Today Concepts

- Polynomial Regression with Multiple Features
- Ridge Regression
- Lasso Regression

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv("https://raw.githubusercontent.com/AP-State-Skill-Developme
```

In [3]:
```
df
```

Out[3]:

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION
0	2014	ACURA	ILX	COMPACT	2.0	4	AS5
1	2014	ACURA	ILX	COMPACT	2.4	4	M6
2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7
3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6
4	2014	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6
...	...	...	...	...	...	...	...
1062	2014	VOLVO	XC60 AWD	SUV - SMALL	3.0	6	AS6
1063	2014	VOLVO	XC60 AWD	SUV - SMALL	3.2	6	AS6
1064	2014	VOLVO	XC70 AWD	SUV - SMALL	3.0	6	AS6
1065	2014	VOLVO	XC70 AWD	SUV - SMALL	3.2	6	AS6
1066	2014	VOLVO	XC90 AWD	SUV - STANDARD	3.2	6	AS6

1067 rows × 13 columns

In [4]:
```
df.columns
```

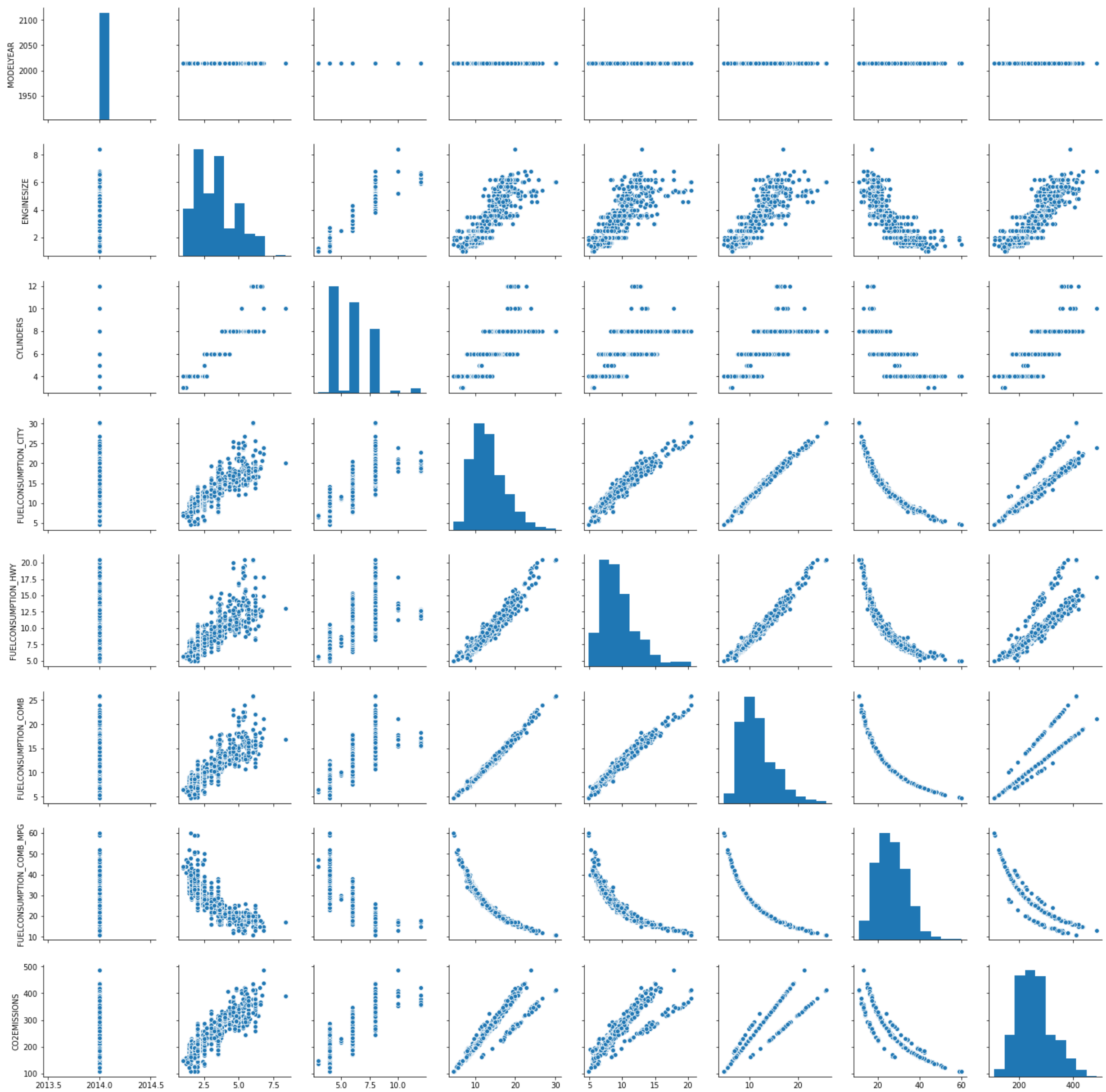
Out[4]:
```
Index(['MODELYEAR', 'MAKE', 'MODEL', 'VEHICLECLASS', 'ENGINESIZE', 'CYLINDERS',
       'TRANSMISSION', 'FUELTYPE', 'FUELCONSUMPTION_CITY',
       'FUELCONSUMPTION_HWY', 'FUELCONSUMPTION_COMB',
       'FUELCONSUMPTION_COMB_MPG', 'CO2EMISSIONS'],
      dtype='object')
```

In [5]:
```
df['CO2EMISSIONS'].head(10)
```

Out[5]:
```
0    196
1    221
2    136
3    255
4    244
5    230
6    232
7    255
8    267
9    212
Name: CO2EMISSIONS, dtype: int64
```

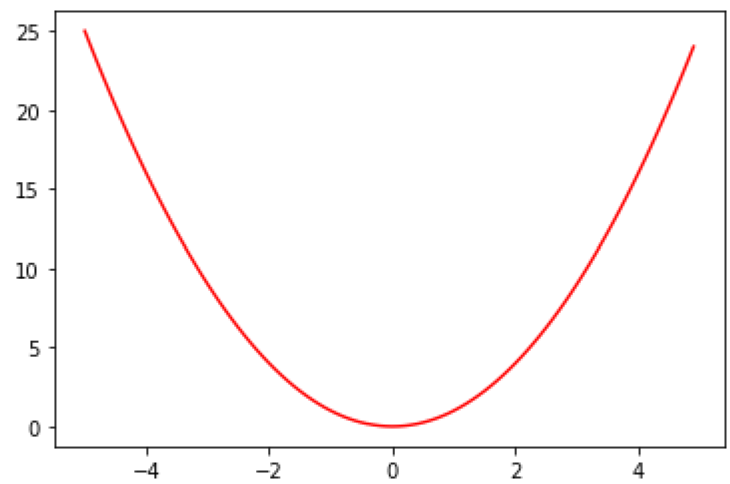
In [6]:
```
import seaborn as sns
sns.pairplot(df)
```

Out[6]: <seaborn.axisgrid.PairGrid at 0x1f7b66dd1c0>



In [7]:
```python
x = np.arange(-5,5,0.1)
plt.plot(x,x**2,c='r')
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x1f7bd7ffdc0>]
```



In [21]:
```python
X = df[['FUELCONSUMPTION_COMB_MPG']].values.reshape(-1,1)
```

In [22]:
```python
X
```

Out[22]:
```
array([[33],
       [29],
       [48],
       ...,
       [24],
       [25],
       [22]], dtype=int64)
```

In [23]:
```python
y = df['CO2EMISSIONS']
```

In [24]:
```python
from sklearn.preprocessing import PolynomialFeatures
```

In [25]:
```python
poly = PolynomialFeatures(degree=2)
```

In [26]:
```python
X_poly = poly.fit_transform(X)
```

In [27]:
```python
X_poly
```

Out[27]:
```
array([[1.000e+00, 3.300e+01, 1.089e+03],
       [1.000e+00, 2.900e+01, 8.410e+02],
       [1.000e+00, 4.800e+01, 2.304e+03],
       ...,
       [1.000e+00, 2.400e+01, 5.760e+02],
       [1.000e+00, 2.500e+01, 6.250e+02],
       [1.000e+00, 2.200e+01, 4.840e+02]])
```

In [28]:
```
from sklearn.linear_model import LinearRegression
```

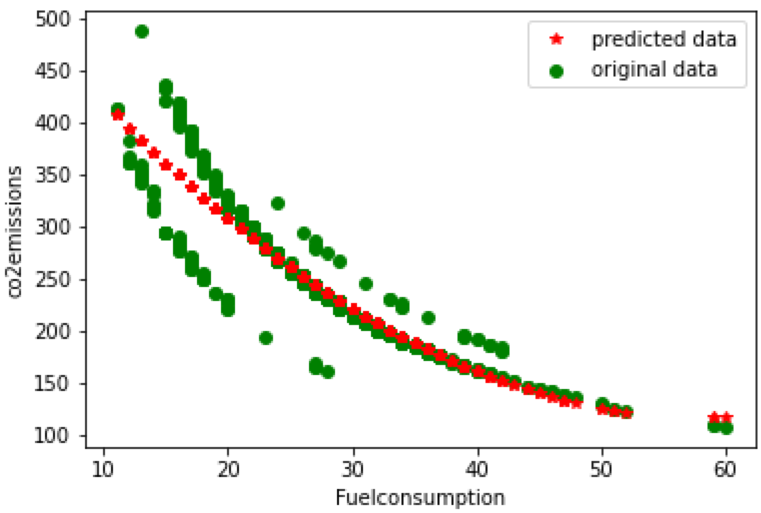
In [29]:
```
model = LinearRegression()
```

In [30]:
```
model.fit(X_poly,y)
```

Out[30]: LinearRegression()

In [31]:
```
y_pred = model.predict(X_poly)
```

In [33]:
```
plt.scatter(X,y,label = 'original data',c='g')
plt.plot(X,y_pred,'r*',label = 'predicted data')
plt.xlabel('Fuelconsumption')
plt.ylabel('co2emissions')
plt.legend()
plt.show()
```



Ridge Regression

In [34]:
```
m=100
x = np.random.rand(m,1)
y = 0.5*x**2+x+2+np.random.randn(m,1)
```

In [35]:
```
print(x)
```

```
[[0.1275804 ]
 [0.56379826]
 [0.92273745]
 [0.63536456]
 [0.52386528]
 [0.10697579]
 [0.55070548]
 [0.73659658]
 [0.44002359]
 [0.45982298]
 [0.79428537]
 [0.9116974 ]
 [0.5458613 ]
 [0.59877542]
 [0.5374324 ]
 [0.43890212]
 [0.93233761]
 [0.47384011]
 [0.59547886]
 [0.8656434 ]
 [0.87900426]
 [0.69395659]
 [0.1980226 ]
 [0.22291654]
 [0.40345386]
 [0.35787618]
 [0.96015506]
 [0.81998997]
 [0.83728784]
 [0.38720258]
 [0.32799929]
 [0.84188215]
 [0.55821378]
 [0.8810462 ]
 [0.47645051]
 [0.07594555]
 [0.95640042]
 [0.39512064]
 [0.68693279]
 [0.63075027]
 [0.93633961]
 [0.98952706]
 [0.00553965]
 [0.30891415]
 [0.30992322]
 [0.06333435]
 [0.81849027]
 [0.3630223 ]
 [0.58041542]
 [0.13714473]
 [0.36788261]
 [0.14730769]
 [0.86602458]
 [0.36858089]
```

```
 [0.6020984 ]
 [0.72639997]
 [0.05977751]
 [0.41982494]
 [0.40627689]
 [0.49813486]
 [0.98589343]
 [0.88899879]
 [0.99063333]
 [0.18343994]
 [0.30986029]
 [0.9045653 ]
 [0.90431475]
 [0.23278418]
 [0.66641164]
 [0.51408005]
 [0.27970614]
 [0.96444049]
 [0.76442354]
 [0.39315697]
 [0.00757536]
 [0.38582148]
 [0.92373111]
 [0.68696396]
 [0.67252587]
 [0.07709121]
 [0.37533877]
 [0.82975991]
 [0.89445485]
 [0.08411864]
 [0.58147322]
 [0.57851751]
 [0.98477089]
 [0.36497697]
 [0.7119468 ]
 [0.64416844]
 [0.76898914]
 [0.66601207]
 [0.16657521]
 [0.33766315]
 [0.42319038]
 [0.30438249]
 [0.11250165]
 [0.8090909 ]
 [0.95969813]
 [0.62076427]]
```

In [36]:
```
print(y)
```

```
[[1.77164985]
 [2.61666142]
 [4.88594726]
 [3.79319673]
 [0.71991947]
 [2.30618844]
 [3.89221617]
 [2.85840739]
 [3.24587906]
 [2.69005347]
 [4.13484452]
 [3.70631309]
 [2.81297893]
 [2.93964509]
 [2.63462184]
 [1.58566038]
 [2.66781196]
 [2.41157247]
 [2.29899402]
 [3.28415413]
 [0.51146693]
 [3.0197024 ]
 [4.40095372]
 [2.47996699]
 [2.47611495]
 [2.06161329]
 [3.43248565]
 [2.7108503 ]
 [4.24002009]
 [2.10475234]
 [2.99275358]
 [2.02793172]
 [1.33056952]
 [1.54219218]
 [3.52025314]
 [2.82511037]
 [4.73017684]
 [3.1844743 ]
 [2.55899585]
 [2.42832857]
 [1.978807  ]
 [3.30621499]
 [1.13086522]
 [3.22213357]
 [1.95825311]
 [2.60019126]
 [3.15595027]
 [1.53692436]
 [2.84056513]
 [1.78002929]
 [1.51350938]
 [2.47172943]
 [0.55894746]
 [1.99632419]
```

```
[4.21242025]
[2.14576266]
[3.94756345]
[2.83364223]
[2.01844412]
[1.75315918]
[3.64138443]
[2.57916464]
[4.37832628]
[0.56120929]
[5.17711629]
[2.44143702]
[3.46837417]
[2.47761714]
[2.68283687]
[2.89336637]
[3.30306414]
[3.08511995]
[3.47874408]
[1.43744338]
[1.69219673]
[0.61281211]
[2.56663093]
[2.93496943]
[2.57310399]
[2.06096387]
[2.86520124]
[4.01524309]
[1.86249689]
[1.37929831]
[2.64952666]
[1.39850018]
[2.80246834]
[1.46213683]
[3.57483405]
[2.71251131]
[1.51342583]
[2.69844427]
[0.72210201]
[2.43477925]
[1.96750145]
[3.37074469]
[3.76013114]
[3.18557085]
[2.42767396]
[3.01974994]]
```

In [37]:
```
x.shape
```

Out[37]: (100, 1)

In [38]:
```
y.shape
```

Out[38]: (100, 1)

In [39]:
```
from sklearn.linear_model import Ridge
rd = Ridge()
```

In [40]:
```
x[1]
```

Out[40]: array([0.56379826])

In [41]:
```
np.ndim(x)
```

Out[41]: 2

In [42]:
```
rd.fit(x,y)
```

Out[42]: Ridge()

In [44]:
```
rd.score(x,y)*100
```

Out[44]: 6.989654113784094

In [46]:
```
rd.predict(x[[1]])
```

Out[46]: array([[2.63566452]])

In [47]:
```
from sklearn.datasets import load_boston
```

In [49]:
```
boston = load_boston()
```

In [50]:
```
boston.keys()
```

Out[50]: dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])

In [51]:
```
df = pd.DataFrame(boston.data,columns = boston.feature_names)
```

In [52]:
```
df.head()
```

Out[52]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

In [53]:
```
df.columns
```

Out[53]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT'],
      dtype='object')
```

In [54]:
```
boston.target
```

Out[54]:
```
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
       18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
       15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
       13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
       21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
       35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
       19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
       20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
       23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
       33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
       21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
       20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
       23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 13.4,
       15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3, 19.4,
       17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. , 22.7,
       25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6, 29.4,
       23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 50. ,
       32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3, 30.3,
       34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5, 24.4,
       20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 23. ,
       26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5, 24.3,
       31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. , 20.1,
       22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8, 29.6,
       42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 31. ,
       36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2, 32.4,
       32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2, 22. ,
       20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6, 27.1,
       20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4, 28.2,
       22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8, 23.1,
       21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3, 22.6,
       19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. , 18.7,
       32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9, 24.1,
       18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9, 20.8,
       16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. , 13.8,
       13.8, 15. , 13.9, 13.3, 13.1, 10.2, 10.4, 10.9, 11.3, 12.3,  8.8,
        7.2, 10.5,  7.4, 10.2, 11.5, 15.1, 23.2,  9.7, 13.8, 12.7, 13.1,
       12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. , 11.9,
       27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5, 10.4,
        8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9, 11. ,
        9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4, 12.8,
       10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. , 13.4,
       15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4, 17.7,
       19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6, 23.2,
       29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. , 21.8,
       20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8, 24.5,
       23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 11.9])
```

In [55]:
```
df['LandPrice']=boston.target
```

In [56]:
```
df.head()
```

Out[56]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

In [57]:
```
df.isnull().sum()
```

Out[57]:
```
CRIM         0
ZN           0
INDUS        0
CHAS         0
NOX          0
RM           0
AGE          0
DIS          0
RAD          0
TAX          0
PTRATIO      0
B            0
LSTAT        0
LandPrice    0
dtype: int64
```

In [58]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 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    float64
 4   NOX        506 non-null    float64
 5   RM         506 non-null    float64
 6   AGE        506 non-null    float64
 7   DIS        506 non-null    float64
 8   RAD        506 non-null    float64
 9   TAX        506 non-null    float64
 10  PTRATIO    506 non-null    float64
 11  B          506 non-null    float64
 12  LSTAT      506 non-null    float64
 13  LandPrice  506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

In [59]:
```
X = df.iloc[:,0:-1]
```

In [60]:
```
y=df['LandPrice']
```

In [61]:
```
from sklearn.linear_model import LinearRegression
linear = LinearRegression()
```

In [62]:
```
linear.fit(X,y)
```

Out[62]:
```
LinearRegression()
```

In [63]:
```
from sklearn.model_selection import train_test_split
```

In [64]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size = 0.75)
```

In [65]:
```
linear.score(X_train,y_train)
```

Out[65]:
```
0.7636799441572235
```

In [66]:
```
linear.score(X_test,y_test)
```

Out[66]:
```
0.6604205887452774
```

In [67]:
```
from sklearn.linear_model import Ridge
```

In [78]:
```
rd = Ridge(alpha = 0.1)

```

In [79]:
```
rd.fit(X_train,y_train)
```

Out[79]:
```
Ridge(alpha=0.1)
```

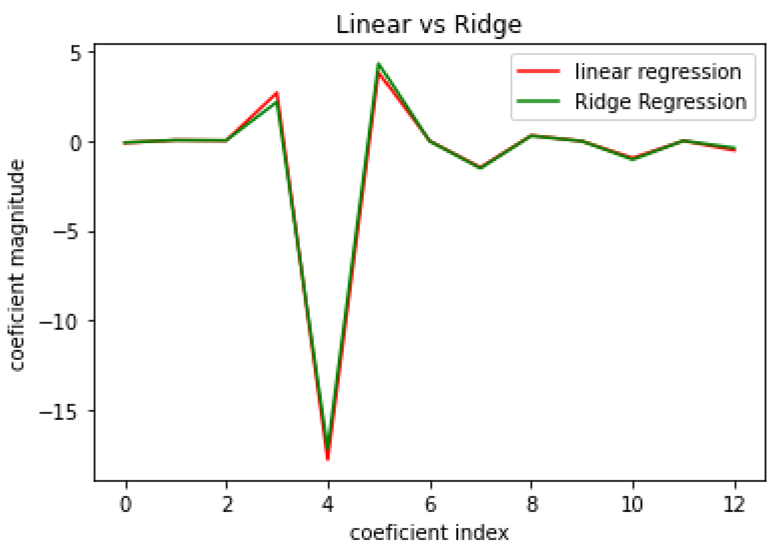
In [80]:
```
rd.score(X_train,y_train)*100
```

Out[80]:
```
77.00211293470278
```

In [81]:
```
rd.score(X_test,y_test)
```

Out[81]:
```
0.6071919674027766
```

In [82]:
```
plt.plot(linear.coef_,c='r',label="linear regression")
plt.plot(rd.coef_,c='g',label = 'Ridge Regression')
plt.title('Linear vs Ridge')
plt.xlabel('coeficient index')
plt.ylabel('coeficient magnitude')
plt.legend()
plt.show()
```



In [90]:
```
from sklearn.linear_model  import Lasso
la = Lasso(alpha =100)
```

In [91]:
```
la.fit(X_train,y_train)
```

Out[91]:
```
Lasso(alpha=100)
```

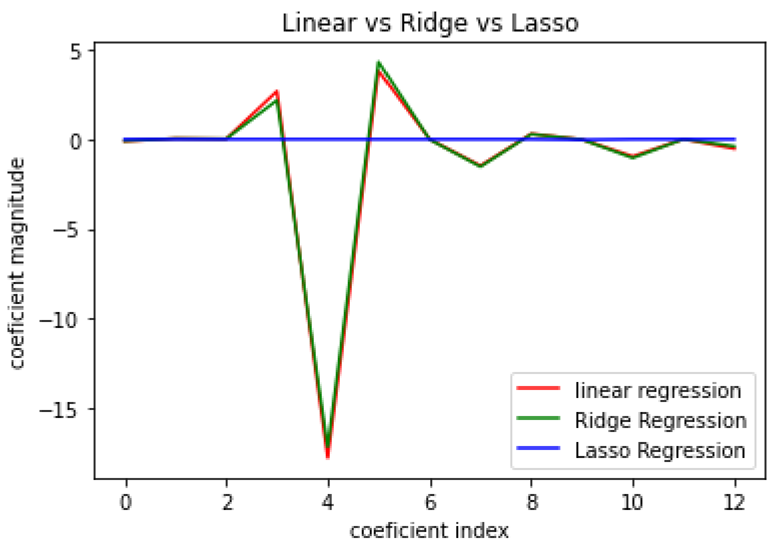
In [92]:
```
la.score(X_train,y_train)
```

Out[92]:
```
0.2879575494019463
```

In [93]:
```
la.score(X_test,y_test)
```

Out[93]: 0.00949562533326187

In [94]:
```
plt.plot(linear.coef_,c='r',label="linear regression")
plt.plot(rd.coef_,c='g',label = 'Ridge Regression')
plt.plot(la.coef_,c='b',label = 'Lasso Regression')
plt.title('Linear vs Ridge vs Lasso')
plt.xlabel('coeficient index')
plt.ylabel('coeficient magnitude')
plt.legend()
plt.show()
```



In [ ]:
```

```