2.4 Exercises

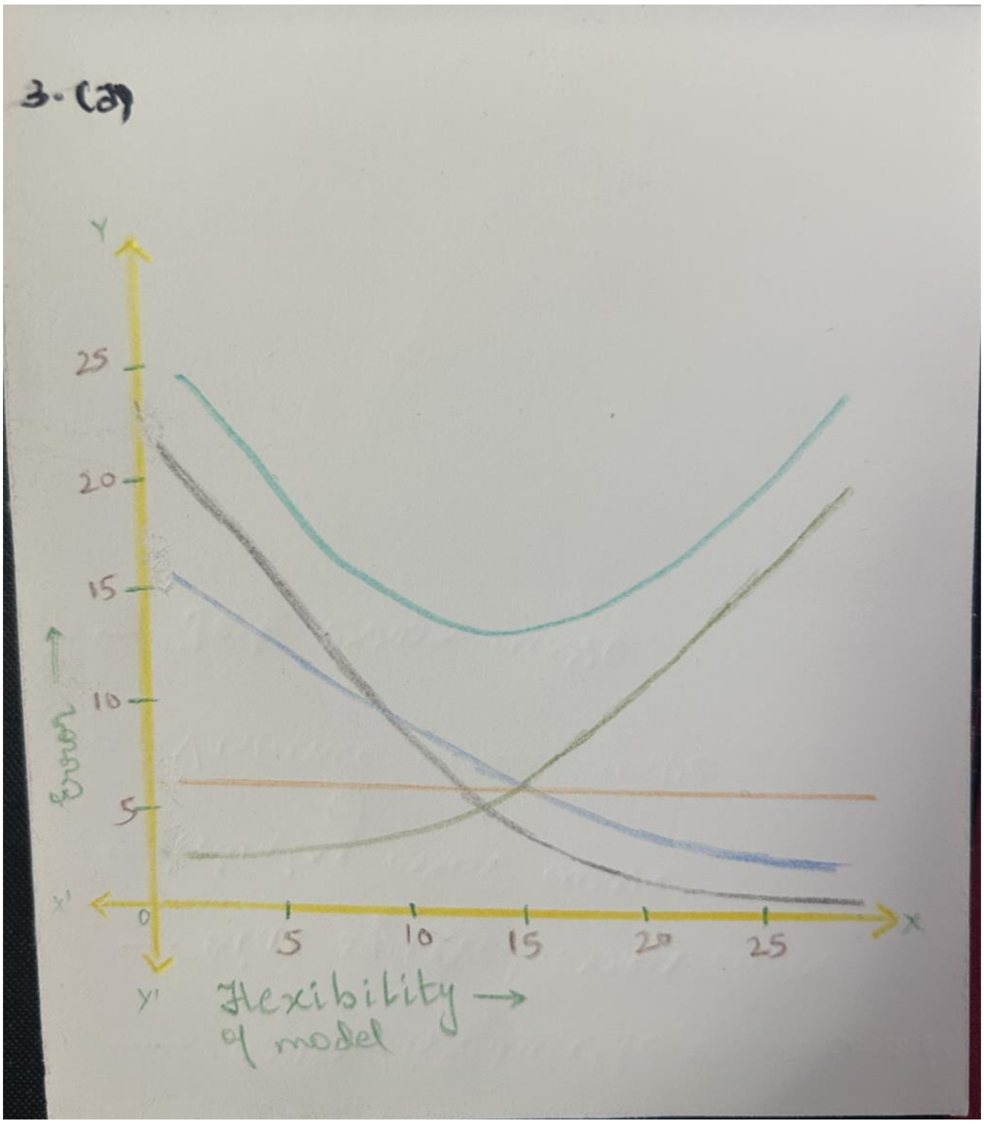
Conceptual

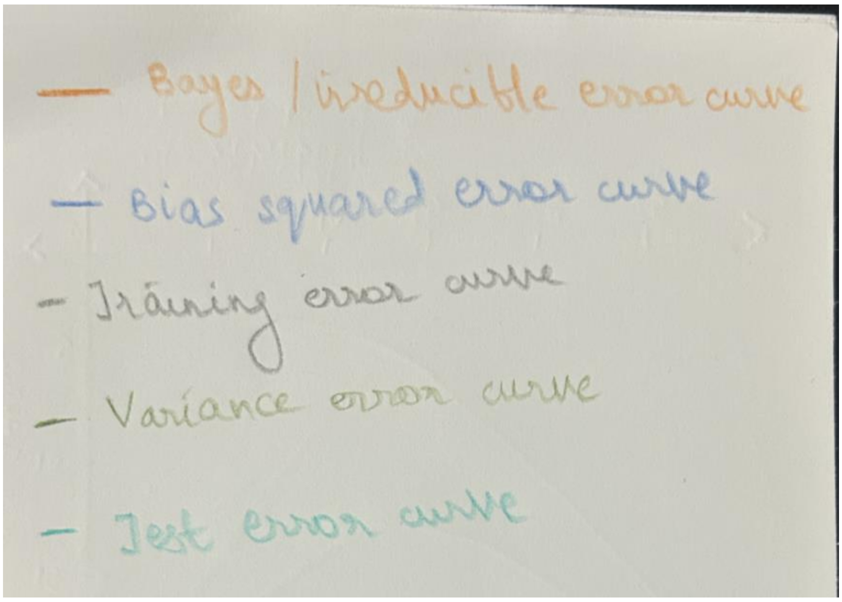
1. (a)When the sample size n is extremely large, and the number of predictors p is small, we would generally expect the performance of a flexible statistical learning method to be better than an inflexible method.
As the number of observations (n) is large, a flexible model would probably be better at capturing the numerous trends provided by the large amount of data and not overfit since a vast amount of data is available for the model, leading to a better performance overall. Also, the small number of predictors (assuming all are relevant) would also help prevent overfitting since the predictors p are all relevant (irrelevant predictors not affecting the response not considered.)

(b) When the number of predictors p is extremely large, and the number of observations n is small, we would generally expect the performance of a flexible statistical learning method to be worse than an inflexible method.
As the number of observations (n) is small, a flexible model would probably overfit on the small number of training samples leading to a small training error but hight test error (as the resultant model is extremely specific to the small range of training data provided) which is what is of significance since it can be equated to real world/unseen data. Also, in the large number of predictors it is possible some of them do not affect the response but would affect the flexible model to a great extent, causing the model to overfit even more.

(c) When the relationship between the predictors and response is highly non-linear, we would generally expect the performance of a flexible statistical learning method to be better than an inflexible method.
Since the relationship between the predictors and the response is highly non-linear, an inflexible model would not capture the trends well, leading to a high training and test error as opposed to a flexible model which can do a better job capturing the non-linear relationship due to a larger number of tuneable parameters.

(d) When the variance of the error terms, i.e. σ2 = Var("), is extremely High, we would generally expect the performance of a flexible statistical learning method to be worse than an inflexible method.
Since the flexible model would tend to find patterns based on the errors too (which are inconsequential and should ideally not be considered in the model), the resulting model would tend to be based on the errors too whose high variance would adversely affect the model performance as opposed to the inflexible model which would be relatively unaffected by the high variance error terms.


3. (a)
Y
25
20
15
10
5
0
Error →
X'
5
10
15
20
25
X
Y'
Flexibility →
of model


— Bayes / irreducible error curve
— Bias squared error curve
— Training error curve
— Variance error curve
— Test error curve

3. (b) The typical (squared) bias curve representing the error introduced due to bias reduces as the flexibility (which can be represented by degrees of freedom) increases.

Bias can be thought as the error when the model, hypothetically is trained on an infinite amount of data. Higer the model flexibility, the more trends and patterns it can capture from the huge (infinite) source of hypothetical training data resulting in a lower bias.

The variance increases as the flexibility (which can be represented by degrees of freedom) increases.

Variance represents the sensitivity of the model to the training data, i.e. the model changes considerably with a small change in the training data. As the flexibility increases, a small change in the training data affects the model more strongly.

The training error decreases as the flexibility (which can be represented by degrees of freedom) increases.

As the flexibility increases, the model can better capture complex underlying patterns due to the increased number of tuneable parameters which can in turn reduce the error of the model on the training data.

In general, the test error initially decreases as flexibility increases up to a certain point, beyond which an increase in flexibility leads to an increase in test error.

This could be attributed to the fact that in in the initial phase (models with low flexibility) would probably lead to a model that has not captured all trends presented in the training data (underfitting). As the flexibility increases, the model captures more information present in the training data till it reaches the optimal flexibility. Beyond this point, the model overfits on the training data leading it to 'memorize' the training data causing large errors on test (real world/unseen) data (leading to an increase in test error with increase in flexibility.

Bayes (or irreducible) error is not affected by flexibility/ degrees of freedom of the model. Bayes/ irreducible error can be attributed to error while reading an instrument and not considering attributes pertinent to the output variable in the model.

6.

	Parametric statistical learning	**Non- parametric statistical learning**
	In a parametric statistical learning approach, a parametric /functional form is selected which is used to model 'nature's model' and the problem is essentially reduced to estimating the parameters of the selected functional form.	In a non-parametric statistical learning approach, no explicit assumption is made about the shape/functional form for the model allowing for a greater variety of shapes for the model potentially.
Advantages	Parametric models are generally more interpretable. Hence, when inference is key, parametric approaches might be preferred to provide clarity.	Owing to the variety in forms, non-parametric models can better capture the trends in complex relationships, especially when a large amount of training data is available as compared to parametric models.
Disadvantages	Nature's model may be complicated in many cases and a non-parametric statistical learning approach may provide a better estimate as compared to a simple parametric approach since the parametric models might not capture all trends and information (underfitting) as well as non-parametric models.	Non- parametric models have poor interpretability and are preferred when the key focus is on prediction and interpretability is secondary. They could also follow the noise in the dataset too closely, leading to high test error especially if the error variance is high. Overfitting is another potential issue with non-parametric approaches, especially when the training dataset is small causing the model to 'memorize' and leading to poor test performance.

Applied

8. (a) Refer to Homework#1_Isha_Jain notebook

8 (a)

In [1]:
```
#Reading data into dataframe
import pandas as pd
college = pd.read_csv('College (1).csv')
college
```

Out[1]:

	Unnamed: 0	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	S.
0	Abilene Christian University	Yes	1660	1232	721	23	52	2885	537	7440	3300	450	2200	70	78	
1	Adelphi University	Yes	2186	1924	512	16	29	2683	1227	12280	6450	750	1500	29	30	
2	Adrian College	Yes	1428	1097	336	22	50	1036	99	11250	3750	400	1165	53	66	
3	Agnes Scott College	Yes	417	349	137	60	89	510	63	12960	5450	450	875	92	97	
4	Alaska Pacific University	Yes	193	146	55	16	44	249	869	7560	4120	800	1500	76	72	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
772	Worcester State College	No	2197	1515	543	4	26	3089	2029	6797	3900	500	1200	60	60	
773	Xavier University	Yes	1959	1805	695	24	47	2849	1107	11520	4960	600	1250	73	75	
774	Xavier University of Louisiana	Yes	2097	1915	695	34	61	2793	166	6900	4200	617	781	67	75	
775	Yale University	Yes	10705	2453	1317	95	99	5217	83	19840	6510	630	2115	96	96	

(b) Refer to Homework#1_Isha_Jain notebook

8 (b)

In [2]:
```
college2 = pd.read_csv('College (1).csv', index_col=0)
college3 = college.rename({'Unnamed: 0': 'College'},
axis=1)
college3 = college3.set_index('College')
college=college3
college
```

Out[2]:

College	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	S.F.Rat
Abilene Christian University	Yes	1660	1232	721	23	52	2885	537	7440	3300	450	2200	70	78	18
Adelphi University	Yes	2186	1924	512	16	29	2683	1227	12280	6450	750	1500	29	30	12
Adrian College	Yes	1428	1097	336	22	50	1036	99	11250	3750	400	1165	53	66	12
Agnes Scott College	Yes	417	349	137	60	89	510	63	12960	5450	450	875	92	97	7
Alaska Pacific University	Yes	193	146	55	16	44	249	869	7560	4120	800	1500	76	72	11
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Worcester State College	No	2197	1515	543	4	26	3089	2029	6797	3900	500	1200	60	60	21
Xavier University	Yes	1959	1805	695	24	47	2849	1107	11520	4960	600	1250	73	75	13
Xavier University of Louisiana	Yes	2097	1915	695	34	61	2793	166	6900	4200	617	781	67	75	14

(c) Refer to Homework#1_Isha_Jain notebook

8 (c)

In [3]:
```
college.describe()
```

Out[3]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	
count	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	77
mean	3001.638353	2018.804376	779.972973	27.558559	55.796654	3699.907336	855.298584	10440.669241	4357.526384	549.380952	1340.642214	7
std	3870.201484	2451.113971	929.176190	17.640364	19.804778	4850.420531	1522.431887	4023.016484	1096.696416	165.105360	677.071454	1
min	81.000000	72.000000	35.000000	1.000000	9.000000	139.000000	1.000000	2340.000000	1780.000000	96.000000	250.000000	
25%	776.000000	604.000000	242.000000	15.000000	41.000000	992.000000	95.000000	7320.000000	3597.000000	470.000000	850.000000	6
50%	1558.000000	1110.000000	434.000000	23.000000	54.000000	1707.000000	353.000000	9990.000000	4200.000000	500.000000	1200.000000	7
75%	3624.000000	2424.000000	902.000000	35.000000	69.000000	4005.000000	967.000000	12925.000000	5050.000000	600.000000	1700.000000	8
max	48094.000000	26330.000000	6392.000000	96.000000	100.000000	31643.000000	21836.000000	21700.000000	8124.000000	2340.000000	6800.000000	10

(d) Refer to Homework#1_Isha_Jain notebook

8 (d)

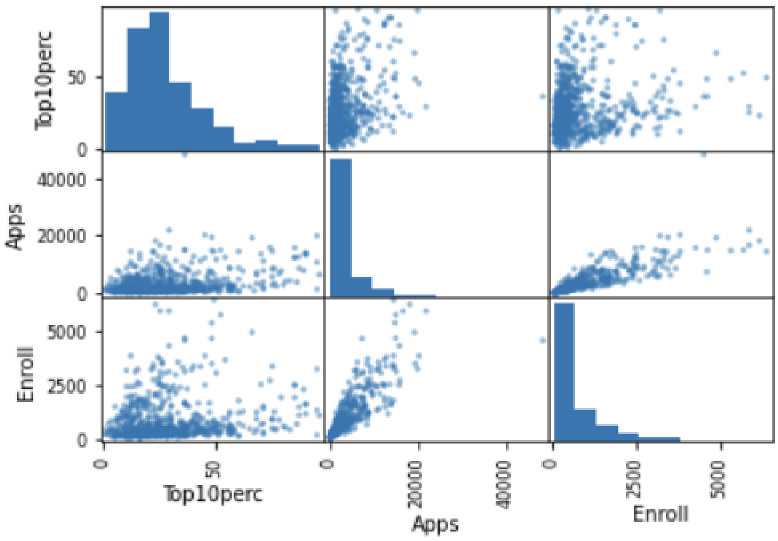
In [6]:
```
small_college_df=college[['Top10perc','Apps', 'Enroll']].copy()
small_college_df.head(5)
```

Out[6]:

College	Top10perc	Apps	Enroll
Abilene Christian University	23	1660	721
Adelphi University	16	2186	512
Adrian College	22	1428	336
Agnes Scott College	60	417	137
Alaska Pacific University	16	193	55

In [7]:
```
pd.plotting.scatter_matrix(small_college_df)
```

Out[7]:
```
array([[<AxesSubplot:xlabel='Top10perc', ylabel='Top10perc'>,
        <AxesSubplot:xlabel='Apps', ylabel='Top10perc'>,
        <AxesSubplot:xlabel='Enroll', ylabel='Top10perc'>],
       [<AxesSubplot:xlabel='Top10perc', ylabel='Apps'>,
        <AxesSubplot:xlabel='Apps', ylabel='Apps'>,
        <AxesSubplot:xlabel='Enroll', ylabel='Apps'>],
       [<AxesSubplot:xlabel='Top10perc', ylabel='Enroll'>,
        <AxesSubplot:xlabel='Apps', ylabel='Enroll'>,
        <AxesSubplot:xlabel='Enroll', ylabel='Enroll'>]], dtype=object)
```



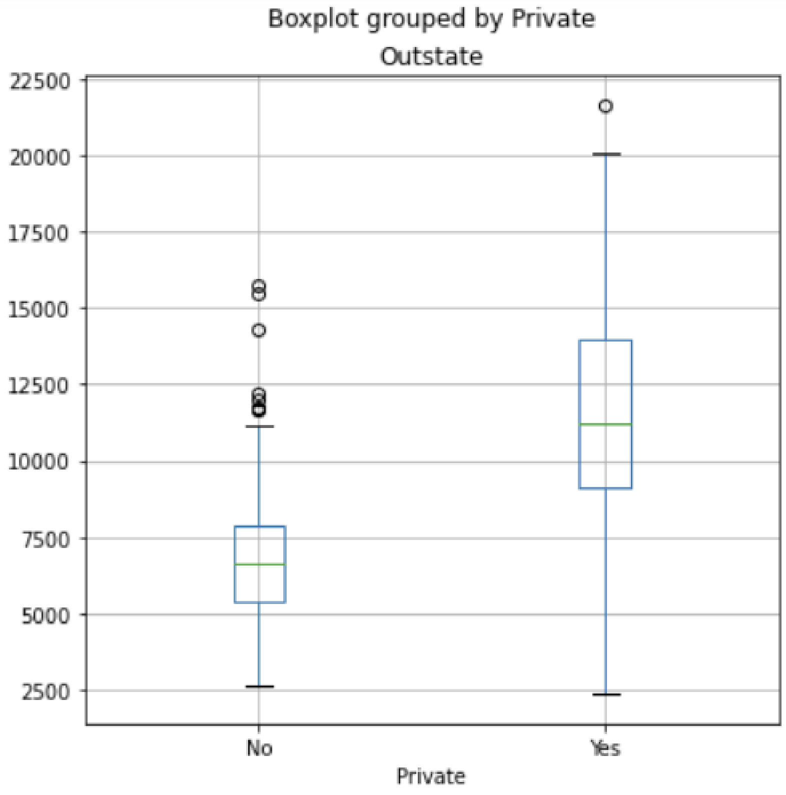
(e) Refer to Homework#1_Isha_Jain notebook

8 (e)

In [10]:
```
college.Private = pd.Series(college.Private, dtype='category')
college.Private.dtype
```

Out[10]:
```
CategoricalDtype(categories=['No', 'Yes'], ordered=False)
```

In [11]:
```
#import subplots from matplotlib.pyplot
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(6, 6))
college.boxplot('Outstate', by='Private', ax=ax);
```



(f) Referring to Homework#1_Isha_Jain notebook, we see that 78 elite universities are there.

8 (f)

In [12]:
```python
college['Elite'] = pd.cut(college['Top10perc'],
[0,50,100],
labels=['No', 'Yes'])
college
```

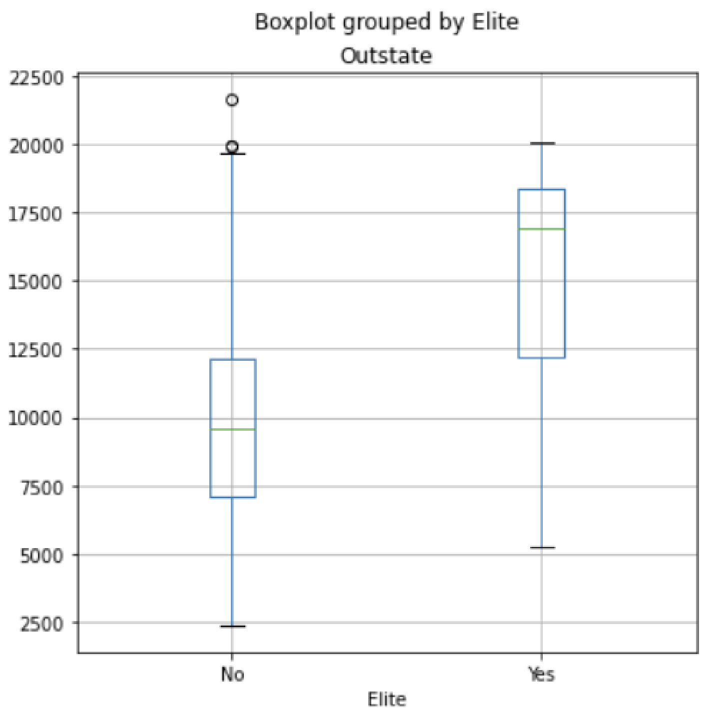
Out[12]:

College	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	S.F.Rat
Abilene Christian University	Yes	1660	1232	721	23	52	2885	537	7440	3300	450	2200	70	78	18
Adelphi University	Yes	2186	1924	512	16	29	2683	1227	12280	6450	750	1500	29	30	12
Adrian College	Yes	1428	1097	336	22	50	1036	99	11250	3750	400	1165	53	66	12
Agnes Scott College	Yes	417	349	137	60	89	510	63	12960	5450	450	875	92	97	7
Alaska Pacific University	Yes	193	146	55	16	44	249	869	7560	4120	800	1500	76	72	11
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Worcester State College	No	2197	1515	543	4	26	3089	2029	6797	3900	500	1200	60	60	21
Xavier University	Yes	1959	1805	695	24	47	2849	1107	11520	4960	600	1250	73	75	13
Xavier University of Louisiana	Yes	2097	1915	695	34	61	2793	166	6900	4200	617	781	67	75	14

In [13]:
```python
college['Elite'].value_counts()
```

Out[13]:
```
No     699
Yes     78
Name: Elite, dtype: int64
```

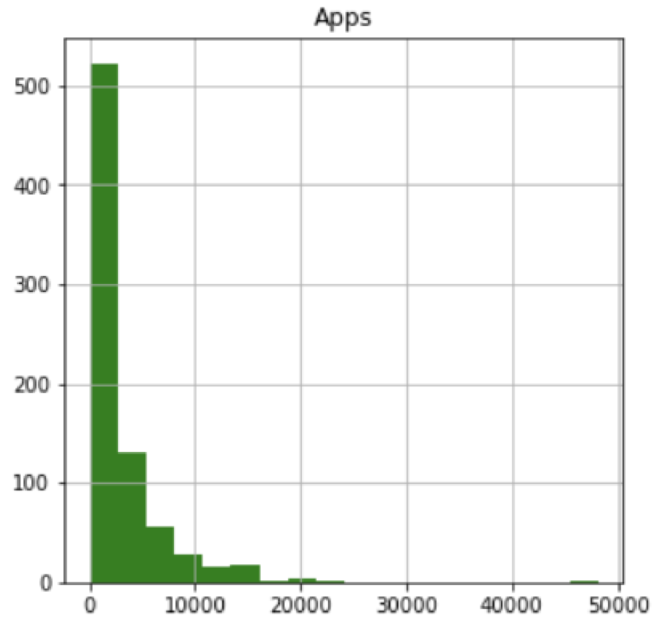
In [14]:
```python
fig, ax = plt.subplots(figsize=(6, 6))
college.boxplot('Outstate', by='Elite', ax=ax);
```



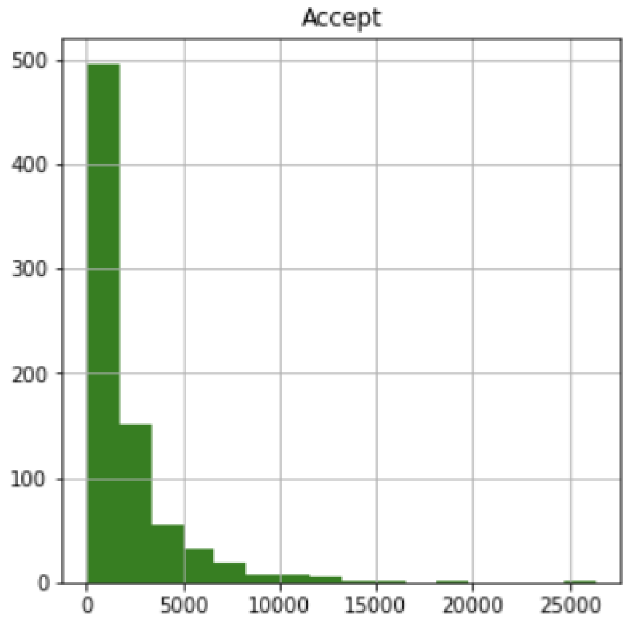
(g) Refer to Homework#1_Isha_Jain notebook

8 (g)

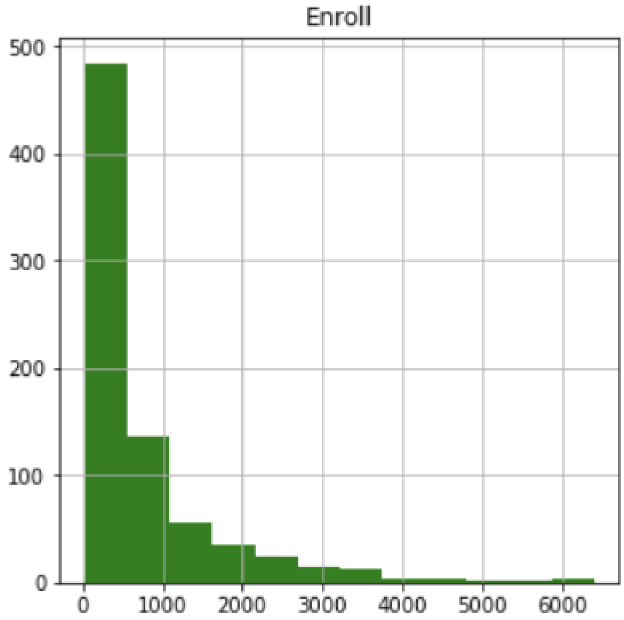
In [15]:
```
fig, ax = plt.subplots(figsize=(5, 5))
college.hist('Apps', color='green', bins=18, ax=ax);
```



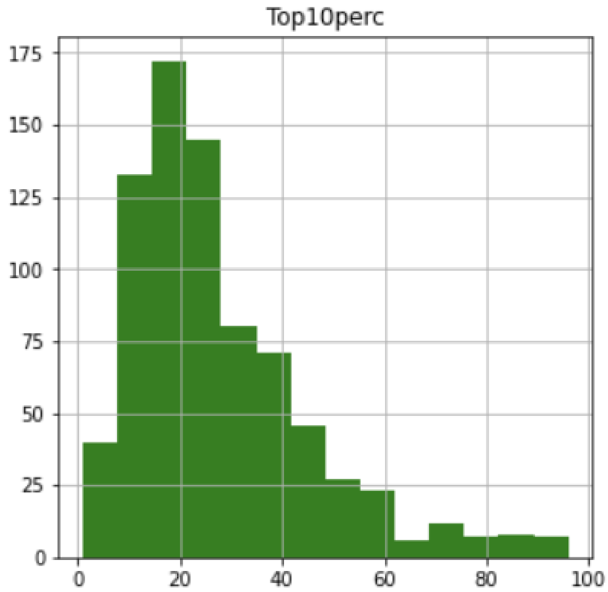
In [16]:
```
fig, ax = plt.subplots(figsize=(5, 5))
college.hist('Accept', color='green', bins=16, ax=ax);
```



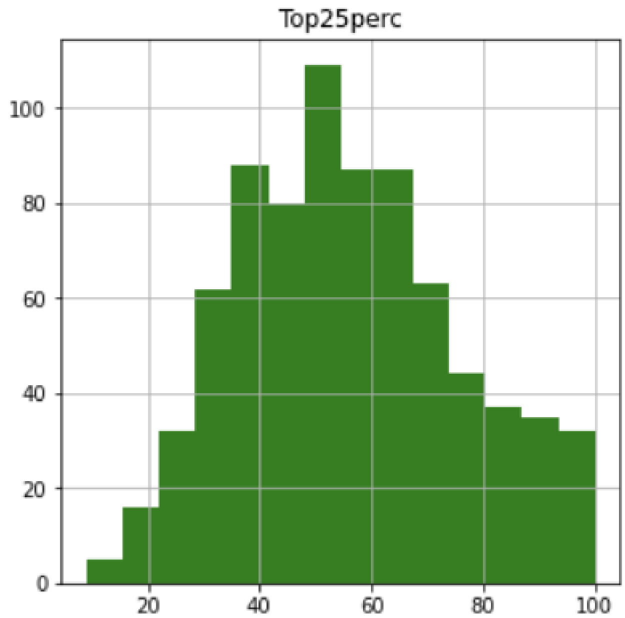
In [17]:
```
fig, ax = plt.subplots(figsize=(5, 5))
college.hist('Enroll', color='green', bins=12, ax=ax);
```



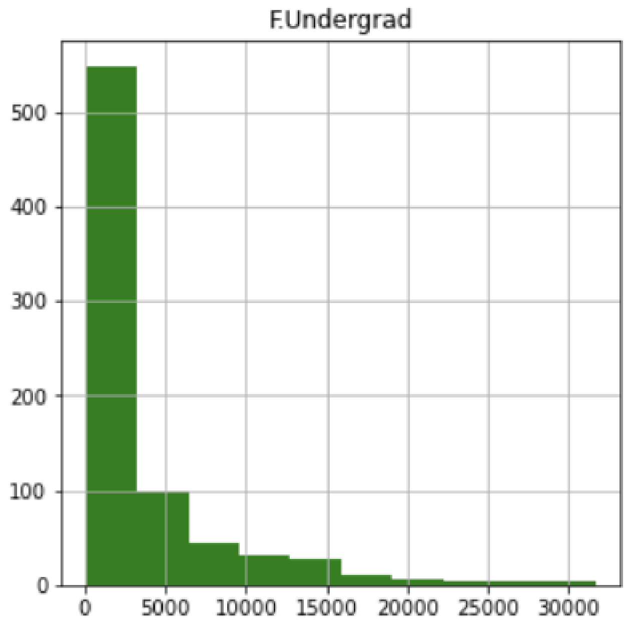
In [18]:
```
fig, ax = plt.subplots(figsize=(5, 5))
college.hist('Top10perc', color='green', bins=14, ax=ax);
```



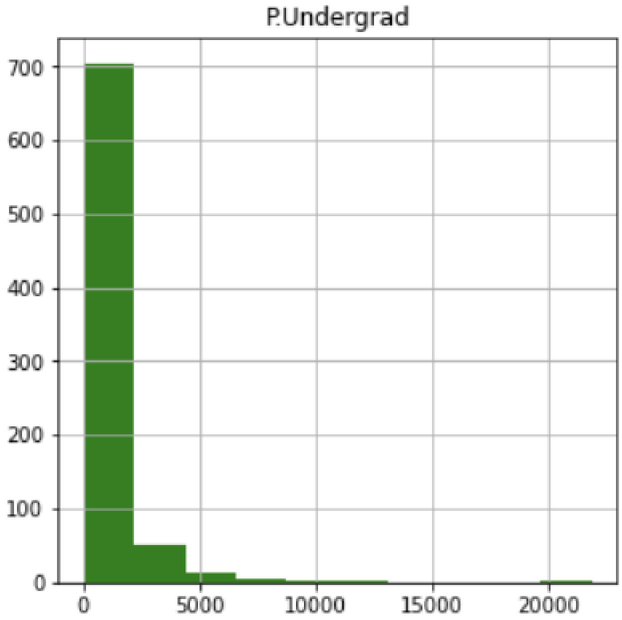
In [19]:
```
fig, ax = plt.subplots(figsize=(5, 5))
college.hist('Top25perc', color='green', bins=14, ax=ax);
```



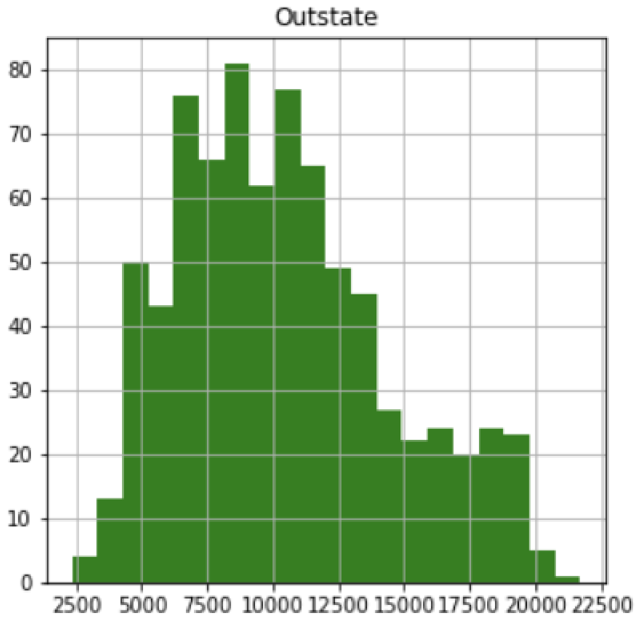
In [20]:
```
fig, ax = plt.subplots(figsize=(5, 5))
college.hist('F.Undergrad', color='green', bins=10, ax=ax);
```



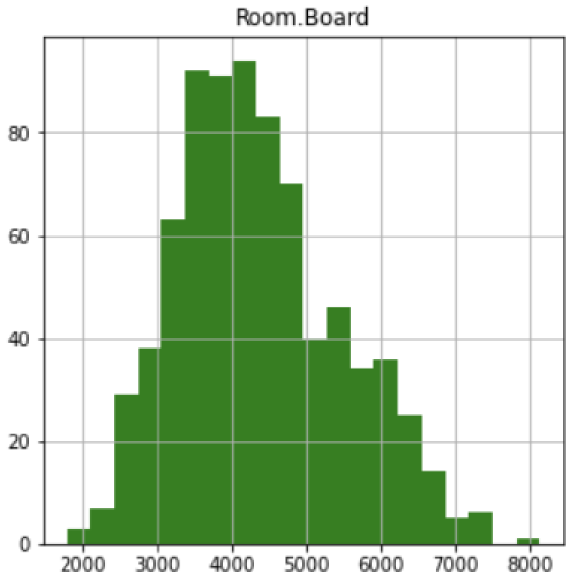
In [21]:
```
fig, ax = plt.subplots(figsize=(5, 5))
college.hist('P.Undergrad', color='green', bins=10, ax=ax);
```



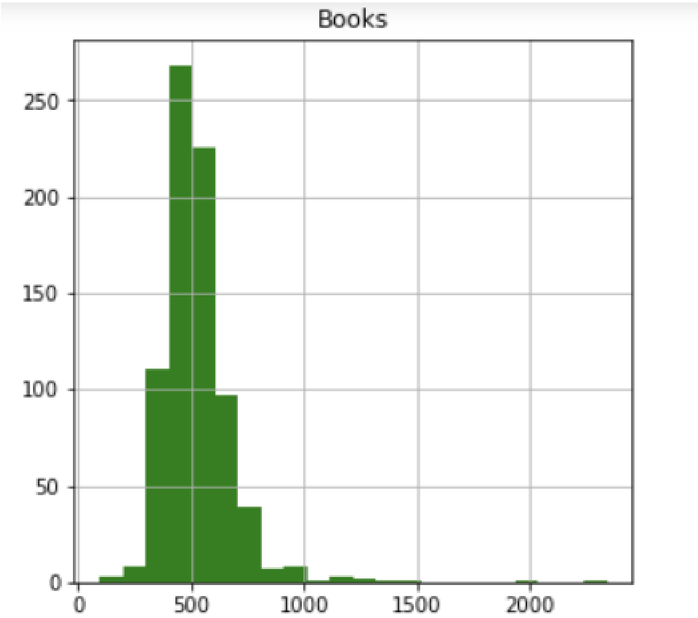
In [22]:
```
fig, ax = plt.subplots(figsize=(5, 5))
college.hist('Outstate', color='green', bins=20, ax=ax);
```

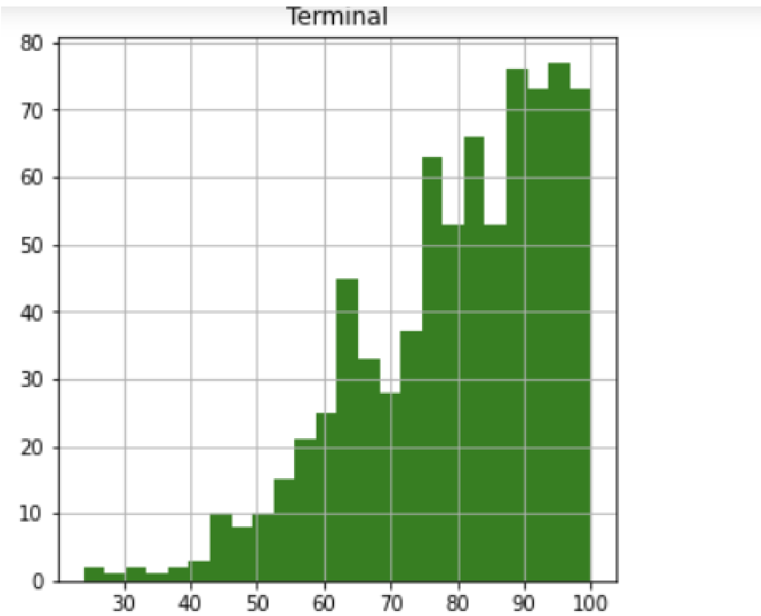


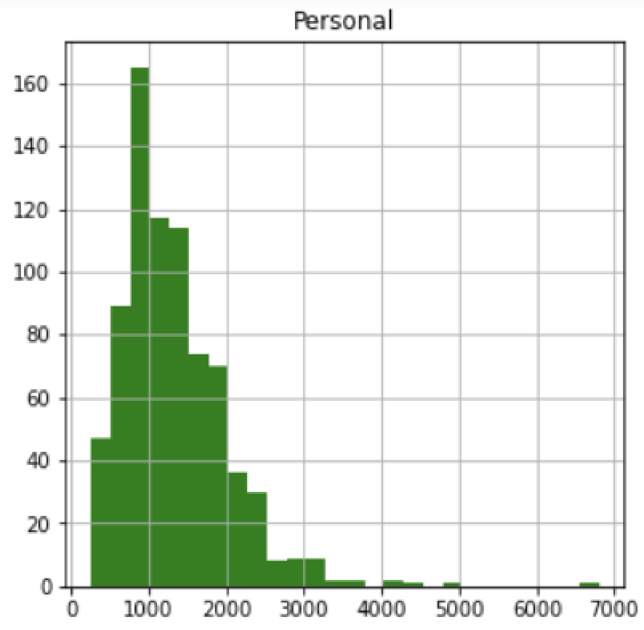
In [23]:
```
fig, ax = plt.subplots(figsize=(5, 5))
college.hist('Room.Board', color='green', bins=20, ax=ax);
```

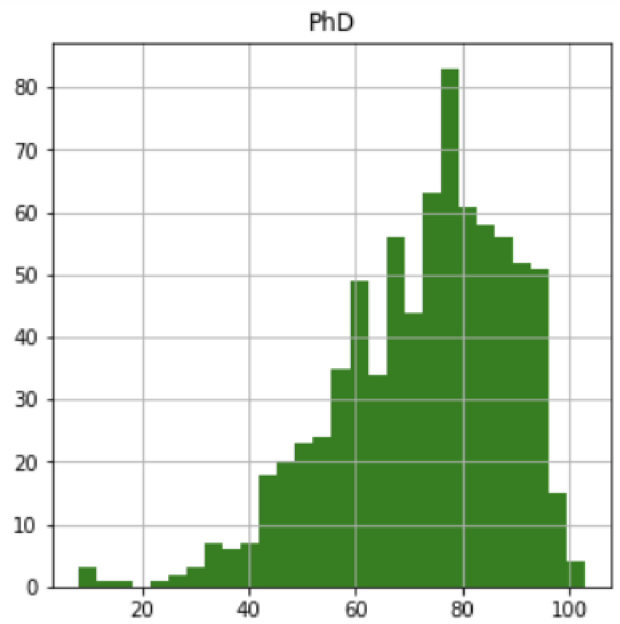


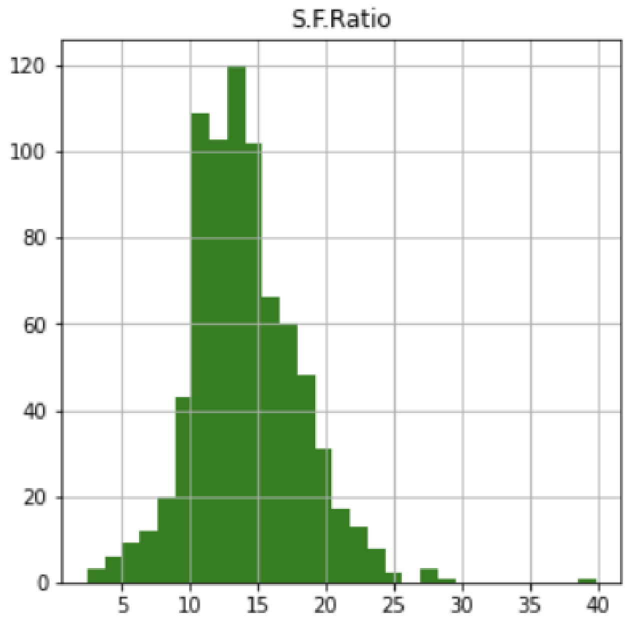
In [24]:
```
fig, ax = plt.subplots(figsize=(5, 5))
college.hist('Books', color='green', bins=22, ax=ax);

fig, ax = plt.subplots(figsize=(5, 5))
college.hist('Terminal', color='green', bins=24, ax=ax);

fig, ax = plt.subplots(figsize=(5, 5))
college.hist('Personal', color='green', bins=26, ax=ax);

fig, ax = plt.subplots(figsize=(5, 5))
college.hist('PhD', color='green', bins=28, ax=ax);

fig, ax = plt.subplots(figsize=(5, 5))
college.hist('S.F.Ratio', color='green', bins=29, ax=ax);

fig, ax = plt.subplots(figsize=(5, 5))
college.hist('perc.alumni', color='green', bins=30, ax=ax);

fig, ax = plt.subplots(figsize=(5, 5))
college.hist('Expend', color='green', bins=32, ax=ax);

fig, ax = plt.subplots(figsize=(5, 5))
college.hist('Grad.Rate', color='green', bins=10, ax=ax);
```

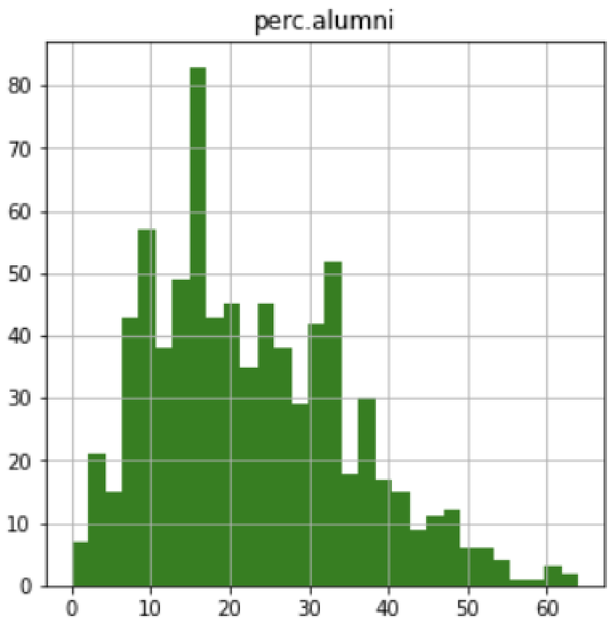

Books
0
50
100
150
200
250
0
500
1000
1500
2000

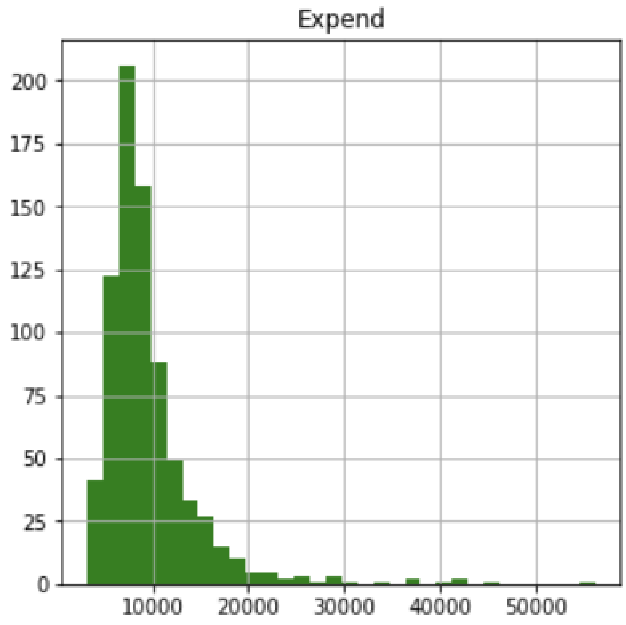

Terminal
0
10
20
30
40
50
60
70
80
30
40
50
60
70
80
90
100

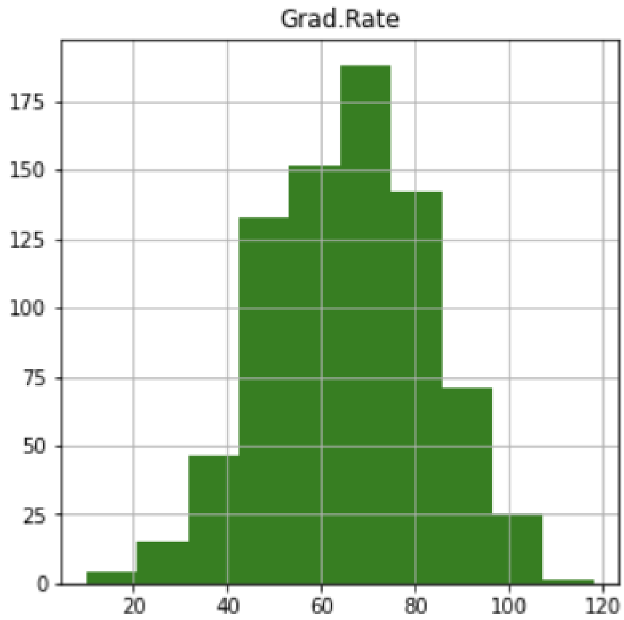

Personal
0
20
40
60
80
100
120
140
160
0
1000
2000
3000
4000
5000
6000
7000


PhD
0
10
20
30
40
50
60
70
80
20
40
60
80
100


S.F.Ratio
120
100
80
60
40
20
0
5
10
15
20
25
30
35
40


perc.alumni
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60





(h)On referring to Homework#1_Isha_Jain notebook, from the correlation matrix we can see that certain attributes such as ‘apps’ and ‘enrolment’, ‘apps’ and ‘accept’, ‘enrol’ and ‘accept’ seem to be highly correlated allowing us to drop certain attributes keeping only one to allow for quicker computation and avoid overfitting.

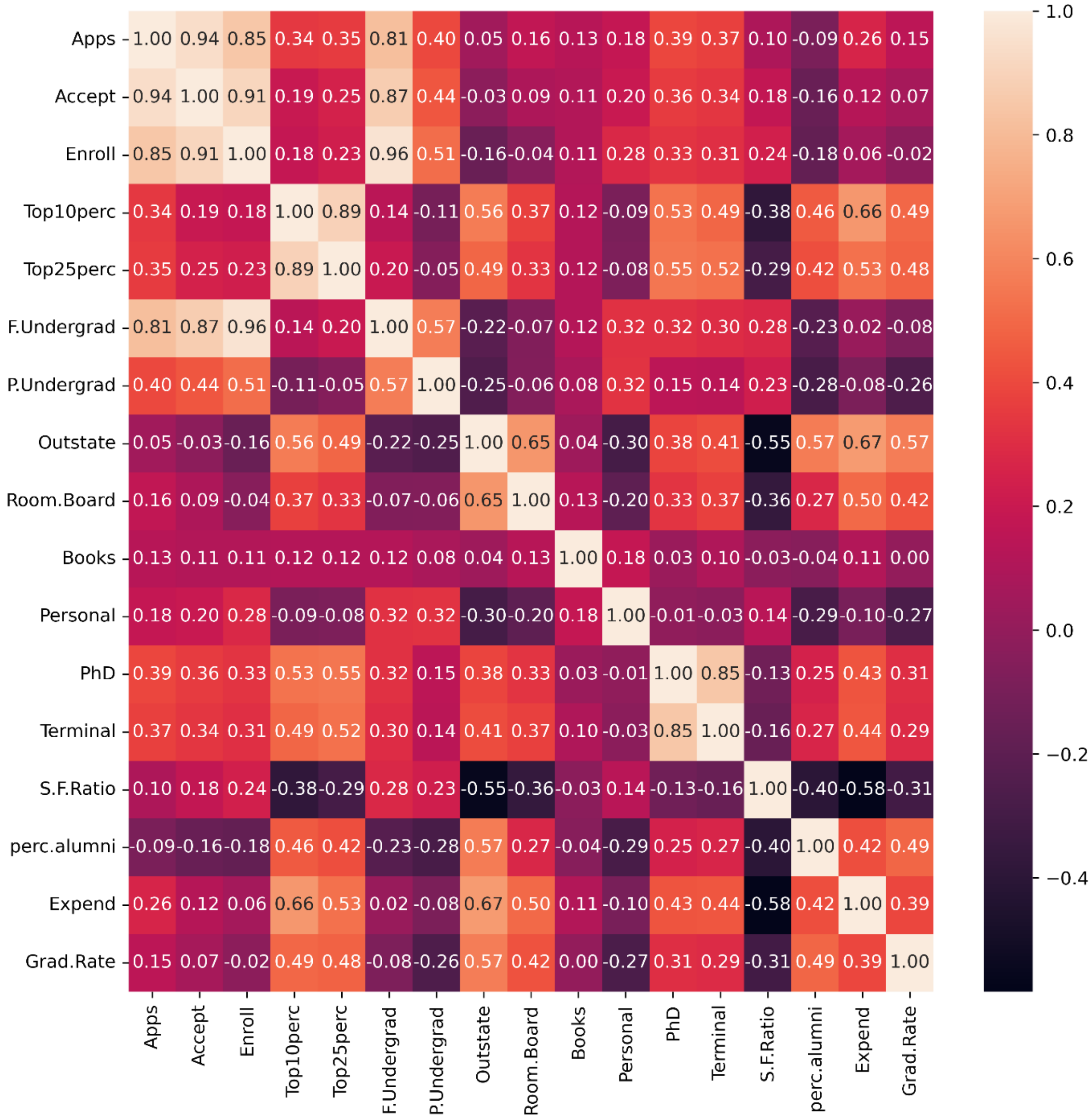
Also, from the boxplot of outstate students segregated by private/public; we can see that private institutions have a larger range with higher mean indicating a larger quantity of the outstate tuition is received by private institutions. This might be due to the possibility that public institutions provide waivers to instate students causing more in state students to apply to non-private institutions.

From the boxplot of outstate tuition vs the elite label built using the Top10perc attribute, we observe that Elite universities (i.e., those where the proportion of students coming from the top 10% of their high school classes exceeds 50%) correspond to higher values of outstate tuition.

8 (h) ¶

In [25]:
```
import seaborn as sns

corr_plot = plt.figure(figsize=(10,10), dpi = 480)
sns.heatmap(college.corr(), annot = True, fmt = '.2f')
```

Out[25]: <AxesSubplot:>



9. (a)We observe that in the Auto dataset, the following attributes are quantitative: mpg, cylinders, displacement, horsepower, weight, acceleration, year and origin. Whereas name is a qualitative variable.

(b) The range of the quantitative variables are illustrated below:

Range of mpg is 37.6
Range of cylinders is 5
Range of displacement is 387.0
Range of weight is 3527
Range of acceleration is 16.8
Range of year is 12
Range of origin is 2

9 (b)

```
In [35]: import numpy as np
for i in ('mpg','cylinders','displacement','weight','acceleration','year','origin'):
    arr=new_Auto[i].index.values
    #print(type(arr))
    range=np.max(new_Auto[i])-np.min(new_Auto[i])
    #print(range)
    #print('Range of', i, 'is', Auto[i].max()-Auto[i].min())
    print('Range of', i, 'is', range)
```

```
Range of mpg is 37.6
Range of cylinders is 5
Range of displacement is 387.0
Range of weight is 3527
Range of acceleration is 16.8
Range of year is 12
Range of origin is 2
```

(c) The mean and standard deviation of each quantitative predictor is illustrated below (rounded to 2 decimal places):

Mean of mpg is 23.45
Standard deviation of mpg is 7.80
Mean of cylinders is 5.47
Standard deviation of cylinders is 1.70
Mean of displacement is 194.41
Standard deviation of displacement is 104.51
Mean of weight is 2977.58
Standard deviation of weight is 848.32
Mean of acceleration is 15.54
Standard deviation of acceleration is 2.76
Mean of year is 75.98
Standard deviation of year is 3.68
Mean of origin is 1.58
Standard deviation of origin is 0.80

9 (c)

```
In [36]: for i in ('mpg','cylinders','displacement','weight','acceleration','year','origin'):
    arr=new_Auto[i].index.values
    #print(type(arr))
    mean=np.mean(new_Auto[i])
    #print(mean)
    print('Mean of', i, 'is', mean)
    #print('Range of', i, 'is', range)
    std=np.std(new_Auto[i])
    print('Standard deviation of', i, 'is', std)
```

(d) The range, mean, and standard deviation of each predictor in the subset of the data that remains on removing the 10th to 85th observation (both, the 10th and 85th observation is excluded from the data) is (rounded to 2 decimal places):

Range of mpg is 35.6
Mean of mpg is 24.40
Standard deviation of mpg is 7.85
Range of cylinders is 5
Mean of cylinders is 5.37
Standard deviation of cylinders is 1.65
Range of displacement is 387.0
Mean of displacement is 187.24
Standard deviation of displacement is 99.52
Range of weight is 3348
Mean of weight is 2935.97
Standard deviation of weight is 810.02
Range of acceleration is 16.3
Mean of acceleration is 15.73
Standard deviation of acceleration is 2.69
Range of year is 12
Mean of year is 77.15
Standard deviation of year is 3.10
Range of origin is 2
Mean of origin is 1.60
Standard deviation of origin is 0.82

9 (d)

In [37]:
```
#getting indices to keep (removing 10th to 85th observation where both the 10th and 85th observation are removed from the dataset
a1=[]
a2=[]
i=0
j=85
while i<9:
    a1.append(i)
    i=i+1
while j<392:

    a2.append(j)
    j+=1
print(a1,a2)
#auto_smaller=new_Auto.iloc[a1,a2]
#auto_smaller
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8] [85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 10
7, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132,
133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 15
8, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183,
184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 20
9, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234,
235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 26
0, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285,
286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 31
1, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336,
337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 36
2, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387,
388, 389, 390, 391]
```

In [38]:
```
auto_smaller1=new_Auto.iloc[a1]
auto_smaller2=new_Auto.iloc[a2]
auto_smaller = pd.concat([auto_smaller1, auto_smaller2], axis=0)
auto_smaller
```

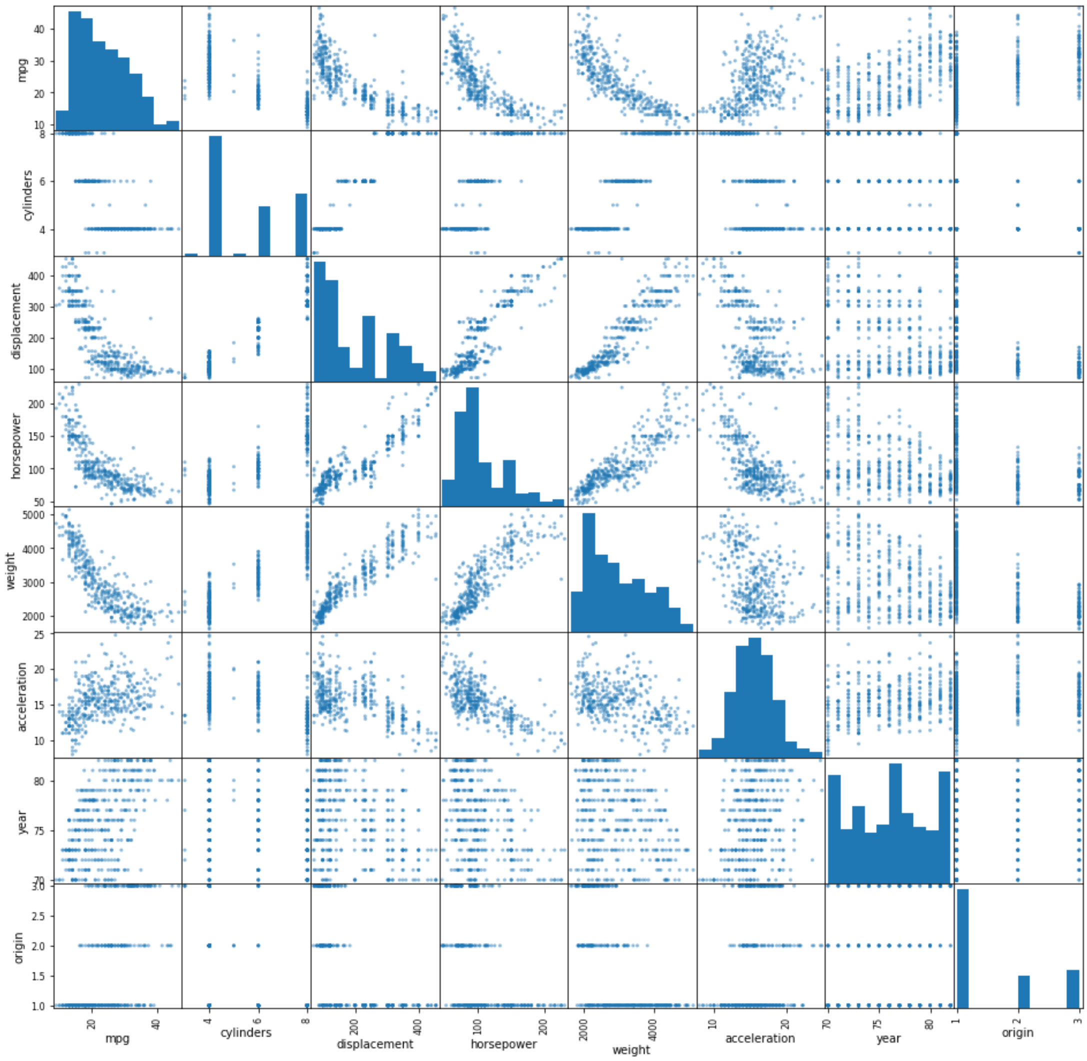
In [39]:
```
for i in ('mpg','cylinders','displacement','weight','acceleration','year','origin'):
    #arr=auto_smaller[i].index.values
    #print(type(arr))
    range=np.max(auto_smaller[i])-np.min(auto_smaller[i])
    #print(range)
    #print('Range of', i, 'is', Auto[i].max()-Auto[i].min())
    print('Range of', i, 'is', range)

    #print(type(arr))
    mean=np.mean(auto_smaller[i])
    #print(mean)
    print('Mean of', i, 'is', mean)
    #print('Range of', i, 'is', range)
    std=np.std(auto_smaller[i])
    print('Standard deviation of', i, 'is', std)
```

(e) From the scatterplots and histograms, we observe the following relationships:

- Cylinders and origin have discrete integer values.
- Mpg is negatively correlated to displacement, horsepower and weight (seemingly exponential)
- Displacements seems to have a positive correlation with horsepower and weight
- Horsepower and weight have a positive linear correlation whereas horsepower and acceleration are negatively correlated (seemingly exponential)

9 (e)

In [40]:
```
#pltsubplots(figsize=(20, 20))
pd.plotting.scatter_matrix(new_Auto, figsize=(16,16));
```

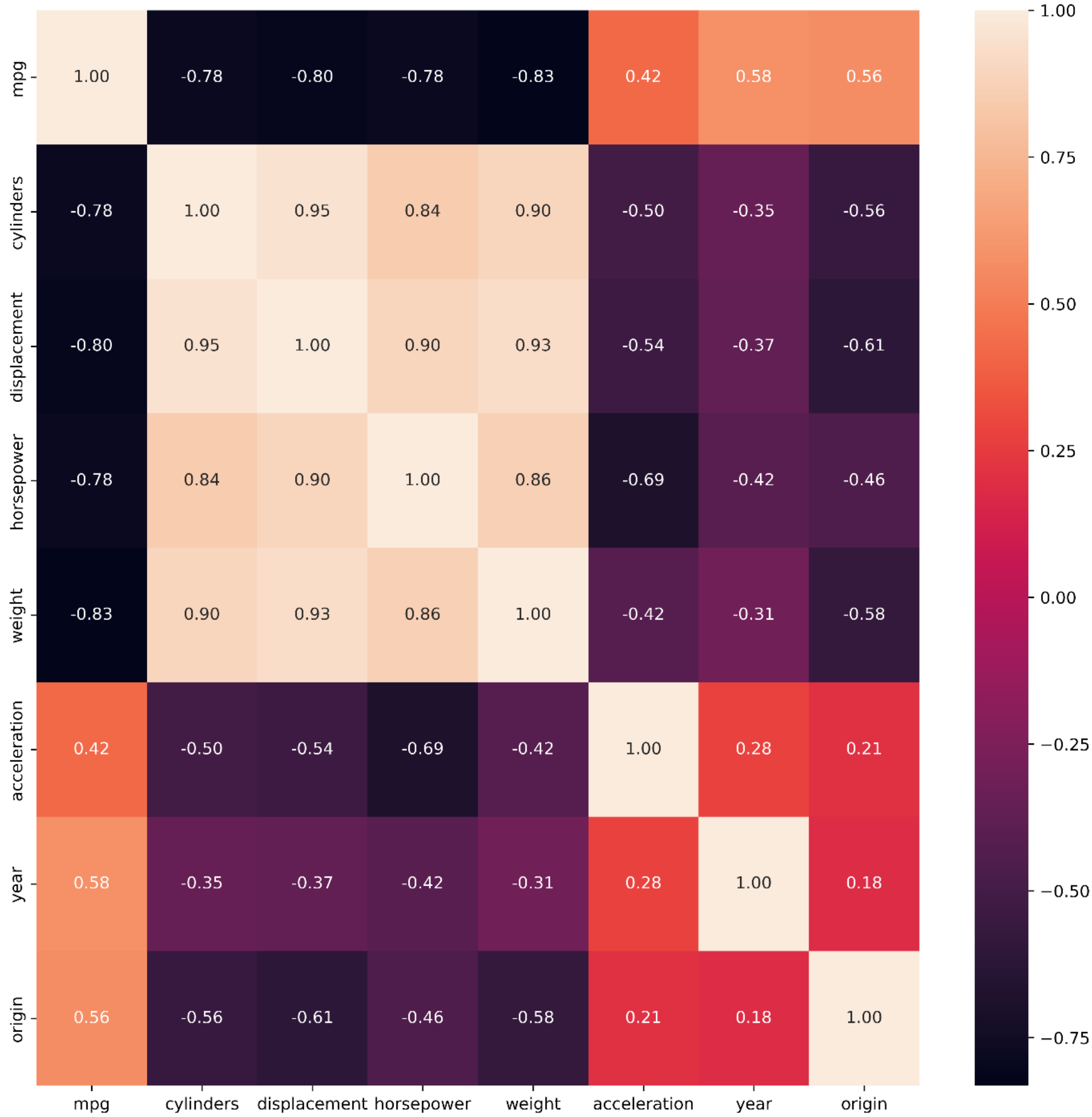


(f) From the correlation matrix, we can observe that the strength of correlation of the predict ors with mpg in descending order of the correlation coefficient (absolute value) is (Order in w hich predictors should be considered for a model):

- Weight (0.83),
- Displacement (0.8),
- Cylinders and horsepower (0.78),
- Year (0.58),
- Origin (0.56),
- Acceleration (0.42)

9 (f)

In [41]:
```
import seaborn as sns

fig = plt.figure(figsize=(12,12), dpi = 480)
sns.heatmap(Auto.corr(), annot = True, fmt = '.2f')
```



3.7 Exercises

Conceptual

1. The null hypothesis in table 3.4 corresponds to the various predictors (radio, TB, newspaper) and the intercept not affecting the response (sales/ number of units sold). Can be mathematically represented as $H_0 : \beta 1 = \beta 2 = \beta 3 = 0$.
We can interpret the p value to quantify the probability of the null hypothesis (intercept and coefficients of the predictors of the multiple regression model to be equal to zero) to be true as that would imply that the predictor has no effect on the response (sales) since it would not affect the response in the equation of multiple linear regression.

   From the table, we can see that the value of the coefficient for the newspaper predictor to have a small value close to zero with a p-value larger than 0.05 (i.e., less than 5% chance that the value of the coefficient was just a matter of chance), leading us to infer that there is a good chance that newspaper does not affect sales.

   The value of coefficients for the other predictors (radio and tv) and the intercept to have values much larger than that for the newspaper coefficient accompanied by a p-value less than 0.05 allowing us to infer that TV and radio affect sales and also that a non-zero intercept value fits the model.

5. $y_i = x_i \hat{\beta}$ [Considering linear regression without an intercept]

where $\hat{\beta} = \left(\sum_{i=1}^{n} x_i y_i\right) / \left(\sum_{i'=1}^{n} x_{i'}^2\right)$

Aim: To get $\hat{y}_i$ in the form of $a_{i'} y_{i'}$

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

and determine a_i

Here $y_{i'}$ represents the response values.

$y_i = x_i \hat{\beta}$

where $\hat{\beta} = \left(\sum_{i=1}^{n} x_i y_i\right)\left(\sum_{i=1}^{n} x_{i'}^2\right)$

$$\therefore y_i = \frac{x_i \left[\sum_{i=1}^{n} x_i y_i\right]}{\sum_{i'=1}^{n} x_{i'}^2}$$

$$\cancel{\therefore y_i = x_i \frac{x_1 y_1 + x_2 y_2 + \dots + x_n y_n}{x_1^2 + x_2^2 + \dots + x_n^2}}$$

$$\cancel{\therefore y_i = \frac{x_i x_1 y_1 + x_i x_2 y_2 + \dots + x_i x_n y_n}{x_1^2 + x_2^2 + \dots + x_n^2}}$$

Using a different notation (i', x' and y') to ~~generalize~~ represent the terms of $\hat{\beta}$ & to get it in the required form containing $y_{i'}$, since x_i can be considered a constant corresponding to y_i

$$\therefore \hat{y}_i = x_i \frac{\sum_{i'=1}^{n} x_{i'} y_{i'}}{\sum_{j=1}^{n} x_j^2} = \frac{\sum_{i'=1}^{n} x_{i'} y_{i'}}{\sum_{j=1}^{n} x_j^2}$$

$$\therefore \hat{y}_i = \sum_{i'=1}^{n} \left(\frac{x_i x_{i'}}{\sum_{j=1}^{n} x_j^2}\right) y_{i'} = \sum_{i'=1}^{n} a_{i'} y_{i'}$$

$$\Rightarrow a_{i'} = \frac{x_i x_{i'}}{\sum_{j=1}^{n} x_j^2}$$

6. As per (3.4),

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \quad — ①$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \quad — ②$$

when minimizing RSS using the least squares approach.

Let y^* be the value of the response calculated using the model satisfying (3.4) at $\bar{x}$ ~~(where~~

$$\therefore y^* = \hat{\beta}_1 \bar{x} + \hat{\beta}_0 \quad \text{[simple linear regression]}$$

$$\cancel{\therefore y^* = \left\{ \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \right\} \bar{x} + \bar{y}}$$

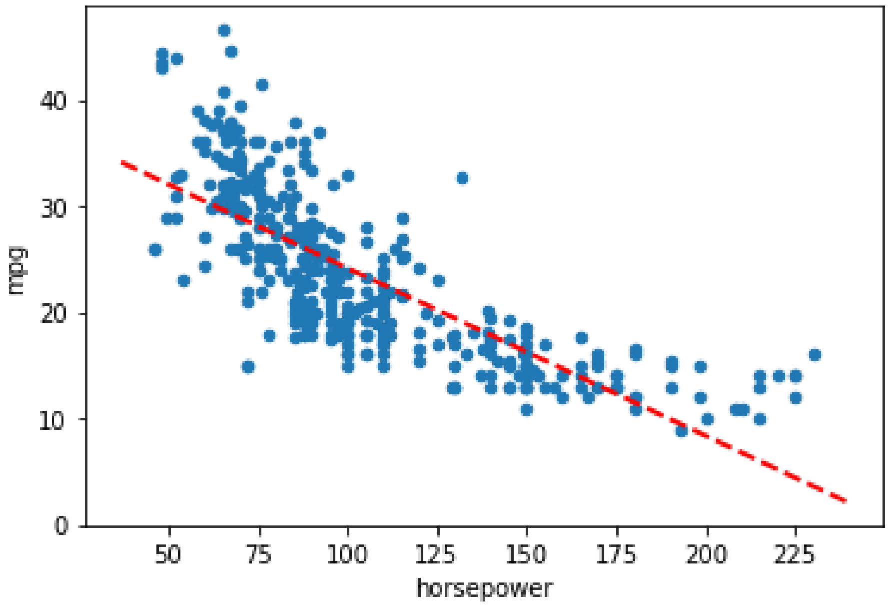
$$\therefore y^* = \hat{\beta}_1 \bar{x} + (\bar{y} - \hat{\beta}_1 \bar{x}) - (\text{from } ②)$$

$$\therefore y^* = \bar{y}$$

Hence proved that in the case of simple linear regression, the least squares line always passes through the point $(\bar{x}, \bar{y})$

Applied

8. (a) i. From the scatterplot we can observe an inverse relationship between the horsepower and mpg.



ii. The value of the coefficient (representing the slope) of horsepower in the linear model is -0.1578 with a standard error of 0.006 and P>|t| (representing probability of coefficient being zero) =0, allowing us to conclude that a relationship between horsepower and mpg exists (rejecting null hypothesis of no relation between predictor and response)

iii. The relationship between the predictor and the response is negative.

In [48]:
```
import numpy as np

# defining the variables
#x = Auto['horsepower'].tolist()
x = pd.DataFrame({'intercept': np.ones(new_Auto.shape[0]),
  'horsepower': new_Auto['horsepower']})

#x=x.tolist()
print(x[:5])
y = new_Auto['mpg']
```

```
   intercept  horsepower
0        1.0       130.0
1        1.0       165.0
2        1.0       150.0
3        1.0       150.0
4        1.0       140.0
```

In [49]:
```
model = sm.OLS(y.astype(float), x.astype(float))
```

In [50]:
```
results = model.fit()
```

In [51]:
```
summarize(results)
```

Out[51]:

	coef	std err	t	P>|t|
intercept	39.9359	0.717	55.660	0.0
horsepower	-0.1578	0.006	-24.489	0.0

iv. The predicted mpg associated with a horsepower of 98 is 24.47 rounded to 2 decimal places.

The associated 95 % confidence interval rounded to 2 decimal places is 23.97 mpg to 24.96 mpg

The associated 95 % prediction interval rounded to 2 decimal places is 14.81 mpg to 34.12 mpg.

8 (a)iv.

In [52]:
```
#design = MS(['horsepower'])
new_df = pd.DataFrame({'horsepower':[98]})
newX=pd.DataFrame({'intercept': np.ones(new_df.sha
  'horsepower': new_df['horsepower']})
#newX = design.transform(new_df)
newX
```

Out[52]:

	intercept	horsepower
0	1.0	98

In [53]:
```
new_predictions = results.get_prediction(newX);
new_predictions.predicted_mean
```

Out[53]: `array([24.46707715])`

In [54]:
```
new_predictions.conf_int(alpha=0.05)
```

Out[54]: `array([[23.97307896, 24.96107534]])`

In [55]:
```
new_predictions.conf_int(obs=True, alpha=0.05)
```

Out[55]: `array([[14.80939607, 34.12475823]])`

(b) Refer to Homework#1_Isha_Jain notebook

8 (b)

In [56]:
```
def abline(ax, b, m):
  "Add a line with slope m and intercept b to ax"
  xlim = ax.get_xlim()
  ylim = [m * xlim[0] + b, m * xlim[1] + b]
  ax.plot(xlim, ylim)
```

In [57]:
```
def abline(ax, b, m, *args, **kwargs):
  "Add a line with slope m and intercept b to ax"
  xlim = ax.get_xlim()
  ylim = [m * xlim[0] + b, m * xlim[1] + b]
  ax.plot(xlim, ylim, *args, **kwargs)
```

In [58]:
```
ax = new_Auto.plot.scatter('horsepower', 'mpg')
abline(ax,
  results.params[0],
  results.params[1],
  'r--',
  linewidth=2)
```

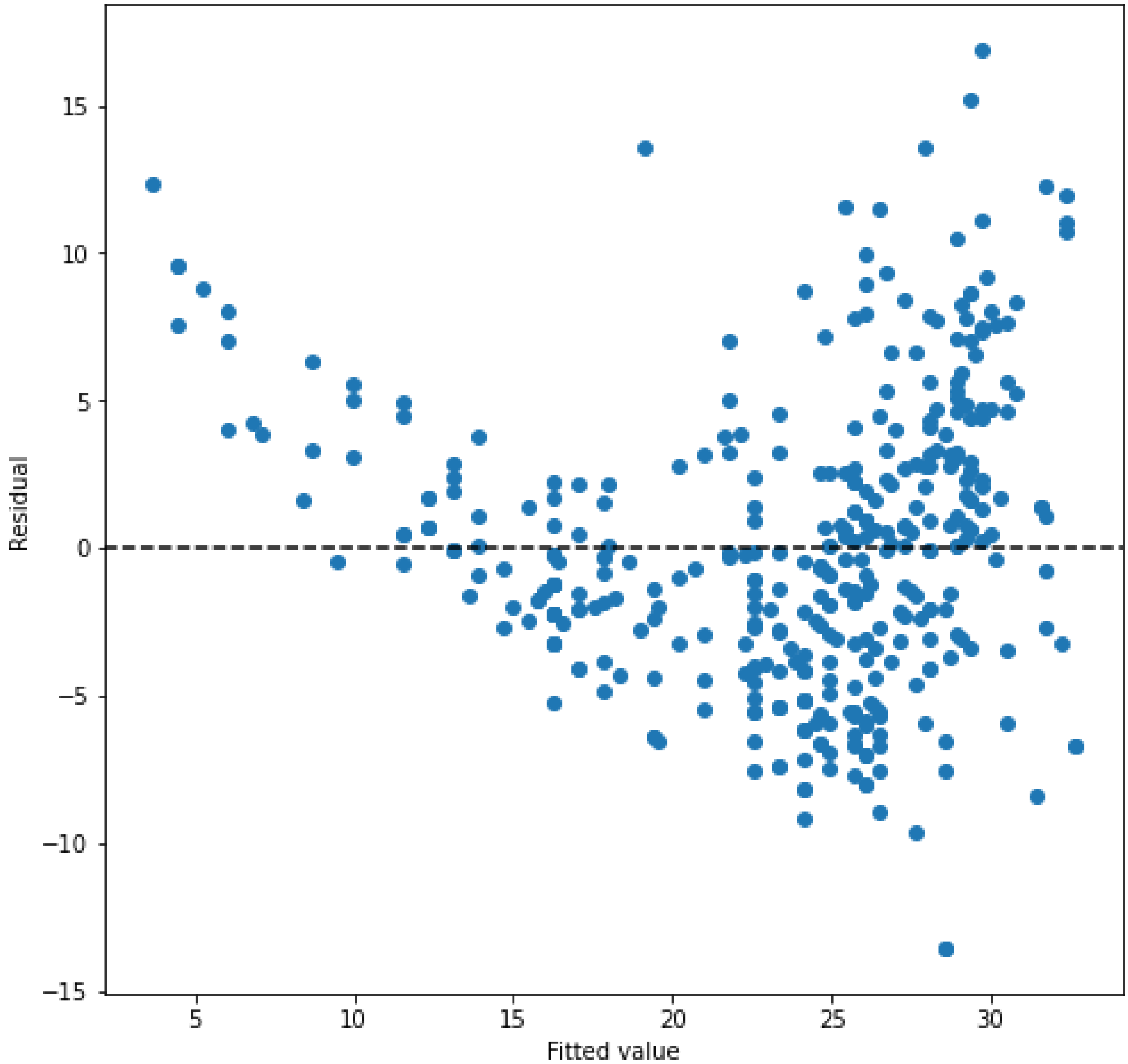
(c) The error plot suggests some non-linearity since there seems to be a trend whereas ideally the residual plot should look somewhat random. From the residual plot we can see that the residual goes progressively from a higher positive value to a negative value to again a positive value and has somewhat of a parabolic shape.

From the horsepower vs mpg scatterplot, we can see that the plot that is somewhat exponentially reducing is approximated linearly leading to the residual plot obtained.

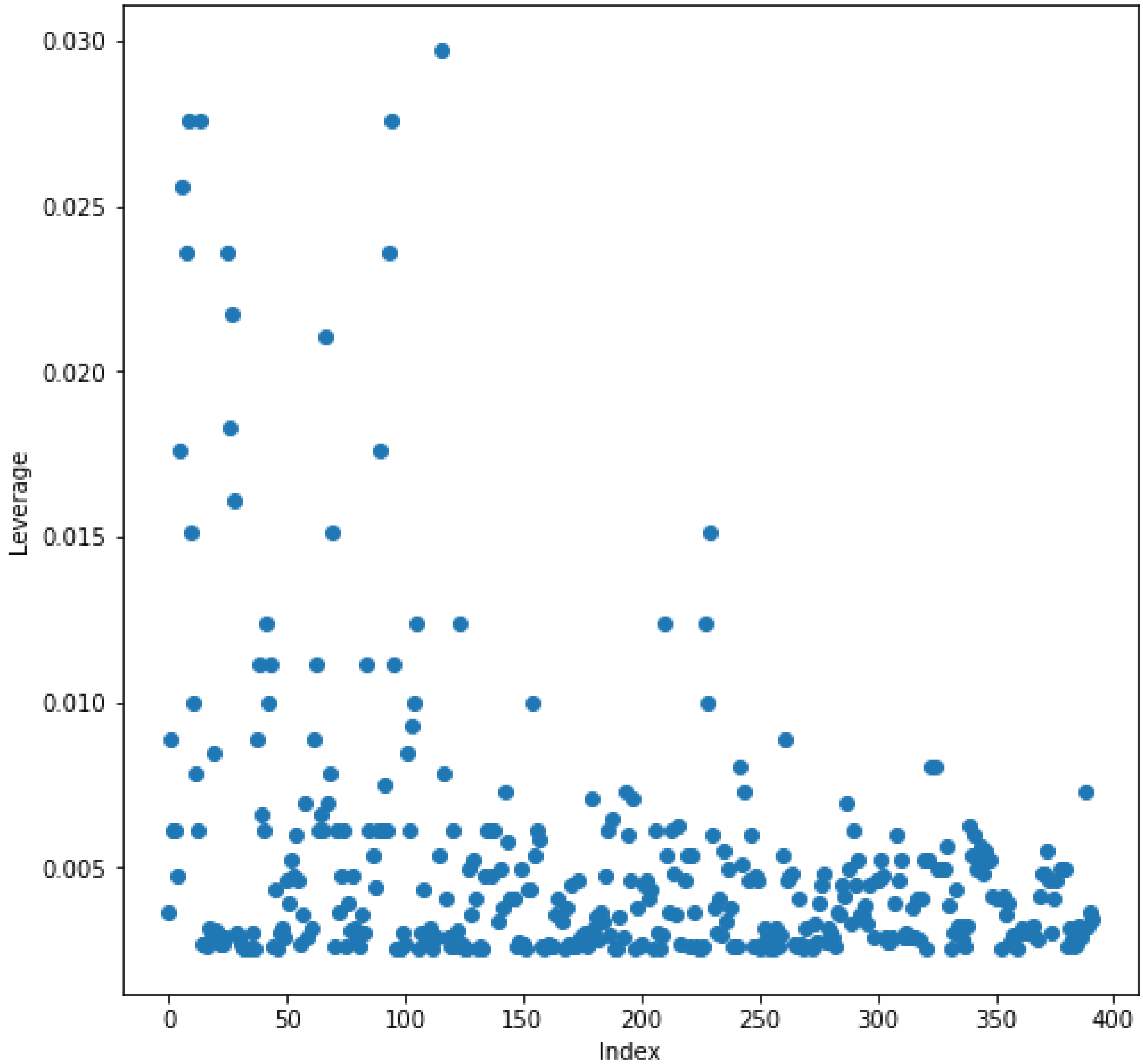
From the leverage plot we can see that the predictor values at lower indices tend to have a higher leverage (quantifier of influence of the observed predictor on model), i.e. are further away from the other observations leading to an excessive effect on the regression model.

8 (c)

In [59]:
```
ax = plt.subplots(figsize=(8,8))[1]
ax.scatter(results.fittedvalues, results.resid)
ax.set_xlabel('Fitted value')
ax.set_ylabel('Residual')
ax.axhline(0, c='k', ls='--');
```



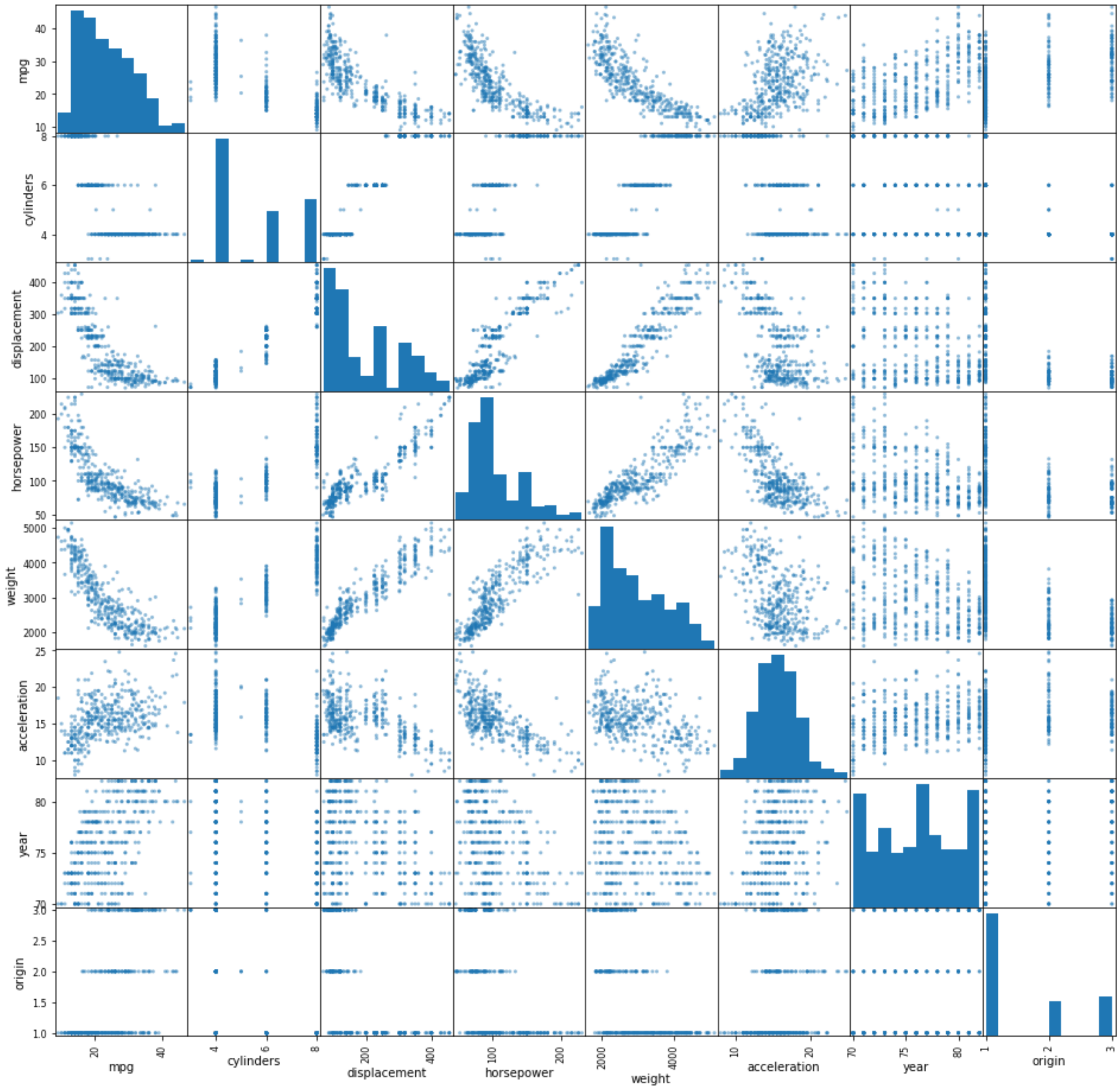
In [62]:
```
infl = results.get_influence()
ax = plt.subplots(figsize=(8,8))[1]
ax.scatter(np.arange(x.shape[0]), infl.hat_matrix_diag)
ax.set_xlabel('Index')
ax.set_ylabel('Leverage')
np.argmax(infl.hat_matrix_diag)
```



9. (a) Refer to Homework#1_Isha_Jain notebook

9 (a)

In [63]:
```
pd.plotting.scatter_matrix(Auto, figsize=(16,16));
```



(b) Refer to Homework#1_Isha_Jain notebook

9 (b)

In [64]:
```
new_Auto.corr(method ='pearson')
```

Out[64]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
mpg	1.000000	-0.777618	-0.805127	-0.778427	-0.832244	0.423329	0.580541	0.565209
cylinders	-0.777618	1.000000	0.950823	0.842983	0.897527	-0.504683	-0.345647	-0.568932
displacement	-0.805127	0.950823	1.000000	0.897257	0.932994	-0.543800	-0.369855	-0.614535
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.416361	-0.455171
weight	-0.832244	0.897527	0.932994	0.864538	1.000000	-0.416839	-0.309120	-0.585005
acceleration	0.423329	-0.504683	-0.543800	-0.689196	-0.416839	1.000000	0.290316	0.212746
year	0.580541	-0.345647	-0.369855	-0.416361	-0.309120	0.290316	1.000000	0.181528
origin	0.565209	-0.568932	-0.614535	-0.455171	-0.585005	0.212746	0.181528	1.000000

(c)i. From the anova result, all predictors considered in the model (Displacement, weight, year, cylinders, horsepower, acceleration and origin) appear to have a statistically significant relationship to the response since they have Pr>|F| values less than 0.05 implying a probability greater than 95% of a relation between the predictor and response and must hence be considered.

ii. Displacement, weight, year, cylinders, horsepower, acceleration and origin (all predictors considered in the model) appear to have a statistically significant relationship to the response since they have Pr>|F| values less than 0.05 implying a probability greater than 95% of a relation between the predictor and response (null hypothesis claiming value of coefficient corresponding to the predictor being equal to zero can be rejected).

iii. The coefficient for the year variable (0.7508) suggests that a unit increase in year would lead to an increase of 0.7508 in mpg (subject to the other predictors being constant).

9 (c)

In [137]:
```
import statsmodels.formula.api as smf
mod = smf.ols(formula='mpg ~ cylinders + displacement + horsepower+weight+acceleration+year+origin', data=new_Auto)

res = mod.fit()

print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.821
Model:                            OLS   Adj. R-squared:                  0.818
Method:                 Least Squares   F-statistic:                     252.4
Date:                Tue, 19 Sep 2023   Prob (F-statistic):          2.04e-139
Time:                        06:50:54   Log-Likelihood:                -1023.5
No. Observations:                 392   AIC:                             2063.
Df Residuals:                     384   BIC:                             2095.
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept      -17.2184      4.644     -3.707      0.000     -26.350      -8.087
cylinders       -0.4934      0.323     -1.526      0.128      -1.129       0.142
displacement     0.0199      0.008      2.647      0.008       0.005       0.035
horsepower      -0.0170      0.014     -1.230      0.220      -0.044       0.010
weight          -0.0065      0.001     -9.929      0.000      -0.008      -0.005
acceleration     0.0806      0.099      0.815      0.415      -0.114       0.275
year             0.7508      0.051     14.729      0.000       0.651       0.851
origin           1.4261      0.278      5.127      0.000       0.879       1.973
==============================================================================
Omnibus:                       31.906   Durbin-Watson:                   1.309
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.100
Skew:                           0.529   Prob(JB):                     2.95e-12
Kurtosis:                       4.460   Cond. No.                     8.59e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.59e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [138]:
```
anova_lm(res)
```

Out[138]:

	df	sum_sq	mean_sq	F	PR(>F)
cylinders	1.0	14403.083079	14403.083079	1300.683788	2.319511e-125
displacement	1.0	1073.344025	1073.344025	96.929329	1.530906e-20
horsepower	1.0	403.408069	403.408069	36.430140	3.731128e-09
weight	1.0	975.724953	975.724953	88.113748	5.544461e-19
acceleration	1.0	0.966071	0.966071	0.087242	7.678728e-01
year	1.0	2419.120249	2419.120249	218.460900	1.875281e-39
origin	1.0	291.134494	291.134494	26.291171	4.665681e-07
Residual	384.0	4252.212530	11.073470	NaN	NaN

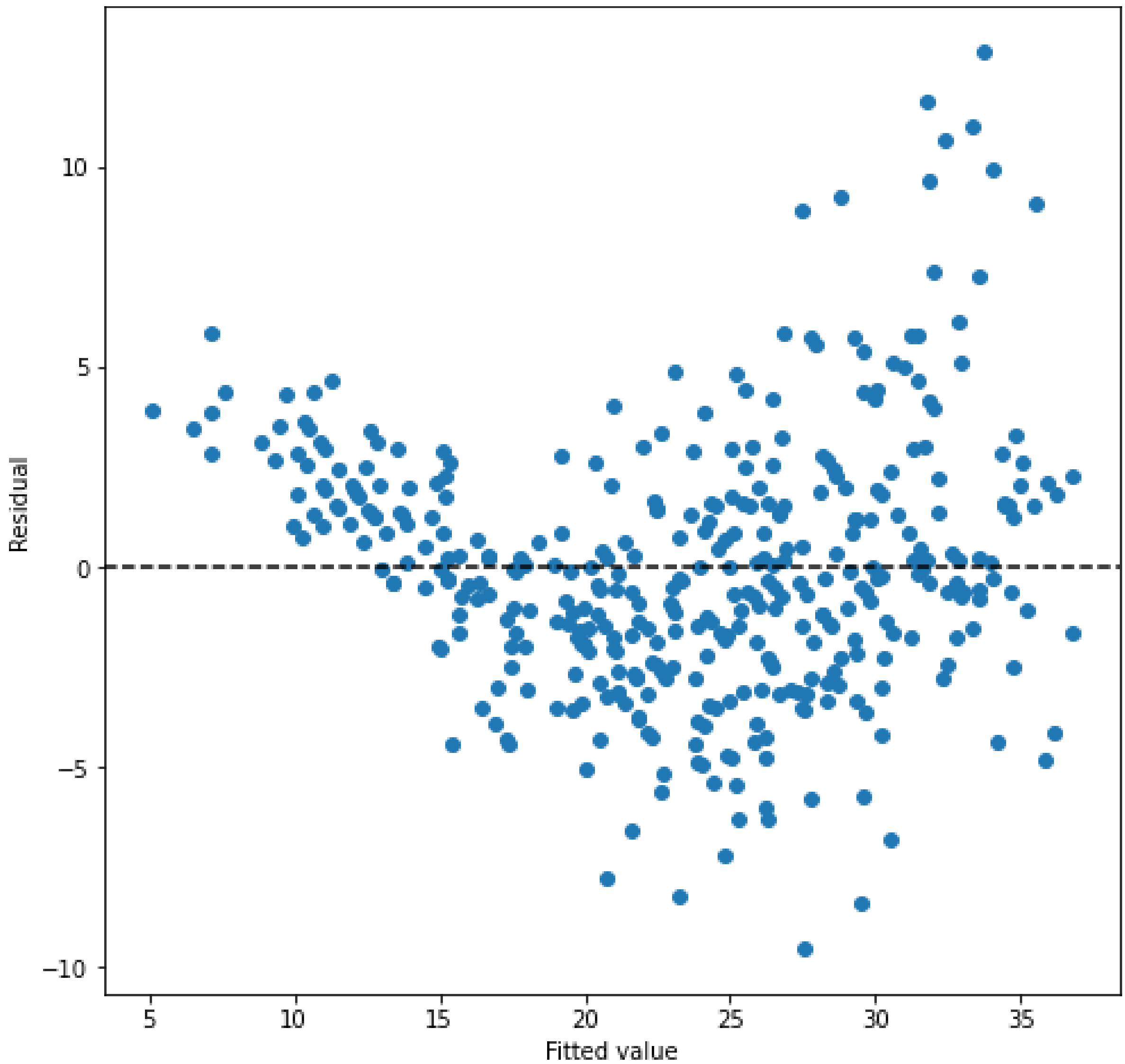
(d) The error plot suggests some non-linearity since there seems to be a trend whereas ideally the residual plot should look somewhat random. From the residual plot we can see that the residual goes progressively from a higher positive value to a negative value to again a positive value and has somewhat of a parabolic shape.

From the leverage plot we can see that 2 predictors at lower indices (between 0 and 50) have a higher leverage (quantifier of influence of the observed predictor on model), i.e. are further away from the other observations leading to an excessive effect on the regression model.

9 (d) ¶

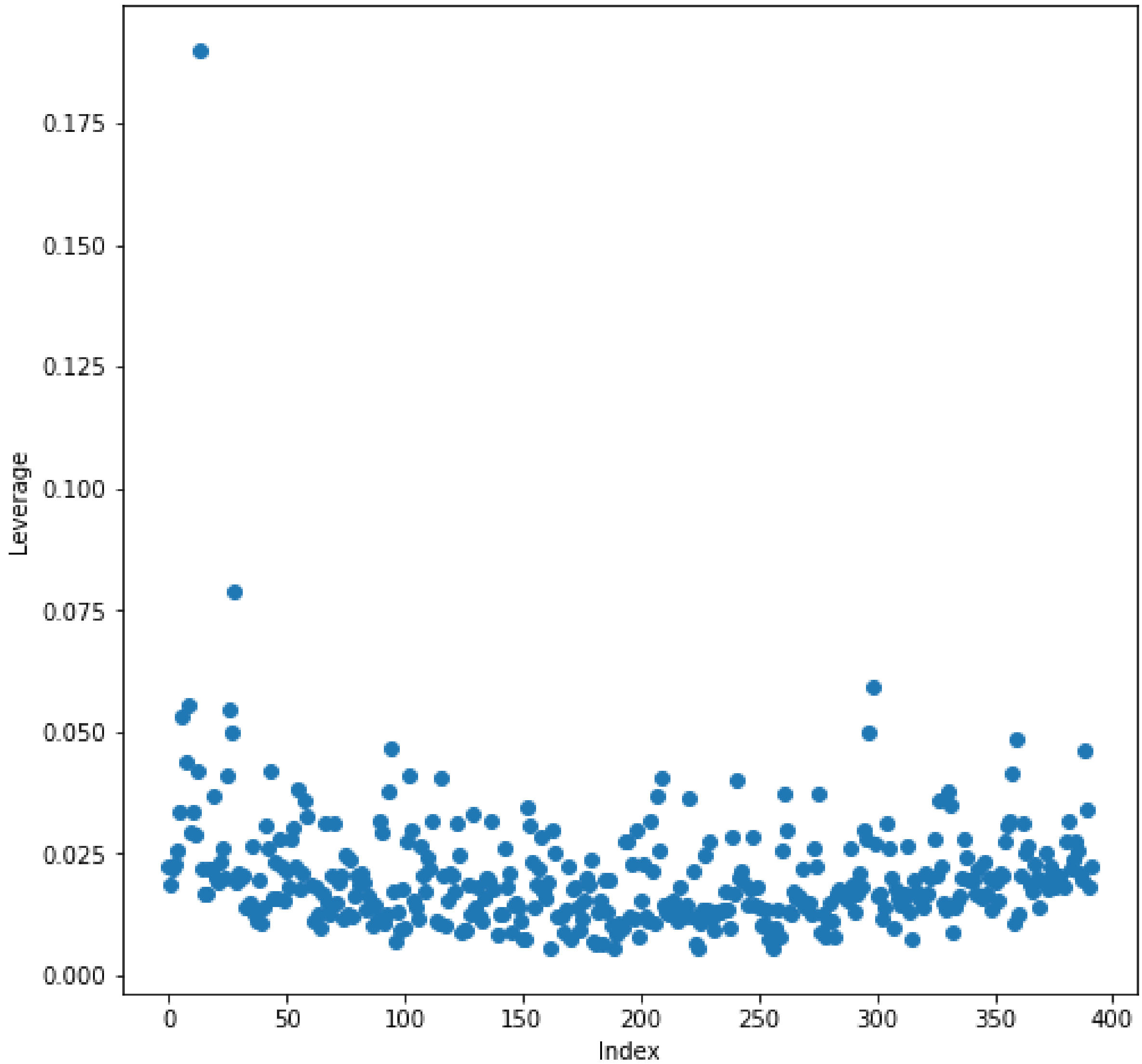
In [109]:
```
ax = plt.subplots(figsize=(8,8))[1]
ax.scatter(results.fittedvalues, results.resid)
ax.set_xlabel('Fitted value')
ax.set_ylabel('Residual')
ax.axhline(0, c='k', ls='--')
```

Out[109]: <matplotlib.lines.Line2D at 0x1d8de6d5370>



In [110]:
```
infl = results.get_influence()
ax = plt.subplots(figsize=(8,8))[1]
ax.scatter(np.arange(x.shape[0]), infl.hat_matrix_diag)
ax.set_xlabel('Index')
ax.set_ylabel('Leverage')
np.argmax(infl.hat_matrix_diag)
```

Out[110]: 13



(e) Referring to the interactions created in Homework#1_Isha_Jain notebook, we can conclude:

- In the first model created with cylinders, weight and cylinders*weight as the interaction term, the cylinder*weight as the interaction term has a P>|t| value of zero implying statistical significance, however the coefficient of the interaction term is 0.0011 which is relatively small implying a weak relation between the interaction term (cylinder*weight) and the response (mpg).
- In the second model created with cylinders, weight, acceleration, cylinders *weight, cylinders*acceleration and weight*acceleration, among the interaction terms, only the

cylinder*weight among the interaction term has a P>|t| value less than 0.05, implying statistical significance, however the coefficient of the interaction term is 0.001300 which is relatively small implying a weak relation between the interaction term (cylinder*weight) and the response (mpg). The other 2 interaction terms (cylinders*acceleration and weight*acceleration) have a P>|t| value greater than 0.05 which validates the null hypothesis (no relation between the interaction terms and response-mpg).

- In the third model created with cylinders, weight, acceleration, cylinders*acceleration, weight*acceleration, weight*acceleration*cylinders and cylinder*weight, the weight*acceleration is the only statistically significant interaction erm having a P>|t| value of 0.05, however the coefficient of the interaction term is 0.0012 which is relatively small implying a weak relation between the interaction term (cylinder*weight) and the response (mpg). The other 3 interaction terms (cylinders*acceleration, cylinders*weight and weight*acceleration*cylinders) have a P>|t| value greater than 0.05 which validates the null hypothesis (no relation between the interaction terms and response-mpg).

9 (e)

In [78]:
```
X = MS(['cylinders',
  'weight',
  ('cylinders', 'weight')]).fit_transform(new_Auto)
model_interaction1 = sm.OLS(y, X)
summarize(model_interaction1.fit())
```

Out[78]:

	coef	std err	t	P>|t|
intercept	65.3865	3.733	17.514	0.0
cylinders	-4.2098	0.724	-5.816	0.0
weight	-0.0128	0.001	-9.418	0.0
cylinders:weight	0.0011	0.000	5.226	0.0

In [79]:
```
X = MS(['cylinders',
  'weight','acceleration',
  ('cylinders', 'weight'),('cylinders', 'acceleration'),('weight', 'acceleration')]).fit_transform(new_Auto)
model_interaction2 = sm.OLS(y, X)
summarize(model_interaction2.fit())
```

Out[79]:

	coef	std err	t	P>|t|
intercept	63.220400	8.519	7.421	0.000
cylinders	-3.937700	1.337	-2.946	0.003
weight	-0.015700	0.004	-3.613	0.000
acceleration	0.300000	0.344	0.871	0.384
cylinders:weight	0.001300	0.000	4.821	0.000
cylinders:acceleration	-0.040600	0.091	-0.448	0.654
weight:acceleration	0.000085	0.000	0.418	0.676

In [80]:
```
X = MS(['cylinders',
  'weight','acceleration',
  ('cylinders', 'weight'),('cylinders', 'acceleration'),('weight', 'acceleration'),('weight', 'acceleration','cylinders')]).fit_t
model_interaction3 = sm.OLS(y, X)
summarize(model_interaction3.fit())
```

Out[80]:

	coef	std err	t	P>|t|
intercept	114.1277	27.665	4.125	0.000
cylinders	-13.0743	4.910	-2.663	0.008
weight	-0.0334	0.010	-3.292	0.001
acceleration	-3.0013	1.742	-1.723	0.086
cylinders:weight	0.0043	0.002	2.754	0.006
cylinders:acceleration	0.5837	0.335	1.741	0.082
weight:acceleration	0.0012	0.001	1.963	0.050
weight:acceleration:cylinders	-0.0002	0.000	-1.933	0.054

(f) On referring to Homework#1_Isha_Jain notebook:

- The first model has horsepower and weight as the predictors with mpg as the response. The predictor horsepower is having degree 2. The resulting model has all terms (intercept, horsepower, horsepower**2) being statistically significant. The values of the coefficient are displayed in the notebook.
- The second model the log of horsepower as the predictors with mpg as the response. The resulting model has the predictor {log(horsepower)} being statistically significant. The value of the coefficient is displayed in the notebook.
- The third model has horsepower raised to the power 0.5 as the predictor with mpg as the response. The resulting model has the predictor {square root(horsepower)} being statistically significant The values of the coefficient are displayed in the notebook.
- The fourth model has acceleration and weight as the predictors with mpg as the response. The predictor acceleration is having degree 3. The resulting model has the intercept, displacement and acceleration**2 being statistically significant. The values of the coefficient are displayed in the notebook.

From the Anova, we can see the first non-linear transformation has the lowest ssr(6201.609295) while the fourth model has the lowest df_resid (387). The Pr(>F) value is extremely small (less than 0.05) allowing us to reject the null hypothesis that the added complexity does not affect model performance and conclude that the fourth model is the best among the options.

9 (f)

In [83]:
```
X_nonlin1 = MS([poly('horsepower', degree=2),'weight']).fit_transform(new_Auto)
model_nonlin1 = sm.OLS(y, X_nonlin1)
results_nonlin1 = model_nonlin.fit()
summarize(results_nonlin1)
```

Out[83]:

	coef	std err	t	P>|t|
intercept	36.7952	1.529	24.069	0.0
poly(horsepower, degree=2)[0]	-55.0379	8.402	-6.551	0.0
poly(horsepower, degree=2)[1]	30.2436	4.296	7.040	0.0
weight	-0.0045	0.001	-8.809	0.0

In [90]:
```
X_log = np.log(new_Auto['horsepower'].values.reshape(-1,1))
#X_nonlin2 = MS([poly('horsepower', degree=math.log),'weight']).fit_transform(new_Auto
model_nonlin2 = sm.OLS(y,X_log )
results_nonlin2 = model_nonlin2.fit()
summarize(results_nonlin2)
```

Out[90]:

	coef	std err	t	P>|t|
x1	9.9574	0.204	48.897	0.0

In [91]:
```
x_pow_half = new_Auto.apply(lambda row: row.horsepower**0.5, axis =1 )
model_nonlin3 = sm.OLS(y,x_pow_half )
results_nonlin3 = model_nonlin3.fit()
summarize(results_nonlin3)
```

Out[91]:

	coef	std err	t	P>|t|
x1	7.1897	0.151	47.474	0.0

In [96]:
```
X_nonlin4 = MS([poly('acceleration', degree=3),'displacement']).fit_transform(new_Auto)
model_nonlin4 = sm.OLS(y, X_nonlin4)
results_nonlin4 = model_nonlin4.fit()
summarize(results_nonlin4)
```

Out[96]:

	coef	std err	t	P>|t|
intercept	36.1916	0.591	61.236	0.000
poly(acceleration, degree=3)[0]	-8.4362	5.519	-1.528	0.127
poly(acceleration, degree=3)[1]	22.2761	4.857	4.586	0.000
poly(acceleration, degree=3)[2]	-1.3544	4.541	-0.298	0.766
displacement	-0.0656	0.003	-23.389	0.000

In [115]:
```
anova_lm(results_nonlin1,results_nonlin2,results_nonlin3,results_nonlin4)
```

Out[115]:

	df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
0	388.0	6201.609295	0.0	NaN	NaN	NaN
1	391.0	33635.101846	-3.0	-27433.492551	445.735208	NaN
2	391.0	35378.002783	-0.0	-1742.900937	inf	NaN
3	387.0	7939.513134	4.0	27438.489649	334.362300	4.068276e-124