1.a

```
[1] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.impute import SimpleImputer
    from sklearn.linear_model import Lasso
    from sklearn.metrics import mean_squared_error as MSE
    from sklearn.pipeline import make_pipeline
    from sklearn.preprocessing import StandardScaler
```

```
[2] data = pd.read_csv('/content/housing22.csv')
    data1 = data[(data['population'] < 20000) & (data['total_rooms'] < 35000)]
```

```
[3] X = data1[['housing_median_age', 'total_rooms', 'total_bedrooms',
           'population', 'households', 'median_income',
           '<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN']]
    y = data1[['median_house_value']]
```

```
imputer = SimpleImputer(strategy='median')
X_filled = imputer.fit_transform(X)
X_filled = pd.DataFrame(X_filled,columns = X.columns)
```

1.b

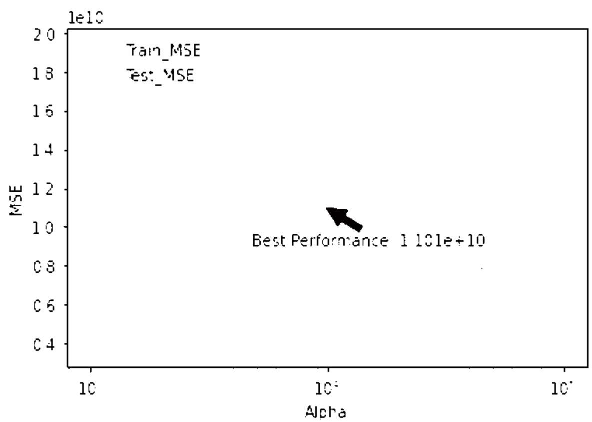
```
X_train = X_filled[:5000]
y_train = y[:5000]
X_test = X_filled[5000:]
y_test = y[5000:]
```

1.c

```
Bookeeping = []
for lambd in np.logspace(3,5,100):
  pip = make_pipeline(StandardScaler(),Lasso(alpha = lambd))
  pip.fit(X_train,y_train)
  train_preds = pip.predict(X_train)
  test_preds = pip.predict(X_test)
  train_mse = MSE(y_train,train_preds)
  test_mse = MSE(y_test,test_preds)
  record = [lambd,train_mse,test_mse] + list(pip[-1].coef_)
  Bookeeping.append(record)
```

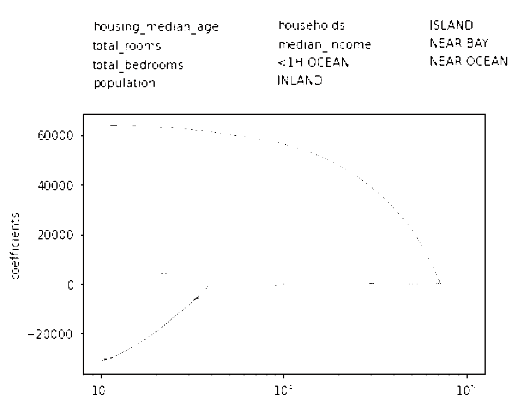
1.d

```python
df = pd.DataFrame(Bookeeping,columns = [
    'alpha','train_mse','test_mse','housing_median_age',
    'total_rooms', 'total_bedrooms','population',
    'households', 'median_income','<1H OCEAN', 'INLAND',
    'ISLAND', 'NEAR BAY', 'NEAR OCEAN'])
plt.plot(df['alpha'],df['train_mse'],label = 'Train_MSE')
plt.plot(df['alpha'],df['test_mse'],label = 'Test_MSE')
lowest = df.sort_values(by = 'test_mse').reset_index(drop = True).iloc[:1,:]
plt.annotate(s = f'Best Performance: 1.101e+10',
             xy = (lowest['alpha'],lowest['test_mse']),
             xytext = (lowest['alpha']-5000,lowest['test_mse'].values.item()-2000000000),
             arrowprops=dict(facecolor='black', shrink=0.05))
plt.xscale('log')
plt.xlabel('Alpha')
plt.ylabel('MSE')
plt.legend()
plt.show()
```



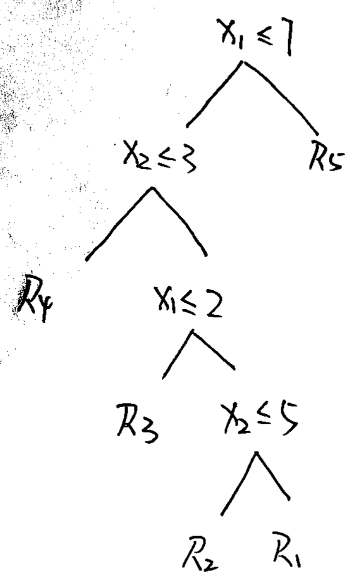
1.e

```python
for col in X.columns:
  plt.plot(df['alpha'],df[col],label = col)
plt.xscale('log')
plt.xlabel('Alpha')
plt.ylabel('coefficients')
plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.4),
          ncol=3, fancybox=True, shadow=True)
plt.show()
```

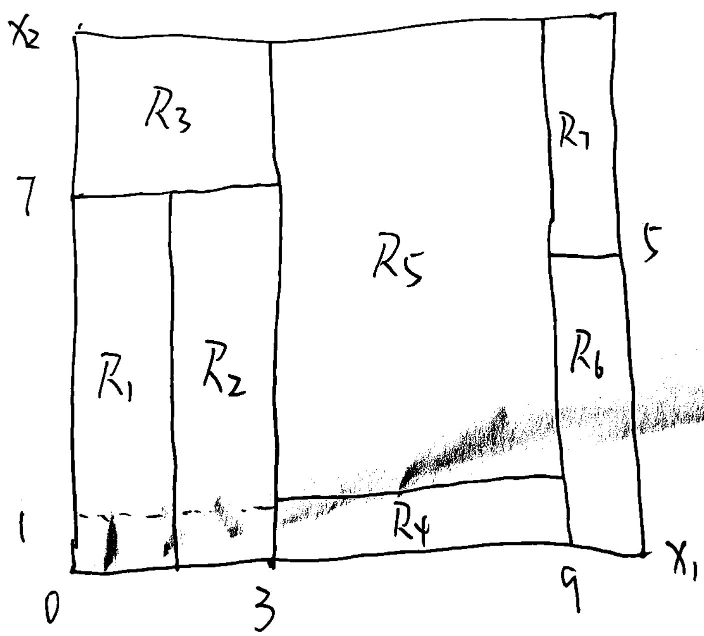


For the best test performance,I have total_bedrooms,median income,1h_ ocean, inland, near_ocean as non-zero coefficients.

2.ab


$x_1 \le 7$
$x_2 \le 3$
R_5
R_4
$x_1 \le 2$
R_3
$x_2 \le 5$
R_2
R_1

2.b.


x_2
7
1
0
3
9
x_1
5
R_3
R_7
R_5
R_1
R_2
R_6
R_4

3a

a.

$x_1 \le 1.5$

1 | 0, 0, 1, 1, 0, 1

3 misclassified

$x_1 \le 3.5$

1, 0, 0, 1 | 1, 0, 1

3 misclassified

$x_1 \le 6$

1, 0, 0, 1, 1, 0 | 1

3 misclassified

$x_2 \le 1.5$

1, 1 | 0, 0, 1, 0, 1

2 misclassified

$x_2 \le 3$

1, 0, 1 | 0, 1, 0, 1

3 misclassified

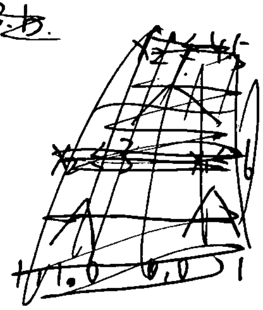
$x_2 \le 4.5$

1, 0, 0, 1, 0, 1 | 1

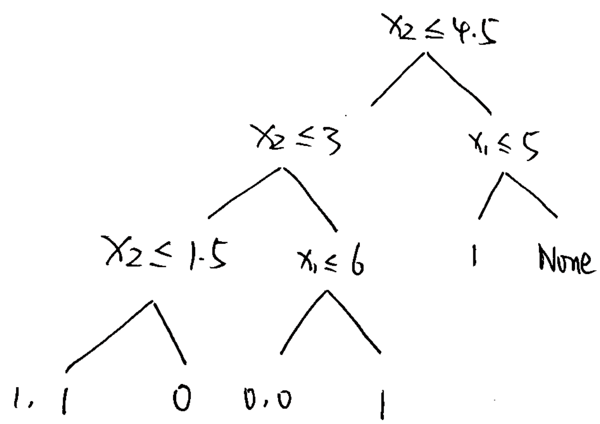
3 misclassified.

$x_2 \le 1.5$ is the best split

3.b.



3.b



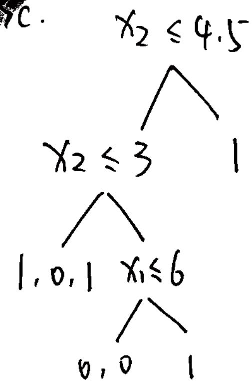
Droping $X_2 \le 1.5$, $X_1 \le 6$ will all resultes in 1 instance misclassified and one less split. Droping $X_1 \le 5$ will not influence accurracy but will have one less split, therefore $X_1 \le 5$ is the best bottom split to prune.

Drop $X_2 \le 1.5$ $\quad$ Loss $= 1 + 4\alpha = 1 + 4\alpha$

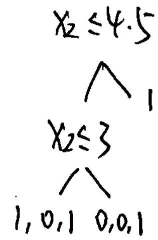
Drop $X_1 \le 6$ $\quad$ Loss $= 1 + 4\alpha$

Drop $X_1 \le 5$ $\quad$ Loss $= 4\alpha = 4\alpha$

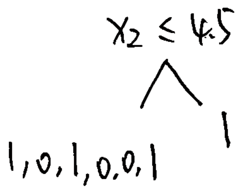
3c.



T_1: 1 misclassified



T_2 : 2 missclassified



T_3: 3 misclacified

1, 0, 1, 0, 0, 1 1 T_4: 3 misclasified

3d. ~~$T_1 = 1 + 3 = 4$~~

$T_1 = 1 + 3 \times 0.5 = 2.5$

$T_2 = 2 + 2 \times 0.5 = 3$

$T_3 = 3 + 1 \times 0.5 = 3.5$

$T_4 = 3 = 3$

~~T_2 or T_4~~

T_1

3e: $T_1 = 1 + 3 \times 2 = 7$

$T_2 = 2 + 2 \times 2 = 6$

$T_3 = 3 + 1 \times 2 = 5$

$T_4 = 3 = 3$

T_4