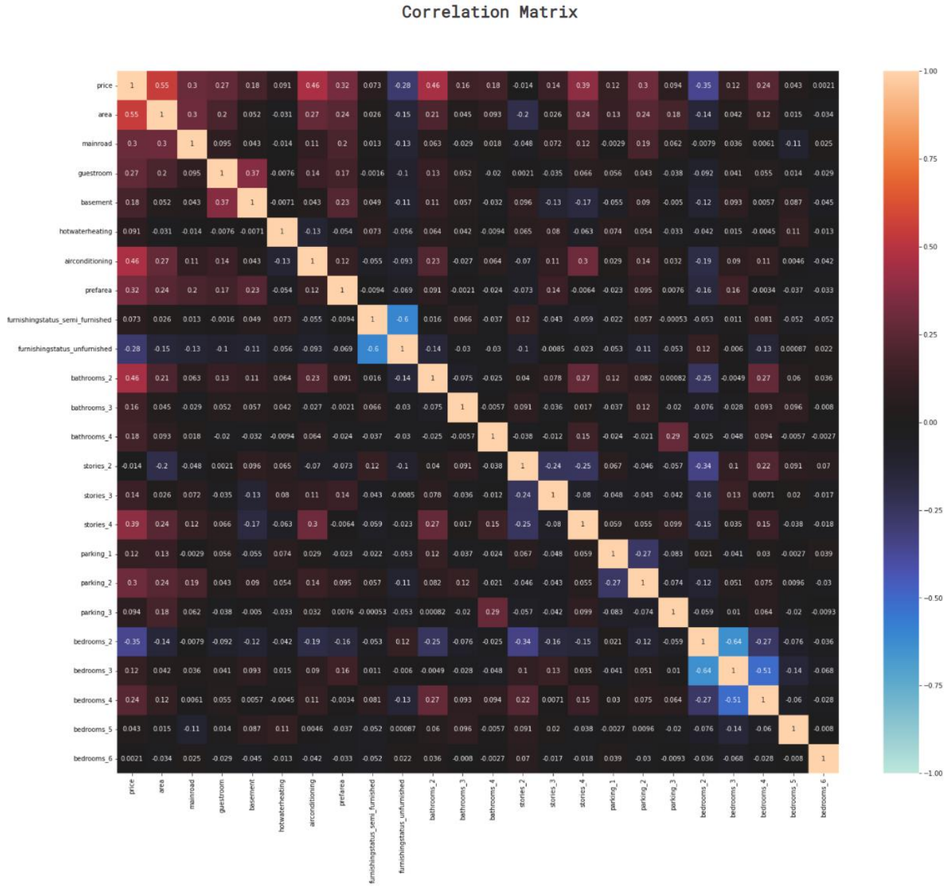
We aim to solve the problem statement by creating a plan of action, Here are some of the necessary steps:

1. Data Exploration
2. Exploratory Data Analysis (EDA)
3. Data Pre-processing
4. Data Manipulation
5. Feature Selection/Extraction
6. Predictive Modelling
7. Project Outcomes & Conclusion

집중해서 확인하고 싶은 파트!!

2. 다중공산성 해결방법

1. 다중공산성 확인 - Heatmap



➡ 다중공산성 확인
추가적으로 OLS 이용해 자세히 확인

2. 다중공산성 해결방법

https://www.statsmodels.org/0.8.0/generated/statsmodels.regression.linear_model.OLS.html

1. 다중공산성 확인 - Linear regression model

- statsmodels.formula에서 제공하는 OLS 사용 `from statsmodels.formula import api`
- 이외에도 statsmodels.regression.linear_model 에서 OLS 제공
  - 여기서는 OLS(Y,X)로 fit하면 된다.
  - from_formula라는 method 존재해 직접 식을 넣어줘도 된다.

```
#Testing a Linear Regression model with statsmodels

Train_xy = pd.concat([Train_X_std,Train_Y.reset_index(drop=True)],axis=1)
a = Train_xy.columns.values

API = api.ols(formula='{} ~ {}'.format(target,' + '.join(i for i in Train_X.columns)), data=Trai
n_xy).fit()
#print(API.conf_int())
#print(API.pvalues)
API.summary()
```

R과 같은 형식

2. 다중공산성 해결방법

https://medium.com/swlh/interpreting-linear-regression-through-statsmodels-summary-4796d359035a

1. 다중공산성 확인 - Linear regression model

OLS Regression Results

Dep. Variable:	price	R-squared:	0.679
Model:	OLS	Adj. R-squared:	0.661
Method:	Least Squares	F-statistic:	36.96
Date:	Wed, 09 Feb 2022	Prob (F-statistic):	2.06e-84
Time:	20:12:46	Log-Likelihood:	-6509.2
No. Observations:	426	AIC:	1.307e+04
Df Residuals:	402	BIC:	1.316e+04
Df Model:	23		
Covariance Type:	nonrobust		

- R-squared : 모델의 설명력
- Adj.R-squared : multiple dependent variables' efficacy
- F-statistic : group of variables are significant?
- AIC/BIC : feature selection에 사용

Omnibus:	96.025	Durbin-Watson:	2.025
Prob(Omnibus):	0.000	Jarque-Bera (JB):	274.474
Skew:	1.058	Prob(JB):	2.51e-60
Kurtosis:	6.315	Cond. No.	26.3

- Omnibus : skew와 kurtosis를 이용한 residual들의 분포를 정규화한 것
  -> 0일수록 좋음
- Prob(Omnibus) : residual 분포가 normal인지 -> 1일수록 좋음
- Skew : symmetry of data -> 0일수록 좋음
- Kurtosis : peakiness of data -> 높을수록 outlier 적음
- Durbin-Watson : homoscedasticity -> 1~2 사이가 안정적
- Prob(JB) : Prob(Omnibus)와 같다고 취급
- Condition number : 데이터의 사이즈에 비해 모델의 sensitivity
  -> condition number이 높을수록 multicollinearity 강함

2. 다중공산성 해결방법

1. 다중공산성 확인 - Linear regression model

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

다중공산성이 있을 경우 coefficient들 간 standard error가 커진다.

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4.717e+06	5.22e+04	90.378	0.000	4.61e+06	4.82e+06
area	4.356e+05	6.41e+04	6.799	0.000	3.1e+05	5.62e+05
mainroad	1.785e+05	5.77e+04	3.092	0.002	6.5e+04	2.92e+05
guestroom	1.197e+05	5.78e+04	2.071	0.039	6082.572	2.33e+05
basement	1.712e+05	6.15e+04	2.784	0.006	5.03e+04	2.92e+05
hotwaterheating	2.006e+05	5.48e+04	3.662	0.000	9.29e+04	3.08e+05
airconditioning	3.635e+05	5.89e+04	6.168	0.000	2.48e+05	4.79e+05
prefarea	2.711e+05	5.75e+04	4.711	0.000	1.58e+05	3.84e+05
furnishingstatus_semi_furnished	1.509e+04	6.71e+04	0.225	0.822	-1.17e+05	1.47e+05
furnishingstatus_unfurnished	-1.688e+05	6.78e+04	-2.489	0.013	-3.02e+05	-3.55e+04
bathrooms_2	3.722e+05	5.98e+04	6.224	0.000	2.55e+05	4.9e+05
bathrooms_3	1.886e+05	5.4e+04	3.492	0.001	8.24e+04	2.95e+05
bathrooms_4	2.801e+05	5.68e+04	4.934	0.000	1.69e+05	3.92e+05
stories_2	1.341e+05	6.97e+04	1.923	0.055	-2986.085	2.71e+05
stories_3	2.289e+05	6.13e+04	3.735	0.000	1.08e+05	3.49e+05
stories_4	3.725e+05	6.46e+04	5.764	0.000	2.45e+05	5e+05
parking_1	1.67e+05	5.78e+04	2.887	0.004	5.33e+04	2.81e+05
parking_2	2.781e+05	5.97e+04	4.662	0.000	1.61e+05	3.95e+05
parking_3	-5.772e+04	5.72e+04	-1.009	0.314	-1.7e+05	5.47e+04
bedrooms_2	-3.385e+04	4.8e+05	-0.070	0.944	-9.78e+05	9.11e+05
bedrooms_3	1.077e+05	5.45e+05	0.197	0.844	-9.64e+05	1.18e+06
bedrooms_4	1.215e+05	4.18e+05	0.291	0.771	-7e+05	9.43e+05
bedrooms_5	3.933e+04	1.66e+05	0.237	0.812	-2.86e+05	3.65e+05
bedrooms_6	8.462e+04	7.49e+04	1.130	0.259	-6.26e+04	2.32e+05

2. 다중공산성 해결방법

https://www.analyticsvidhya.com/blog/2020/03/what-is-multicollinearity/

2. VIF – Manual method

- R^2 : 한 독립변수가 다른 독립변수들을 얼마나 잘 설명하는지 보여주는 지수

  -> R^2가 높을수록 다른 변수들과의 상관도가 높음.

  => R^2가 1에 가까울수록 VIF는 높고, 다중공산성 또한 높아짐

$$VIF = \frac{1}{1-R^2}$$

- VIF는 1에서 시작 + 최대치는 없음
- VIF=1 : 다중공산성 없음
- VIF가 5 또는 10 이상 : 다중공산성 매우 높음

- 해결방법

  ① VIF가 높은 순서대로!! 변수들 drop

  ② 여러 변수들을 합쳐 하나로 만들고 나머지 drop

2. 다중공산성 해결방법

2. VIF – Manual method

```
if vif.loc[0][1]>1:
    DROP.append(vif.loc[0][0])
    LR = LinearRegression()
    LR.fit(Train_X_std.drop(DROP,axis=1), Train_Y)

    pred1 = LR.predict(Train_X_std.drop(DROP,axis=1))
    pred2 = LR.predict(Test_X_std.drop(DROP,axis=1))

    Trr.append(np.sqrt(mean_squared_error(Train_Y, pred1)))
    Tss.append(np.sqrt(mean_squared_error(Test_Y, pred2)))

    #Trd.loc[i,'ord-'+str(k)] = round(np.sqrt(mean_squared_error(Train_Y, pred1)),2)
    #Tsd.loc[i,'ord-'+str(k)] = round(np.sqrt(mean_squared_error(Test_Y, pred2)),2)
```

여기서는 1 이상이기만 하면 바로 drop?!?

	Features	VIF
0	bedrooms_3	109.20
1	bedrooms_2	84.76
2	bedrooms_4	64.05
3	bedrooms_5	10.07
4	bedrooms_6	2.06
5	stories_2	1.78
6	furnishingstatus_unfurnished	1.69
7	furnishingstatus_semi_furnished	1.65
8	stories_4	1.53
9	area	1.51
10	basement	1.39
11	stories_3	1.38
12	bathrooms_2	1.31
13	parking_2	1.31
14	airconditioning	1.28
15	parking_1	1.23
16	guestroom	1.23
17	mainroad	1.22
18	prefarea	1.22
19	parking_3	1.20
20	bathrooms_4	1.18
21	hotwaterheating	1.10
22	bathrooms_3	1.07

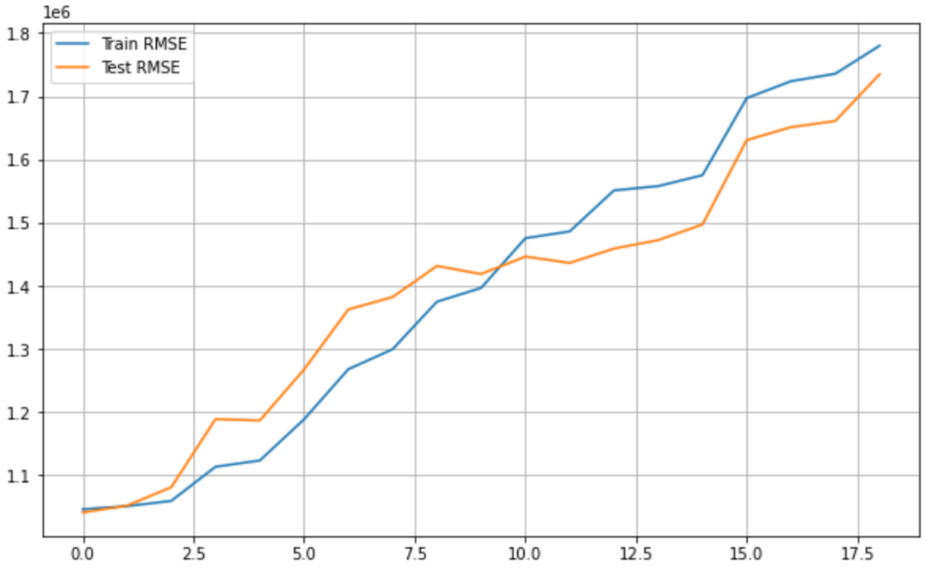
모든 피처들을 포함해 VIF 모델 돌렸을 때 나온 결과 =>

2. 다중공산성 해결방법

2. VIF – Manual method

```
Dropped Features -->  ['bedrooms_3', 'stories_2', 'furnishingstatus_unfurnished', 'area', 'b
asement', 'bathrooms_2', 'airconditioning', 'bedrooms_2', 'parking_2', 'bathrooms_4', 'prefa
rea', 'hotwaterheating', 'mainroad', 'furnishingstatus_semi_furnished', 'bathrooms_3', 'stor
ies_4', 'stories_3', 'parking_1', 'bedrooms_4']
```

⇒ 19개의 column drop
⇒ 모든 피처들을 갖고 VIF를 돌렸을 때 결국 4개의 피처로만 예측모델 만들었다는 말..?



2. 다중공산성 해결방법

https://machinelearningmastery.com/rfe-feature-selection-in-python/

https://www.analyticsvidhya.com/blog/2020/10/a-comprehensive-guide-to-feature-selection-using-wrapper-methods-in-python/#:~:text=In%20wrapper%20methods%2C%20the%20feature,features%20against%20the%20evaluation%20criterion.

https://www.guru99.com/greedy-algorithm.html

2. RFE (Recursive Feature Elimination) – automatic method

- **wrapper-type feature selection**

  : 데이터셋에 사용할 머신러닝 알고리즘을 기반으로 feature selection 진행

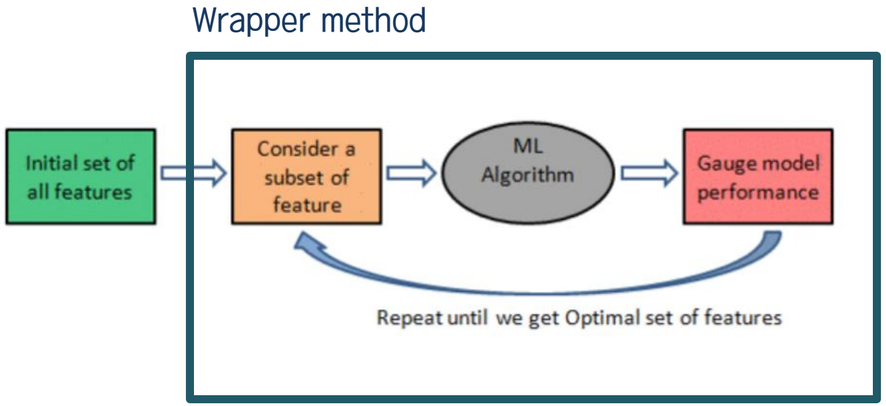
  : 특정 평가지표에 따라 greedy search approach 사용 (세번째 링크 참고)

  (ex1) regression – p-value, R.squared, Adj.R.squared

  (ex2) classification – accuracy, precision, recall, f1-score

  : 종류에 따라 forward selection, backward elimination, stepwise selection

  (두번째 링크 참고)



- 중요한 configuration option

  ① 피처 개수

  ② 피처 선택하는데 있어 사용하는 모델 종류

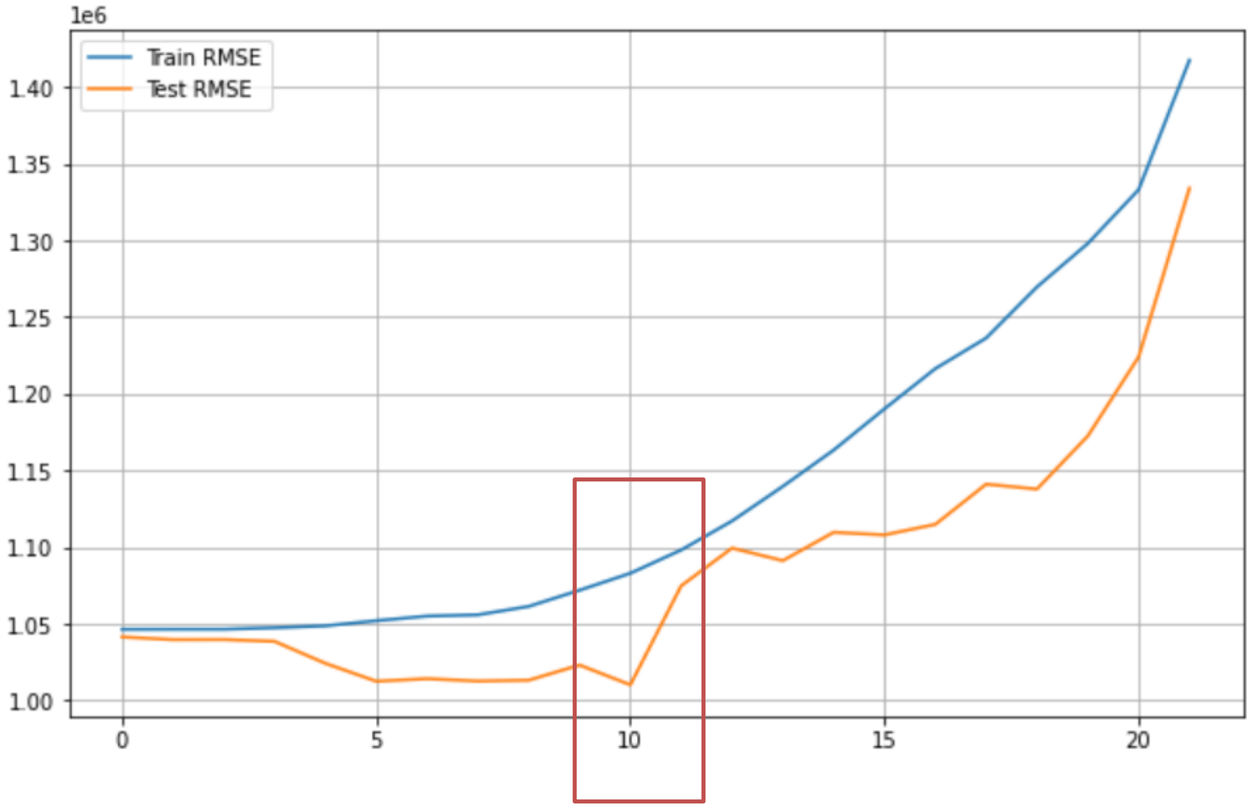
  -> 사용하는 모델이 classification인지 regression인지에 따라 달라짐

  -> 여기서는 regression 모델 사용

```
m=df.shape[1]-2
for i in range(m):
    lm = LinearRegression()
    rfe = RFE(lm,n_features_to_select=Train_X_std.shape[1]-i)        # running RFE
    rfe = rfe.fit(Train_X_std, Train_Y)

    LR = LinearRegression()
    LR.fit(Train_X_std.loc[:,rfe.support_], Train_Y)

    pred1 = LR.predict(Train_X_std.loc[:,rfe.support_])
    pred2 = LR.predict(Test_X_std.loc[:,rfe.support_])

    Trr.append(np.sqrt(mean_squared_error(Train_Y, pred1)))
    Tss.append(np.sqrt(mean_squared_error(Test_Y, pred2)))
```

2. 다중공산성 해결방법

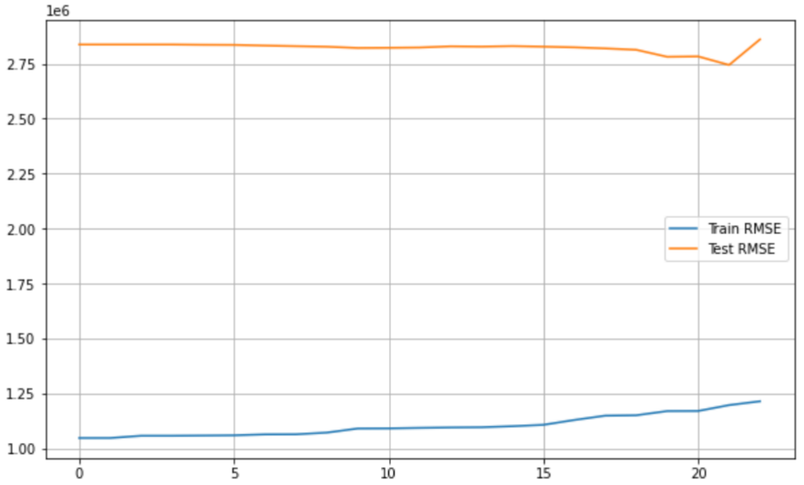
2. RFE (Recursive Feature Elimination) – automatic method



2. 다중공산성 해결방법

2. PCA – feature elimination

```
m=df.shape[1]-1

for i in range(m):
    pca = PCA(n_components=Train_X_std.shape[1]-i)
    Train_X_std_pca = pca.fit_transform(Train_X_std)
    Test_X_std_pca = pca.fit_transform(Test_X_std)

    LR = LinearRegression()
    LR.fit(Train_X_std_pca, Train_Y)

    pred1 = LR.predict(Train_X_std_pca)
    pred2 = LR.predict(Test_X_std_pca)

    Trr.append(round(np.sqrt(mean_squared_error(Train_Y, pred1)),2))
    Tss.append(round(np.sqrt(mean_squared_error(Test_Y, pred2)),2))
```



3. 아쉬운 점

1. VIF로 다중공산성을 확인할 때 왜 굳이 1을 썼는지
2. 데이터셋마다의 특성이 서로 다를 텐데 이를 확인하고 같은 프로세스를 사용한 건지