```
from pyspark.sql import SparkSession
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
import seaborn as sns
import sklearn
from sklearn.preprocessing import scale
import sklearn.linear_model as skl_lm
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
import statsmodels.formula.api as smf

df1 =
spark.read.format("csv").load("dbfs:/FileStore/shared_uploads/MichaelRocchio2023@u.northwestern.ed
u/boston.csv", header=True)

df=df1.toPandas()
train_var_fin=df

del train_var_fin['neighborhood']
for i in list(train_var_fin.columns):
  train_var_fin[i]=pd.to_numeric(train_var_fin[i])
y=train_var_fin['mv'].to_numpy()
del train_var_fin['mv']
X = train_var_fin.to_numpy()
X

Out[36]: array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 2.9600e+02, 1.5300e+01,
        4.9800e+00],
       [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 2.4200e+02, 1.7800e+01,
        9.1400e+00],
       [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 2.4200e+02, 1.7800e+01,
        4.0300e+00],
       ...,
       [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.7300e+02, 2.1000e+01,
        5.6400e+00],
       [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.7300e+02, 2.1000e+01,
        6.4800e+00],
       [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.7300e+02, 2.1000e+01,
        7.8800e+00]])

train_var_fin

Out[39]:
```

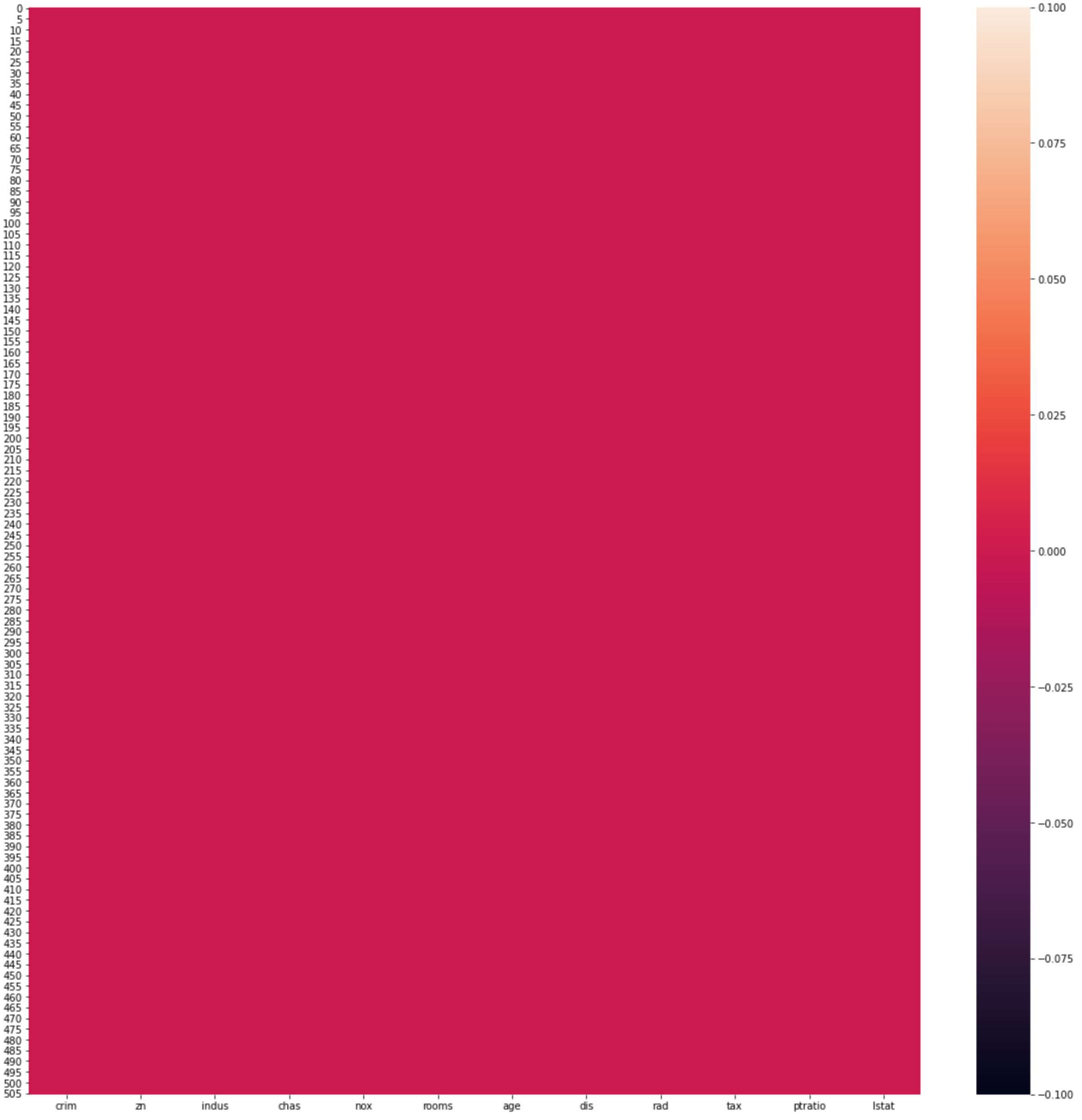
	crim	zn	indus	chas	nox	rooms	age	dis	rad	tax	ptratio	lstat
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	4.03
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	2.94
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	5.33
...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	9.67
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	9.08
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	5.64
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	6.48
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	7.88

506 rows × 12 columns

```
train_var_fin.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 12 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64  
 4   nox      506 non-null    float64
 5   rooms    506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64  
 9   tax      506 non-null    int64  
 10  ptratio  506 non-null    float64
 11  lstat    506 non-null    float64
dtypes: float64(9), int64(3)
memory usage: 47.6 KB
```

```
plt.figure(figsize=(20, 20))
sns.heatmap(train_var_fin.isnull())
plt.show()
```


0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
260
265
270
275
280
285
290
295
300
305
310
315
320
325
330
335
340
345
350
355
360
365
370
375
380
385
390
395
400
405
410
415
420
425
430
435
440
445
450
455
460
465
470
475
480
485
490
495
500
505
crim
zn
indus
chas
nox
rooms
age
dis
rad
tax
ptratio
lstat
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100

```
from sklearn.linear_model import ElasticNet
elastic_net = ElasticNet()
elastic_net.fit(X, y)

elastic_net.coef_
resulte = train_var_fin.mul(elastic_net.coef_, axis='columns')
resulte['beta_zero']=elastic_net.intercept_
resulte['elastic_house_price']=resulte.sum(axis=1)
resulte=resulte.reset_index()
resulte=resulte[['index', 'elastic_house_price']]
final_result=resulte
final_result['actual_house_price']=y
final_result
```

Out[49]:

	index	elastic_house_price	actual_house_price
0	0	31.054209	24.0
1	1	25.679992	21.6
2	2	29.966521	34.7
3	3	29.543117	33.4
4	4	27.984297	36.2
...	...	...	...
501	501	23.986619	22.4
502	502	24.346976	20.6
503	503	28.202926	23.9
504	504	27.170687	22.0
505	505	25.127949	19.0

506 rows × 3 columns

```
from sklearn.linear_model import Lasso

lasso_reg = Lasso(alpha=0.1)
lasso_reg.fit(X, y)

resultl = train_var_fin.mul(lasso_reg.coef_, axis='columns')
resultl['beta_zero']=lasso_reg.intercept_
resultl['lasso_house_price']=resultl.sum(axis=1)
resultl=resultl.reset_index()
resultl=resultl[['index', 'lasso_house_price']]
final_result=pd.merge(final_result, resultl, how='inner')
final_result=final_result[['index', 'elastic_house_price', 'lasso_house_price',
'actual_house_price']]
display(final_result)
```

	index	elastic_house_price	lasso_house_price	actual_house_price
1	0	31.05420939247718	30.772545131622906	24

2	1	25.679992347229245	24.715345884143503	21.6
3	2	29.966520774492583	30.645558644708444	34.7
4	3	29.543116990271223	29.609866327085072	33.4
5	4	27.984296993785424	28.61745690572944	36.2
6	5	27.54651435351851	26.12989097073054	28.7
7	6	24.157543272087274	22.816065345482013	22.9

Showing all 506 rows.

```
lasso_reg = Lasso(alpha=0.1)
lasso_reg.fit(X, y)

resultl = train_var_fin.mul(lasso_reg.coef_, axis='columns')
resultl['beta_zero']=lasso_reg.intercept_
resultl['lasso_house_price']=resultl.sum(axis=1)
resultl=resultl.reset_index()
resultl=resultl[['index', 'lasso_house_price']]
final_result=pd.merge(final_result, resultl, how='inner')
final_result=final_result[['index', 'elastic_house_price', 'lasso_house_price',
'actual_house_price']]
display(final_result)


lasso_mse = sklearn.metrics.mean_squared_error(final_result['actual_house_price'].to_numpy(),
final_result['lasso_house_price'].to_numpy())
elastic_mse = sklearn.metrics.mean_squared_error(final_result['actual_house_price'].to_numpy(),
final_result['elastic_house_price'].to_numpy())
print('lasso_mse: ', lasso_mse)
print('elastic_mse: ', elastic_mse)

lasso_mse:  23.29498433612626
elastic_mse:  26.617920352737336

X

Out[57]: array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 2.9600e+02, 1.5300e+01,
         4.9800e+00],
        [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 2.4200e+02, 1.7800e+01,
         9.1400e+00],
        [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 2.4200e+02, 1.7800e+01,
         4.0300e+00],
        ...,
        [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.7300e+02, 2.1000e+01,
         5.6400e+00],
        [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.7300e+02, 2.1000e+01,
         6.4800e+00],
        [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.7300e+02, 2.1000e+01,
         7.8800e+00]])
```

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score

tree_reg = DecisionTreeRegressor()
tree_reg.fit(X, y)
regressor = DecisionTreeRegressor(random_state=0)
cross_val_score(regressor, X, y)

Out[77]: array([ 0.62968671,  0.42436098,  0.62503004,  0.35857816, -1.45998688])

from sklearn.metrics import accuracy_score
y_pred = tree_reg.predict(X)

def reg_metrics(y, y_pred, X):
    from sklearn.metrics import mean_squared_error, r2_score
    rmse = np.sqrt(mean_squared_error(y,y_pred))
    r2 = r2_score(y,y_pred)
    n = y_pred.shape[0]
    k = X.shape[1]
    adj_r_sq = 1 - (1 - r2)*(n-1)/(n-1-k)
    print('RMSE: ', rmse)
    print('R2: ',r2)
    print('ADJUSTED R_SQR: ', adj_r_sq)
    return rmse, r2, adj_r_sq
reg_metrics(y, y_pred, X)

###OVER FITTED

RMSE:  0.0
R2:  1.0
ADJUSTED R_SQR:  1.0
Out[91]: (0.0, 1.0, 1.0)

predict=pd.DataFrame(
{
  'y': y,
  'y_pred': y_pred
})
predict

Out[74]:
```

	y	y_pred
0	24.0	24.0
1	21.6	21.6
2	34.7	34.7
3	33.4	33.4
4	36.2	36.2
...	...	...
501	22.4	22.4
502	20.6	20.6
503	23.9	23.9
504	22.0	22.0
505	19.0	19.0

506 rows × 2 columns

```
from sklearn import tree
text_representation = tree.export_text(tree_reg)
print(text_representation)
```

```
|--- feature_5 <= 6.84
|   |--- feature_11 <= 14.40
|   |   |--- feature_7 <= 1.38
|   |   |   |--- feature_0 <= 10.59
|   |   |   |   |--- value: [50.00]
|   |   |   |--- feature_0 >  10.59
|   |   |   |   |--- value: [27.90]
|   |   |--- feature_7 >  1.38
|   |   |   |--- feature_5 <= 6.54
|   |   |   |   |--- feature_11 <= 7.57
|   |   |   |   |   |--- feature_9 <= 222.50
|   |   |   |   |   |   |--- value: [28.70]
|   |   |   |   |   |--- feature_9 >  222.50
|   |   |   |   |   |   |--- feature_2 <= 5.48
|   |   |   |   |   |   |   |--- feature_10 <= 19.95
|   |   |   |   |   |   |   |   |--- feature_0 <= 0.04
|   |   |   |   |   |   |   |   |   |--- feature_11 <= 7.47
|   |   |   |   |   |   |   |   |   |   |--- feature_6 <= 29.95
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |--- feature_6 >  29.95
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
```

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
import numpy as np
def createList(r1, r2):
    return np.arange(r1, r2+1, 1)

depth_tries = createList(1,20)

for i in depth_tries:
  tree_reg2 = DecisionTreeRegressor(random_state=42, max_depth=i)
  tree_reg2.fit(X, y)
  y_pred = tree_reg2.predict(X)
  print('depth = ',i)
  reg_metrics(y, y_pred, X)
  print('')
```

```
depth =  1
RMSE:  6.781452485868371
R2:  0.45347079983071137
ADJUSTED R_SQR:  0.4401678578387611

depth =  2
RMSE:  4.9601852791465255
R2:  0.7076091873459209
ADJUSTED R_SQR:  0.7004921695936919

depth =  3
RMSE:  3.7808197781187625
R2:  0.8301209293555389
ADJUSTED R_SQR:  0.8259859418347812

depth =  4
RMSE:  3.05461079325196
R2:  0.8891132415208693
ADJUSTED R_SQR:  0.8864141723489636

depth =  5
```

```
tree_regfin = DecisionTreeRegressor(random_state=42, max_depth=5)
tree_regfin.fit(X, y)

final=pd.DataFrame({
  'tree_predict': tree_regfin.predict(X),
  'actual': y
})
final
```

Out[113]:

	tree_predict	actual
0	26.012500	24.0
1	21.011184	21.6
2	34.286364	34.7
3	30.290323	33.4
4	34.286364	36.2
...	...	...
501	26.012500	22.4
502	21.011184	20.6
503	30.290323	23.9
504	26.012500	22.0
505	21.011184	19.0

506 rows × 2 columns

```
from sklearn import tree
text_representation = tree.export_text(tree_regfin)
print(text_representation)
```

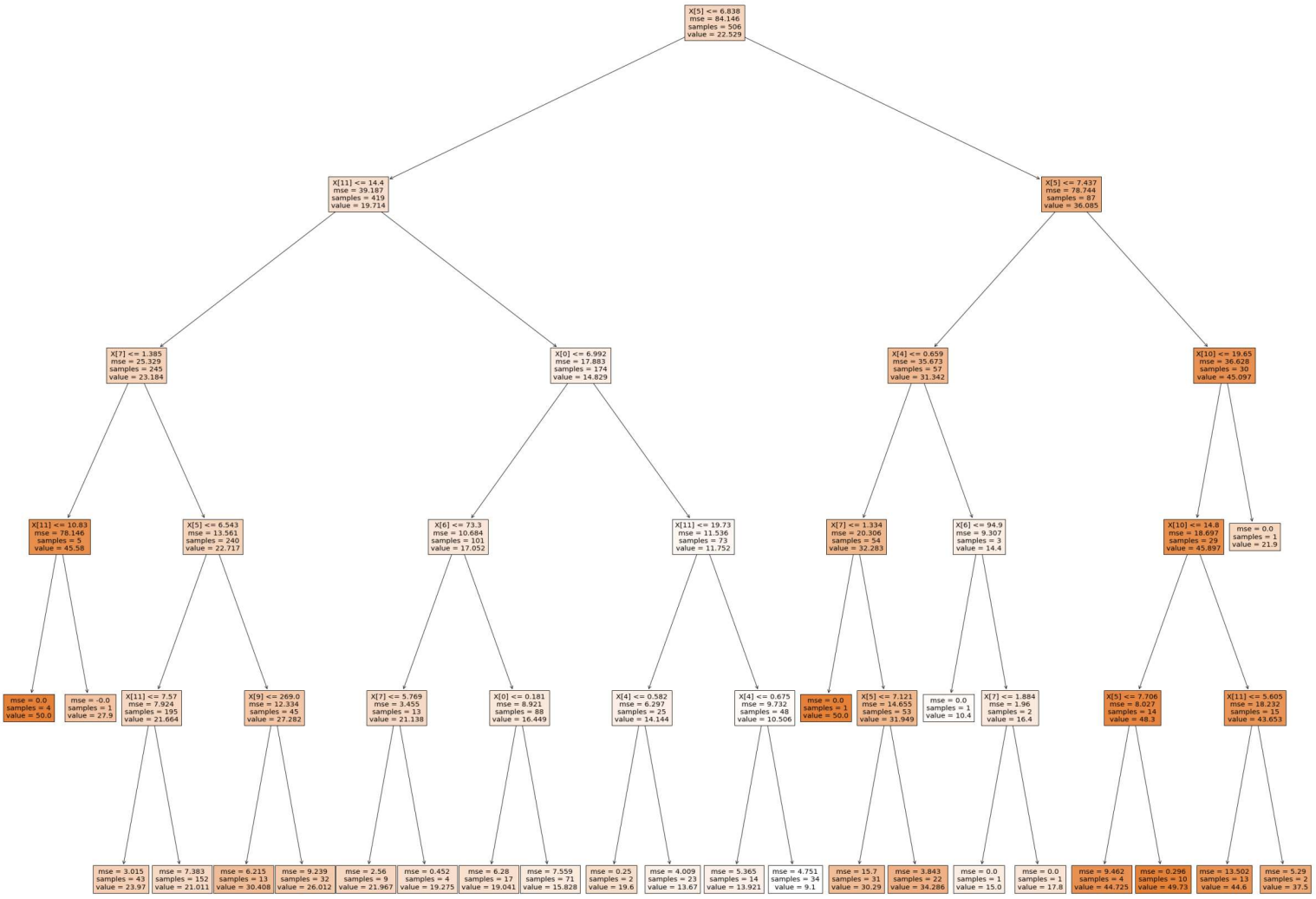
```
|--- feature_5 <= 6.84
|   |--- feature_11 <= 14.40
|   |   |--- feature_7 <= 1.38
|   |   |   |--- feature_11 <= 10.83
|   |   |   |   |--- value: [50.00]
|   |   |   |--- feature_11 >  10.83
|   |   |   |   |--- value: [27.90]
|   |   |--- feature_7 >  1.38
|   |   |   |--- feature_5 <= 6.54
|   |   |   |   |--- feature_11 <= 7.57
|   |   |   |   |   |--- value: [23.97]
|   |   |   |   |--- feature_11 >  7.57
|   |   |   |   |   |--- value: [21.01]
|   |   |   |--- feature_5 >  6.54
|   |   |   |   |--- feature_9 <= 269.00
|   |   |   |   |   |--- value: [30.41]
|   |   |   |   |--- feature_9 >  269.00
|   |   |   |   |   |--- value: [26.01]
|   |--- feature_11 >  14.40
|   |   |--- feature_0 <= 6.99
|   |   |   |--- feature_6 <= 73.30
```

```
print(tree_regfin.n_features_)
```

```
12
```

```
fig = plt.figure(figsize=(50,40))
_ = tree.plot_tree(tree_regfin,
                   filled=True)

### I AM SEEING SAMPLING ISSUES TOWARDS THE BOTTOM OF THE TREE AFTER THIS ASSESMENT I WOULD REDUCE THE DEPTH TO EITHER 3 OR 4
```



Show cell