A Project Report

On

FECAL COLIFORM ESTIMATION USING SVMs

BY

SREEKAR CHIGURUPATI

2014AAPS0274H

Under the supervision of

DR. JAGADEESH ANMALA

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

CE F367: DESIGN PROJECT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

HYDERABAD CAMPUS

(NOVEMBER 2017)

ACKNOWLEDGMENTS

I would like to thank Dr. Jagadeesh Anmala for giving me this opportunity to pursue a project under his guidance. I would also like to thank the Project Jupyter for providing the perfect open source web application to execute my scripts in.

Birla Institute of Technology and Science-Pilani,

Hyderabad Campus

Certificate

This is to certify that the project report entitled "**FECAL COLIFORM ESTIMATION USING SVMs"** submitted by Mr. SREEKAR CHIGURUPATI (ID No. 2014AAPS0274H) in partial fulfillment of the requirements of the course CE F367 Design Project Course, embodies the work done by him under my supervision and guidance.

Date: 30-11-2017

(DR. JAGADEESH ANMALA)

BITS- Pilani, Hyderabad Campus

ABSTRACT

The prediction of water stream quality using specific water quality parameters like precipitation, land use factor and temperature. This is done by determining the amount of a specific kind of bacteria found in water called fecal coliform. The modelling is done using a genre of machine learning algorithms called support vector machines.

CONTENTS

1. INITIAL WORK PLAN

Objective	Finish Date
Literature Review	Sep-25th
SVM Programming	Oct-25th
Parameter Adjustment	Nov-10th
Comparison-ANN/SVM	Nov-25th
Final Presentation	Dec-1st

All objectives except ANN/SVM Comparison are done.

2. LITERATURE REVIEW

A)

- URL: http://green2.kingcounty.gov/streamsdata/Bacteria.aspx?Locator=B319
- Website Title: Stream Bacteria Chart - King County
- Article Title: Stream Bacteria Chart

B)

Jagadeesh Anmala, O.W.Meier, A.J. Meier and S.Grubbs 2015, A GIS and an Artificial Neural Network Based Water Quality Model for a Stream Network in Upper Green River Basin, Kentucky, USA, ASCE, *Journal of Environmental Engineering*, 141(5),

04014082

DOI: 10.1061/(ASCE)EE. 1943-7870.0000801.

C)

- URL: http://scikit-learn.org/stable/modules/svm.html#svr
- Website Title: 1.4. Support Vector Machines — scikit-learn 0.19.1 documentation
- Article Title: 1.4. Support Vector Machines

3. WATER QUALITY

We would like to predict the water quality of water streams from the upper green river. This has been previously done using artificial neural networks [Jagadeesh Anmala et al.].

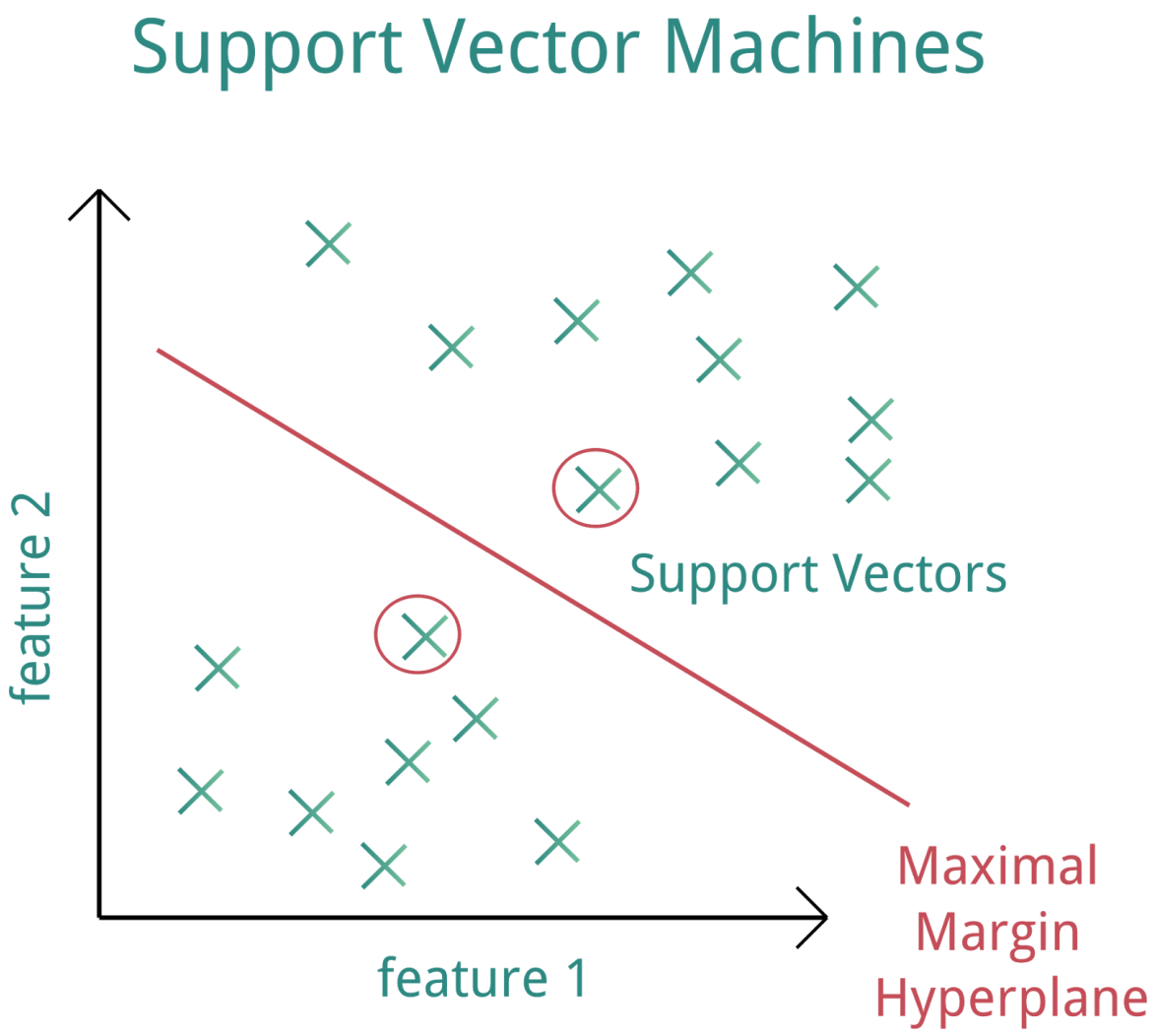
Stream water quality is important because it's the source most used for human consumption. It's also the one most prone to human effluents.

AGENT	HEALTH EFFECTS	
	Acute	Chronic
Bacteria		
E. coli (O157:H7)	Diarrhea	Adults: death from thrombocytopenia Children: death from kidney failure
Campylobacter	Diarrhea	Death from Guillian-Barre syndrome
Salmonella	Diarrhea	Reactive arthritis
Shigella	Diarrhea	Reactive arthritis
Protozoa		
Giardia lamblia	Diarrhea	Failure to thrive Severe hypothyroidism Lactose intolerance Chronic joint pain
Cryptosporidium	Diarrhea	Death in immuno-compromised humans
Viruses		
Hepatitis viruses	Liver infections	Liver failure
Norwalk virus	Diarrhea	

Coliform bacteria are the ones most known to cause illness in humans from water contamination.

4. SUPPORT VECTOR MACHINES

Support vector machines are a class of machine learning algorithms typically used for classification.



They were conceptualized by Vapnik and Chervonenkis. Given an N-dimensional dataset, it outputs a hyperplane that's one dimension lesser. This hyperplane splits the data into the two labelled classifications.

The name support vector machine is self-explanatory. It is a machine which works only on support vectors. Support vectors are those points that are closest to the classifier boundary. In a sense they are the defining force behind the classifier.

This kind of dimensionality reduction helps in running large data sets with ease but might underfit small ones.

The same algorithms can be used for solving the regression problem too

The typical libraries implement what is known as linear epsilon-insensitive SVM (ε-SVM) regression.

Given below is the concept behind SVR.

The dual formula for nonlinear SVM regression replaces the inner product of the predictors ($x_i'x_j$) with the corresponding element of the Gram matrix (g_{ij}).

Nonlinear SVM regression finds the coefficients that minimize

$$L(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)G(x_i, x_j) + \varepsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) - \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i^*)$$

subject to

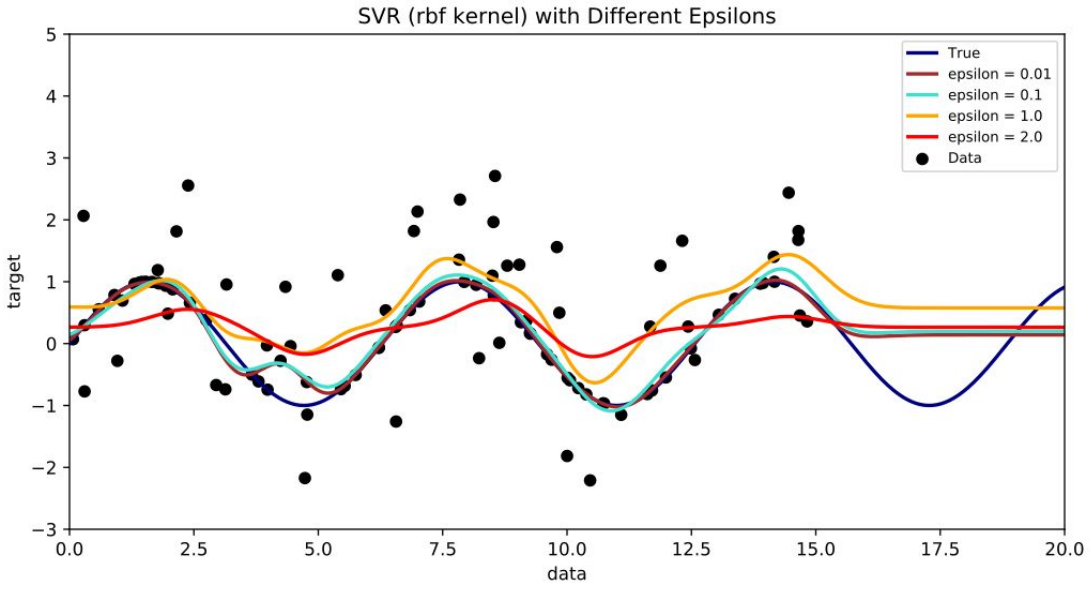
$$\sum_{n=1}^{N}(\alpha_n - \alpha_n^*) = 0$$
$$\forall n : 0 \le \alpha_n \le C$$
$$\forall n : 0 \le \alpha_n^* \le C\,.$$

Predictions are done using the following equation

$$f(x) = \sum_{n=1}^{N}(\alpha_n - \alpha_n^*)G(x_n, x) + b$$

5. SUPPORT VECTOR REGRESSION

Support vector regression is achieved by drawing an analogy to classifier algorithm



Regression is done by a cost function which ignores any points beyond a margin from the classifier and close to the model prediction.

It is a kernel based method and works by adjusting hyper parameters over an initial kernel.

Usual kernels using rbf, linear and polynomial.

6. TOOLS USED

Jupyter

Scikit-learn

7. ANALYSIS USING SVM

Fecal Coliform estimation using Support Vector Machines

Code for fitting water parameter data using SVMs

Data import

Import the parameters and FC data from csv

```
from sklearn.svm import SVR
import matplotlib.pyplot as plt
#from sklearn import svm
#import numpy as np
#from sklearn.svm import NuSVR
import csv

Temps=[]
Precs=[]
Means=[]
Cumuls=[]
Urban_LUFs=[]
Forest_LUFs=[]
Agris=[]
FCs=[]

#Data read
with open('coli_final.csv','r') as csvfile:
    coli = csv.reader(csvfile,delimiter=",")
    for row in coli:
        #[float(i) for i in row]
        #[value for value in row]
        Temp,Cumul,Urban_LUF,Forest_LUF,Agri,FC = [float(row[0]),float(row[1]),float(row[2]),float(row[3]),float(row[4]),float(row[5])]
        Temps.append(Temp)
        Cumuls.append(Cumul)
        Urban_LUFs.append(Urban_LUF)
        Forest_LUFs.append(Forest_LUF)
        Agris.append(Agri)
        FCs.append(FC)
```

SVM Regression

SVM Regression using scikit SVR

```
X=[Temps,Cumuls,Urban_LUFs,Forest_LUFs,Agris]
#X=zip(*X)
X=[list(p) for p in zip(*X)]
Y=FCs
svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)
svr_lin = SVR(kernel='linear', C=1e3)
svr_poly = SVR(kernel='poly', C=1e3, degree=2)
y_rbf = svr_rbf.fit(X, Y).predict(X)
y_lin = svr_lin.fit(X, Y).predict(X)
y_poly = svr_poly.fit(X, Y).predict(X)
# clf=svm.SVC()
# clf.fit(X,Y)
# clf.predict
print ("Original Value : 3080")
print ("rbf:", end=" ")
print (svr_rbf.predict([[13.24,11.15,0.03707,0.683092,0.872937]])[0])
print ("lin:", end=" ")
print (svr_lin.predict([[13.24,11.15,0.03707,0.683092,0.872937]])[0])
print ("poly:", end=" ")
print (svr_poly.predict([[13.24,11.15,0.03707,0.683092,0.872937]])[0])
```

```
Original Value : 3080
rbf: 3579.9066802
lin: 3343.29562377
poly: 4934.78233245
```

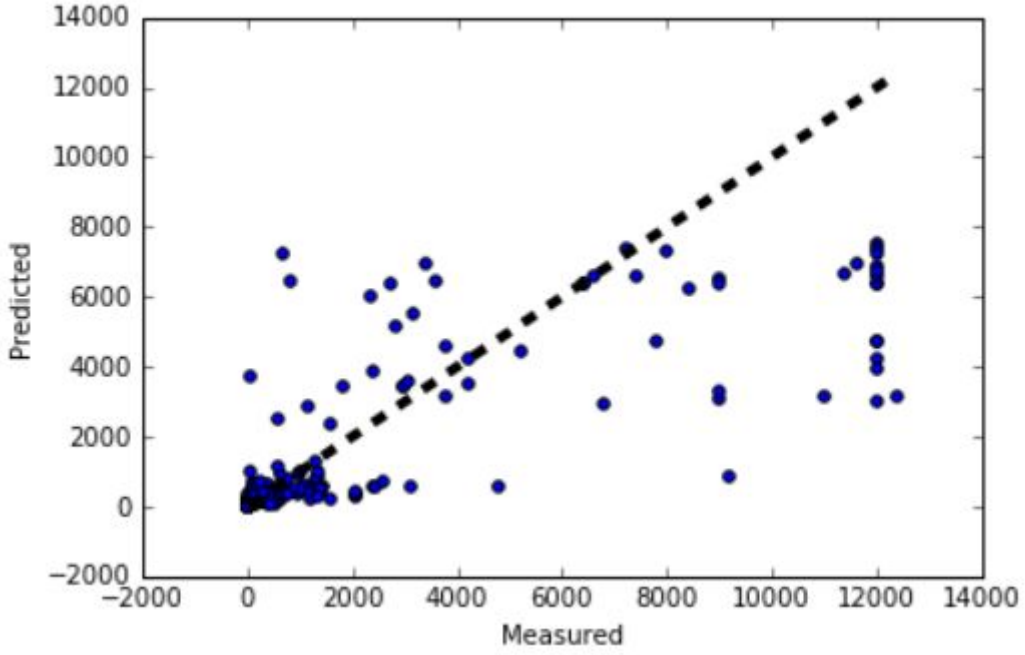
SVR Prediction for entire dataset

```
%matplotlib inline
RbfPreds=[]
LinPreds=[]
PolyPreds=[]
for row in X:
    RbfPreds.append(svr_rbf.predict([row]))
    LinPreds.append(svr_lin.predict([row]))
    PolyPreds.append(svr_poly.predict([row]))
```

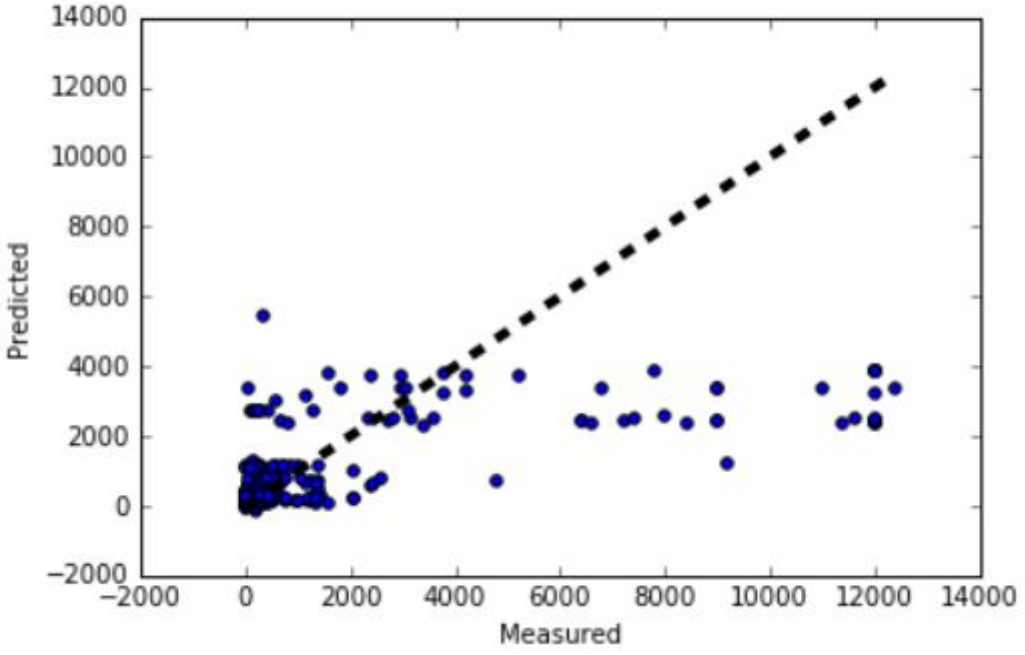
Scatter plots for each model

```
from sklearn.cross_validation import cross_val_predict
from sklearn import linear_model
fig, ax = plt.subplots()
ax.scatter(Y, RbfPreds, edgecolors=(0, 0, 0))
ax.plot([min(Y), max(Y)], [min(Y), max(Y)], 'k--', lw=4)
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
print("RBF")
plt.show()
fig, ax = plt.subplots()
ax.scatter(Y, LinPreds, edgecolors=(0, 0, 0))
ax.plot([min(Y), max(Y)], [min(Y), max(Y)], 'k--', lw=4)
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
print("LINEAR")
plt.show()
fig, ax = plt.subplots()
ax.scatter(Y, RbfPreds, edgecolors=(0, 0, 0))
ax.plot([min(Y), max(Y)], [min(Y), max(Y)], 'k--', lw=4)
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
print("POLY")
plt.show()
```

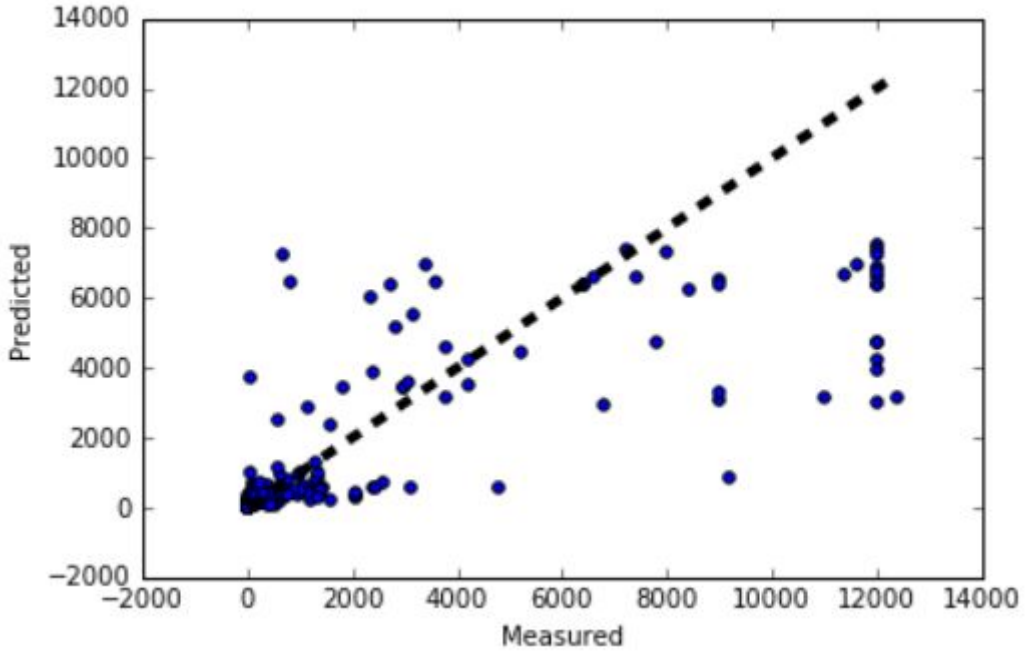
RBF


14000
12000
10000
8000
6000
4000
2000
0
−2000
Predicted
−2000
0
2000
4000
6000
8000
10000
12000
14000
Measured

LINEAR


14000
12000
10000
8000
6000
4000
2000
0
−2000
Predicted
−2000
0
2000
4000
6000
8000
10000
12000
14000
Measured

POLY



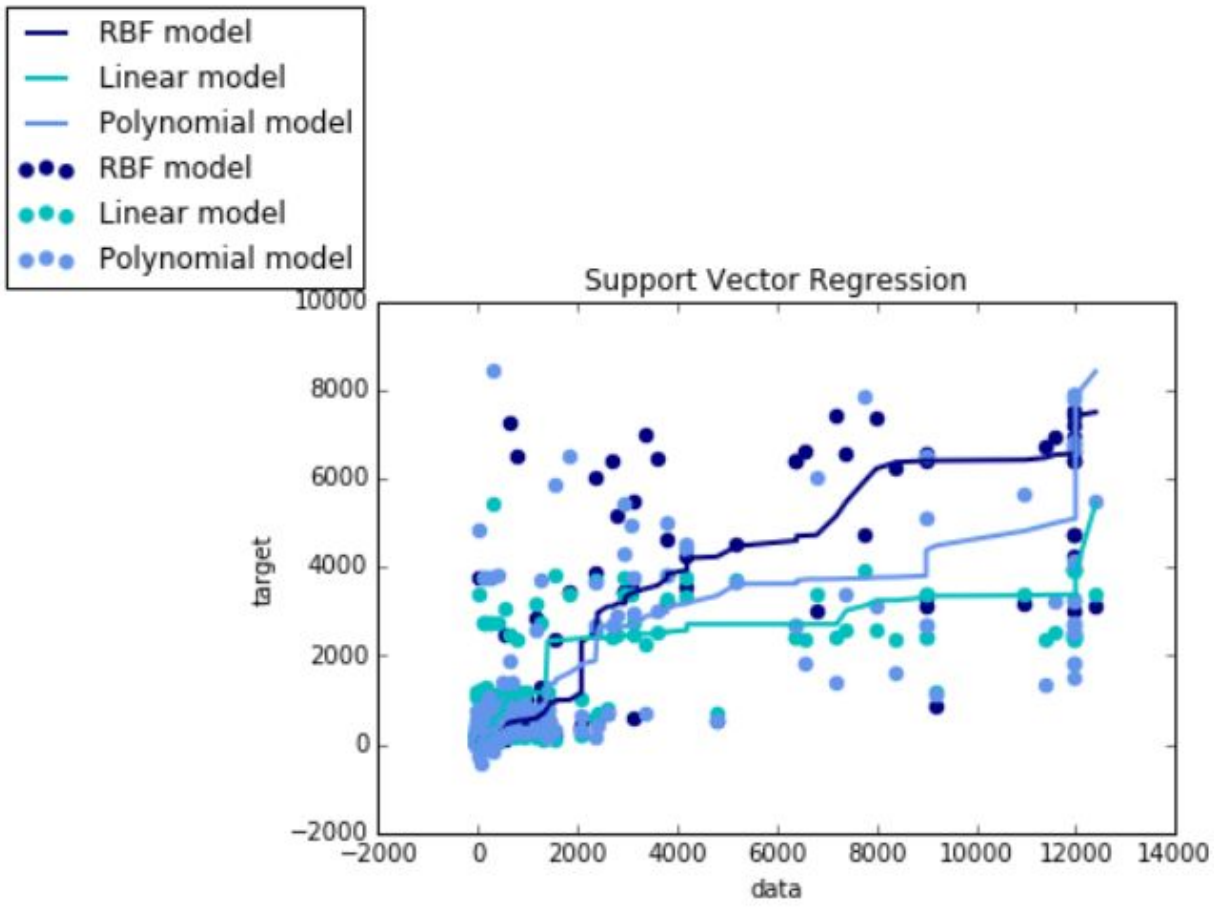
Pearson Correlation

```
import math

def average(x):
    assert len(x) > 0
    return float(sum(x)) / len(x)

def pearson(x, y):
    assert len(x) == len(y)
    n = len(x)
    assert n > 0
    avg_x = average(x)
    avg_y = average(y)
    diffprod = 0
    xdiff2 = 0
    ydiff2 = 0
    for idx in range(n):
        xdiff = x[idx] - avg_x
        ydiff = y[idx] - avg_y
        diffprod += xdiff * ydiff
        xdiff2 += xdiff * xdiff
        ydiff2 += ydiff * ydiff

    return (diffprod / math.sqrt(xdiff2 * ydiff2))[0]
print("Pearson Correlation")
print("RBF",end=" ")
print(pearson(Y,RbfPreds))
print("LINEAR",end=" ")
print(pearson(Y,LinPreds))
print("POLY",end=" ")
print(pearson(Y,PolyPreds))
```

```
Pearson Correlation
RBF 0.803605104878
LINEAR 0.651948964823
POLY 0.631786132189
```

Scatter plot for all points

```
lw = 2
#plt.scatter(X, y, color='darkorange', label='data')
plt.plot(sorted(Y), sorted(RbfPreds), color='navy', lw=lw, label='RBF model')
plt.plot(sorted(Y), sorted(LinPreds), color='c', lw=lw, label='Linear model')
plt.plot(sorted(Y), sorted(PolyPreds), color='cornflowerblue', lw=lw, label='Polynomial model')
plt.scatter(Y, RbfPreds, color='navy', lw=lw, label='RBF model')
plt.scatter(Y, LinPreds, color='c', lw=lw, label='Linear model')
plt.scatter(Y, PolyPreds, color='cornflowerblue', lw=lw, label='Polynomial model')
plt.xlabel('data')
plt.ylabel('target')
plt.title('Support Vector Regression')
plt.legend(bbox_to_anchor=(0, 1), loc='lower right', ncol=1)
plt.show()
```



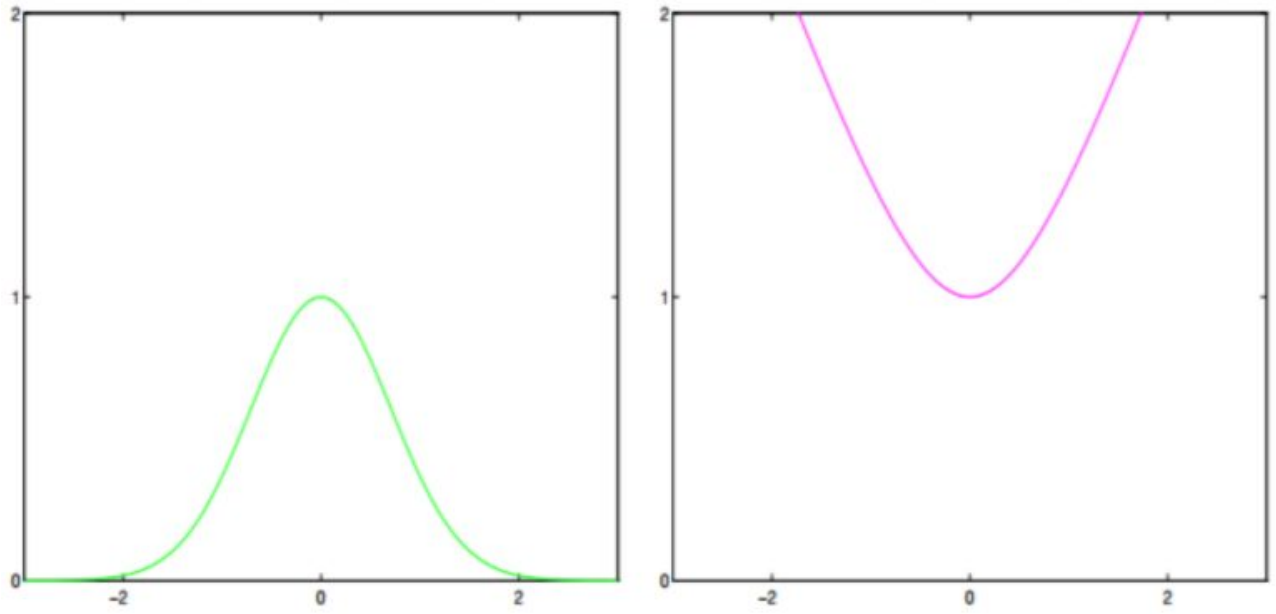
Data is clustered upto 500

Rest is too sparse

8. RBF

Radial basis functions are those which output a real valued function basing their predictions on clusters centered at a point. Hence the name.

It has given the best results so far for this dataset.



Gaussian and Multiquadratic Radial Basis Functions

9. CONCLUSION

We are able to achieve a pearson correlation of 0.8 with an rbf based kernel after hyper parameter tuning. Although this means almost all the data is well represented, the graph shows otherwise. The data is unevenly sparse after 500 and is exceedingly clustered before. Hence a simple SVM might not be able to represent it well even after tuning. Considering more parameters or applying transfer learning on similar data in different time frames can help.

10. REFERENCES

- https://pubs.usgs.gov/wri/wri004139/pdf/wrir00-4139.pdf
- http://green2.kingcounty.gov/streamsdata/watershedinfo.aspx?Locator=B319
- http://www.ecy.wa.gov/programs/eap/fw_riv/index.html
- https://en.wikipedia.org/wiki/Support_vector_machine#Regression
- http://scikit-learn.org/stable/modules/svm.html#svr
- http://green2.kingcounty.gov/streamsdata/Bacteria.aspx?Locator=B319
- **Jagadeesh Anmala**, O.W.Meier, A.J. Meier and S.Grubbs 2015, A GIS and an Artificial Neural Network Based Water Quality Model for a Stream Network in Upper Green River Basin, Kentucky, USA, ASCE, *Journal of Environmental Engineering*, 141(5), 04014082 DOI: 10.1061/(ASCE)EE. 1943-7870.0000801.