MATH2319 Machine Learning - Assignment 3 - Part 2: My Exam, My Questions (MemQu)

Full Name: Harini Mylanahally Sannaveeranna - s3755660

Conceptual and/ or numerical questions :

Question 1

a. What is Bias and give examples? (1 Mark)

Bias is the simplifying assumptions made by a model to make the target function easier to learn.

Low Bias: Predicting less assumption about Target Function. Examples of low-bias machine learning algorithms include Decision Trees, k-Nearest Neighbors and Support Vector Machines.

High Bias: Predicting more assumption about Target Function. Examples of high-bias machine learning algorithms include Linear Regression, Linear Discriminant Analysis, and Logistic Regression.

b. What is Variance and give examples? (1 Mark)

Variance is the amount that the estimate of the target function will change if different training data was used.

Low Variance: Predicting small changes to the estimate of the target function with changes to the training dataset. Examples of low-variance machine learning algorithms include Linear Regression, Linear Discriminant Analysis, and Logistic Regression.

High Variance: Predicting large changes to the estimate of the target function with changes to the training dataset. Examples of high-variance machine learning algorithms include Decision Trees, k-Nearest Neighbors and Support Vector Machines.

c. What is Underfitting and overfitting? (1 Mark)

underfitting happens when a model incapable to catch the hidden example of the information. These models as a rule have high predisposition and low difference. It happens when we have exceptionally less measure of information to assemble a precise model or when we attempt to construct a direct model with a nonlinear information. Additionally, these sort of models are exceptionally easy to catch the intricate examples in information like Linear and logistic regression.

overfitting happens when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over the noisy dataset. These models have low bias and high variance. These models are extremely mind boggling like Decision trees which are inclined to overfitting.

d. Explain Bias-Variance Tradeoff and why it is used? (2 Marks)

If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand, if our model has a large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data.

This tradeoff in multifaceted nature is the reason there is a tradeoff among inclination and fluctuation. An algorithm can't be more complex and less complex at the same time.

- Increasing the bias will decrease the variance.
- Increasing the variance will decrease the bias.

In [1]:

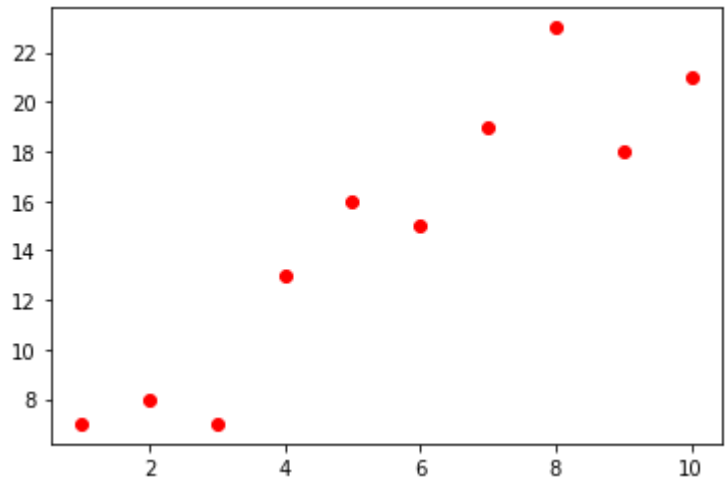
```
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
X = np.arange(1, 11).reshape(10, 1)
y = np.array([7, 8, 7, 13, 16, 15, 19, 23, 18, 21]).reshape(10, 1)
```

In [3]:

```
plt.plot(X, y, 'ro')
plt.show()
```



In [4]:

```
model = LinearRegression()
```

In [5]:

```
model.fit(X, y)
```

Out[5]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, norma
lize=False)
```

In [6]:

```
model.coef_
```

Out[6]:

```
array([[1.77575758]])
```

In [7]:

```
model.intercept_
```

Out[7]:

```
array([4.93333333])
```

we will only focus on coef*, which stores the weight parameter, and intercept*, which stores the bias parameter.

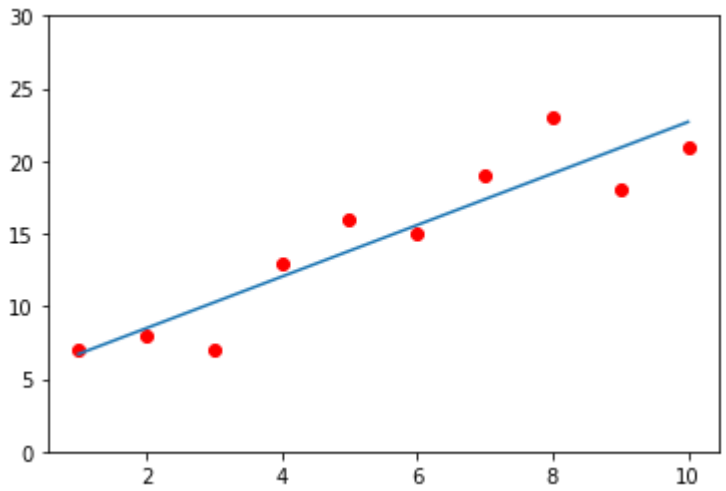
In [8]:

```
#Next, let's compute the prediction vector a, using the obtained weight and bia
s:

a = model.coef_ * X + model.intercept_
```

In [9]:

```
#let's draw all X, y and a on the same plot. Here we got a straight line, which
fit the data better than what we did before (which is easy to understand, since
we only went through 4 iterations).
plt.plot(X, y, 'ro', X, a)
axes = plt.gca()
axes.set_ylim([0, 30])
plt.show()
```



In [10]:

```
print(model.score(X, y))
```

```
0.849880708423668
```

Our model now has the accuracy of 85% over the training data.

It is resulted in a very low accuracy, which is not what we expected, either. It is called UNDERFITTING (or High Bias). if we have more Features, then we will likely have a better fit model.

Polynomial Features:

The easiest way to add more Features, is to computing polynomial features from the provided features. That means if we have X, then we can use X2, X3, etc as additional features. Now we can improve the current Model from this. First, we have to modify our X matrix by adding X2:

In [11]:

```
X = np.c_[X, X**2]
X
```

Out[11]:

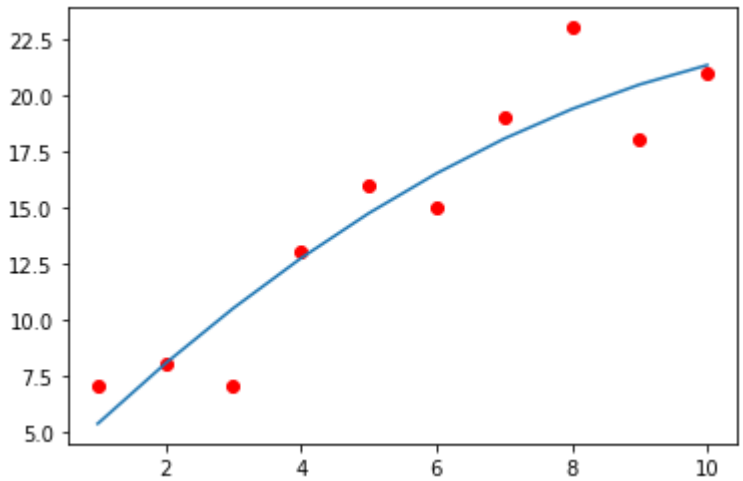
```
array([[  1,   1],
       [  2,   4],
       [  3,   9],
       [  4,  16],
       [  5,  25],
       [  6,  36],
       [  7,  49],
       [  8,  64],
       [  9,  81],
       [ 10, 100]])
```

In [12]:

```
model.fit(X, y)
x = np.arange(1, 11, 0.1)
x = np.c_[x, x**2]
a = np.dot(X, model.coef_.transpose()) + model.intercept_
```

In [13]:

```
plt.plot(X[:, 0], y, 'ro', X[:, 0], a)
plt.show()
```



In [14]:

```
model.score(X, y)
```

Out[14]:

```
0.8721550691495155
```

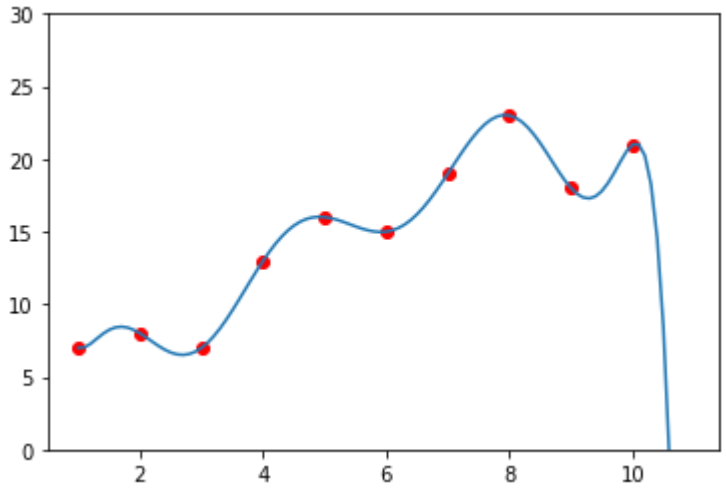
Now we got a new accuracy of 87%. we can improve it a lot more by continuing to add more polynomial features to it.

In [15]:

```
X = np.arange(1, 11)
X = np.c_[X, X**2, X**3, X**4, X**5, X**6, X**7, X**8, X**9]
x = np.arange(1, 11, 0.1)
x = np.c_[x, x**2, x**3, x**4, x**5, x**6, x**7, x**8, x**9]

model.fit(X, y)
a = np.dot(x, model.coef_.transpose()) + model.intercept_

plt.plot(X[:, 0], y, 'ro', x[:, 0], a)
axes = plt.gca()
axes.set_ylim([0, 30])
plt.show()
```



Now we just obtained a new curve which fit our training data perfectly. Let's use the score function again to get an exact number:

In [16]:

```
model.score(X, y)
```

Out[16]:

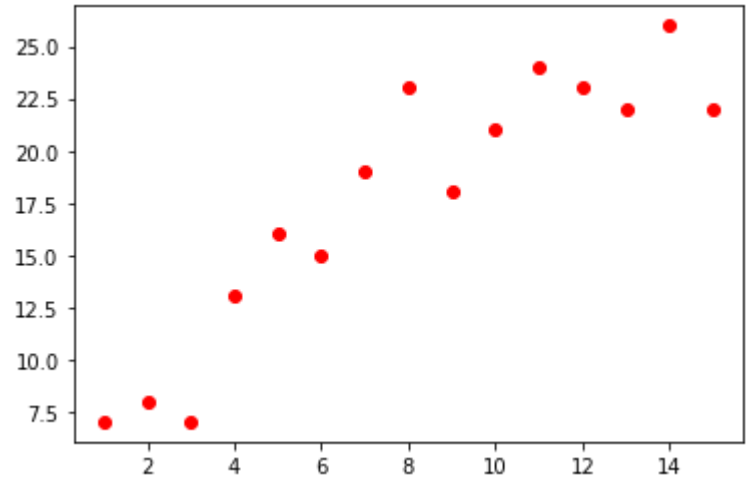
```
0.9999999999769595
```

Now we can see an accuracy of 100%.

In [17]:

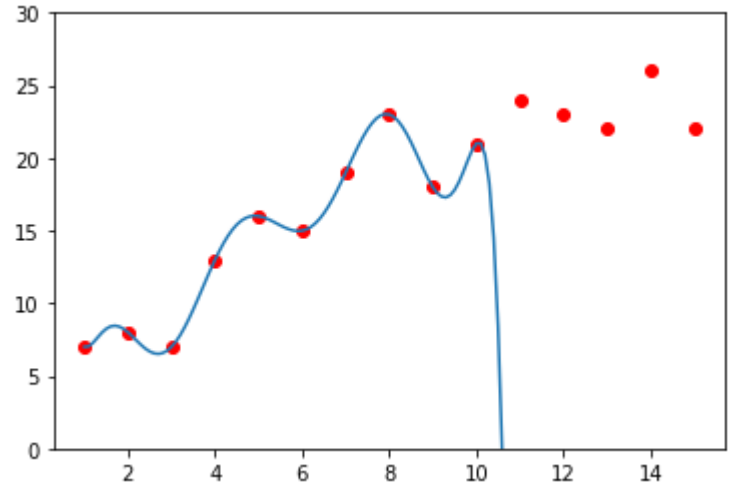
```
X = np.arange(1, 16)
y = np.append(y, [24, 23, 22, 26, 22])

plt.plot(X, y, 'ro')
plt.show()
```



In [18]:

```
plt.plot(X, y, 'ro', x[:, 0], a)
axes = plt.gca()
axes.set_ylim([0, 30])
plt.show()
```



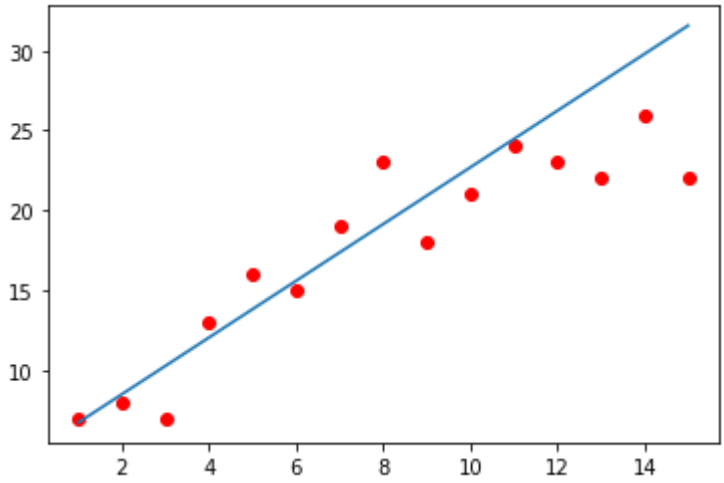
In contrast, we use just one feature, it resulted in a very low accuracy, which is not what we expected, either. We call this problem UNDERFITTING (or High Bias).

Model 1

In [19]:

```
X = np.arange(1, 16).reshape(15, 1)
model.fit(X[:10], y[:10])
model.score(X[10:], y[10:])
-12.653810835629017

a = np.dot(X, model.coef_.transpose()) + model.intercept_
plt.plot(X, y, 'ro', X, a)
plt.show()
```

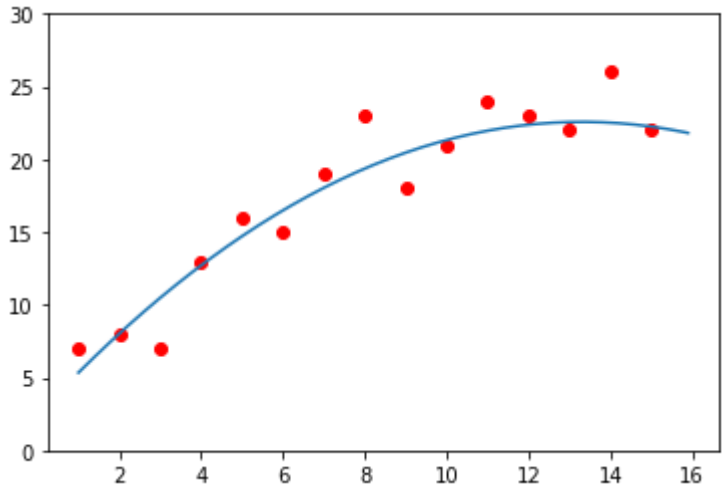


Model 2

In [20]:

```
X = np.arange(1, 16).reshape(15, 1)
X = np.c_[X, X**2]
x = np.arange(1, 16, 0.1)
x = np.c_[x, x**2]
model.fit(X[:10], y[:10])
model.score(X[10:], y[10:])
-0.51814820280729812

a = np.dot(x, model.coef_.transpose()) + model.intercept_
plt.plot(X[:, 0], y, 'ro', x[:, 0], a)
axes = plt.gca()
axes.set_ylim([0, 30])
plt.show()
```

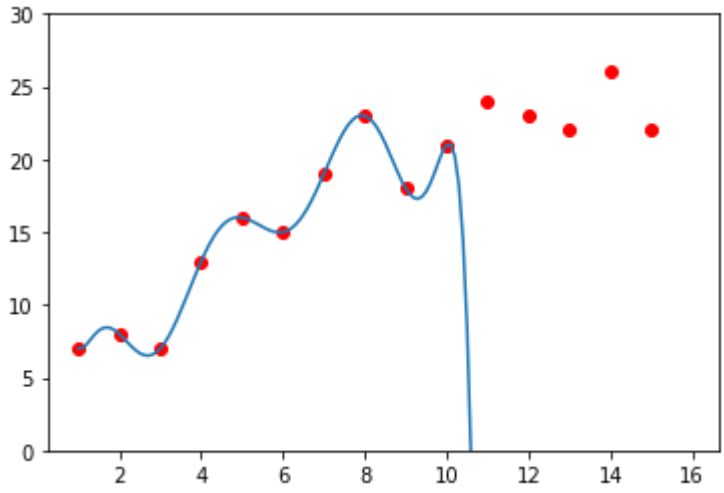


Model 3

In [21]:

```
X = np.arange(1, 16).reshape(15, 1)
X = np.c_[X, X**2, X**3, X**4, X**5, X**6, X**7, X**8, X**9]
x = np.arange(1, 16, 0.1)
x = np.c_[x, x**2, x**3, x**4, x**5, x**6, x**7, x**8, x**9]
model.fit(X[:10], y[:10])
model.score(X[10:], y[10:])
-384608869887696.81

a = np.dot(x, model.coef_.transpose()) + model.intercept_
plt.plot(X[:, 0], y, 'ro', x[:, 0], a)
axes = plt.gca()
axes.set_ylim([0, 30])
plt.show()
```



Our first model is excessively basic, which didn't accommodate our information well. This is a case of Underfitting issue. Interestingly, our third model is excessively confused, which performed very well on preparing information, however neglected to fit the testing information. This is the thing that we called Overfitting issue.

The subsequent model may not fit just as the third model, however the one really realized, which brings about great execution over the testing information. Furthermore, we can some way or another state that, it will likewise anticipate well with whatever other information which it has never observed during preparing.

Question 2

a) What is an Outlier? (1 Mark)

An outlier is a data point in a data set that is distant from all other observations or it lies outside the overall distribution of the dataset. If we don't remove outlier, it can result in wildly wrong estimations.

b) What are the types of Outliers? (1 Mark)

There are two types of outliers:

- Univariate
- Multivariate.

Univariate outliers can be found, when we look at distribution of a single variable. Multi-variate outliers are outliers in an n-dimensional space with distributions in multi-dimensions.

c) What causes Outliers? (1 Mark)

An outlier could exist in a dataset due to Variability in the data and/or experimental measurement error

d) How can we detect an outlier? Explain it. (2 Marks)

we can detect or identify an outlier(s) using scatter plots using Z score using the IQR interquartile range

In [22]:

```
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
from sklearn.datasets import load_boston
import seaborn as sns
```

In [23]:

```
boston = load_boston()
x = boston.data
y = boston.target
columns = boston.feature_names
print(columns)
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRA
TIO'
 'B' 'LSTAT']
```

In [24]:

```
#create the dataframe
boston_df = pd.DataFrame(boston.data)
boston_df_o = boston_df
boston_df.shape
boston_df.columns = columns
boston_df.head()
```

Out[24]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	

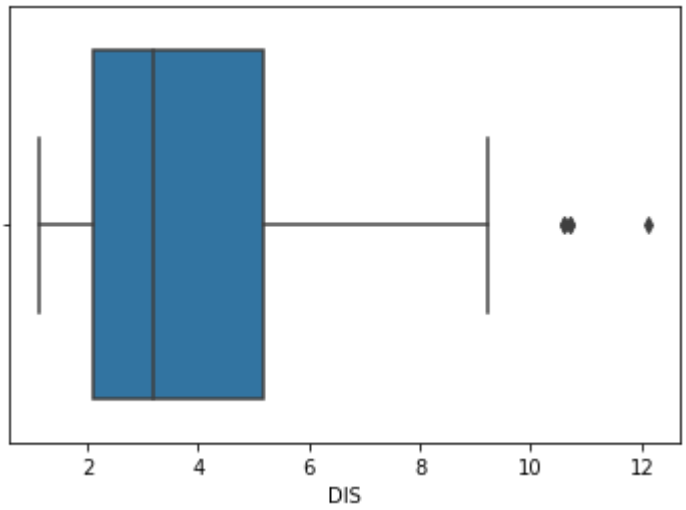
BoxPlot

In [25]:

```
sns.boxplot(x=boston_df['DIS'])
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fe44dc450>
```

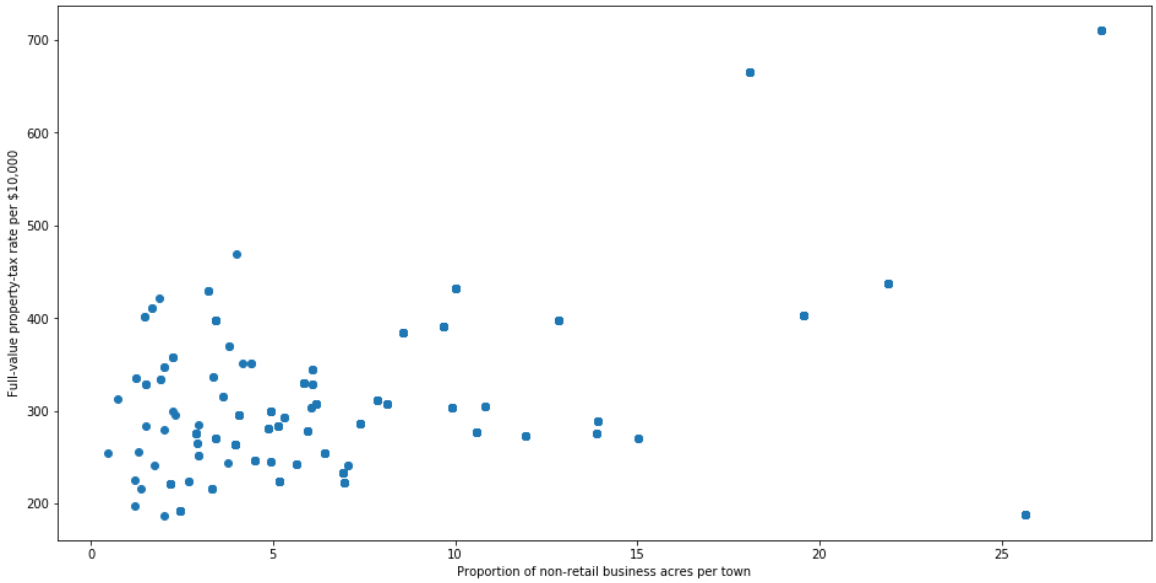


Above plot shows three points between 10 to 12, these are outliers as there are not included in the box of other observation.

Scatter Plot

In [26]:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(16,8))
ax.scatter(boston_df['INDUS'], boston_df['TAX'])
ax.set_xlabel('Proportion of non-retail business acres per town')
ax.set_ylabel('Full-value property-tax rate per $10,000')
plt.show()
```



Looking at the plot above, we can most of data points are lying bottom left side but there are points which are far from the population like top right corner.

Z-Score

In [27]:

```
from scipy import stats
import numpy as np
z = np.abs(stats.zscore(boston_df))
print(z)
```

```
[[0.41978194 0.28482986 1.2879095  ... 1.45900038 0.44105193 1.07556
23 ]
 [0.41733926 0.48772236 0.59338101 ... 0.30309415 0.44105193 0.49243
937]
 [0.41734159 0.48772236 0.59338101 ... 0.30309415 0.39642699 1.20872
74 ]
 ...
 [0.41344658 0.48772236 0.11573841 ... 1.17646583 0.44105193 0.98304
761]
 [0.40776407 0.48772236 0.11573841 ... 1.17646583 0.4032249  0.86530
163]
 [0.41500016 0.48772236 0.11573841 ... 1.17646583 0.44105193 0.66905
833]]
```

Looking the code and the output above, it is difficult to say which data point is an outlier. Let's define a threshold to identify an outlier.

In [28]:

```
threshold = 3
print(np.where(z > 3))
```

```
(array([ 55,  56,  57, 102, 141, 142, 152, 154, 155, 160, 162, 163,
199,
       200, 201, 202, 203, 204, 208, 209, 210, 211, 212, 216, 218, 2
19,
       220, 221, 222, 225, 234, 236, 256, 257, 262, 269, 273, 274, 2
76,
       277, 282, 283, 283, 284, 347, 351, 352, 353, 353, 354, 355, 3
56,
       357, 358, 363, 364, 364, 365, 367, 369, 370, 372, 373, 374, 3
74,
       380, 398, 404, 405, 406, 410, 410, 411, 412, 412, 414, 414, 4
15,
       416, 418, 418, 419, 423, 424, 425, 426, 427, 427, 429, 431, 4
36,
       437, 438, 445, 450, 454, 455, 456, 457, 466]), array([ 1,  1,
1, 11, 12,  3,  3,  3,  3,  3,  3,  3,  1,  1,  1,  1,  1,
        1,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  5,  3,  3,
1,  5,
        5,  3,  3,  3,  3,  3,  3,  1,  3,  1,  1,  7,  7,  1,  7,
7,  7,
        3,  3,  3,  3,  3,  5,  5,  5,  3,  3,  3, 12,  5, 12,  0,
0,  0,
        0,  5,  0, 11, 11, 11, 12,  0, 12, 11, 11,  0, 11, 11, 11, 1
1, 11,
       11,  0, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11]))
```

The first array contains the list of row numbers and second array respective column numbers, which mean z[55][1] have a Z-score higher than 3.

IQR score

In [29]:

```
boston_df_o1 = boston_df
```

In [30]:

```
Q1 = boston_df_o1.quantile(0.25)
Q3 = boston_df_o1.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
CRIM         3.595038
ZN          12.500000
INDUS       12.910000
CHAS         0.000000
NOX          0.175000
RM           0.738000
AGE         49.050000
DIS          3.088250
RAD         20.000000
TAX        387.000000
PTRATIO      2.800000
B           20.847500
LSTAT       10.005000
dtype: float64
```

Question 3

a. What is Feature Selection? (1 Mark)

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in.

b. What is Normalization and Explain it (with an example) (3 Marks)

Normalization is a good technique to use when you do not know the distribution of your data or when you know the distribution is not Gaussian (a bell curve).

In [31]:

```
import warnings

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [32]:

```
# (loan.csv) dataset link: https://www.kaggle.com/wendykan/lending-club-loan-data/data?select=loan.csv
cols = ['loan_amnt', 'int_rate', 'installment']
data = pd.read_csv('/Users/harini/Downloads/loan.csv', nrows = 30000, usecols = cols)
data.head()
```

Out[32]:

	loan_amnt	int_rate	installment
0	2500	13.56	84.92
1	30000	18.94	777.23
2	5000	17.97	180.69
3	4000	18.94	146.51
4	30000	16.14	731.78

In this approach, the data is scaled to a fixed range—usually 0 to 1.

A Min-Max scaling is typically done via the following equation:

In [33]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
```

In [34]:

```
print('means (Loan Amount, Int rate and Installment): ', data_scaled.mean(axis=0))
print('std (Loan Amount, Int rate and Installment): ', data_scaled.std(axis=0))
```

```
means (Loan Amount, Int rate and Installment):  [0.38312667 0.279737
96 0.27125369]
std (Loan Amount, Int rate and Installment):  [0.26301581 0.19646036
0.18102978]
```

After MinMaxScaling, the distributions are not centered at zero and the standard deviation is not 1.

In [35]:

```
print('Min (Loan Amount, Int rate and Installment): ', data_scaled.min(axis=0))
print('Max (Loan Amount, Int rate and Installment): ', data_scaled.max(axis=0))
```

```
Min (Loan Amount, Int rate and Installment):  [0. 0. 0.]
Max (Loan Amount, Int rate and Installment):  [1. 1. 1.]
```

But the minimum and maximum values are standardized across variables, different from what occurs with standardization

c.When Should You Use Normalization ? (1 Mark)

Normalization is useful when your data has varying scales and the algorithm you are using does not make assumptions about the distribution of your data, such as k-nearest neighbors and artificial neural networks.

Question 4

a. What is Standardization and explain it with an example? (3 Mark)

“Standardizing” a vector most often means subtracting a measure of location and dividing by a measure of scale. For example, if the vector contains random values with a Gaussian distribution, you might subtract the mean and divide by the standard deviation, thereby obtaining a “standard normal” random variable with mean 0 and standard deviation 1.

In [36]:

```
#using the same dataset (loan.csv) for standardization (as we used this same dat
aset for normalization)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)
```

In [37]:

```
print(data_scaled.mean(axis=0))
print(data_scaled.std(axis=0))
```

```
[-8.71599089e-17  1.77635684e-18  3.03164901e-17]
[1. 1. 1.]
```

As expected, the mean of each variable is now around zero and the standard deviation is set to 1. Thus, all the variable values lie within the same range.

In [38]:

```
print('Min values (Loan Amount, Int rate and Installment): ', data_scaled.min(ax
is=0))
print('Max values (Loan Amount, Int rate and Installment): ', data_scaled.max(ax
is=0))
```

```
Min values (Loan Amount, Int rate and Installment):  [-1.4566678  -
1.42389012 -1.49839262]
Max values (Loan Amount, Int rate and Installment):  [2.34538496 3.6
6619529 4.02556036]
```

However, the minimum and maximum values vary according to how spread out the variable was, to begin with, and is highly influenced by the presence of outliers.

c. When Should You Use Standardization ? (1 Mark)

Standardization is useful when your data has varying scales and the algorithm you are using does make assumptions about your data having a Gaussian distribution, such as linear regression, logistic regression, and linear discriminant analysis.

d. Why Should You Standardize Variables ? (1 Mark)

Standardizing the features around the center and 0 with a standard deviation of 1 is important when we compare measurements that have different units. Variables that are measured at different scales do not contribute equally to the analysis and might end up creating a bais.

MCQ's :

Question 5 (3 Marks)

A feature B1 can take certain value: 1, 2, 3 and 4 and represents levels of educational experience.

Which of the following statement is true in following case?

a) Feature B1 is an example of nominal variable. b) Feature B1 is an example of ordinal variable. c) It doesn't belong to any of the above category. d) Both of these

Answer: b)

Ordinal variables are the variables, that has some order associated with it

For example, level 1 should be consider as high level of education than level 2.

Question 6 (3 Marks)

what will happen(what is the result), if we add a non-important feature to a linear regression model.

- Increase in R-square
- Decrease in R-square

A) Only 1 is correct

B) Only 2 is correct

C) Either 1 or 2

D) None of these

answer: A)

The R-Squared always increase, after adding the feature in feature space either that added feature is important or unimportant feature

Question 7 (3 Marks)

Suppose, you are working with categorical features and you have not looked at the distribution of the categorical variable in the test data.

Apply one hot encoding (OHE) on the categorical features. What challenges you may confront in the event that you have applied OHE on a straight out factor of train dataset?

A) Train and Test always have same distribution.

B) Frequency distribution of categories is different in train as compared to the test dataset.

C) All categories of categorical variable are not present in the test dataset.

D) Both A and B

Answer : D)

Both B and C, since the OHE will neglect to encode the classifications which is available in test however not in train so it could be one of the fundamental difficulties while applying OHE. The test given in choice B is additionally evident you have to increasingly cautious while applying OHE if recurrence circulation doesn't same in train and test.

Question 8 (3 Marks)

For which of the following hyperparameters, higher value is better for decision tree algorithm?

- Samples for leaf
- Depth of tree
- Number of samples used for split

A) 1 and 3

B) 1, 2 and 3

C) 1 and 2

D) 2 and 3

E) None of these

Answer : E)

Answer is A, B and C options.

it is not essential that on the off chance that you increment the estimation of boundary the presentation may increment. For instance, in the event that we have an extremely high estimation of profundity of tree, the subsequent tree may overfit the information, and would not sum up well. Then again, in the event that we have a low worth, the tree may underfit the information. In this way, we can not state without a doubt that "higher is better".

Question 9 (3 Marks)

Pearson correlation between two variables is zero but, still their values can still be related to each other.

A) TRUE

B) FALSE

Answer A)

they are not only associated, but one is a function of the other and Pearson correlation between them is 0.

Question 10 (3 Marks)

PCA is most useful for non linear type models.

a) True

b) False

Answer: b

PCA is most useful for linear type models.