```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
import statsmodels.formula.api as smf
from sklearn.metrics import mean_squared_error
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
Auto = pd.read_csv('/content/gdrive/MyDrive/ISLR/Auto.csv',na_values=
Auto.head(2)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320

```
X = Auto.iloc[:,1:8]
Y = Auto.iloc[:,0]
```

```
lm = smf.ols('mpg~cylinders+displacement+horsepower+weight+accelerat
lm.summary().tables[1]
```

	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

```
lm_fit = LinearRegression()
lm_fit.fit(X,Y)
lm_fit.coef_
```

```
array([-0.49337632,  0.01989564, -0.01695114, -0.00647404,  0.08057584,
       0.75077268,  1.4261405 ])
```

```
regi = Ridge()
regi.fit(X,Y)
regi.coef_
```

```
array([-0.48766469,  0.01973747, -0.01682864, -0.00647685,  0.08058275,
       0.75062291,  1.41586743])
```

```
lasso = Lasso(alpha=1)
lasso.fit(X,Y)
lasso.coef_
```

```
array([-0.        ,  0.        , -0.00734394, -0.00646937,  0.        ,
       0.66308442,  0.        ])
```

```
alphas = 10**np.linspace(10,-2,100)*0.5
alphas
```

```
array([5.00000000e+09, 3.78231664e+09, 2.86118383e+09, 2.16438064e+09,
       1.63727458e+09, 1.23853818e+09, 9.36908711e+08, 7.08737081e+08,
       5.36133611e+08, 4.05565415e+08, 3.06795364e+08, 2.32079442e+08,
       1.75559587e+08, 1.32804389e+08, 1.00461650e+08, 7.59955541e+07,
       5.74878498e+07, 4.34874501e+07, 3.28966612e+07, 2.48851178e+07,
       1.88246790e+07, 1.42401793e+07, 1.07721735e+07, 8.14875417e+06,
       6.16423370e+06, 4.66301673e+06, 3.52740116e+06, 2.66834962e+06,
       2.01850863e+06, 1.52692775e+06, 1.15506485e+06, 8.73764200e+05,
       6.60970574e+05, 5.00000000e+05, 3.78231664e+05, 2.86118383e+05,
       2.16438064e+05, 1.63727458e+05, 1.23853818e+05, 9.36908711e+04,
       7.08737081e+04, 5.36133611e+04, 4.05565415e+04, 3.06795364e+04,
       2.32079442e+04, 1.75559587e+04, 1.32804389e+04, 1.00461650e+04,
       7.59955541e+03, 5.74878498e+03, 4.34874501e+03, 3.28966612e+03,
       2.48851178e+03, 1.88246790e+03, 1.42401793e+03, 1.07721735e+03,
       8.14875417e+02, 6.16423370e+02, 4.66301673e+02, 3.52740116e+02,
       2.66834962e+02, 2.01850863e+02, 1.52692775e+02, 1.15506485e+02,
       8.73764200e+01, 6.60970574e+01, 5.00000000e+01, 3.78231664e+01,
       2.86118383e+01, 2.16438064e+01, 1.63727458e+01, 1.23853818e+01,
       9.36908711e+00, 7.08737081e+00, 5.36133611e+00, 4.05565415e+00,
       3.06795364e+00, 2.32079442e+00, 1.75559587e+00, 1.32804389e+00,
       1.00461650e+00, 7.59955541e-01, 5.74878498e-01, 4.34874501e-01,
       3.28966612e-01, 2.48851178e-01, 1.88246790e-01, 1.42401793e-01,
       1.07721735e-01, 8.14875417e-02, 6.16423370e-02, 4.66301673e-02,
       3.52740116e-02, 2.66834962e-02, 2.01850863e-02, 1.52692775e-02,
       1.15506485e-02, 8.73764200e-03, 6.60970574e-03, 5.00000000e-03])
```

```
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
ridge = Ridge()
coefs = []

for a in alphas:
    ridge.set_params(alpha=a)
    ridge.fit(X, Y)
    coefs.append(ridge.coef_)

np.shape(coefs)
```

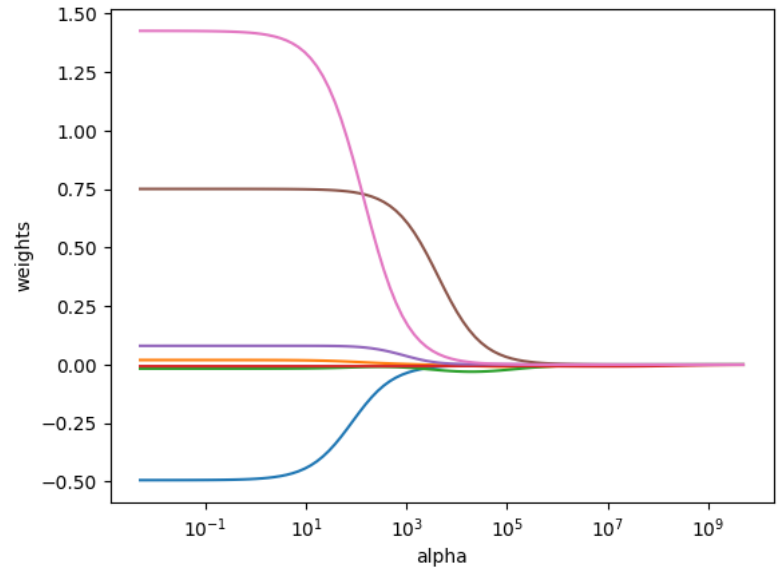
```
(100, 7)
```

```
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
ridge = Lasso()
coefs = []

for a in alphas:
    ridge.set_params(alpha=a)
    ridge.fit(X, Y)
    coefs.append(ridge.coef_)

np.shape(coefs)
```

```
(100, 7)
```

```
import matplotlib.pyplot as plt
ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('weights')
```

```
Text(0, 0.5, 'weights')
```




```
from sklearn.model_selection import train_test_split
# Split data into training and test sets


X_train, X_test , y_train, y_test = train_test_split(X, Y, test_size=
```

```
ridge2 = Ridge(alpha = 4)
ridge2.fit(X_train, y_train)             # Fit a ridge regression on
pred2 = ridge2.predict(X_test)           # Use this model to predict
print(pd.Series(ridge2.coef_, index = X.columns)) # Print coefficient
print(mean_squared_error(y_test, pred2))          # Calculate the te
```

```
cylinders      0.205844
displacement   -0.023716
horsepower     -0.012694
weight         -0.004830
acceleration   -0.011398
dtype: float64
20.932107857712253
```

```
ridge3 = Ridge(alpha = 10**10)
ridge3.fit(X_train, y_train)             # Fit a ridge regression on
pred3 = ridge3.predict(X_test)           # Use this model to predict
print(pd.Series(ridge3.coef_, index = X.columns)) # Print coefficient
print(mean_squared_error(y_test, pred3))          # Calculate the te
```

```
cylinders      -1.921595e-07
displacement   -1.207963e-05
horsepower     -4.141260e-06
weight         -1.030826e-04
acceleration    1.659249e-07
dtype: float64
63.721895414957395
```

```
ridgecv = RidgeCV(alphas = alphas, scoring = 'neg_mean_squared_error
ridgecv.fit(X_train, y_train)
ridgecv.alpha_
```

```
16.372745814388658
```

```
ridge4 = Ridge(alpha = ridgecv.alpha_)
ridge4.fit(X_train, y_train)
mean_squared_error(y_test, ridge4.predict(X_test))
```

```
12.633914106000457
```

```
ridge4.fit(X, Y)
pd.Series(ridge4.coef_, index = X.columns)
```

```
cylinders      -0.413218
displacement    0.017622
horsepower     -0.015192
weight         -0.006513
acceleration    0.080460
year            0.748272
origin          1.275031
dtype: float64
```

```
df = pd.read_csv('/content/gdrive/MyDrive/ISLR/Boston.csv',index_col=
df.head(2)
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lsta
1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.9	4.9
2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.1

```
df.isna().sum()
```

```
crim       0
zn         0
```

```
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
black      0
lstat      0
medv       0
dtype: int64
```

```
X = df.iloc[:,0:13]
Y = df.iloc[:,-1]
```

```
X
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black
1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90
2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90
3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83
4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63
5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90
...	...	...	...	...	...	...	...	...	...	...	...	...
502	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99
503	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90
504	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90
505	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45
506	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90

506 rows × 12 columns

```
from sklearn.model_selection import train_test_split
# Split data into training and test sets


X_train, X_test , y_train, y_test = train_test_split(X, Y, test_size=

alphas = 10**np.linspace(10,-2,100)*0.5


lassocv = LassoCV(alphas = alphas)
lassocv.fit(X_train, y_train)
lassocv.alpha_
```

```
0.005
```

```
ridgecv = RidgeCV(alphas = alphas, scoring = 'neg_mean_squared_error
ridgecv.fit(X_train, y_train)
ridgecv.alpha_
```

```
0.0466301673441609
```

```
Lasso1 = Lasso(alpha = lassocv.alpha_)
Lasso1.fit(X_train, y_train)
```

```
mean_squared_error(y_test, Lasso1.predict(X_test))
```

```
18.55336719429718
```

```
redgi1 = Ridge(alpha = ridgecv.alpha_)
redgi1.fit(X_train, y_train)
mean_squared_error(y_test, redgi1.predict(X_test))
```

```
18.5204959732037
```

```
Lasso1.fit(X, Y)
pd.Series(Lasso1.coef_, index = X.columns)
```

```
crim       -0.107119
zn          0.046640
indus       0.013518
chas        2.595461
nox       -16.080464
rm          3.812029
age        -0.000563
dis        -1.448856
```