Course Title (과목명)	기계학습기초 (01)
HW Number (HW 번호)	Project
Submit Date (제출일)	2020-12-09
Grade (학년)	3rd Grade
ID (학번)	20161482
Name (이름)	박준용

인구조사 데이터를 이용한 주택 가격 예측 모델

- 주어진 인구 데이터를 사용하여 각 구역의 중간 주택 가격을 예측하는 모델 구현
  - 데이터 분석 방법
  - 모델 구현 방법
  - 성능 평가 방법

이 프로젝트에서는 캘리포니아의 인구조사 데이터를 이용하여 주택의 가격을 예측하는 모델을 제작하는 것을 목표로 한다. 우선, 주어진 데이터의 특성을 파악하고 특성 공학을 활용한 전처리를 수행한 뒤, 적합한 머신러닝 알고리즘을 찾아 구현하여 오류를 최소화하고 정확한 예측을 가능하게 한다.

해당하는 데이터에 대한 전처리 및 분석에는 outliers 제거를 통한 skewness 감소 및 null data imputation, categorical value의 변환 및 cross correlation을 이용한 연관 데이터 작성 등을 사용하였다. 또한, cross validation 방법을 사용하여 적합한 머신러닝 알고리즘을 분석하여, Random Forest 방법을 사용한 뒤 hyperparameter를 가장 효과적인 hyperparameter를 적용시켜 모델에 가장 적합한 알고리즘을 직접 구현할 수 있었다.

프로젝트에 사용된 코드에는 데이터의 형식을 지정하기 위한 NumPy와 Pandas, plotting을 위한 matplotlib.pyplot과 seaborn, 그리고 전처리와 머신러닝 라이브러리의 활용을 위한 Scikit-learn의 라이브러리를 사용하였다.

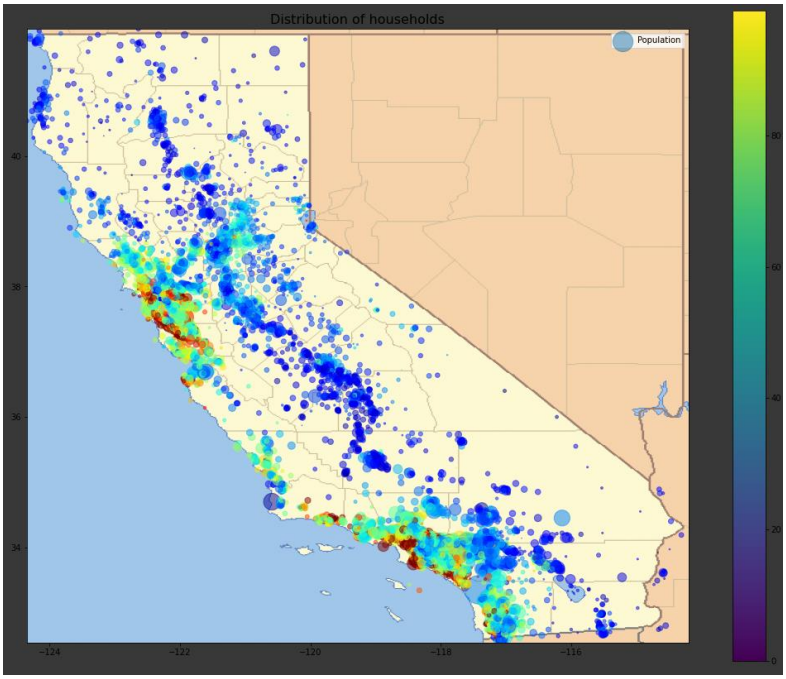
1) 데이터 전처리 및 분석

(a) 사전 데이터 처리

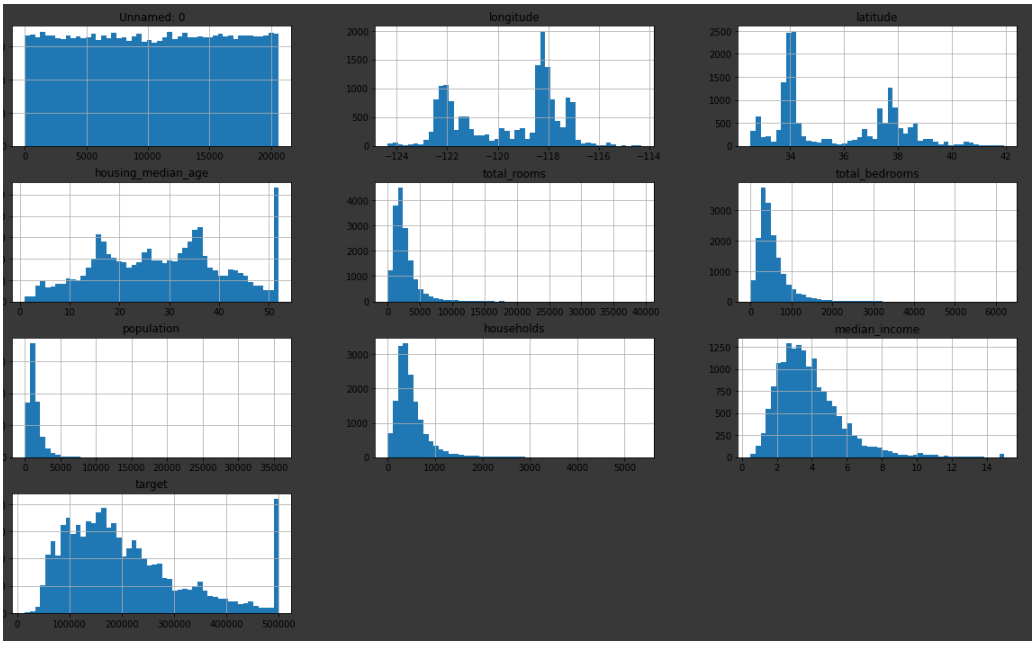
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16512 entries, 0 to 16511
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Unnamed: 0          16512 non-null  int64
 1   longitude           16512 non-null  float64
 2   latitude            16512 non-null  float64
 3   housing_median_age  16512 non-null  int64
 4   total_rooms         16512 non-null  int64
 5   total_bedrooms      16354 non-null  float64
 6   population          16512 non-null  int64
 7   households          16512 non-null  int64
 8   median_income       16512 non-null  float64
 9   ocean_proximity     16512 non-null  object
 10  target              16512 non-null  int64
dtypes: float64(4), int64(6), object(1)
memory usage: 1.4+ MB
```

	Unnamed: 0	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	target
count	16512.000000	16512.000000	16512.000000	16512.000000	16512.000000	16354.000000	16512.000000	16512.000000	16512.000000	16512.000000
mean	10332.352108	-119.575834	35.639577	28.653101	2622.728319	534.973890	1419.790819	497.060380	3.875589	206990.920724
std	5979.473431	2.001860	2.138058	12.574726	2138.458419	412.699041	1115.686241	375.720845	1.904950	115703.014830
min	0.000000	-124.350000	32.540000	1.000000	6.000000	2.000000	3.000000	2.000000	0.499900	14999.000000
25%	5157.750000	-121.800000	33.940000	18.000000	1443.000000	295.000000	784.000000	279.000000	2.566775	119800.000000
50%	10341.000000	-118.510000	34.260000	29.000000	2119.500000	433.000000	1164.000000	408.000000	3.540900	179500.000000
75%	15522.500000	-118.010000	37.720000	37.000000	3141.000000	644.000000	1719.250000	602.000000	4.744475	263900.000000
max	20638.000000	-114.310000	41.950000	52.000000	39320.000000	6210.000000	35682.000000	5358.000000	15.000100	500001.000000

우선, 적합한 머신러닝 알고리즘을 선택하기 위하여 주어진 데이터셋을 분석할 수 있다. pandas의 info 함수와 describe 함수를 통하여 raw data의 특성을 살펴볼 수 있다.
첫 번째로 feature 중 total_bedroom 값에 null value가 있어 다른 feature보다 count가 적은 것을 확인할 수 있다. 또한, feature 중 ocean_proximity가 연속성을 띄는 다른 feature와 달리 categorical한 것 역시 확인할 수 있다.

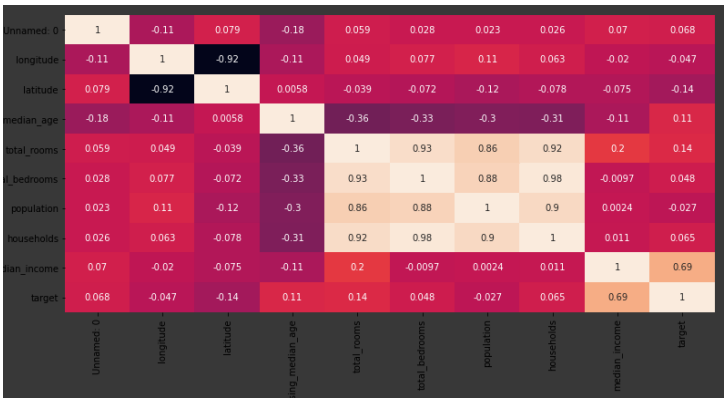


matplotlib의 scatter plot을 이용하여 Project 안내문에 표기되어 있는 figure를 작성할 수 있었다. 가격의 고저차는 색깔로 표현하였으며, 인구의 많고 적음은 plot의 크기를 통해 표현하였다. 이를 캘리포니아 주의 외부 이미지와 일치시켜 전체적인 경향을 파악할 수 있다.



Histogram을 활용하여 continuous한 값의 경향성을 알아볼 수 있다. 우선 gaussian 분포와 가까운 값을 띄는 feature들의 경우 값의 치우침이 심한 것을 확인할 수 있다. 또한, housing_median_age와 target의 큰 값에 데이터가 몰려 있는데, 이는 일정 상한의 값을 하나의 값으로 치환하여 정리한 것임을 추측할 수 있다.

```
target                1.000000
median_income         0.687160
total_rooms           0.135097
housing_median_age    0.114110
Unnamed: 0            0.067723
households            0.064506
total_bedrooms        0.047689
population           -0.026920
longitude            -0.047432
latitude             -0.142724
Name: target, dtype: float64
```



또한, 위와 같이 correlation matrix를 사용하여 target과 다른 feature 사이에 어느 정도의 correlation이 존재하는지 역시 파악할 수 있다. 0.15보다 낮은 correlation coefficient를 갖는 여타 feature와 달리, median_income feature는 약 0.68의 훨씬 큰 correlation coefficient를 가져, median_income이 주택 가격에 큰 영향을 미친다는 사실을 알 수 있다.

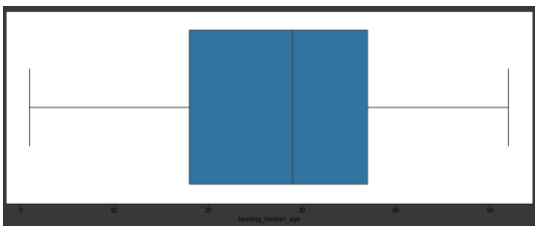
```
<1H OCEAN     7276
INLAND        5263
NEAR OCEAN    2124
NEAR BAY      1847
ISLAND           2
Name: ocean_proximity, dtype: int64
```

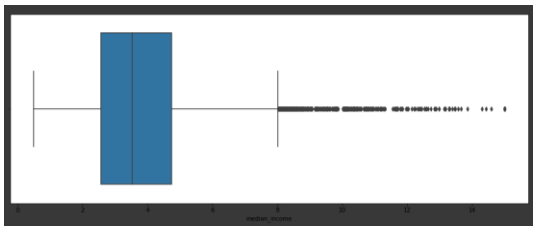
마지막으로 유일한 categorical feature인 ocean_proximity의 value를 살펴보았다. 이 경우 island의 value가 매우 적어, one-hot encoding을 적용하였을 시 test set을 나누거나 k-fold validation 방법을 사용할 때 dimension이 맞지 않는 경우가 생길 수 있다.

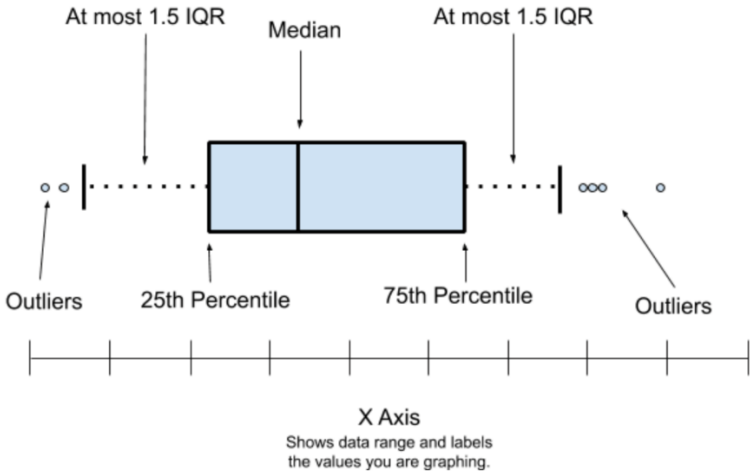
위의 정보를 토대로 아무런 정보를 나타내지 않으며 correlation도 없는 Unnamed:0의 feature를 제거하고, ocean_proximity의 ISLAND 값을 대체하기로 하였다. 위의 지도에서 island의 위치를 확인한 결과 bay(만) 주변이었으므로 NEAR BAY 값으로 대체할 수 있다. 이렇게 가공된 DataFrame을 train_test_split 함수를 통하여 무작위로 train set과 test set으로 분할할 수 있으며, 분할 크기는 4:1로 분할하였다.

(b) Outlier 제거

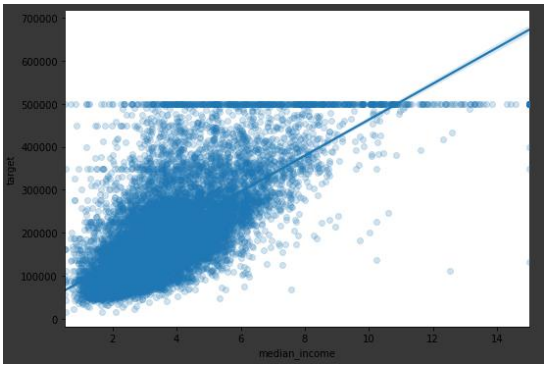
먼저 train set의 가공을 할 수 있다. 우선적으로, 위의 histogram에서 관찰하였듯 feature의 편향이 매우 심하므로 정규화를 위해서는 먼저 outlier를 제거하여 skewness를 줄일 필요가 있다. seaborn의 boxplot 함수를 통해 outlier를 시각적으로 확인할 수 있다.

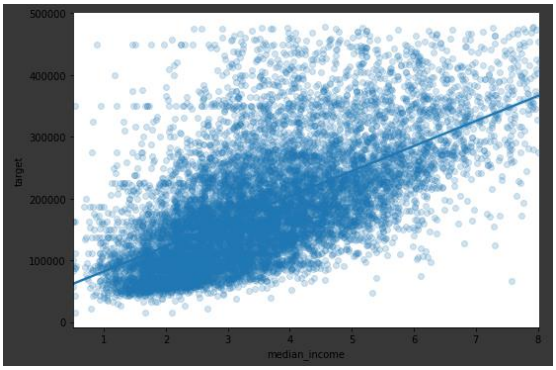






Outlier의 정의는 box의 너비에 해당하는 Interquartile Ranges의 1.5배만큼의 값이 양쪽 quartile에 각각 extended 되었을 때, 그 이상 또는 이하의 값을 뜻한다. 왼쪽 feature (housing_median_age)는 outlier에 해당하는 값이 없지만, 오른쪽 feature(median_income)의 경우에는 maximum 값보다 큰 outliner가 존재하는 것을 확인할 수 있다. 이러한 값을 전체 dataset에서 제거하기 위하여 하나라도 outlier가 존재하는 column을 제거하는 code를 작성하였다.





	skewness
population	5.027668
total_rooms	4.159476
total_bedrooms	3.358739
households	3.313178
median_income	1.604107
target	0.987234
latitude	0.454154
housing_median_age	0.061392
longitude	-0.288570

	skewness
target	0.783150
total_rooms	0.671702
total_bedrooms	0.649427
population	0.627362
households	0.604922
median_income	0.563819
latitude	0.408314
housing_median_age	0.018955
longitude	-0.251755

위의 그래프는 target과 가장 큰 correlation을 보이는 median_income과의 graph 변화이다. 500k 주변에 밀집되어 있는 데이터가 정리되어 정규화에 가까워지는 것을 볼 수 있다. 아래는 outlier를 제거한 후의 skewness 변화를 skewness 함수를 통하여 정리하였다. 매우 큰 skewness 값을 가지던 population, total_rooms 등의 feature가 크게 감소한 것을 확인할 수 있다.

(c) Null Value Imputation

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms      108
population            0
households            0
median_income         0
ocean_proximity       0
target                0
dtype: int64
```

위의 null value가 존재하는 feature를 나타낸 값이다. total_bedrooms의 값에 null value가 존재하므로 산입(imputation)이 필요하다. sklearn의 SimpleImputer 함수는 해당 값에 특정 strategy에 의거하여 imputation을 시행하며, 이 경우에는 해당 feature의 median으로 impute하는 code를 수행하였다.

(d) 새로운 Feature 추가

```
data_num['rooms_per_household']=data_num['total_rooms']/data_num['households']
data_num['bedrooms_per_room']=data_num['total_bedrooms']/data_num['total_rooms']
data_num['population_per_household']=data_num['population']/data_num['households']
```

Feature 중 total_rooms와 total_bedrooms는 단순히 room과 bedroom의 총계이므로 해당 주택의 가치를 평가하는 데에 크게 영향을 미치지 못하며, 가구(household)당 room과 bedroom의 개수가 더욱 유의미한 관계를 나타낼 것이다. 또한, room 대비 bedroom의 개수 역시 개별적인 feature보다 더 큰 상관관계를 나타낼 것으로 예상할 수 있다. 이를 통하여 3가지의 새로운 feature를 추가할 수 있다.

```
target                     1.000000
median_income              0.625075
total_rooms                0.197688
households                 0.126746
housing_median_age         0.099117
rooms_per_household        0.098587
total_bedrooms             0.098518
population                -0.006662
longitude                 -0.040372
latitude                  -0.158286
bedrooms_per_room         -0.185926
population_per_household  -0.231579
```

추가된 feature의 correlation coefficient를 조사하였더니 bedrooms_per_room과 population_per_household feature의 경우 각각 -0.186과 -0.232의 연관성을 나타내어 기존 feature보다 유의미한 값을 얻는 데 성공하였다.

(e) Categorical 값의 One-Hot Encoding

```
[array(['<1H OCEAN', 'INLAND', 'NEAR BAY', 'NEAR OCEAN'], dtype=object)]
```

앞서 살펴보았던 ocean_proximity data를 continuous한 값으로 사용하기 위해서 one hot encoding을 사용할 수 있다. sklearn의 OneHotEncoder 함수를 통하여 각 4개의 category별로 encode된 array를 얻을 수 있다.

(f) Standard Scaling

마지막으로 sklearn의 StandardScaler 함수를 통하여 numerical value에 해당되는 값을 정규변환하여 z = (x - u)/s의 형태로 변환할 수 있다.

```
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   longitude               11256 non-null  float64
 1   latitude                11256 non-null  float64
 2   housing_median_age      11256 non-null  float64
 3   total_rooms             11256 non-null  float64
 4   total_bedrooms          11256 non-null  float64
 5   population              11256 non-null  float64
 6   households              11256 non-null  float64
 7   median_income           11256 non-null  float64
 8   rooms_per_household     11256 non-null  float64
 9   bedrooms_per_room       11256 non-null  float64
 10  population_per_household  11256 non-null  float64
 11  <1H OCEAN               11256 non-null  float64
 12  INLAND                  11256 non-null  float64
 13  NEAR BAY                11256 non-null  float64
 14  NEAR OCEAN              11256 non-null  float64
dtypes: float64(15)
```

이렇게 생성된 array와 one hot encoded된 array를 concatenate한 뒤, feature에 해당하는 column을 labeling 하여 위와 같이 전처리가 완성된 DataFrame을 만들 수 있다.

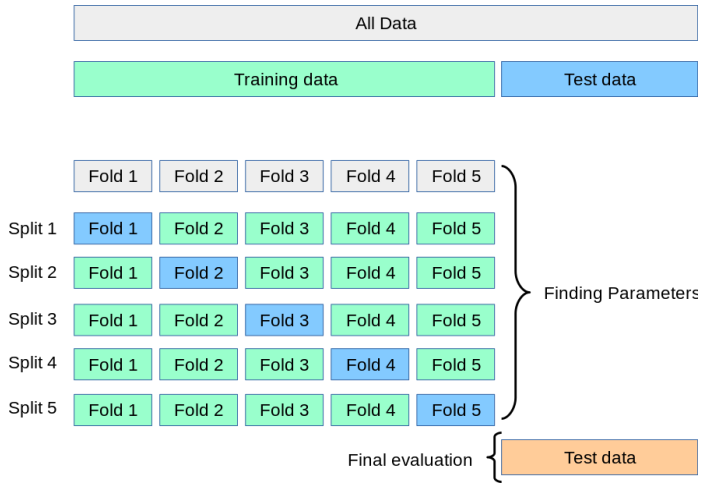
2) 머신러닝 알고리즘 분석

(a) 최적 알고리즘 선택

sklearn 라이브러리는 다양한 regression model을 제공한다. 이에 자주 사용되는 6개의 model을 선정하였으며, 아래와 같다.

Linear Regression	Ridge Regression
Decision Tree Regressor	Random Forest Regressor
SVR (Support Vector Regression)	Gradient Boosting Regressor

위의 regression model에 위의 dataframe과 target value를 입력한 뒤 RMSE를 각각 비교하여 최적의 알고리즘을 선택할 수 있다. 이 경우, 각 regression model의 hyperparameter는 설정하지 않은 채로 진행하였다.

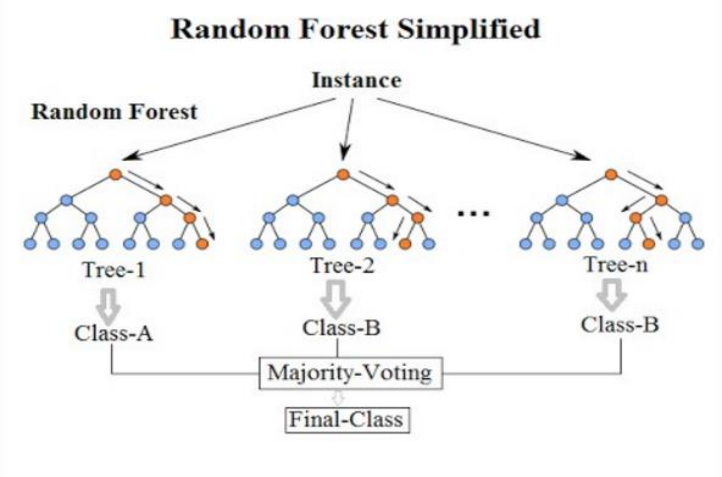


이때, 성능을 평가할 때는 train error RMSE가 아닌 test error RMSE가 중요한 parameter이기 때문에 sklearn의 cross_val_score 함수를 통한 cross validation RMSE를 사용하여 model의 성능을 평가할 수 있다. cross_val_score 함수는 위의 figure와 같이 k-fold validation 방법을 통하여 training dataset 내에서 split별로 validation set을 생성하여 이를 training set와 비교, 각각의 validation RMSE를 저장한다. k번의 과정을 마친 뒤에는 저장된 RMSE의 평균을 계산하여 cross-validation RMSE를 도출한다. 계산된 cross-validation RMSE를 아래 표에 정리하였다.

Model	Train Error	Cross-Validation Error
Linear	RMSE(Train Error): 56503.97826146843	RMSE: 56607.94741870197
Ridge	RMSE(Train Error): 56503.993788437	RMSE: 56607.80815533913
Decision Tree	RMSE(Train Error): 0.0	RMSE: 63124.18769410842
Random Forest	RMSE(Train Error): 16444.644226120556	RMSE: 44608.02903095743
SVR	RMSE(Train Error): 94392.43706585978	RMSE: 94456.5898099344
Gradient Boosting	RMSE(Train Error): 44197.23661814588	RMSE: 47161.05952184643

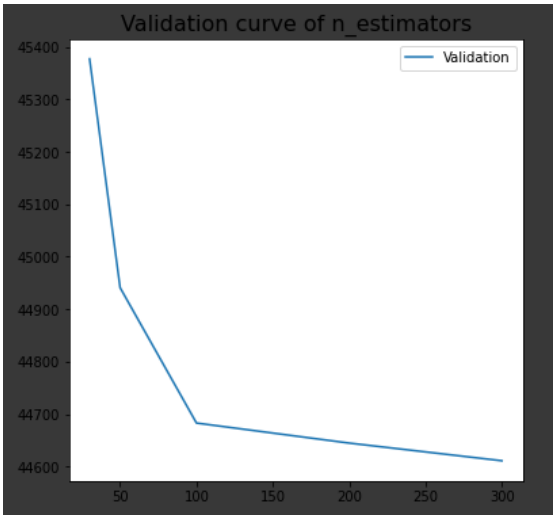
cross-validation RMSE가 가장 낮은 regression model은 Random Forest Regressor이므로 해당 모델을 선정하여 hyperparameter의 미세한 설정을 할 수 있다.

(b) Hyperparameter 설정

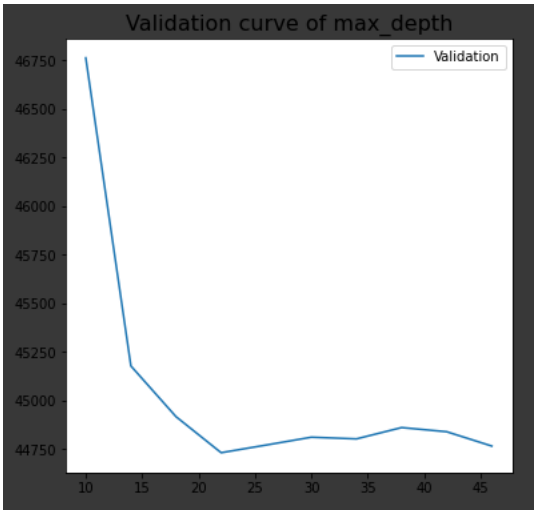


Random Forest 방법은 model에서 구성한 임의로 작동하는 다수의 decision tree로부터 평균 예측값을 얻어 regression을 가능하게 하는 방법이다. 이 때 가장 큰 영향을 미치는 parameter는 forest의 크기 (decision tree의 개수)와 maximum depth이다. 또한, 사용되는 dataset에 여러 feature가 있으므로 maximum feature의 값을 설정하여 random node의 최적화를 할 수 있다.

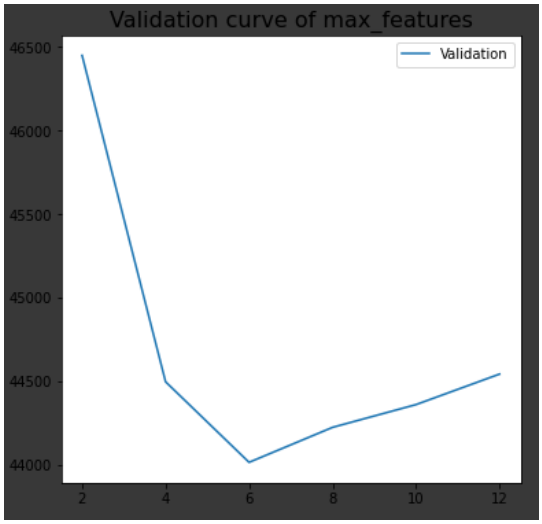
우선, plot을 위하여 validation curve를 그리는 함수를 지정하였다. 이 함수는 cross_val_score 함수에서 출력하는 neg_mean_squared_error를 시각화하여 어느 한 parameter의 증감에 따라 Validation RMSE를 plot한다. 이를 통하여 sklearn에서 제공하는 RandomForestRegressor의 parameter인 n_estimators, max_depth, max_features의 가장 적합한 hyperparameter를 예측할 수 있다.



param_range = [30, 50, 100, 200, 300]



param_range = range(10,50,4)



param_range = range(2,14,2)

위의 함수를 보았을 때 n_estimators의 경우 값이 늘어날수록 미세하게 Validation RMSE가 감소하는 것을 확인할 수 있었으며, max_depth의 경우 n=21~23 정도, max_features의 경우 n=5~7 정도에서 최소값을 보인 후 overfitting이 진행되는 것을 확인할 수 있었다.

이를 정확하게 확인하기 위해서 GridSearchCV 함수를 사용할 수 있다. n_estimators의 값을 300으로 고정하고 위의 범위 int value에 대해서 함수를 사용하였다. best_params_ 명령어를 통해서 시도한 조합 중 가장 RMSE가 낮은 parameter를 얻을 수 있다.

```
{'max_depth': 23, 'max_features': 6, 'n_estimators': 300}

43941.97036536455 {'max_depth': 21, 'max_features': 5, 'n_estimators': 300}
43944.6101066475 {'max_depth': 21, 'max_features': 6, 'n_estimators': 300}
43965.215097853536 {'max_depth': 21, 'max_features': 7, 'n_estimators': 300}
43950.48025343575 {'max_depth': 22, 'max_features': 5, 'n_estimators': 300}
43895.450393987776 {'max_depth': 22, 'max_features': 6, 'n_estimators': 300}
43906.349291575 {'max_depth': 22, 'max_features': 7, 'n_estimators': 300}
44059.29087493717 {'max_depth': 23, 'max_features': 5, 'n_estimators': 300}
43844.209818832074 {'max_depth': 23, 'max_features': 6, 'n_estimators': 300}
43972.77249189659 {'max_depth': 23, 'max_features': 7, 'n_estimators': 300}
```

cv_results_ 값을 통하여 시도하였던 모든 경우의 수를 볼 수 있으며, best_estimator_는 해당 parameter가 적용된 model을 반환한다. 따라서, 최종적으로 선정된 regression model은 **RandomForestRegressor(n_estimators=300,max_depth= 23, max_features=6)** 이다.

3) 성능 평가

이 project에서는 test dataset이 주어지지 않았으므로 임의로 train_test_split 함수를 사용하여 train set과의 비율이 4:1이 되도록 test set을 제작하였다. 이 test dataset을 사용하여 prediction할 시 cross-validation에서 예상하였던 RMSE와 같은 성능 평가가 이루어지는지 다시 한 번 확인하였다.

우선, test dataset에 train dataset에 적용하였던 전처리 과정을 적용한다. (Outliner 제거, Imputation, 새로운 Feature 추가, One-Hot Encoding, Standard Scale) 이후, 선정된 regression model에 predict 함수를 사용하여 Test Error RMSE를 출력할 수 있다.

```
RMSE(Train Error): 16066.554255950394
RMSE(Cross-Validation Error): 43886.48836752757
RMSE(Test Error): 46236.37061414899
```

Cross-Validation Error와 Test Error가 크게 차이가 나지 않는 모습을 볼 수 있으며 이를 통해 model이 training 과정에서 노출되지 않은 dataset에 대해서도 좋은 regression을 보여준다는 것을 확인할 수 있다.

마지막으로, 위에서 서술한 내용을 토대로 새로운 Test dataset에 대응하기 위하여 train set과 test set을 나누지 않고 모든 dataset을 train set으로 활용하여 데이터 전처리 및 선정된 regression model을 사용하였으며, 해당하는 모델을 joblib 라이브러리를 통하여 model.sav로 저장하였다. 또한, 새로운 test dataset의 전처리를 위한 code 역시 첨부하였다.

참고문헌

House Price Prediction (Regression) with Tensorflow — Keras
(https://medium.com/analytics-vidhya/house-price-prediction-regression-with-tensorflow-keras-4fc49fae7123)

Predicting House Prices using Machine Learning
(https://medium.com/@manilwagle/predicting-house-prices-using-machine-learning-cab0b82cd3f)

California Housing Price Analysis
(https://rpubs.com/Frason/CA_House_Prediction)

Random forest, Wikimedia Foundations.
(https://en.wikipedia.org/wiki/Random_forest)