Stochastic Gradient Descent

1. Loading Boston Dataset from Sklearn Library

2. Custom SGD Regressor implementation (PART-1)

2. Sklearn's SGD Regressor implementation (PART-2)

4. Comparing Both the implementaton

NOTE :- In each implementation plot the graph for the predicted vs actual values in the dataset(part1, part 2)

In [126]:

```
#importing libraries
import warnings
warnings.filterwarnings("ignore")
from sklearn.datasets import load_boston
from random import seed
from random import randrange
from csv import reader
from math import sqrt
from sklearn import preprocessing
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from prettytable import PrettyTable
from sklearn.linear_model import SGDRegressor
from sklearn import preprocessing
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn import linear_model
import seaborn as sns
from sklearn.model_selection import train_test_split
from numpy import c_
```

1. Loading Boston Dataset from Sklearn Library

In [127]:

```
dataset = load_boston()
X = pd.DataFrame(dataset.data)
y = dataset.target

print("="*100)
print("Dataset\n")
print(X[:4])
print(type(X))

print("="*100)
print("Actual price\n")
print(y[:4])
print(type(y))

print("="*100)
```

```
====================================================================================================

Dataset

         0     1     2    3      4      5     6       7    8      9    10  \
0  0.00632  18.0  2.31  0.0  0.538  6.575  65.2  4.0900  1.0  296.0  15.3
1  0.02731   0.0  7.07  0.0  0.469  6.421  78.9  4.9671  2.0  242.0  17.8
2  0.02729   0.0  7.07  0.0  0.469  7.185  61.1  4.9671  2.0  242.0  17.8
3  0.03237   0.0  2.18  0.0  0.458  6.998  45.8  6.0622  3.0  222.0  18.7
```

```
       11    12
0  396.90  4.98
1  396.90  9.14
2  392.83  4.03
3  394.63  2.94
<class 'pandas.core.frame.DataFrame'>
===================================================================================================

Actual price

[24.  21.6 34.7 33.4]
<class 'numpy.ndarray'>
===================================================================================================
```

In [128]:

```
X_tr, X_te, Y_train, Y_test = train_test_split(X, y, test_size = 0.3, random_state = 5)
```

In [129]:

```
# Standardize the data
sc = StandardScaler()
X_train = sc.fit_transform(X_tr)
X_test = sc.transform(X_te)
```

2. Custom SGD Regressor implementation (PART-1)

In [130]:

```
#Fitting the train Dataset in the custom SGD Regressor model


# Modifying dataset for getting the random row according to batch size in each iteration in the cu
stom SGD model
X_tr = np.c_[np.ones((len(X_train),1)),X_train]
X_train = X_tr

X_te = np.c_[np.ones((len(X_test),1)),X_test]
X_test = X_te

# initialising the values for fitting the model
max_iter = 1000 #initialising no of iterations
learning_rate=0.01 #initialising the learning
length = len(Y_train) #  length of the data set
error = []

# Generating normally distributed values of the train dataset
weight = np.random.normal(0,1,X_train.shape[1])
weight = np.c_[weight]


# MODEL
for k in range(max_iter):
    for i in range(length):
        batch_size = np.random.randint(0,length)
        X_batch = X_train[batch_size,:].reshape(1,X_train.shape[1])
        y_batch = Y_train[batch_size].reshape(1,1)
        prediction = np.dot(X_batch,weight)
        weight = weight -(2/length)*learning_rate*( X_batch.T.dot((prediction - y_batch)))
```

In [131]:

```
pred_custom = X_test.dot(weight)
pred_custom = pred_custom.ravel()
```

In [132]:

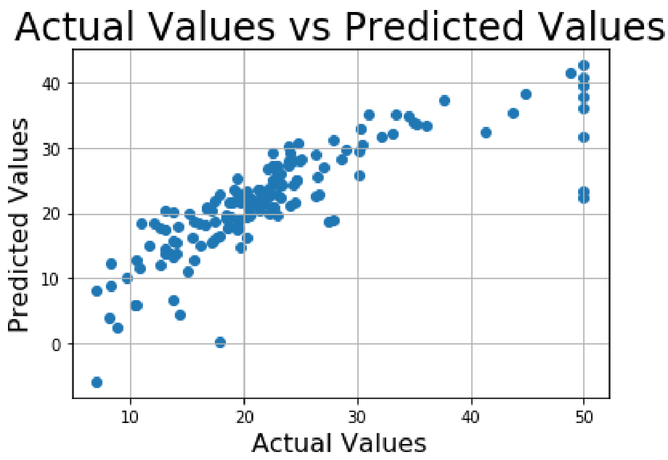
```
#Weights from Custom SGD Model
# weight custom = []
```

```
# weight_custom = []
# for i in weight:
#     weight_custom.append(i)
weight_custom = weight.ravel()
print(c_[weight_custom])
```

```
[[22.54034743]
 [-1.22434741]
 [ 0.89977223]
 [-0.28205546]
 [ 0.19912563]
 [-1.34718894]
 [ 2.8248531 ]
 [-0.30299525]
 [-2.71286166]
 [ 2.44520023]
 [-1.81463673]
 [-2.04418207]
 [ 1.1216619 ]
 [-3.28606046]]
```

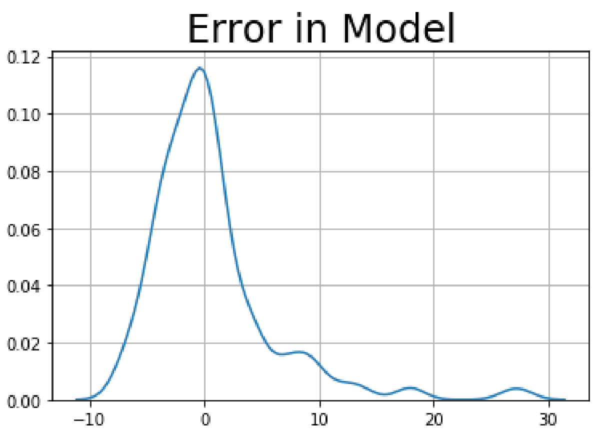
In [133]:

```
# Plotting graph for actual vs predicted price
plt.scatter(Y_test, pred_custom)
plt.xlabel("Actual Values",size=16)
plt.ylabel("Predicted Values",size=16)
plt.title("Actual Values vs Predicted Values",size=24)
plt.grid()
plt.show()
```



In [134]:

```
error = Y_test - pred_custom; #error = actual_price - predicted_price
sns.kdeplot(error)
plt.title("Error in Model",size=24)
plt.grid()
plt.show()
```



In [135]:

```
# Obtaining Mean Squared Error (MSE)
MSE_CUSTOM = mean_squared_error(Y_test, pred_custom)
print("Mean Squared Error (MSE) = ",MSE_CUSTOM)
```

```
Mean Squared Error (MSE) =  30.52018558400602
```

3. Sklearn's SGD Regressor implementation (PART-2)

In [136]:

```
model = linear_model.SGDRegressor(penalty='none', max_iter=1000, learning_rate='constant')
model.fit(X_train,Y_train)
pred_sklearn = model.predict(X_test)
```

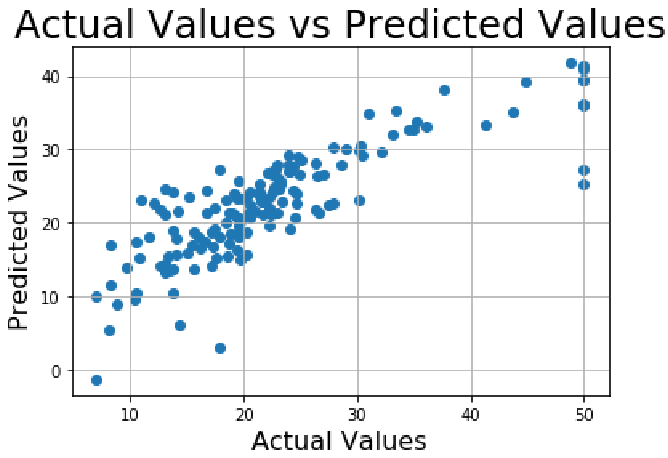
In [137]:

```
#Weights from Sklearn Model
weight_sklearn = model.coef_
print(c_[weight_sklearn])
```

```
[[11.66921283]
 [-0.71637651]
 [ 0.76089547]
 [ 0.13184507]
 [ 0.25055041]
 [-1.24878103]
 [ 3.11008312]
 [-0.22604507]
 [-3.02407402]
 [ 3.24388265]
 [-1.64174153]
 [-1.77595867]
 [ 1.49545237]
 [-3.20159722]]
```

In [138]:

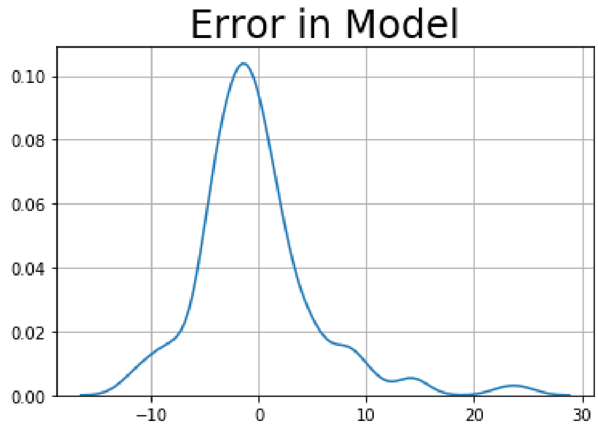
```
# Plotting graph for actual vs predicted price
plt.scatter(Y_test, pred_sklearn)
plt.xlabel("Actual Values",size=16)
plt.ylabel("Predicted Values",size=16)
plt.title("Actual Values vs Predicted Values",size=24)
plt.grid()
plt.show()
```



In [139]:

```
error = Y_test - pred_sklearn; #error = actual_price - predicted_price
sns.kdeplot(error)
plt.title("Error in Model",size=24)
plt.grid()
```

```
plt.show()
```



In [140]:

```python
# Obtaining Mean Squared Error (MSE)
MSE_SKLEARN = mean_squared_error(Y_test, pred_sklearn)
print("Mean Squared Error (MSE) = ",MSE_SKLEARN)
```

```
Mean Squared Error (MSE) =  30.089575756454817
```

4. Comparing Both the implementaton

In [141]:

```python
# Creating the table using PrettyTable library
from prettytable import PrettyTable

# Initializing prettytable
preety_table = PrettyTable()

slno = [1,2,3,4,5,6,7,8,9,10,11,12,13,14]


# Adding columns
preety_table.add_column("SL NO.",slno)
preety_table.add_column("Weights of Manual SGD",weight_custom)
preety_table.add_column("Weights of Sklearn's SGD",weight_sklearn)

# Printing the Table
print(preety_table)
```

```
+--------+----------------------+-------------------------+
| SL NO. | Weights of Manual SGD | Weights of Sklearn's SGD |
+--------+----------------------+-------------------------+
|   1    |   22.540347429394803  |    11.669212830204986    |
|   2    |  -1.2243474130631211  |   -0.7163765061264418    |
|   3    |   0.8997722286520311  |    0.7608954664247759    |
|   4    |  -0.2820554593186188  |   0.13184507146017782    |
|   5    |   0.1991256302227841  |   0.25055040542801027    |
|   6    |  -1.3471889373772865  |   -1.2487810324327755    |
|   7    |   2.8248531020463727  |    3.110083115199914     |
|   8    |  -0.3029952450970284  |   -0.22604507263175322   |
|   9    |  -2.7128616606927336  |   -3.0240740233393377    |
|   10   |    2.4452002261934    |    3.243882650604810 6   |
|   11   |   -1.814636731115359  |   -1.6417415250485121    |
|   12   |  -2.0441820664854395  |   -1.7759586709623616    |
|   13   |   1.1216618963717684  |    1.4954523650534282    |
|   14   |  -3.2860604557159188  |    -3.201597217478671    |
+--------+----------------------+-------------------------+
```

In [142]:

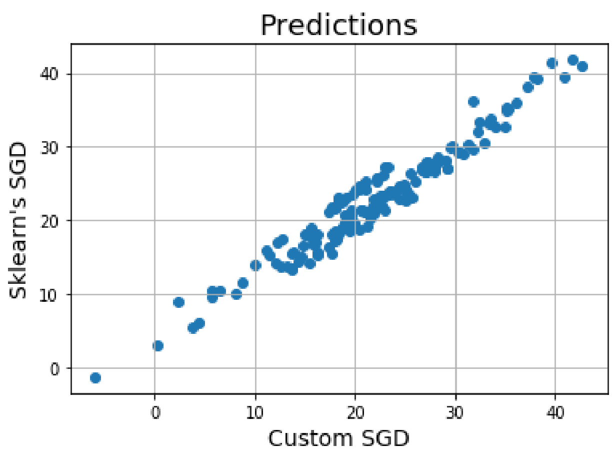
```python
# Creating the table using PrettyTable library
from prettytable import PrettyTable
```

```
# Initializing prettytable
preety_table = PrettyTable()

preety_table.field_names = ["SGD Type","Mean Squared Error"]
preety_table.add_row(["Custom SGD",MSE_CUSTOM])
preety_table.add_row(["Sklearn SGD",MSE_SKLEARN])


print(preety_table)
```

```
+-------------+--------------------+
|   SGD Type  | Mean Squared Error |
+-------------+--------------------+
|  Custom SGD | 30.52018558400602  |
| Sklearn SGD | 30.089575756454817 |
+-------------+--------------------+
```

In [143]:

```
# Scatter Plot of the predictions of both manual SGD Regression and Sklearn's SGD Regression
plt.scatter(pred_custom, pred_sklearn)
plt.xlabel("Custom SGD",size=14)
plt.ylabel("Sklearn's SGD",size=14)
plt.title("Predictions",size=18)
plt.grid()
plt.show()
```



Conclusion

- MSE of Custom SGD is slightly higher than the Sklearn SGD, therefore the custom model is performing well here

- Weights of both the models are approximately similar

- As we can see on the above prediction graph that the predicted value in sklearn and custom are giving aprroximately same value of predicted price

END