Data Science IFT6758 - Assignment 2

October 8, 2019

0.1 Data Wrangling and Visualization

Question 4. Selection bias. Download the IMDB and Rotten Tomatoes data.

a. Is there any missingness in this dataset? Which columns have the most missingness?

b. Make a hexbin scatterplot of rotten tomatoes vs. imdb scores against one another. Comment on the relationship between these variables.

c. Filter the movies to those made before 1970, and remake the scatterplot. What do you notice?

d. Propose explanations for what you see in part (c).

[1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
df_4 = pd.read_csv("https://gist.githubusercontent.com/krisrs1128/
↪9276aa2a5d9fa7ab0786bbc75f93d77a/raw/
↪1aa5220f9e140515d04601a6c114fc43cecf1e21/movies.csv")
```

a) Is there any missingness in this dataset? Which columns have the most missingness?

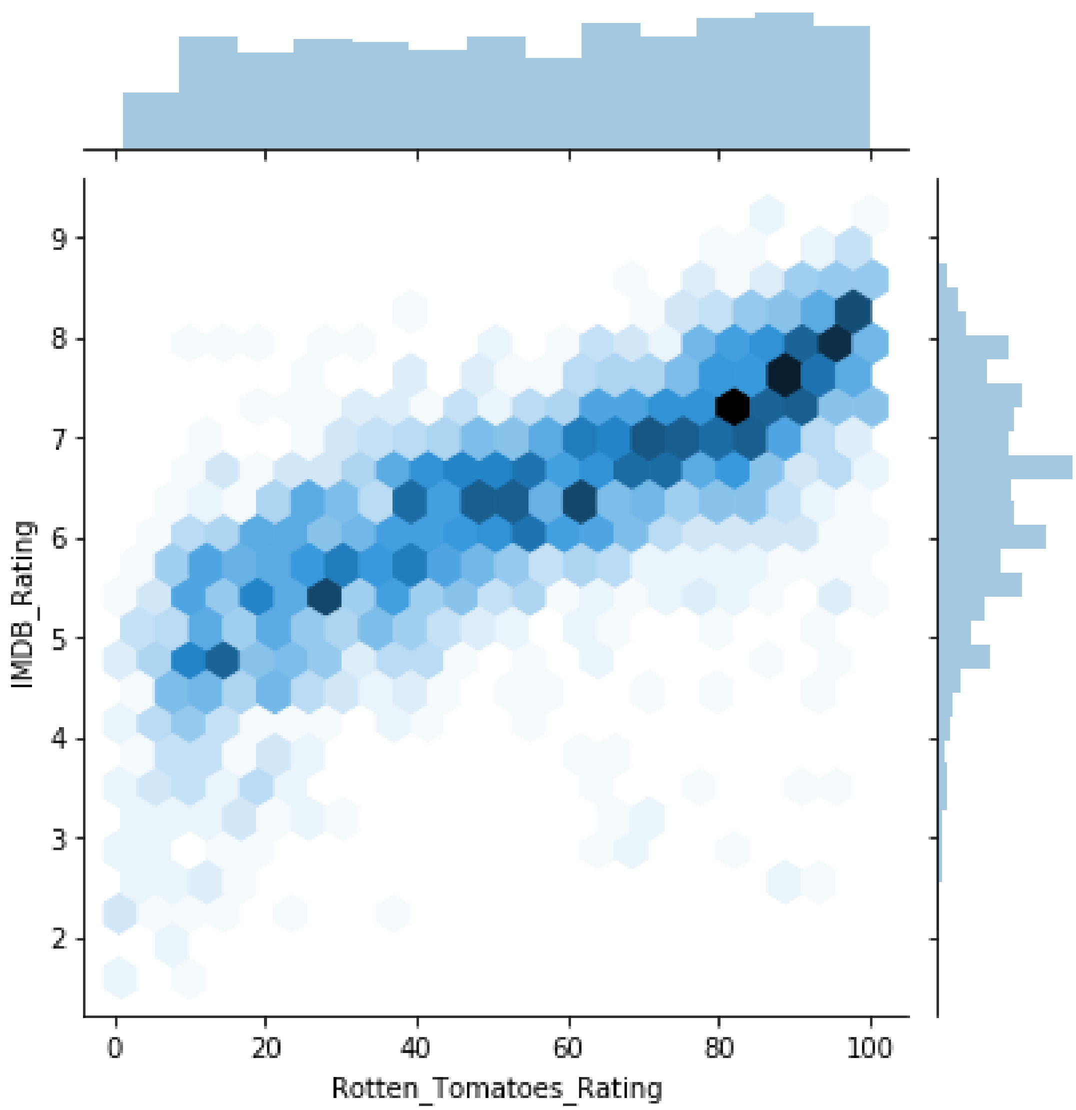
[2]:
```python
df_4.isna().sum()
```

[2]:
```
Title                     1
US_Gross                  7
Worldwide_Gross           7
US_DVD_Sales           2637
Production_Budget         1
Release_Date              0
MPAA_Rating             605
Running_Time_min       1992
Distributor             232
Source                  365
Major_Genre             275
Creative_Type           446
Director               1331
Rotten_Tomatoes_Rating  880
IMDB_Rating             213
IMDB_Votes              213
dtype: int64
```

Answer 4a: From the data above, it seems like *US_DVD_Sales* has the highest number of missing data points, followed by *Running_Time_min*

b). Make a hexbin scatterplot of rotten tomatoes vs. imdb scores against one another. Comment on the relationship between these variables.

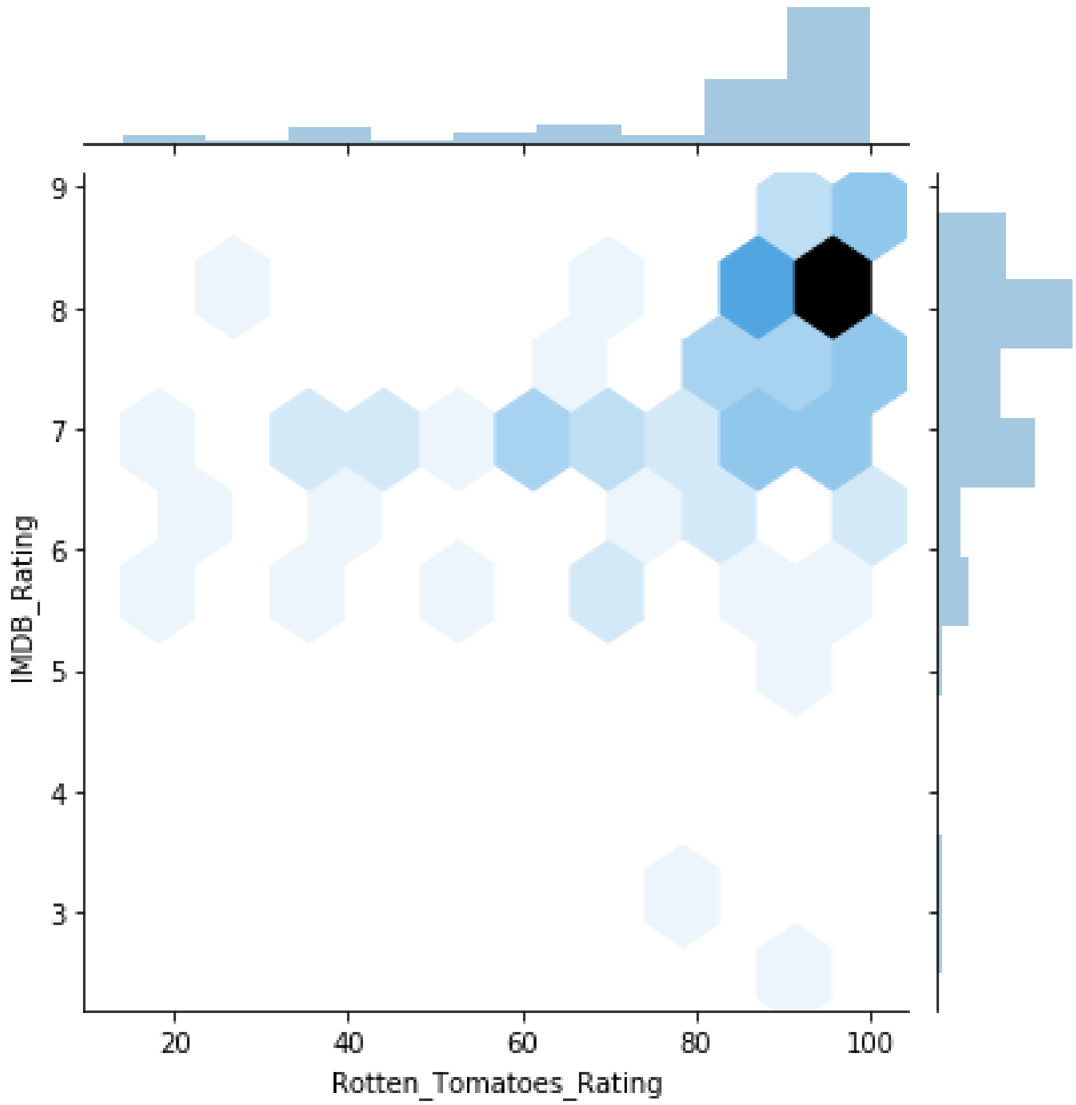
[3]:
```
import seaborn as sns
sns.jointplot('Rotten_Tomatoes_Rating', 'IMDB_Rating', df_4, kind="hex");
```



Answer 4b: Both variables seems to have a linear relationship after IMDB Scores of 5+. People tend to rate movies similarily on both the platforms. Moreover, the number of movies with high ratings are present in good numbers compared to the number of movies with low scores.

c). Filter the movies to those made before 1970, and remake the scatterplot. What do you notice?

[4]:
```
df_4["Release_Date_conv"] = df_4['Release_Date'].apply(lambda x : int(x.split("␣
↪")[2]))
sns.jointplot('Rotten_Tomatoes_Rating', 'IMDB_Rating',␣
↪df_4[df_4["Release_Date_conv"]<1970], kind="hex");
```



Answer 4c: The plot shows that there arent many films rated below 7-8 in Imdb Rating and below 80 in Rotten Tomatoes. Most of the films before 1970 are rated very high on both the platforms

d). Propose explanations for what you see in part (c).

Answer 4d: I believe since both the companies IMDB and Rotten Tomatoes started their operations in 90s and got famous in the 2000. People would have not been able to rate all the movies from 1970s, but only the famous ones. Since everybody would like to view good rated films from the past and hence the high rating on IMDB & Rotten Tomatoes.

Question 9. You have the following table in a variable called "test_scores",

Student	Physics	Chemistry	English	Math
John	78	79	56	95
Alice	58	72	91	81
Rachel	22	61	88	64
Tom	78	89	56	83

(a) Explain the format of the table after running this code,

```
test_scores_clean = pd.melt(
    test_scores,
    id_vars=['Student'],
    var_name=['Subject'],
    value_name='Score'
)
```

(b) Explain what the following code does.

```
test_scores.assign(
    overall=lambda df: df.drop("Student", axis=1).sum(axis=1),
    quant=lambda df: df["Math"] + df["Physics"],
)
```

[10]:
```
df9 = pd.DataFrame({"Student":["John", "Alice", "Rachel", "Tom"],
                    "Physics":[78, 58, 22, 78],
                    "Chemistry":[79, 72, 61, 89],
                    "English":[56, 91, 88, 56],
                    "Math":[95, 81, 64, 83]})
df9_melt = pd.melt(df9,
                   id_vars = ['Student'],
                   var_name = ['Subject'],
                   value_name = 'Score'
                  )
df9_melt
```

[10]:
```
   Student    Subject  Score
0     John    Physics     78
1    Alice    Physics     58
2   Rachel    Physics     22
3      Tom    Physics     78
4     John  Chemistry     79
5    Alice  Chemistry     72
6   Rachel  Chemistry     61
7      Tom  Chemistry     89
8     John    English     56
9    Alice    English     91
10  Rachel    English     88
11     Tom    English     56
12    John       Math     95
```

```
13   Alice        Math     81
14  Rachel        Math     64
15     Tom        Math     83
```

Answer 9a: Melt is the opposite operation of Pivot and it converts the data from a wide format to a long format. This code block orders the table based on marks from the same subject i.e. you can see the marks obtained for the subject Physics by all the students and then followed by other subjects.

[6]:
```
df9.assign(
    overall=lambda df: df.drop("Student", axis=1).sum(axis=1),
    quant=lambda df: df["Math"] + df["Physics"],
)
```

[6]:

	Student	Physics	Chemistry	English	Math	overall	quant
0	John	78	79	56	95	308	173
1	Alice	58	72	91	81	302	139
2	Rachel	22	61	88	64	235	86
3	Tom	78	89	56	83	306	161

Answer 9b: Two new columns i.e. *overall* and *quant* have been added and their values have been calculated from their respective rows.

Overall: This is calculated by summing all the marks from different subjects of the respective student.

Quant: This is calculated by summing only the marks from Physics and Mathematics.

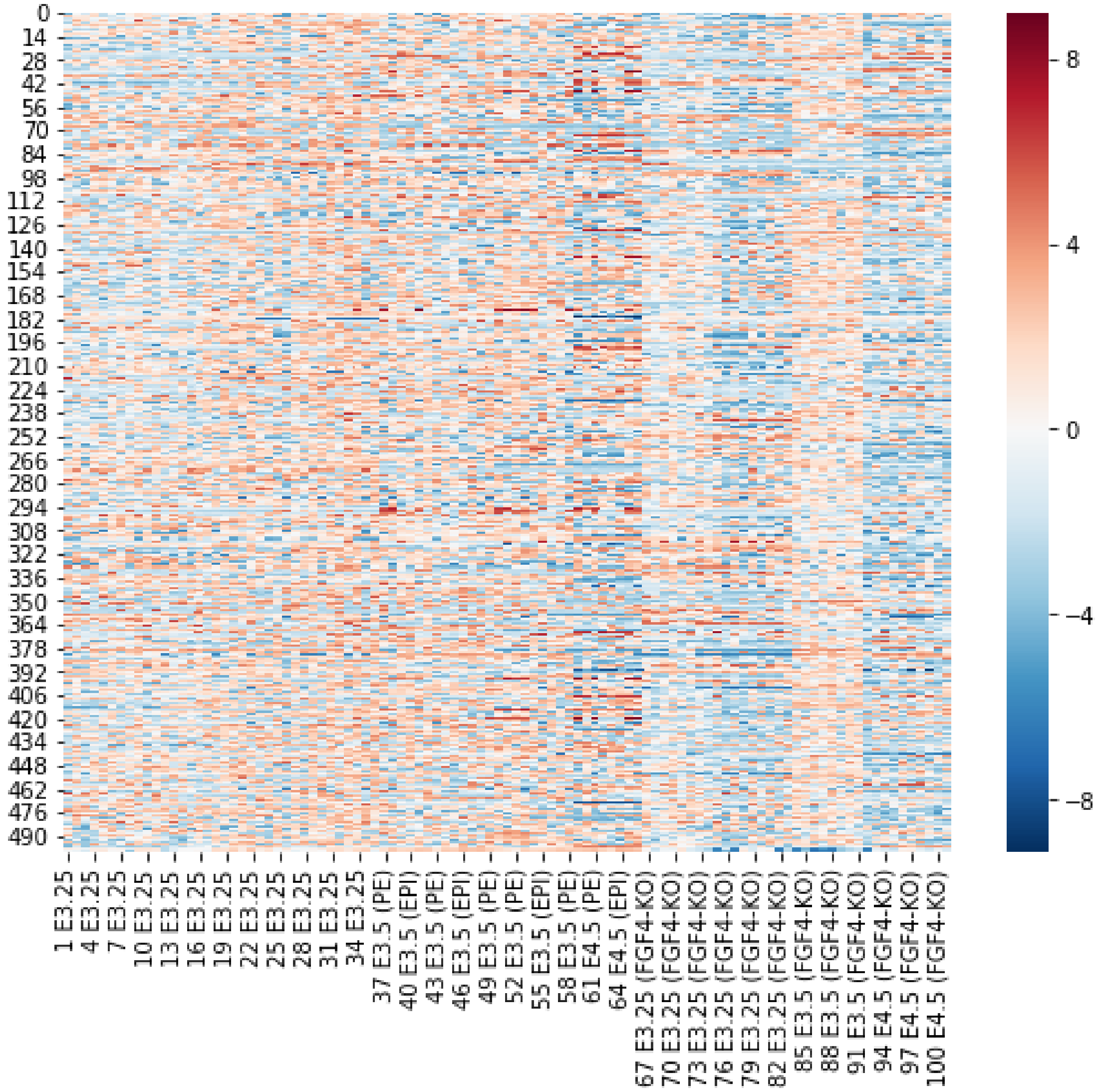
Question 12. Making a heatmap. Heatmaps are a way of plotting continuous values against combinations of categorical variables. We'll use them to analyze a gene expression dataset, collected to study changes in expression after the first symmetry breaking event of the embryo. The rows of the matrix correspond to genes, and the columns are different experimental samples.

a. Make a heatmap of the raw data, using `sns.heatmap`. Make sure to use a diverging color scale, centered around zero.

b. The heatmap is not particularly informative. It's hard to make comparisons across either genes or samples, since there are so many of them. To remedy this, order them using a clustering method (the details are unimportant), as implemented in the `clustermap` function.

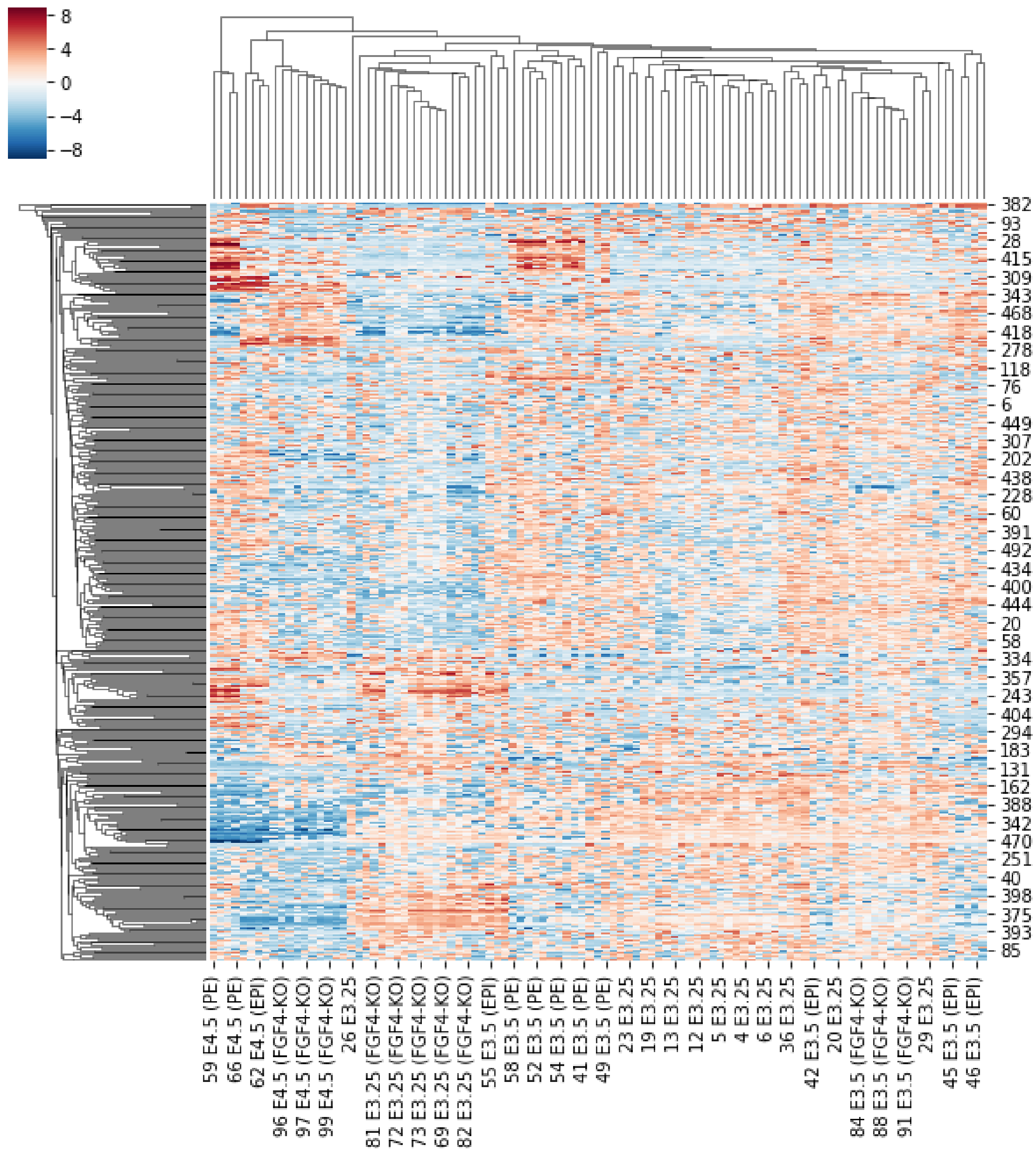
[14]:
```
df_12 = pd.read_csv("https://gist.githubusercontent.com/krisrs1128/
↪b8dc85b659186259715f8efe950ffce6/raw/
↪77669ebdda4d6f8a029c2fdf506f4599277b50f4/hiiragi.csv")

plt.figure(figsize=(9,7))
del df_12['gene']
sns.heatmap(df_12, center = 0, cmap = "RdBu_r");
# Diverging Palatte and centered around 0
```



Answer 12a: Heatmaps provide a clearer visual for large datasets by displaying gradient colors for values in a grid. But since the data is so huge, we can cluster them to dig further in the next part.

[15]:
```
sns.clustermap(df_12, center = 0, cmap = "RdBu_r");
```



Answer 12b: The clustermap() method uses a hierarchical clusters to order data by similarity. This reorganizes the data for the rows and columns and displays similar content next to one another for even more depth of understanding the data.

As we can see the data is no more ordered in the same way we have before, and is ordered based on the similarity.

1 Function Fitting

1.1 Linear Regression

Question 1. [ISLR 3.7.5] Consider the fitted values that result from performing linear regression without an intercept. In this setting, the i^{th} fitted value takes the form, $\hat{y}_i = x_i\hat{\beta}$ where

$$\hat{\beta} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$$

Show that we can write

$$\hat{y}_i = \sum_{i'} a_{i'} y_{i'}.$$

What is $a_{i'}$? Note: We interpret this result by saying that the fitted values from linear regression are **linear combinations** of the response values.

Answer 1: To avoid confusion, changing the notation in the numerator and denominator (to illustrate the different summations. The numerator needs to be divided by the total summation of squares)

$$\hat{\beta} = \frac{\sum_{i'=1}^{n} x_{i'} y_{i'}}{\sum_{k=1}^{n} x_k^2}$$

Since $\hat{y}_i = x_i\hat{\beta}$

This x_i is a constant in the equation and has nothing to do with the existing notations.

$$\hat{y}_i = x_i \frac{\sum_{i'=1}^{n} x_{i'} y_{i'}}{\sum_{k=1}^{n} x_k^2}$$

$$\hat{y}_i = \sum_{i'=1}^{n} \frac{x_{i'} y_{i'}}{\sum_{k=1}^{n} x_k^2} x_i$$

Since x_i doesnt depend on i' and i_k, we can move it in or out of summation like a constant.

$$\hat{y}_i = \sum_{i'=1}^{n} \frac{x_{i'} x_i}{\sum_{k=1}^{n} x_k^2} y_{i'}$$

Therefore,

$$\hat{y}_i = \sum_{i'=1}^{n} a_{i'} y_{i'}$$

where

$$a_{i'} = \frac{x_{i'} x_i}{\sum_{k=1}^{n} x_k^2}$$

1.2 Extending Linear Regression

Question 8. [ISLR 7.9.3] Suppose we fit a curve with basis functions $b_1(x) = x, b_2(x) = (x-1)^2 \mathbb{1}\{x \geq 1\}$. (Note that $\mathbb{1}\{x \geq 1\}$ equals 1 for $x \geq 1$ and 0 otherwise.) We fit the linear regression model,

$$y = \beta_0 + \beta_1 b_1(x) + \beta_2 b_2(x) + \epsilon$$

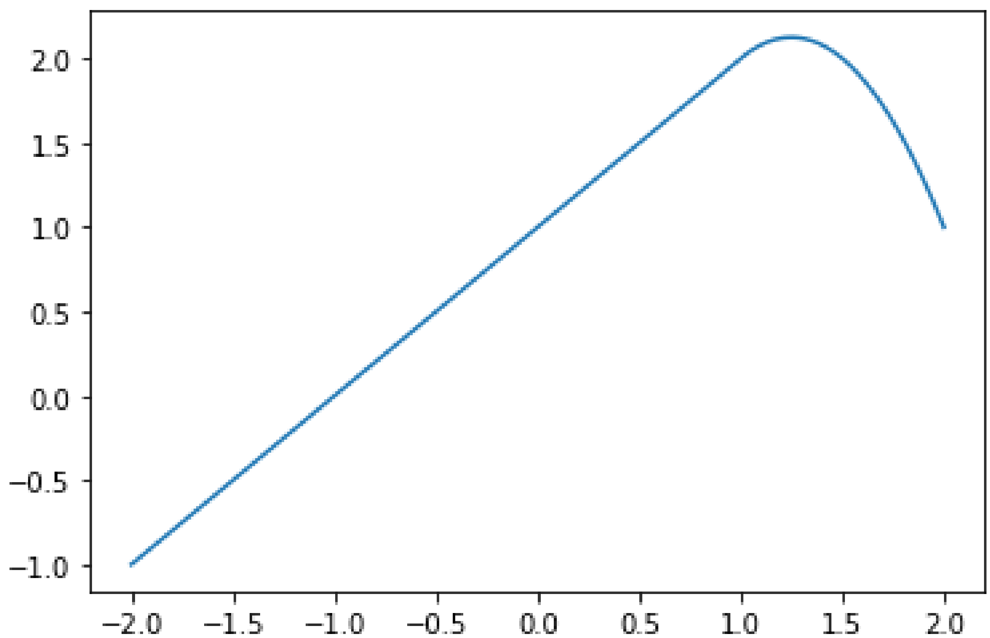
and obtain coefficient estimates

$$\hat{\beta}_0 = 1, \hat{\beta}_1 = 1\hat{\beta}_2 = -2.$$

Sketch the estimated curve between $x = -2$ and $x = 2$. Note the intercepts, slopes, and other relevant information.

[9]:
```
def b_2(x_list):
    y = []
    for x in x_list:
        if x >= 1:
            y.append((x-1)**2)
        else:
            y.append(0)
    return y


import matplotlib.pyplot as plt
import random
import numpy as np
x = np.linspace(-2, 2, 500)
plt.plot(x, 1 + x + np.multiply(-2,b_2(x)));
```



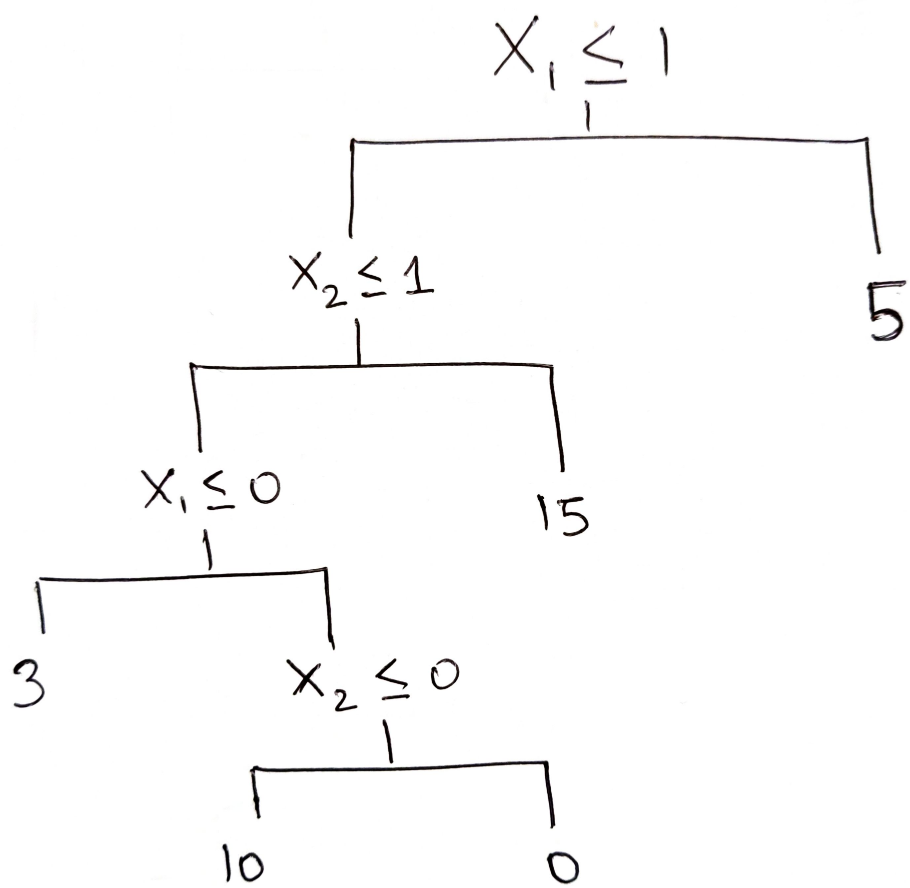
The curve is linear between $x = -2$ and $x = 1$, $y = 1 + x$ and quadratic between $x = 1$ and $x = 2$, $y = 1 + x - 2(x-1)^2$.

1.3 Trees

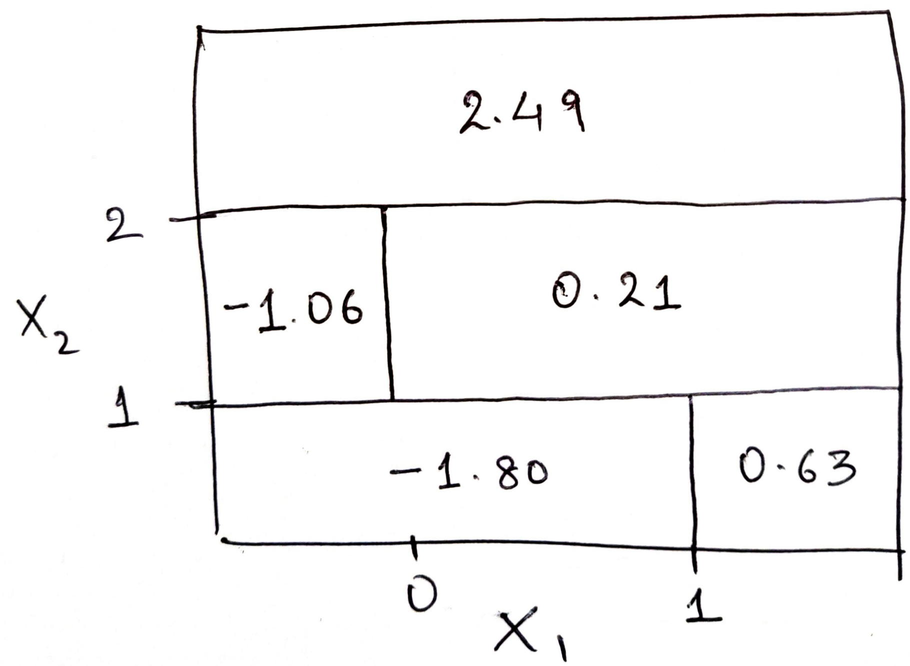
Question 11. [ISLR 8.4.4] Consider the figure below.

a. Sketch the tree corresponding to the partition of the predictor space illustrated in the left-hand panel. The numbers inside the boxes indicate the mean of Y within each region.

b. Create a diagram similar to the left-hand panel, using the tree illustrated in the right-hand panel. You should divide up the predictor space into the correct regions, and indicate the mean for each region.



Answer 11a



Answer 11b