In [27]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.cross_validation import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from pandas import DataFrame,Series
import seaborn as sns
%matplotlib inline
from sklearn import datasets,svm
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

In [28]:

```
from sklearn.datasets import load_diabetes
data= load_diabetes()
```

In [29]:

```
db= pd.DataFrame(data.data,columns=data.feature_names)
```

In [30]:

```
db.head()
```

Out[30]:

	age	sex	bmi	bp	s1	s2	s3	s
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.00259
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.03949
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.00259
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.03430
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.00259

In [31]:

```
y= pd.DataFrame(data.target)
```

In [32]:

```
x_train,x_test,y_train,y_test= train_test_split(db,y, test_size=0.25,
random_state=10)
```

In [33]:

```
from sklearn.kernel_ridge import KernelRidge

classifier= KernelRidge(kernel ='linear', alpha=1.0)

classifier.fit(x_train,y_train)

kr_pred = classifier.predict(x_test)
y_pred = np.exp(kr_pred)-1

r = np.sqrt(mean_squared_error(y_test,kr_pred))
```

In [34]:
```
# Finding the mean square error

print("MSE: %.2f"% mean_squared_error(y_test,kr_pred))

# Finding the Variance Score

print('Variance Score: %.2f' % (r2_score(y_test,kr_pred)))
```

```
MSE: 27483.50
Variance Score: -3.38
```

In [35]:
```
classifier.score(x_test,y_test)
```

Out[35]:
```
-3.3763604794152817
```

In [36]:
```
from sklearn.linear_model import Ridge
ridge_regres=Ridge(alpha=0.3, normalize=True).fit(x_train,y_train)
ridge_regres.coef_
ridge_regres.intercept_
```

Out[36]:
```
array([ 151.69207589])
```

In [37]:
```
pred_regres = ridge_regres.predict(x_test)
```

In [38]:
```
ridge_regres.score(x_test,y_test)
```

Out[38]:
```
0.50115529533842318
```

In [41]:
```
#CONCLUSION

import pandas as pd
data = [['Kernel Ridge ', classifier.score(x_test,y_test)],['Ridge Reg
ression ',ridge_regres.score(x_test,y_test)]]
frame = pd.DataFrame(data,columns=['Regression','Variance'])
frame.reset_index(drop = True, inplace = True)
print(frame)
```

```
          Regression  Variance
0      Kernel Ridge  -3.376360
1  Ridge Regression   0.501155
```

In [19]:
```
#Since, the variance of Ridge regression is higher than Kernel ridge,
we conclude that Kernel ridge will perform better for regression model
.
```

In [189]:
```
# SVM Linear, RBF(Gaussian), Polynomial
```

In [42]:
```
data = datasets.load_iris()

v1 = data.data
r1 = data.target
```

In [43]:
```
dataset = DataFrame(v1)
dataset.columns=['sepal length','sepal width','petal length','petal wi
dth']
```

In [46]:
```
t1 = DataFrame(r1,columns=['species'])
```

In [47]:
```
def flower(num):
    if(num==0):
        return "setosa"
    elif(num==1):
        return "versicolor"
    else:
        return "virginica"
```

In [48]:
```
t1['species']=t1['species'].apply(flower)
```

In [49]:
```
dataset_new = pd.concat([dataset,t1],axis=1)

dataset_new.head()
```

Out[49]:

	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [51]:
```
#Linear Regression Model
lin_reg = svm.SVC(kernel='linear', C=1, gamma=1)

x_train,x_test,y_train,y_test = train_test_split(v1,r1,test_size=0.25,
random_state=4)

lin_reg.fit(x_train,y_train)

p2 = lin_reg.predict(x_test)

print(metrics.accuracy_score(y_test,p2))
```

```
0.973684210526
```

In [52]:
```
# Gaussian Radial Bassis Function
rad_bas = svm.SVC(kernel='rbf', gamma=0.1, C=1)

rad_bas.fit(x_train,y_train)

p3 = rad_bas.predict(x_test)

print(metrics.accuracy_score(y_test,p3))
```

```
0.973684210526
```

In [53]:
```
# Polynomial Model

poly = svm.SVC(kernel='poly', degree=3, C=1)

poly.fit(x_train,y_train)

p4=poly.predict(x_test)

print(metrics.accuracy_score(y_test,p4))
```

```
0.947368421053
```

In [54]:
```
import pandas as pd
data = [['Linear ', metrics.accuracy_score(y_test,p2)],['Gaussian ',me
trics.accuracy_score(y_test,p3)],['Polynomial', metrics.accuracy_score
(y_test,p4)]]
df = pd.DataFrame(data,columns=['Regression','Accuracy Score'])
df.reset_index(drop = True, inplace = True)
print(df)
```

```
   Regression  Accuracy Score
0     Linear         0.973684
1   Gaussian         0.973684
2  Polynomial        0.947368
```

In [205]:
```
#CONCLUSION
#So, Linear and Gaussian models perform better as compared to Polynomi
al model since the accuracy is higher.
```