BOSTON HOUSING PRICES EXPLORATORY DATA ANALYSIS

09.11.2022

—

AKRITI SOOD

Overview

You are a Data Scientist with a housing agency in Boston MA, you have been given access to a previous dataset on housing prices derived from the U.S. Census Service to present insights to higher management. Based on your experience in Statistics, what information can you provide them to help with making an informed decision?
Using the appropriate graphs and charts, generate basic statistics and visualizations that you think will be useful for the upper management to give them important insight given the question they are asking, in your graphs, include an explanation of each statistic.

.

Goals

1. Is there a significant difference in the median value of houses bounded by the Charles river or not?
2. Is there a difference in median values of houses of each proportion of owner-occupied units built before 1940?
3. Can we conclude that there is no relationship between Nitric oxide concentrations and the proportion of non-retail business acres per town?
4. What is the impact of an additional weighted distance to the five Boston employment centers on the median value of owner-occupied homes?

Data Collection

The Boston Housing Prices is available in various sites such as:

1. Kaggle
2. Sklearn.load_boston
3. Tornoto.edu etc

This dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Mass. It was obtained from the StatLib archive (http://lib.stat.cmu.edu/datasets/boston), and has been used extensively throughout the literature to benchmark algorithms. However, these comparisons were primarily done outside of **Delve** and are thus somewhat suspect. The dataset is small in size with only 506 cases.

The data was originally published by Harrison, D. and Rubinfeld, D.L. `Hedonic prices and the demand for clean air', J. Environ. Economics & Management, vol.5, 81-102, 1978

The dataset contains columns as follows:

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

The target variable is MEDV.

Plan for Data Exploration

1. Installing and Importing Packages
2. Loading Dataset
3. Generating Descriptive Analytics
4. Correlations
5. Log Transformation
6. Feature Scaling
7. Handling Outliers
8. Data Visualizations
9. Hypothesis testing
10. Feature Selection
11. Feature Engineering
12. Conclusion
13. Further Steps
14. Recommendations

Installation and Importing Packages

The language used for this computation is Python in Jupyter Notebooks.

To install packages use

!pip install pandas

The following python packages are required are:

1. Pandas for managing data.
2. Numpy for mathematical calculations.
3. Seaborn for statistical focused plotting methods.
4. Matplotlib for data visualization.
5. Scipy for statistical computation.
6. Statsmodel for statistics model.
7. Sklearn for machine learning.
8. Plotly.express for data visualization.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats
import statsmodels.api as sm
import plotly.express as px
from sklearn.preprocessing import OneHotEncoder, MinMaxScaler, StandardScaler
from sklearn.decomposition import PCA
```

Loading Dataset

```
boston_url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ST0151EN-Skills
boston_df=pd.read_csv(boston_url, index_col = 0)
```

Descriptive Analytics

The descriptive analytics include the description of the dataset, there are various methods in this analytics:

1. Head(): Overviewing first 5 columns of the dataset.

```
boston_df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	5.33	36.2

2. Shape(): This determines the size of the dataset.

```
boston_df.shape
```

```
(506, 13)
```

3. Describe(): It gives the full description of each column of the dataset such as: count, maximum, minimum, quartile range etc.

```
boston_df.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV
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	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	12.653063	22.532806
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	7.141062	9.197104
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	6.950000	17.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	11.360000	21.200000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	37.970000	50.000000

4. Info(): It gives a brief overview about the columns such as data types, null values, count of values, memory used etc.

```
boston_df.info()
```

```
Int64Index: 506 entries, 0 to 505
Data columns (total 13 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    float64
 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    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  LSTAT    506 non-null    float64
 12  MEDV     506 non-null    float64
dtypes: float64(13)
memory usage: 55.3 KB
```

Correlations

Before proceeding with the data cleaning, it is useful to establish a correlation between the response variable (in our case the MEDV) and other predictor variables, as some of them might not have any major impact in determining the price of the house and will not be used in the analysis. There are many ways to discover correlation between the target variable and the rest of the features. Building pair plots, scatter plots, heat maps, and correlation matrices are the most common ones. Below, we will use the corr() function to list the top features based on the Pearson correlation coefficient (measures how closely two sequences of numbers are correlated). Correlation coefficient can only be calculated on the numerical attributes (floats and integers), therefore, only the numeric attributes will be selected.

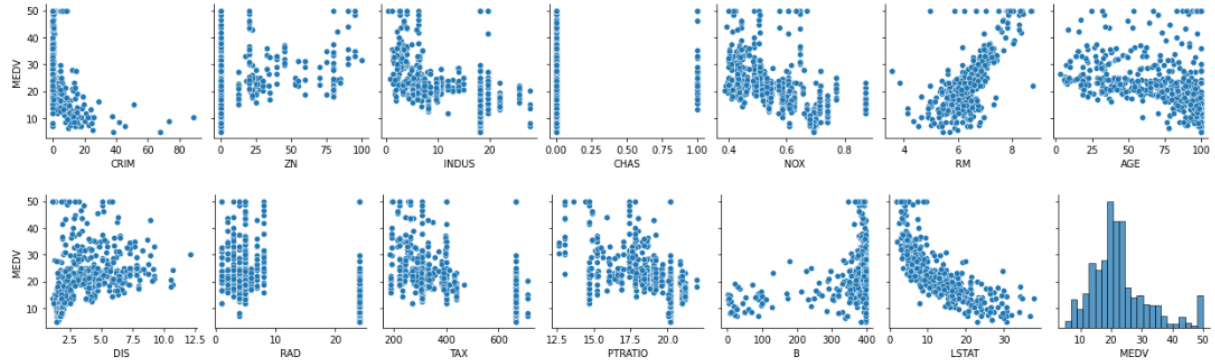
```
boston_num = boston_df.select_dtypes(include = ['float64', 'int64'])
boston_num_corr = boston_num.corr()['MEDV'][:-1]
top_features = boston_num_corr[abs(boston_num_corr) > 0.5].sort_values(ascending=False)
print("There is {} strongly correlated values with MEDV:\n{}".format(len(top_features), top_features))
```

```
There is 3 strongly correlated values with MEDV:
RM         0.695360
PTRATIO   -0.507787
LSTAT     -0.737663
Name: MEDV, dtype: float64
```

Above are 3 features which are correlated with MEDV.

Next, let's generate some par plots to visually inspect the correlation between some of these features and the target variable. We will use seaborns sns.pairplot() function for this analysis. Also, building pair plots is one of the possible ways to spot the outliers that might be present in the data.

```
for i in range(0, len(boston_num.columns),7):
    sns.pairplot(data=boston_num,
                 x_vars=boston_num.columns[i:i+7],
                 y_vars=['MEDV'])
```



Few conclusions drawn by pairplot about correlation of features are: RM, DIS,LSTAT etc are correlated.

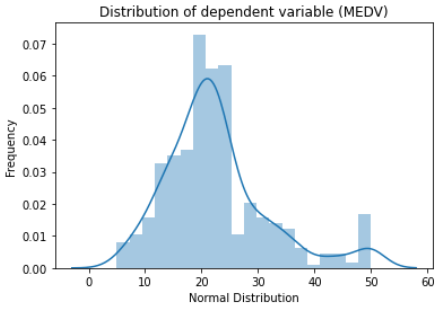
Log Transformation

In this section, we are going to inspect whether our 'MEDV'' data are normally distributed. The assumption of the normal distribution must be met in order to perform any type of regression analysis. There are several ways to check for this assumption, however here, we will use the visual method, by plotting the 'MEDV'' distribution using the distplot() function from the seaborn library.

```
ax = sns.distplot(x = boston_df["MEDV"], bins = 20, kde = True)
ax.set(xlabel='Normal Distribution', ylabel='Frequency', title='Distribution of dependent variable (MEDV)')
```

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be r
emoved in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-l
evel function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
[Text(0.5, 0, 'Normal Distribution'),
 Text(0, 0.5, 'Frequency'),
 Text(0.5, 1.0, 'Distribution of dependent variable (MEDV)')]
```



As the plot shows, our 'MEDV'' deviates from the normal distribution. It has a longer tail to the right, so we call it a positive skew. In statistics *skewness* is a measure of asymmetry of the distribution.

Here, we can simply use the skew() function to calculate our skewness level of the MEDV

```
print("Skewness: %f" % boston_df['MEDV'].skew())
```

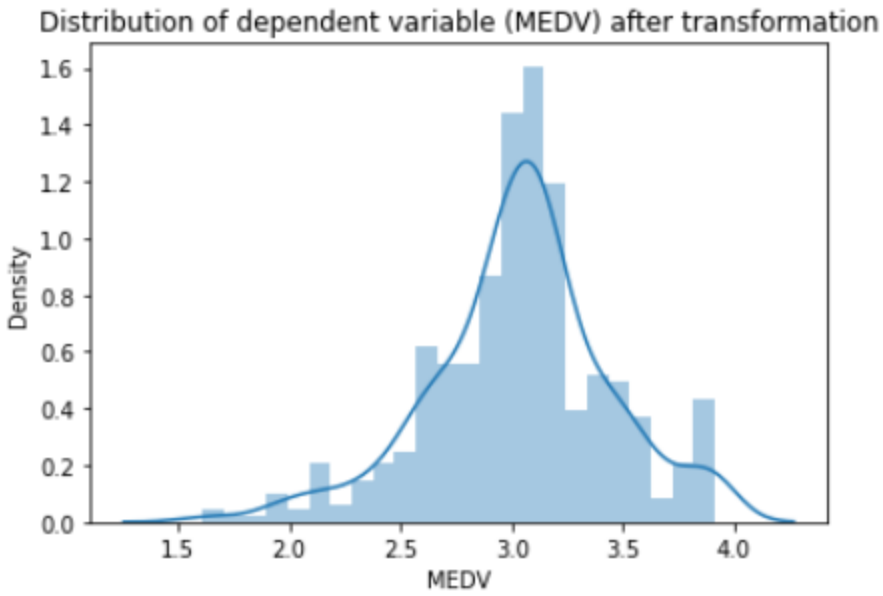
```
Skewness: 1.108098
```

The range of skewness for a fairly symmetrical bell curve distribution is between -0.5 and 0.5; moderate skewness is -0.5 to -1.0 and 0.5 to 1.0; and highly skewed distribution is < -1.0 and > 1.0. In our case, we have 1.1, so it is considered highly skewed data.

Now, we can try to transform our data, so it looks more normally distributed. We can use the np.log() function from the numpy library to perform log transform.

```
log = np.log(boston_df['MEDV'])
medv_transformed  = sns.distplot(log)
medv_transformed.set(title='Distribution of dependent variable (MEDV) after transformation')
print("Skewness: %f" % (log).skew())
```

```
Skewness: -0.330321
```



As we can see, the log method transformed the 'MEDV' distribution into a more symmetrical bell curve and the skewness level now is -0.33, well within the range.

Feature Scaling

One of the most important transformations we need to apply to our data is feature scaling. There are two common ways to get all attributes to have the same scale: min-max scaling and standardization.

Min-max scaling (or normalization) is the simplest: values are shifted and rescaled so they end up ranging from 0 to 1. This is done by subtracting the min value and dividing by the max minus min.

Standardization is different: first it subtracts the mean value (so standardized values always have a zero mean), and then it divides by the standard deviation, so that the resulting distribution has unit variance.

First, we will normalize our data.

```
norm_data = MinMaxScaler().fit_transform(boston_num)
norm_data
```

```
array([[0.00000000e+00, 1.80000000e-01, 6.78152493e-02, ...,
        1.00000000e+00, 8.96799117e-02, 4.22222222e-01],
       [2.35922539e-04, 0.00000000e+00, 2.42302053e-01, ...,
        1.00000000e+00, 2.04470199e-01, 3.68888889e-01],
       [2.35697744e-04, 0.00000000e+00, 2.42302053e-01, ...,
        9.89737254e-01, 6.34657837e-02, 6.60000000e-01],
       ...,
       [6.11892474e-04, 0.00000000e+00, 4.20454545e-01, ...,
        1.00000000e+00, 1.07891832e-01, 4.20000000e-01],
       [1.16072990e-03, 0.00000000e+00, 4.20454545e-01, ...,
        9.91300620e-01, 1.31070640e-01, 3.77777778e-01],
       [4.61841693e-04, 0.00000000e+00, 4.20454545e-01, ...,
        1.00000000e+00, 1.69701987e-01, 1.53333333e-01]])
```

Note the data is now a ndarray we can also standardize our data.

```
scaled_data = StandardScaler().fit_transform(boston_num)
scaled_data
```

```
array([[-0.41978194,  0.28482986, -1.2879095 , ...,  0.44105193,
        -1.0755623 ,  0.15968566],
       [-0.41733926, -0.48772236, -0.59338101, ...,  0.44105193,
        -0.49243937, -0.10152429],
       [-0.41734159, -0.48772236, -0.59338101, ...,  0.39642699,
        -1.2087274 ,  1.32424667],
       ...,
       [-0.41344658, -0.48772236,  0.11573841, ...,  0.44105193,
        -0.98304761,  0.14880191],
       [-0.40776407, -0.48772236,  0.11573841, ...,  0.4032249 ,
        -0.86530163, -0.0579893 ],
       [-0.41500016, -0.48772236,  0.11573841, ...,  0.44105193,
        -0.66905833, -1.15724782]])
```

Handling Outliers

Finding the Outliers

In statistics, an outlier is an observation point that is distant from other observations. An outlier can be due to some mistakes in data collection or recording, or due to natural high variability of data points. How to treat an outlier highly depends on our data or the type of analysis to be performed. Outliers can markedly affect our models and can be a valuable source of information, providing us insights about specific behaviours.

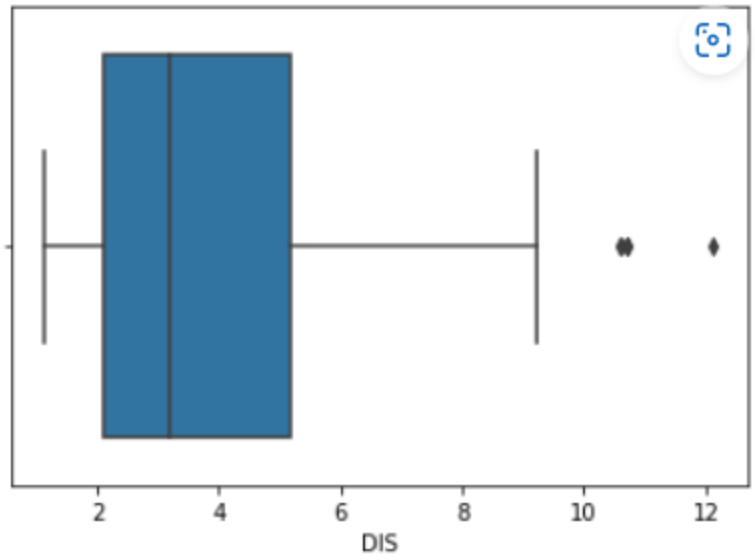
There are many ways to discover outliers in our data. We can do Uni-variate analysis (using one variable analysis) or Multi-variate analysis (using two or more variables). One of the simplest ways to detect an outlier is to inspect the data visually, by making box plots or scatter plots.

Uni-variate Analysis

A box plot is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles.

```
sns.boxplot( x= "DIS", data = boston_df)
```

```
<AxesSubplot:xlabel='DIS'>
```



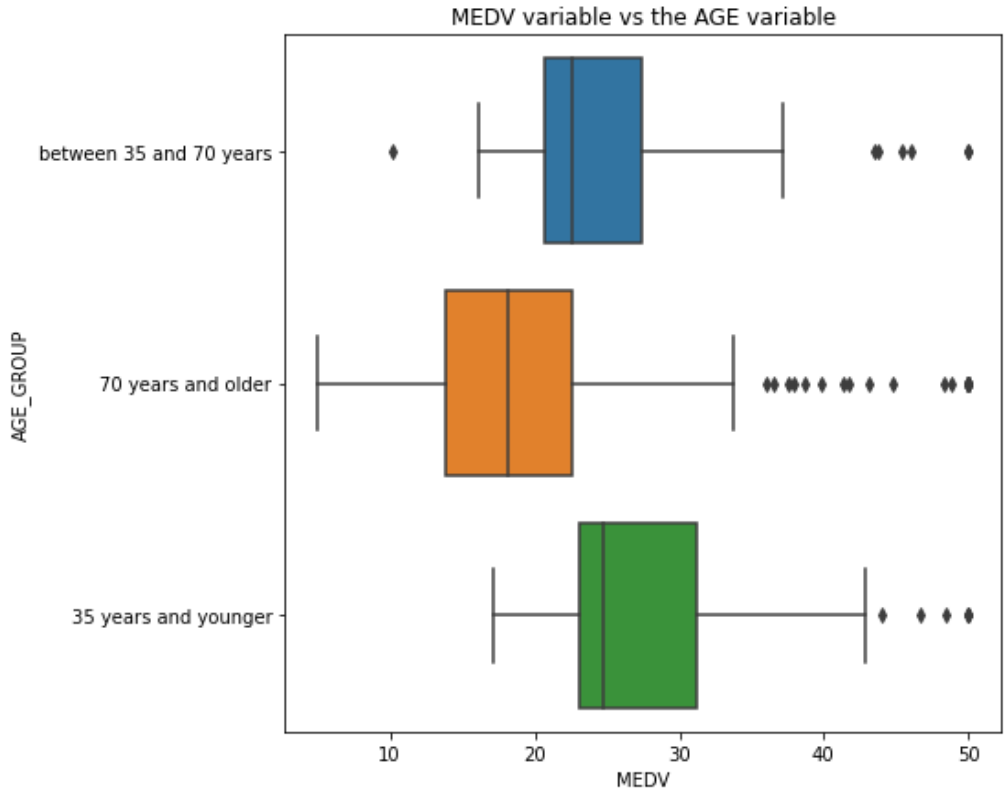
As we can see from these two plots, we have some points that are plotted outside the box plot area and that greatly deviate from the rest of the population. Whether to remove or keep them will greatly depend on the understanding of our data and the type of analysis to be performed. In this case, the points that are outside of our box plots in the 'MEDV'' might be the actual true data points and do not need to be removed.

Bi-variate Analysis

Next, we will look at the bi-variate analysis of the two features, the sale price, 'SalePrice', and the ground living area, 'GrLivArea', and plot the scatter plot of the relationship between these two parameters.

```
# Discretize the age variable into three groups of 35 years and younger, between 35 and 70 years and 70 years and older
boston_df.loc[(boston_df["AGE"] <= 35), "AGE_GROUP"] = "35 years and younger"
boston_df.loc[(boston_df["AGE"] > 35) & (boston_df["AGE"] < 70), "AGE_GROUP"] = "between 35 and 70 years"
boston_df.loc[(boston_df["AGE"] >= 70), "AGE_GROUP"] = "70 years and older"
```

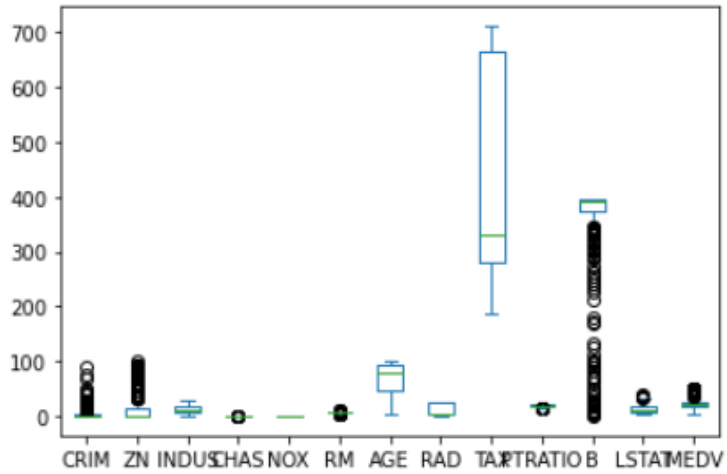
```
#Provide a boxplot for the MEDV variable vs the AGE variable
plt.figure(figsize=(7,7))
plt.title("MEDV variable vs the AGE variable")
sns.boxplot(x= "MEDV", y= "AGE_GROUP", data = boston_df)
```



There are approx 15 - 20 values which are outliers.

Deleting the Outliers

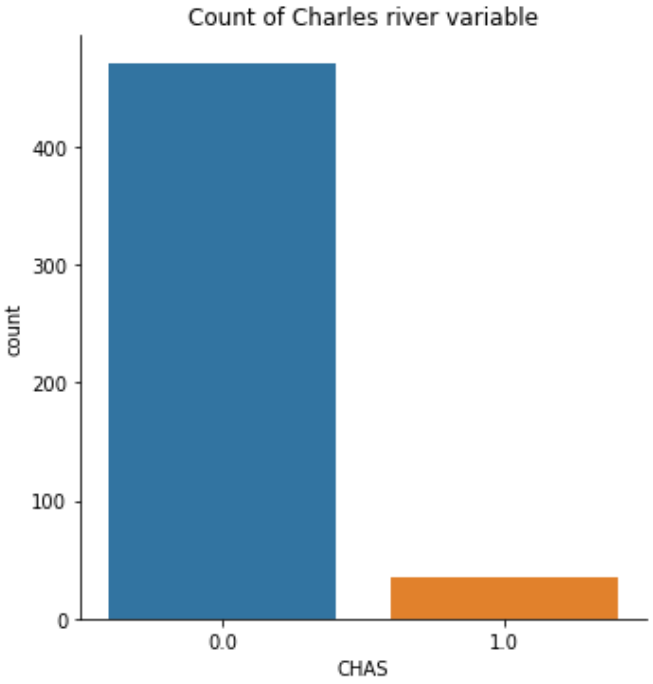
```
outliers_dropped = boston_df.drop(boston_df.index[[351,352,353]])
new_plot = outliers_dropped.plot.box(x='DIS')
```



Data Visualization

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

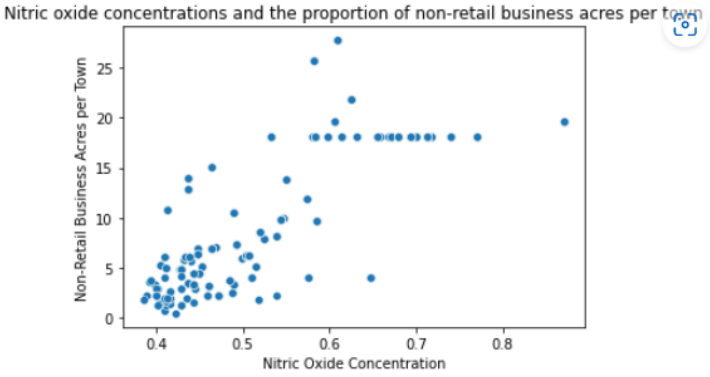
```
ax = sns.catplot(x = "CHAS", kind ="count", data = boston_df)
ax.set(xlabel='CHAS', ylabel='count', title = "Count of Charles river variable")
```



The chart shows that more people live far from the Charles river. Less than 100 people resides near Charles river.T

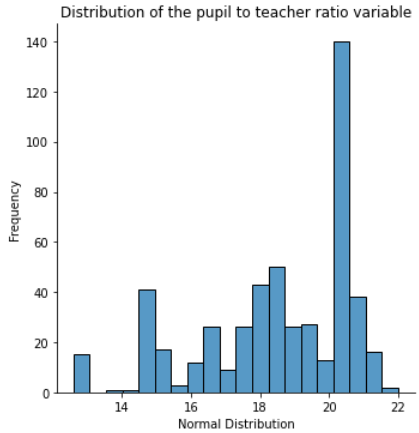
```
ax = sns.scatterplot(x = "NOX", y = "INDUS", data = boston_df)
ax.set(xlabel='Nitric Oxide Concentration',
       ylabel='Non-Retail Business Acres per Town',
       title='Nitric oxide concentrations and the proportion of non-retail business acres per town')
```

```
[Text(0.5, 0, 'Nitric Oxide Concentration'),
 Text(0, 0.5, 'Non-Retail Business Acres per Town'),
 Text(0.5, 1.0, 'Nitric oxide concentrations and the proportion of non-retail business acres per town')]
```



The NOX is positively correlated to INDUS. The graph show the increase in Nitric Oxide concentration as there is increase in non-retail business, but once the non-retail business is greater than 15 the Nitric Oxide concentration become constant.

```
# Create a histogram for the pupil to teacher ratio variable
ax = sns.displot(x = boston_df["PTRATIO"], bins = 20, kde = False)
ax.set(xlabel='Normal Distribution', ylabel='Frequency', title = "Distribution of the pupil to teacher ratio variable")
```



The graph shows the distribution is left skewed distribution.

The graph shows the left skewed distribution.

Hypothesis Testing

Hypothesis 1 :**Is there a significant difference in median value of houses bounded by the Charles river or not? (T-test for independent samples)**

We will be using the t-test for independent samples. For the independent t-test, the following assumptions must be met.

- One independent, categorical variable with two levels or group
- One dependent continuous variable
- Independence of the observations. Each subject should belong to only one group. There is no relationship between the observations in each group.
- The dependent variable must follow a normal distribution
- Assumption of homogeneity of variance

State the Hypothesis

H_0 : μ_1 = μ_2 (There is no significant difference in median values of houses bounded by the Charles river)

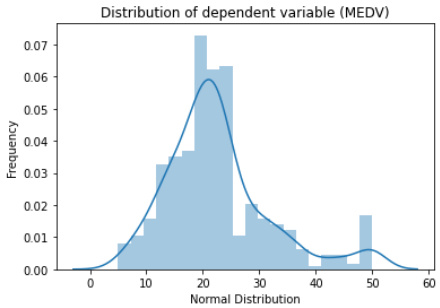
H_1: μ_1 ≠ μ_2 (There is significant difference in median values of houses bounded by the Charles river)

Assumption 1: One independent, categorical variable with two levels or groups, the independent variable is CHAS, but it is a numerical variable. So, we need to convert it to categorical variables.

```
boston_df.loc[(boston_df['CHAS'] == 0), 'CHAS_T'] = 'FAR'
boston_df.loc[(boston_df['CHAS'] == 1), 'CHAS_T'] = 'NEAR'
```

```
# We can plot the dependent variable with a historgram
ax = sns.distplot(x = boston_df["MEDV"], bins = 20, kde = True)
ax.set(xlabel='Normal Distribution', ylabel='Frequency', title='Distribution of dependent variable (MEDV)')
```

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be r
emoved in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-l
evel function for histograms).
  warnings.warn(msg, FutureWarning)
[Text(0.5, 0, 'Normal Distribution'),
 Text(0, 0.5, 'Frequency'),
 Text(0.5, 1.0, 'Distribution of dependent variable (MEDV)')]
```



Levene's test

State the Hypothesis:

H_0: Equality of variance

H_1: At Least one variance is not equal

```
scipy.stats.levene(boston_df[boston_df['CHAS_T'] == 'FAR']['MEDV'],
                   boston_df[boston_df['CHAS_T'] == 'NEAR']['MEDV'], center = "mean")
```

```
LeveneResult(statistic=8.751904896045998, pvalue=0.003238119367639829)
```

Note:

The p value is less than alpha i.e. 0.05>0.03. So, we reject the null hypothesis i.e. Inequality of variance

```
scipy.stats.ttest_ind(boston_df[boston_df['CHAS_T'] == 'FAR']['MEDV'],
                   boston_df[boston_df['CHAS_T'] == 'NEAR']['MEDV'], equal_var = False)
```

```
Ttest_indResult(statistic=-3.113291312794837, pvalue=0.003567170098137517)
```

Result:

Since the value of p is less than alpha which is 0.05. So, we reject the null hypothesis, i.e. There is enough statistical proof that there is statistical difference in median values by Charles river.

Hypothesis 2 : **Is there a difference in Median values of houses (MEDV) for each proportion of owner occupied units built prior to 1940 (AGE)? (ANOVA)**

State the Hypothesis:

H_0: μ_1=μ_2=μ_3 (the three population means are equal) i.e. There is no significant difference in mean of Median values of houses (MEDV) for each proportion of owner occupied units built prior to 1940.

H_1: At least one of the means differ. i.e. There is significant difference in mean of Median values of houses (MEDV) for each proportion of owner occupied units built prior to 1940

```
boston_df.loc[(boston_df["AGE"] <= 35), "AGE_GROUP"] = "35 years and younger"
boston_df.loc[(boston_df["AGE"] > 35) & (boston_df["AGE"] < 70), "AGE_GROUP"] = "between 35 and 70 years"
boston_df.loc[(boston_df["AGE"] >= 70), "AGE_GROUP"] = "70 years and older"
```

Levene's test

State the Hypothesis:

H_0: Equality of variance

H_1: At Least one variance is not equal

```
# Test for equality of variance
scipy.stats.levene(boston_df[boston_df['AGE_GROUP'] == '35 years and younger']['MEDV'],
                   boston_df[boston_df['AGE_GROUP'] == 'between 35 and 70 years']['MEDV'],
                   boston_df[boston_df['AGE_GROUP'] == '70 years and older']['MEDV'],
                   center='mean')
```

```
LeveneResult(statistic=2.780620029374844, pvalue=0.06295337343259205)
```

Note:

The p value is greater than alpha i.e. 0.06>0.05. So, we fail to reject the null hypothesis i.e. Eqaulity of variance

```
thirtyfive_lower   = boston_df[boston_df['AGE_GROUP'] == '35 years and younger']['MEDV']
thirtyfive_seventy = boston_df[boston_df['AGE_GROUP'] == 'between 35 and 70 years']['MEDV']
seventy_older      = boston_df[boston_df['AGE_GROUP'] == '70 years and older']['MEDV']
```

```
# One way ANOVA
f_statistic, p_value = scipy.stats.f_oneway(thirtyfive_lower, thirtyfive_seventy, seventy_older)
print("F_Statistic: {0}, P-Value: {1}".format(f_statistic,p_value))
```

```
F_Statistic: 36.40764999196599, P-Value: 1.7105011022702984e-15
```

Result:

The p-value is less than alpha (0.05) So, we need to reject the null hypothesis. i.e. There is significant difference in mean of Median values of houses (MEDV) for each proportion of owner occupied units built prior to 1940

Hypothesis 3 : **What is the impact of an additional weighted distance to the five Boston employment centers on the median value of owner occupied homes? (Regression analysis)**

State the Hypothesis:

H_0: Weighted distance to the five Boston employment centers on the median value of owner occupied homes are not correlated.

H_1: Weighted distance to the five Boston employment centers on the median value of owner occupied homes are correlated.

```
## X is the input variables (or independent variables)
X = boston_df['DIS']
## y is the target/dependent variable
y = boston_df['MEDV']
## add an intercept (beta_0) to our model
X = sm.add_constant(X)

model = sm.OLS(y, X).fit()
predictions = model.predict(X)

# Print out the statistics
model.summary()
```

OLS Regression Results

Dep. Variable:	MEDV	**R-squared:**	0.062
Model:	OLS	**Adj. R-squared:**	0.061
Method:	Least Squares	**F-statistic:**	33.58
Date:	Fri, 28 Oct 2022	**Prob (F-statistic):**	1.21e-08
Time:	10:49:28	**Log-Likelihood:**	-1823.9
No. Observations:	506	**AIC:**	3652.
Df Residuals:	504	**BIC:**	3660.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	18.3901	0.817	22.499	0.000	16.784	19.996
DIS	1.0916	0.188	5.795	0.000	0.722	1.462

Omnibus:	139.779	**Durbin-Watson:**	0.570
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	305.104
Skew:	1.466	**Prob(JB):**	5.59e-67
Kurtosis:	5.424	**Cond. No.**	9.32

Result:

Since the p-value is less than alpha(0.05), we reject the null hypothesis. i.e. Weighted distance to the five Boston employment centers on the median value of owner occupied homes are correlated.

Feature Selection

Here, we will select only those attributes which best explain the relationship of the independent variables with respect to the target variable, 'MEDV''. There are many methods for feature selection, building the heatmap and calculating the correlation coefficient scores are the most commonly used ones.

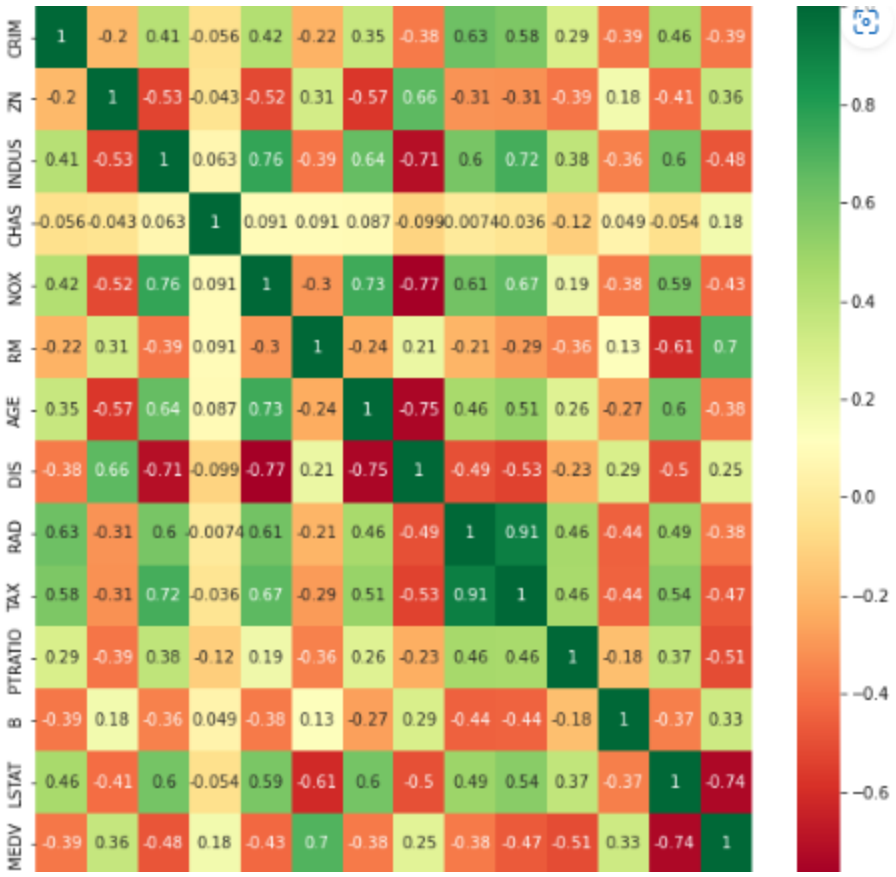
First, we will select only the relevant and newly transformed variables and place them into a 'new_data' data frame.

```
boston_df.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

```
new_boston = boston_df.loc[:,['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV']]


plt.figure(figsize=(18,18))
sns.heatmap(new_boston.corr(),annot=True,cmap='RdYlGn')
plt.show()
```



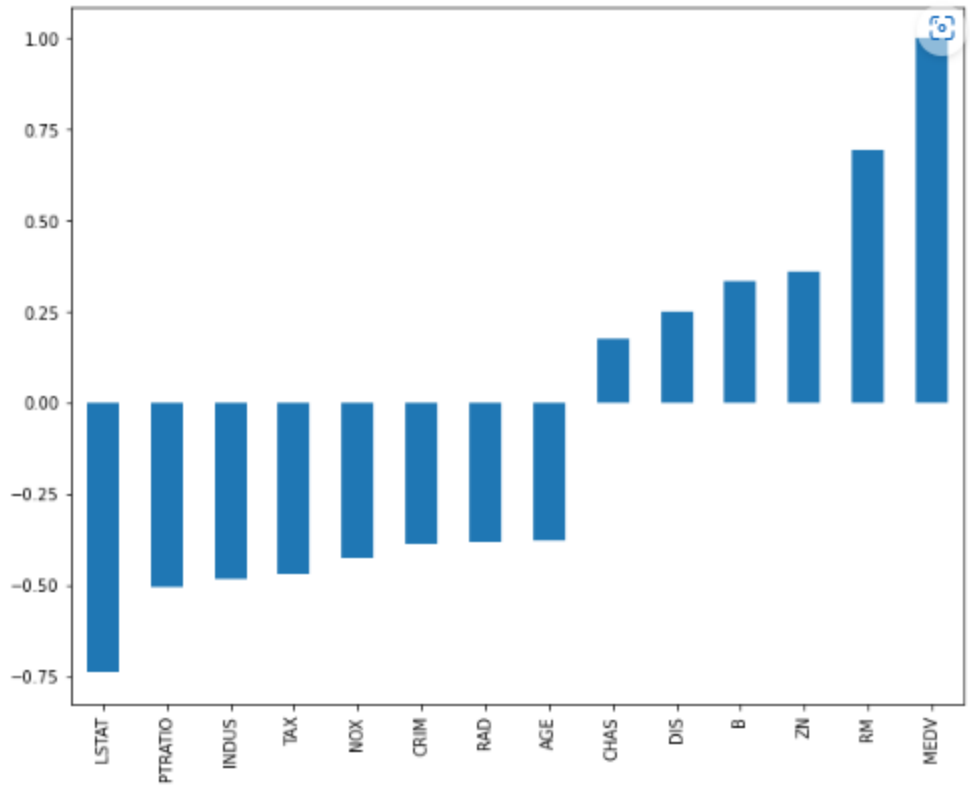
From the heatmap above, extreme green means highly positively correlated features (relationship between two variables in which both variables move in the same direction), extreme red means negatively correlated features (relationship between two variables in which an increase in one variable is associated with a decrease in the other).

```
features = new_boston.corr()['MEDV'].sort_values()
features
```

```
LSTAT     -0.737663
PTRATIO   -0.507787
INDUS     -0.483725
TAX       -0.468536
NOX       -0.427321
CRIM      -0.388305
RAD       -0.381626
AGE       -0.376955
CHAS       0.175260
DIS        0.249929
B          0.333461
ZN         0.360445
RM         0.695360
MEDV       1.000000
Name: MEDV, dtype: float64
```

```
features.plot(kind='bar',figsize=(10,8))
```

```
<AxesSubplot:>
```



From the graph above, we can deduct some of the highly correlated features and select only those ones for any future analysis.

Feature Extraction

PCA - Principal Component Analysis

Dimensionality reduction is part of the feature extraction process that combines the existing features to produce more useful ones. The goal of dimensionality reduction is to simplify the data without losing too much information. Principal Component Analysis (PCA) is one of the most popular dimensionality reduction algorithms. First, it identifies the hyperplane that lies closest to the data, and then it projects the data onto it. In this way, a few multidimensional features are merged into one.

We will assign all the independent variables to x, and the dependent variable, 'MEDV;', to y.

```
x = boston_df.loc[:,['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT']]
y = boston_df.MEDV
scaler = StandardScaler()
x=scaler.fit_transform(x.astype(np.float64))
x
```

```
array([[-0.41978194,  0.28482986, -1.2879095 , ..., -1.45900038,
         0.44105193, -1.0755623 ],
       [-0.41733926, -0.48772236, -0.59338101, ..., -0.30309415,
         0.44105193, -0.49243937],
       [-0.41734159, -0.48772236, -0.59338101, ..., -0.30309415,
         0.39642699, -1.2087274 ],
       ...,
       [-0.41344658, -0.48772236,  0.11573841, ...,  1.17646583,
         0.44105193, -0.98304761],
       [-0.40776407, -0.48772236,  0.11573841, ...,  1.17646583,
         0.4032249 , -0.86530163],
       [-0.41500016, -0.48772236,  0.11573841, ...,  1.17646583,
         0.44105193, -0.66905833]])
```

Instead of arbitrarily choosing the number of dimensions to reduce down to, it is simpler to choose the number of dimensions that add up to a sufficiently large proportion of the variance, let's say 95%.

The following code performs PCA without reducing dimensionality, then computes the minimum number of dimensions required to preserve 95% of the variance.

```
pca = PCA()
pca.fit(x)
cumsum = np.cumsum(pca.explained_variance_ratio_)
d = np.argmax(cumsum >=0.95) + 1
d
```

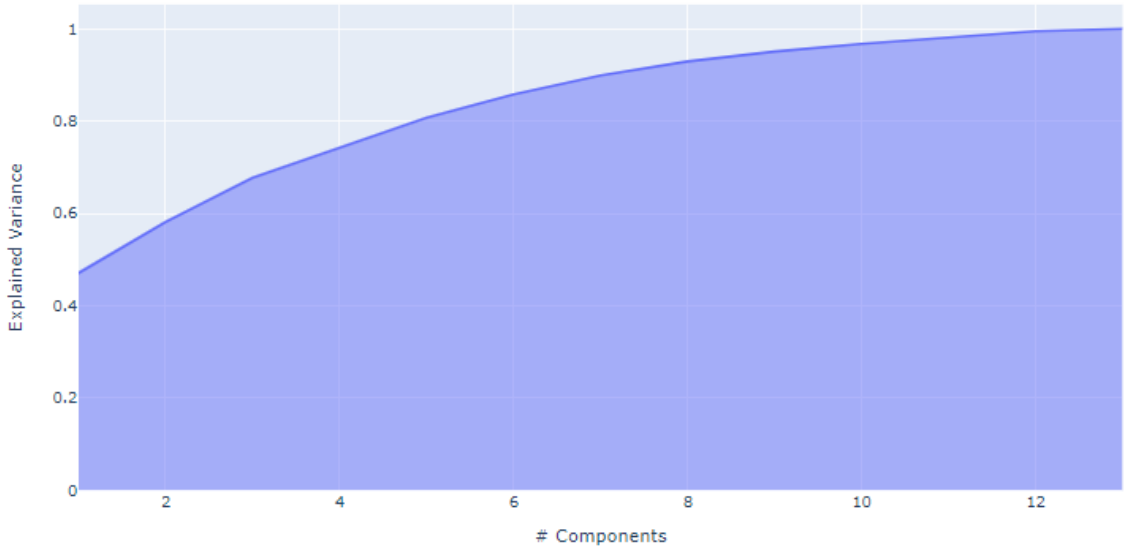
```
9
```

There are 9 components required to meet 95% variance. Therefore, we could set n_components = 9 and run PCA again.

```
pca = PCA(n_components=0.95)
x_reduced = pca.fit_transform(x)
```

There is also a graphical way to determine the number of principal components in your analysis. It is to plot the explained variance as a function of the number of dimensions. There will usually be an elbow in the curve, where the explained variance stops growing fast. That point is usually the optimal point for the number of principal components.

```
px.area(
    x=range(1, cumsum.shape[0] + 1),
    y=cumsum,
    labels={"x": "# Components", "y": "Explained Variance"}
)
```



Conclusion

The following conclusions can be drawn:

1. The target variable MEDV is strongly correlated to only three features which are: RM, PTRATIO and LSTAT.
2. The target variable MEDV was right skewed, we used log transformation to fulfill normal distribution.
3. The data is scaled using MinMaxScaler and standardized using Standard Scaler.
4. There were various outliers present in the dataset, we solved them by dropping the outliers.

5. There is enough statistical proof that there is statistical difference in median values by Charles river.
6. There is significant difference in mean of Median values of houses (MEDV) for each proportion of owner occupied units built prior to 1940
7. Weighted distance to the five Boston employment centers on the median value of owner occupied homes are correlated
8. The features with positive correlation for analysis are: CHAS, B, RM, DIS, ZN.
9. The features with negative correlation are: CRIM, INDUS, NOX, AGE, RAD, TAX, PTRATIO, LSTAT.
10. Only 9 features are selected using PCA.

Further Steps

1. Finding appropriate machine learning algorithms for prediction of target variable MEDV.
2. Building machine learning models for predictions.
3. Making predictions using machine learning models.
4. Enhancing the models for better prediction rate.
5. Deploying the models using Flask or streamlit library.

Recommendations

1. Visualize with more graphs.
2. Using more and latest data.
3. Remove the outliers in a more effective way.