Regularization (Ridge Regression)

- it is an extremely important concept in machine learning. It's a way to prevent overfitting, and thus, improve the likely generalization performance of a model by reducing the complexity of the final estimated model.
- In regularization, what we do is normally we keep the same number of features, but reduce the magnitude of the coefficients.
- Main Objective of regularization is to scale down the coefficient value.
- Regularisation can be applied to simple linear, multiple linear, polynomial regression as well as on all type of gradient descent and almost every algorithm.
- In general, regularization is a versatile tool used across a wide range of machine learning models and algorithms to manage model complexity and improve generalization.

Ex:- Real Value

0.9+1.2x1+20x2+39x3

- as you all see there are 1.2,20,39 are three coefficient. we easily say which have high coefficient this effect highly our target variable as x3 has high coefficient it has highly corelate with target variable.
- so but this may cause overfit in real life problem. so we need to scale down these coefficient value to decrease the complexity of model

After Scale down (By applying some regularization technique)

0.9+0.7x1+2x2+5x3

- see here i apply regularization and scaled down the value of coefficient. as this is low value so here the model give good accuracy. so we always tries to scaled down the coefficient value.
- Now let discuss how we scaled down these value.
- In regularization two type of method

    1. Ridge Regression
    2. Lasso Regression

Ridge Regression :

Formula : Ridge = Loss + α M^2

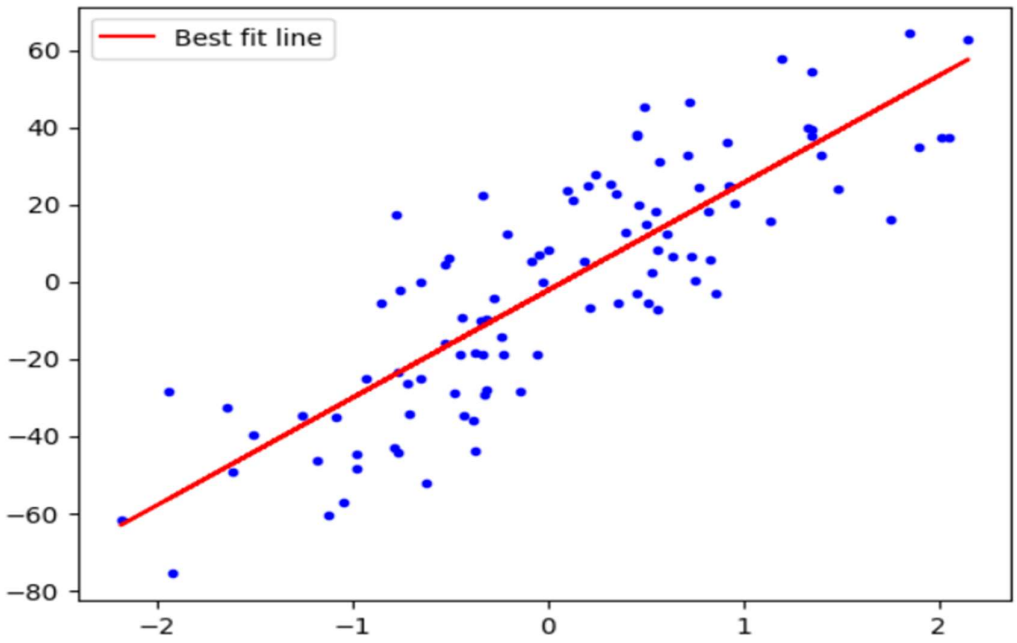
Ridge = Loss + α ||W||2

- Where Loss = Difference between predict and actual value (Or Cost Function)
- W = slope
- α = constant

(1) Simple linear regression

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X,y)
print(lr.coef_)
print(lr.intercept_)
plt.plot(X,y,'b.')
plt.plot(X,lr.predict(X),color='red',label='Best fit line')
plt.legend()
```

```
[27.82809103]
-2.29474455867698
<matplotlib.legend.Legend at 0x78bb9f1cb9d0>
```



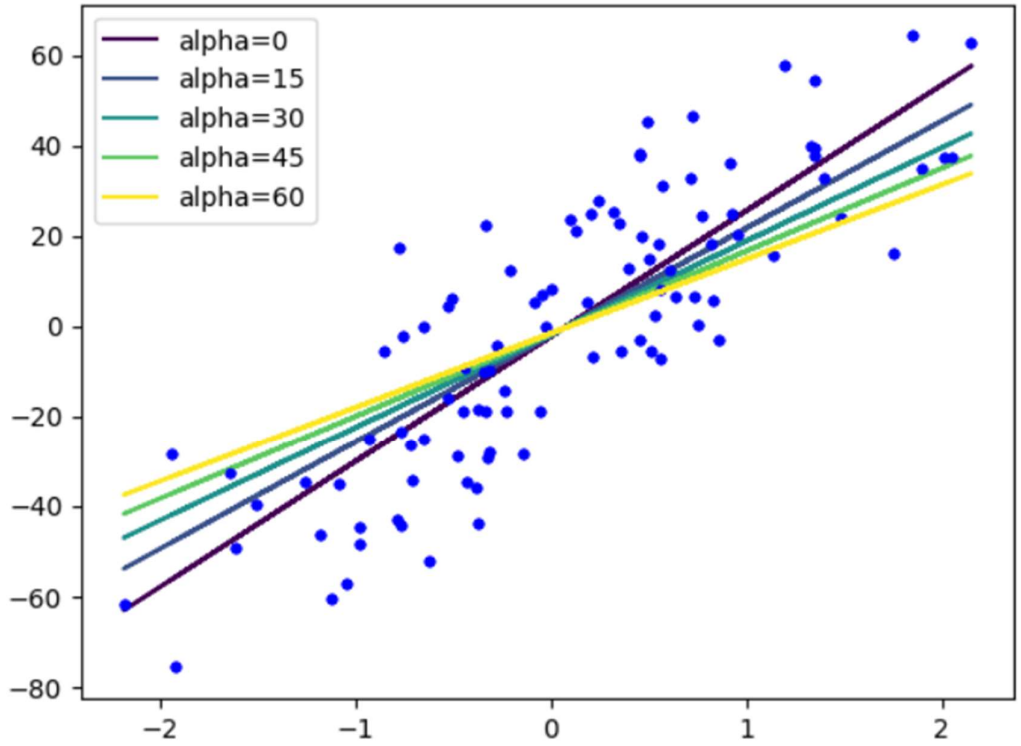
Here slope is 27.82 and intercept is -2.2

(2) Use ridge regression(code from scratch)

```
class MyRidge:
    def __init__(self,alpha):
        self.m=None
        self.b=None
        self.alpha=alpha

    def fit(self,X_train,y_train):

        num=0
        den=0

        for i in range(X_train.shape[0]):

            x_mean=X_train.mean()
            y_mean=y_train.mean()

            num = num + ((X_train[i] - x_mean)*(y_train[i] - y_mean))
            den = den + ((X_train[i] - x_mean)*(X_train[i] - x_mean))

        self.m=num/(den+self.alpha)
        self.b=y_mean-(self.m*x_mean)
        print('slope :',self.m,'intercept :',self.b)

    def predict(self,X_test):
        return (self.m*X_test + self.b)
```

```
alphas = [i * 15 for i in range(5)]
colors = plt.cm.viridis(np.linspace(0, 1, len(alphas)))

for i in range(len(alphas)):
    plt.plot(X,y,'b.')
    rr = MyRidge(alpha=alphas[i])
    rr.fit(X,y)
    plt.plot(X,rr.predict(X),color=colors[i],label='alpha={}'.format(alphas[i]))
    plt.legend()
```

```
slope : [27.82809103] intercept : [-2.29474456]
slope : [23.72950005] intercept : [-2.05535655]
slope : [20.68322312] intercept : [-1.87743146]
slope : [18.33009531] intercept : [-1.73999141]
slope : [16.45770427] intercept : [-1.63062993]
```

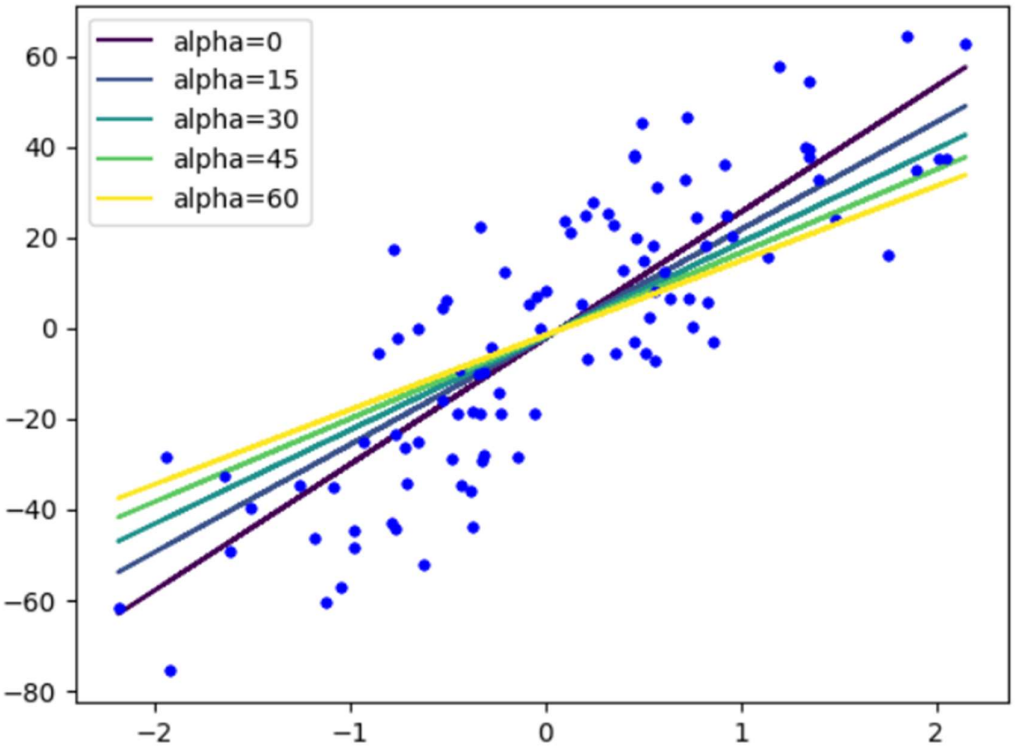


Here closest value of slope and intercept to linear regression is

Slope=23.73 and intercept=-2.05

(3) Use ridge regression(built-in)

```
from sklearn.linear_model import Ridge

alphas = [i * 15 for i in range(5)]
colors = plt.cm.viridis(np.linspace(0, 1, len(alphas)))
for i in range(len(alphas)):
    plt.plot(X,y,'b.')
    rr = Ridge(alpha=alphas[i])
    rr.fit(X,y)
    plt.plot(X,rr.predict(X),color=colors[i],label='alpha={}'.format(alphas[i]))
    plt.legend()
```

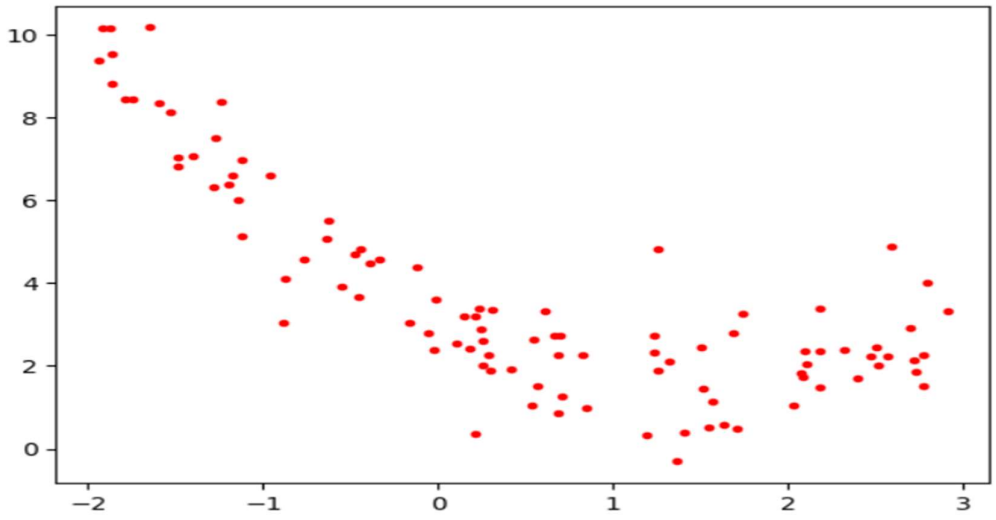


Here closest value of slope and intercept to linear regression is

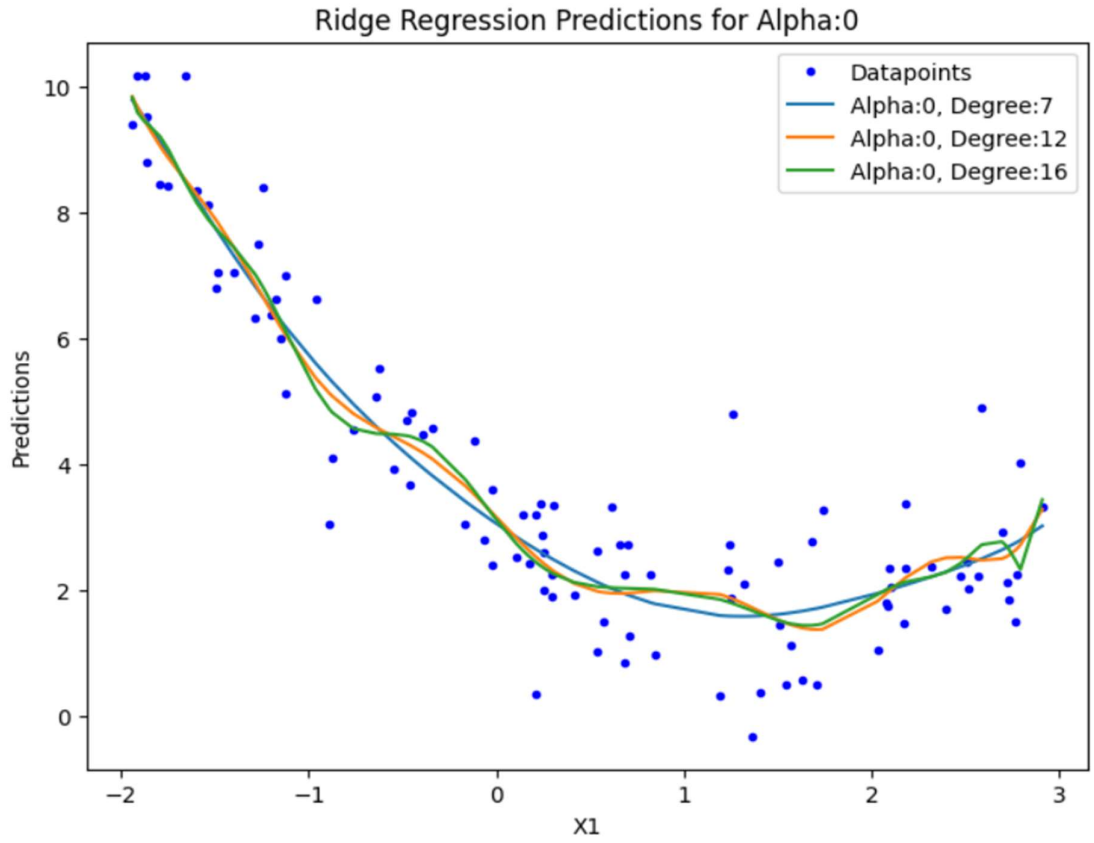
Slope=23.73 and intercept=-2.05

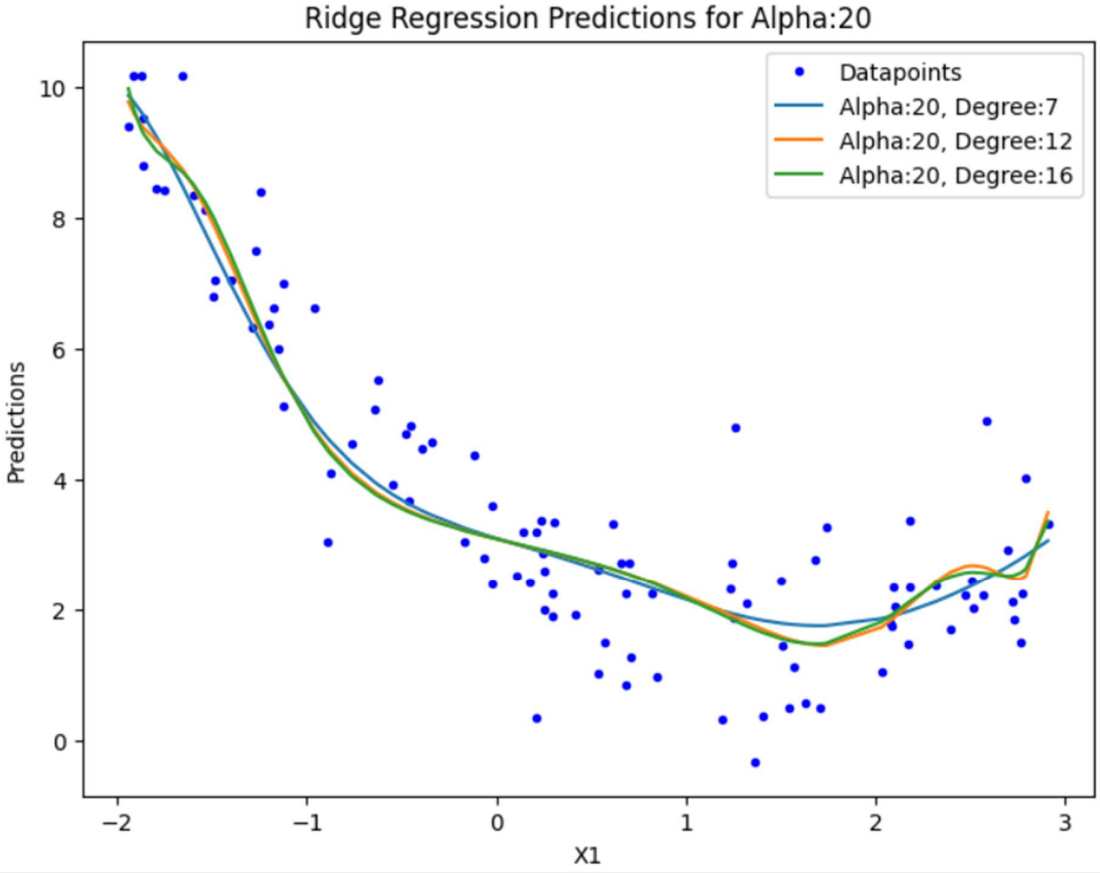
Ridge Regression for polynomial linear regression

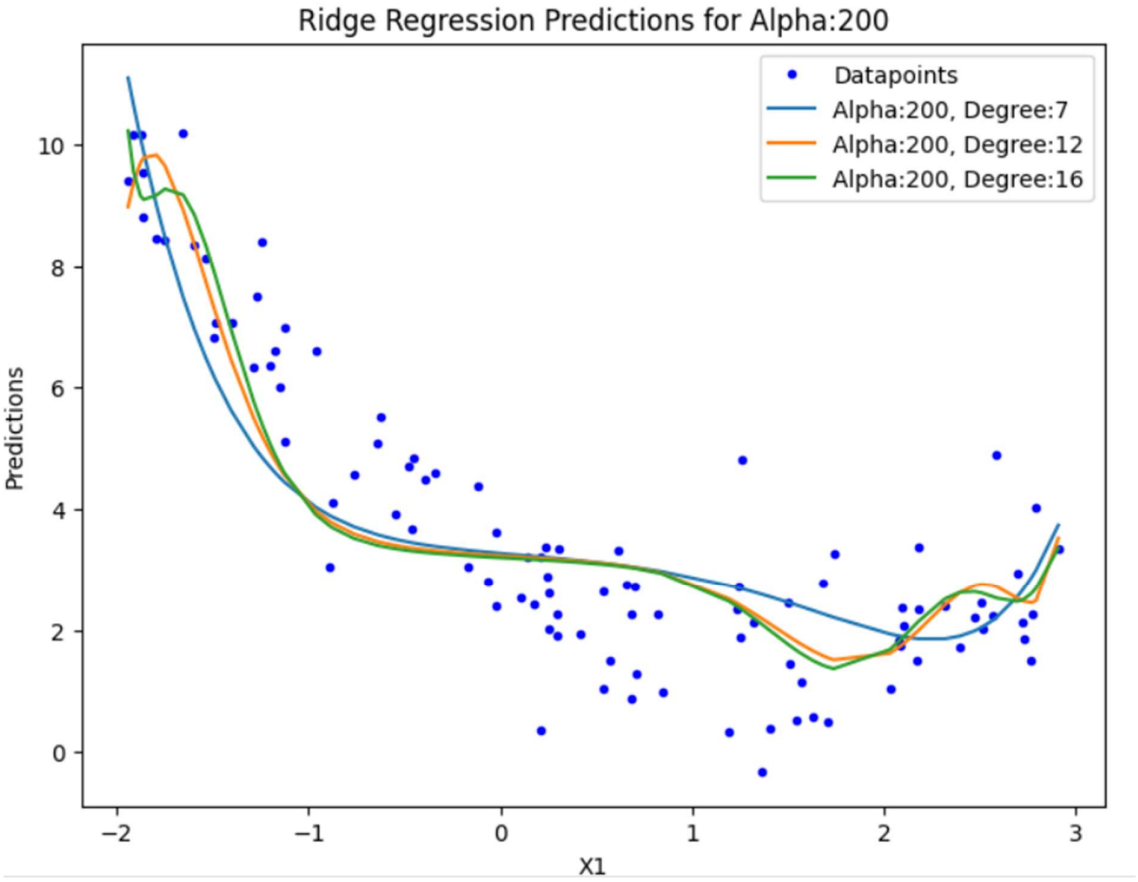
```
m = 100
x1 = 5 * np.random.rand(m, 1) - 2
x2 = 0.7 * x1 ** 2 - 2 * x1 + 3 + np.random.randn(m, 1)

plt.plot(x1, x2,'r.')
plt.show()
```




```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge
def get_preds_ridge(x1, x2, alpha, degree):
    model = Pipeline([
        ('poly_feats', PolynomialFeatures(degree=degree)),
        ('ridge', Ridge(alpha=alpha))
    ])
    model.fit(x1, x2)
    return model.predict(x1)

degree=[7,12,16]
alphas = [0, 20, 200]
cs = ['blue', 'red', 'yellow']

for alpha, c in zip(alphas, cs):
    plt.figure()  # Create a new figure for each alpha
    plt.figure(figsize=(8, 6))
    plt.plot(x1, x2, 'b.', label='Datapoints')
    for d in degree:
        preds = get_preds_ridge(x1, x2, alpha, d)
        plt.plot(sorted(x1[:, 0]), preds[np.argsort(x1[:, 0])],label='Alpha:{}, Degree:{}'.format(alpha,
    plt.legend()
    plt.xlabel('X1')
    plt.ylabel('Predictions')
    plt.title('Ridge Regression Predictions for Alpha:{}'.format(alpha))
    plt.show()
```


Ridge Regression Predictions for Alpha:0
Datapoints
Alpha:0, Degree:7
Alpha:0, Degree:12
Alpha:0, Degree:16
Predictions
X1
10
8
6
4
2
0
−2
−1
0
1
2
3


Ridge Regression Predictions for Alpha:20
Datapoints
Alpha:20, Degree:7
Alpha:20, Degree:12
Alpha:20, Degree:16
Predictions
X1
10
8
6
4
2
0
−2
−1
0
1
2
3



Here , for alpha=0 , there is under fitting

for alpha=20 , best fitting

for alpha=200 , there is over fitting

Note :

1. Regularization is mainly used for Scaled down the coeficient value so that overfit not happen in model.
2. Regularization reduce complexity of model
3. in Ridge Regression magnitude of coefficient is almost zero , but in Lasso magnitude of coefficient is exaclty zero.
4. we can find appropriate α value by doing Cross validation.
5. α value is always 0 to any +ve number.
6. Lasso is used for feature selection.
7. Ridge Also called as L2 Regularization, and Lasso Is L1 Regularization.