Information of Case

Brief Description

My target variable is continuous, not classification. That's why I used Regression and KNN models in my homework.

Number of Instances: 506

Attribute Information (in order):

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

Missing Attribute Values: None

Data Exploration

I return top n (5 by default) rows of a data frame with .head()

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

We use describe() to get basic summary statistics for each of the columns.

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.0
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.6
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.1
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.7
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.9
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.3
75%	3.677082	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.9
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.9

There is no null value in any variable and it consists of integer and float values.

```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
black      0
lstat      0
medv       0
dtype: int64
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64
 9   tax      506 non-null    int64
 10  ptratio  506 non-null    float64
 11  black    506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

I separated 20% of my data as test data using train_test_split (random_state=5).

```
x_train.shape
```

```
(404, 13)
```

```
x_test.shape
```

```
(102, 13)
```

Models

a) Multiple Linear Regression Model

This model is regression (multiple variables). Assigns a coefficient to all x features we have to estimate Y (our predicted value). I trained my model using train data from my data that I separated as test-train. Then I calculated the scores for my model with the test data. I also printed the coefficient of each features.

Mean Squared Error: 34.4976

R-square: 0.4925

Conclusion: My r-square value is 0.49. This means that in the model I have set up, we can explain the value of y by 49% with x. Also, in my prediction model, the average squared difference between the estimated values and the actual value is 34.5 (in test data).

b) OLS Model

I built our model on linear regression using OLS method. I fitted the same data in the OLS model and calculated MSE and R2 with the test data.

Mean Squared Error: 35.7552

R-square: 0.4740

Conclusion: My r-square value is 0.47. This means that in the model I have set up, we can explain the value of y by 47% with x. Also, in my prediction model, the average squared difference between the estimated values and the actual value is 35.75 (in test data).

c) Best Subset Selection Model

I calculated the score for each variable with the Best Subset method. Our best score is 43.72, and our variables are 'zn', 'indus', 'nox', 'dis', 'rad', 'ptratio', 'medv'. I will use these 7 features ('zn', 'indus', 'nox', 'dis', 'rad', 'ptratio', 'medv') to make predictions in my model. The remaining 6 variables were not used because my model considered them to be meaningless.

Mean Squared Error: 33.9896

R-square: 0.5000

Conclusion: My r-square value is 0.50. This means that in the model I have set up, we can explain the value of y by 50% with x. Also, in my prediction model, the average squared difference between the estimated values and the actual value is 33.98 (in test data).

d) Forward Selection Model

I calculated the score for each variable with the forward selection method. Our best score is 43.72, and our variables are 'zn', 'indus', 'nox', 'dis', 'rad', 'ptratio', 'medv'. I will use these 7 features ('zn', 'indus', 'nox', 'dis', 'rad', 'ptratio', 'medv') to make predictions in my model. The remaining 6 variables were not used because my model considered them to be meaningless.

Mean Squared Error: 33.9896

R-square: 0.5000

Conclusion: My r-square value is 0.50. This means that in the model I have set up, we can explain the value of y by 50% with x. Also, in my prediction model, the average squared difference between the estimated values and the actual value is 33.98 (in test data).

e) Backward Selection Model

I calculated the score for each variable with the backward selection method. Our best score is 43.72, and our variables are 'zn', 'indus', 'nox', 'dis', 'rad', 'ptratio', 'medv'. I will use these 7 features ('zn', 'indus', 'nox', 'dis', 'rad', 'ptratio', 'medv') to make predictions in my model. The remaining 6 variables were not used because my model considered them to be meaningless.

Mean Squared Error: 33.9896

R-square: 0.5000

Conclusion: My r-square value is 0.50. This means that in the model I have set up, we can explain the value of y by 50% with x. Also, in my prediction model, the average squared difference between the estimated values and the actual value is 33.98 (in test data).

f) Ridge Regression Model

Features are not eliminated in Ridge regression. That's why all my features will be used in my model. I trained my model with train data and calculated my score values using my test data. I also printed the coefficient of each features. Features are not eliminated here, but it can be seen from the coefficients which features are more important. My "age", "tax" and "black" features have less impact. However, I used all the features in my model (including these 3 features with little impact).

Best alpha: 7.5996

Mean Squared Error: 34.2990

R-square: 0.4955

Conclusion: My r-square value is 0.4955. This means that in the model I have set up, we can explain the value of y by 49.5% with x. Also, in my prediction model, the average squared difference between the estimated values and the actual value is 34.299 (in test data).

h) Lasso Regression Model

In Lasso regression, unnecessary features are eliminated. Here, my features with a coefficient of 0 have been removed from the model. The coefficients of "age" and "tax" features were found to be 0 in Lasso regression. This means that "age" and "tax" have no effect on our estimation model. I used the remaining 11 features ('zn', 'indus', 'chas', 'nox', 'rm', 'dis', 'rad', 'ptratio', 'black', 'lstat', 'medv') in my model. I also printed the coefficient of each features.

Mean Squared Error: 34.3226

R-square: 0.4951

Conclusion: My r-square value is 0.4951. This means that in the model I have set up, we can explain the value of y by 49.5% with x. Also, in my prediction model, the average squared difference between the estimated values and the actual value is 34.3226 (in test data).

i) KNN Regression

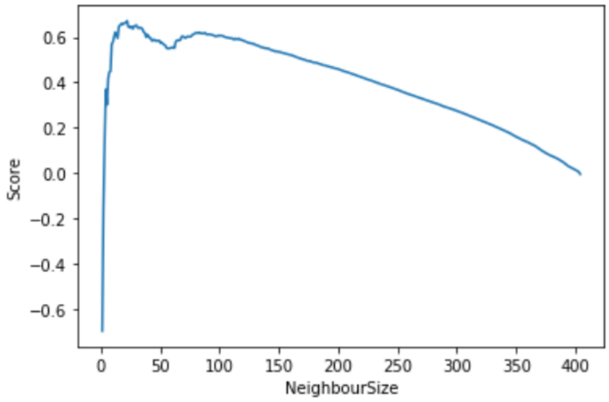
To find the best value of k, I calculated the scores of all k values one by one using the for loop. First of all, I created k values as much as the train data I have. Then I added them to the a_ list with .append (). After, I compute my knn_score for each k (number of neighbors).Then I recorded them in df and sorted them in ascending order according to the scores by giving ascending = False.

In [146]:

```
#for to find best k
a_=[]
for a in range(1,405):
    a_.append(a)
df=pd.DataFrame()
scr=[]
df["k"]=a_
for k in a_:
    knn = KNeighborsRegressor(n_neighbors = k)
    knn.fit(x_train,y_train)
    scr_=knn.score(x_test, y_test)
    scr.append(scr_)
df["score"]=scr
df.sort_values(by="score",ascending=False)
```

Out[146]:

	k	score
21	22	0.671416
20	21	0.665141
19	20	0.661580
18	19	0.660067
16	17	0.659402
...	...	...



The best k value I have found is 22 and the score is 0.671416.

Best k: 22

Mean Squared Error: 11.7033

R-square: 0.6714

Conclusion: While making predictions in my model, I will look at my 22 closest neighbors and make my prediction based on them. My r-square value is 0.6714. This means that in the model I have set up, we can explain the value of y by 67.1% with x. Also, in my prediction model, the average squared difference between the estimated values and the actual value is 11.703 (in test data).

Comparison

I trained all methods with test-train data sets and then calculated the score.

Validation Set MSE and R-square is used for evaluating model performance. Below is the table of different approaches with the MSE and R-squared obtained for Validation Data set.

Model	R-square	MSE
Multiple Linear Regression	0.49	34.49
OLS	0.47	35.75
Best Subset Selection	0.50	33.99
Forward Selection	0.50	33.99
Backward Selection	0.50	33.99
Ridge Regression	0.4955	34.29
Lasso Regression	0.4951	34.32
KNN Regression (k=22)	0.67	11.70

Sorting by MSE and R-square values:

No	Model	R-square	MSE
1	**KNN Regression (k=22)**	0.67	11.70
2	**Best Subset Selection**	0.50	33.99
3	**Forward Selection**	0.50	33.99
4	**Backward Selection**	0.50	33.99
5	**Ridge Regression**	0.4955	34.29
6	**Lasso Regression**	0.4951	34.32
7	**Multiple Linear Regression**	0.49	34.49
8	**OLS**	0.47	35.75

The minimum validation score is obtained for KNN Regression Model with k=22.
Our closest 2^{nd}, 3^{rd} and 4^{th} models are best subset selection, forward selection and backward selection (These gave the same result.)

Result

Since I use the KNN method, my model gives the lowest MSE score at k = 22. And this model includes all the features. So, looking at the closest 22 neighbors while predicting will give us the best forecast model.
Some of my models contain all the features (Example: Ridge) while some don't (Example: Lasso, Subset Selection.). I mentioned above the detailed descriptions of all models (their scores, whether they include all features, my comments…).