Import Data

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
```

In [2]:
```
data = pd.read_csv('data.csv')
```

In [3]:
```
data.head()
#Age: 나이
#Experience: 경력
#Income: 수입
#Family: 가족단위
#CCAvg: 월 카드 사용량
```

Out[3]:

	Age	Experience	Income	Family	CCAvg
0	25	1	49	4	1.6
1	45	19	34	3	1.5
2	39	15	11	1	1.0
3	35	9	100	1	2.7
4	35	8	45	4	1.0

In [4]:
```
#결측치 확인(없음)
data.isnull().sum()
```

Out[4]:
```
Age           0
Experience    0
Income        0
Family        0
CCAvg         0
dtype: int64
```

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2500 entries, 0 to 2499
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Age         2500 non-null   int64
 1   Experience  2500 non-null   int64
 2   Income      2500 non-null   int64
 3   Family       2500 non-null   int64
 4   CCAvg        2500 non-null   float64
dtypes: float64(1), int64(4)
memory usage: 97.8 KB
```

In [6]:
```
#Y = 수입, X = 나이, 경력, 가족단위, 월 카드 사용량
y = data['Income']
X = data.drop(['Income'], axis = 1)
x_train, x_test, y_train, y_test = train_test_split(X, y, train_size =
0.7, random_state = 1000)
#7:3으로 train-test split
```

Linear Regression

In [7]:
```
reg = LinearRegression()
results1 = reg.fit(x_train, y_train)
```

In [8]:
```
reg.coef_
```

Out[8]:
```
array([-3.07793956,  2.89401562, -3.37220023, 16.09065086])
```

Ridge Regression

In [9]:
```
#alpha, 즉 lambda가 0일 때에는 LSE와 같은 값이 나온다!
rreg = Ridge(alpha = 0) # alpha = Lambda
rreg.fit(x_train, y_train)
```

Out[9]:
```
Ridge(alpha=0)
```

In [10]:
```
rreg.coef_
```

Out[10]:
```
array([-3.07793956,  2.89401562, -3.37220023, 16.09065086])
```

In [11]:
```
#alpha를 -3부터 3까지 log를 취해 7개 생성
alpha = np.logspace(-3,3,7)
alpha
```

Out[11]:
```
array([1.e-03, 1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03])
```

In [12]:
```
df = []
acc_table = []


for i, a in enumerate(alpha):
        rreg = Ridge(alpha=a).fit(x_train, y_train)
        df.append(pd.Series(np.hstack([rreg.intercept_, rreg.coef_])))
        pred_y = rreg.predict(x_test)

df_ridge = pd.DataFrame(df,index = alpha).T
df_ridge
#가로축은 lambda 값, 세로축은 coefficients. lambda가 커질수록 더 shrink
```

Out[12]:

	0.001	0.010	0.100	1.000	10.000	100.000	1000.000
0	132.296084	132.295649	132.291303	132.247877	131.817002	127.823048	105.704966
1	-3.077937	-3.077919	-3.077732	-3.075864	-3.057321	-2.884607	-1.883048
2	2.894014	2.893995	2.893806	2.891920	2.873198	2.698718	1.681685
3	-3.372199	-3.372192	-3.372122	-3.371422	-3.364435	-3.295822	-2.731156
4	16.090648	16.090622	16.090363	16.087768	16.061871	15.807207	13.634454

Lasso Regression

In [13]:

```
lreg = Lasso(alpha = 0 ) # alpha = Lambda
lreg.fit(x_train, y_train)
```

```
<ipython-input-13-b25edddec866>:2: UserWarning: With alpha=0, this algorithm does not
converge well. You are advised to use the LinearRegression estimator
  lreg.fit(x_train, y_train)
C:₩Users₩smile₩anaconda3₩lib₩site-packages₩sklearn₩linear_model₩_coordinate_descent.p
y:530: UserWarning: Coordinate descent with no regularization may lead to unexpected r
esults and is discouraged.
  model = cd_fast.enet_coordinate_descent(
C:₩Users₩smile₩anaconda3₩lib₩site-packages₩sklearn₩linear_model₩_coordinate_descent.p
y:530: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations. Duality gap: 1105890.1320882086, tolerance: 373.84840920000005
  model = cd_fast.enet_coordinate_descent(
```

Out[13]:

```
Lasso(alpha=0)
```

In [14]:

```
lreg.coef_
```

Out[14]:

```
array([-3.07790231,  2.8939786 , -3.37220244, 16.09065156])
```

In [15]:

```
df = []
acc_table = []

for i, a in enumerate(alpha):
        lreg = Lasso(alpha=a).fit(x_train, y_train)
        df.append(pd.Series(np.hstack([lreg.intercept_, lreg.coef_])))
        pred_y = lreg.predict(x_test)
```

```
df_lasso = pd.DataFrame(df,index = alpha).T
df_lasso
#Ridge와 유사한 방식. lambda가 커질수록 shrink. 그러나 Ridge와는 달리
Lasso는 0까지 shrink.
```

```
C:\Users\smile\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.p
y:530: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations. Duality gap: 3094.379392363131, tolerance: 373.84840920000005
  model = cd_fast.enet_coordinate_descent(
```

Out[15]:

	0.001	0.010	0.100	1.000	10.000	100.000	1000.000
0	132.261976	131.960877	128.945930	98.937749	54.569493	73.876	73.876
1	-3.076625	-3.065044	-2.949074	-1.794975	-0.134206	-0.000	-0.000
2	2.892703	2.881139	2.765340	1.612913	-0.000000	-0.000	-0.000
3	-3.371595	-3.366136	-3.311548	-2.765340	-0.000000	-0.000	-0.000
4	16.090400	16.088142	16.065558	15.839618	13.184919	0.000	0.000

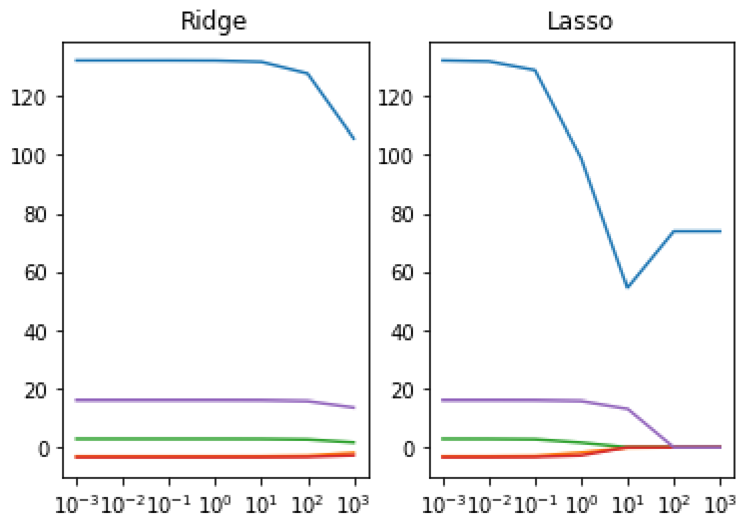
In [16]:

```
import matplotlib.pyplot as plt

ax1 = plt.subplot(121)
plt.semilogx(df_ridge.T)
plt.xticks(alpha)
plt.title("Ridge")

ax2 = plt.subplot(122)
plt.semilogx(df_lasso.T)
plt.xticks(alpha)
plt.title("Lasso")

plt.show()
```



```
In [ ]:
```