

From Linear Regression to Support Vector

Shuai Zhao

2016/10/18

Part 1: Mathematical Elaboration

1. Problem Definition:.....	2
2. Starting with LGR.....	2
3. Think more about $h\theta x = f\theta T x$	2
4. Functional Margin and Geometrical Margin	3
5. Maximum Margin Classifier	3
6. Optimization Problem	3
6.1 LR.....	4
6.2 SVM	4
6.3 SVR.....	4
6.4 LGR.....	5
7. References:.....	5

Part 2: Experimental Tutorial

1. Experiment Environment.....	6
1.1 Python 3.5.2 (https://www.python.org/).....	6
1.2 scikit-learn package (http://scikit-learn.org/stable/index.html).....	6
2. Data Set for Classification.....	6
2.1 Source Link	6
2.2 Data Set Information.....	6
2.3 Attribute Information.....	6
2.4 Understanding Data Set	7
3. Logistic Regression vs SVM	7
3.1 Data Pre-clean.....	7
3.2 Data Import to Python and Process Missing values	8
3.3 Split Train and Test Data.....	9
3.4 Logistic Regression with scikit learn	10
3.5 SVM with scikit learn.....	10
3.6 Analysis of Logistic Regression and SVM.....	11
4. Data Set for Regression.....	11
4.1 Source Link	11
4.2 Data Set Information.....	11
4.3 Attribute Information.....	12
5. Linear Regression vs SVR	12
5. 1 Data Import to Python	12
5.2 Preparing Train and Test Data	13
5.3 SVR model with scikit learn.....	14
5.4 Linear Regression with scikit learn	15
5.5 Analysis of Linear Regression and SVR.....	16
6. Limitation	16

Part 1: Mathematical Elaboration

1. Problem Definition:

Regression analysis is defined as a statistical process for estimating the relationships among variables (https://en.wikipedia.org/wiki/Regression_analysis). The focus is on the relationship between a dependent variable (DV) and one or more independent variables (or 'predictors'). From this perspective of view, SVM, logistic regression (LGR), linear regression (LR), and SVR are all regression models.

But traditionally, LR and SVR are regarded as regression methods and SVM and LGR are regarded as classification models. Especially, SVM/LGR is used when DVs are binary values and LR/SVR is used when DVs are continuous values. Also, LGR is regarded as a special case of LR. SVR is regarded as an adaption of SVM as well.

Thus what is the difference between SVM and LGR? And what is the difference between LR and SVR? This report explores the relationship between those four data mining methods.

2. Starting with LGR

LGR is also called Sigmoid Regression because of the usage of sigmoid function:

$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

Take a look at the above function and we find that $h_{\theta}(x)$ is determined by $\theta^T x$ only. When

- $\theta^T x > 0 \rightarrow h_{\theta}(x) > 0.5 \rightarrow$ the result is (+)
- $\theta^T x < 0 \rightarrow h_{\theta}(x) < 0.5 \rightarrow$ the result is (-)

θ^T is the transpose of a parameter vector. And it is what we want to fit. When training, we want $\theta^T x \gg 0$ when the result is (+), and want $\theta^T x \ll 0$ when the result is (-). Thus in fact $h_{\theta}(x)$ is a function of $\theta^T x$. In other words, $h_{\theta}(x) = f(\theta^T x)$

3. Think more about $h_{\theta}(x) = f(\theta^T x)$

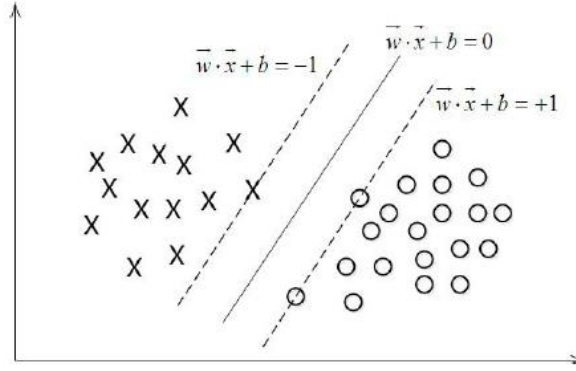
$$\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n (x_0 = 1)$$

Use b to replace θ_0 . Therefore:

$$\theta^T x = w^T x + b$$

When the result is (+), let $h_{\theta}(x) = +1$

When the result is (-), let $h_{\theta}(x) = -1$



4. Functional Margin and Geometrical Margin

Functional Margin is defined as:

$$\hat{\gamma} = y(w^T x + b) = yf(x)$$

This is because we can judge the correctness of our classification by comparing the signs of y and $\theta^T x + b$. Thus our objective is to get the minimum of $\hat{\lambda}_i$.

$$\hat{\gamma} = \min \hat{\gamma}_i, \quad (i = 1, \dots, n)$$

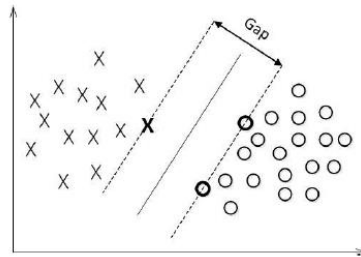
But there is a problem of functional margin. If we change w and b at the same ratio (for example, to be $2w$ and $2b$), $f(x)$ is doubled but the hyperplane is not changed.

Thus there comes the Geometrical Margin, which is defined as:

$$\tilde{\gamma} = y\gamma = \frac{\hat{\gamma}}{\|w\|}$$

5. Maximum Margin Classifier

Because we want to maximize the gap between support vector and hyperplane.



Mathematically, Our objective is $\max \tilde{\gamma}$.

Due to $\hat{\gamma} \leq \hat{\gamma}_i = y_i(\theta^T x + b)$, $i = 1, 2, \dots, n$ and $y_i(\theta^T x + b) \geq 1$, $i = 1, 2, \dots, n$, the objective function is transformed as

$$\max \frac{1}{\|w\|}, \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1, i = 1, \dots, n$$

6. Optimization Problem

Mathematically, all those four models are turned into optimization problems. And the core

difference among them is the objective function and the restriction. I use objective functions and loss functions interchangeably in this report. Loss Function = Loss Term + Regularization Term. Regularization term is used to avoid over-fitting. We can only consider the loss term in this case.

6.1 LR

The objective function for LR is:

$$\min \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i - b)^2$$

The commonest way to fit linear regression models is to use the least square approach.

6.2 SVM

The objective function for SVM is as follows:

$$\max \frac{1}{\|w\|}, \quad s.t. \quad y_i(w^T x_i + b) \geq 1, i = 1, \dots, n$$

This function is equal to:

$$\min \frac{1}{2} \|w\|^2, \quad s.t. \quad y_i(w^T x_i + b) \geq 1, i = 1, \dots, n$$

The solution is firstly to get its dual problem based on Lagrange Duality,

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i (y_i (w^T x_i + b) - 1)$$

Then the objective function is changed to:

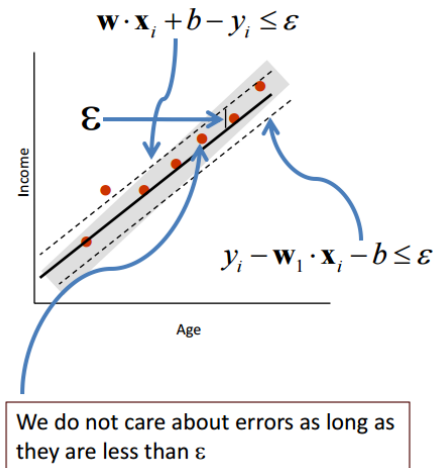
$$\theta(w) = \max_{\alpha_i \geq 0} \mathcal{L}(w, b, \alpha)$$

Till now, we can find that this is a nonlinear programming question and more detailed solutions is not covered in this report.

6.3 SVR

The objective function of SVR ($\min \frac{1}{2} \|w\|^2$) is equal to the objective function of SVM ($\max \frac{1}{\|w\|}$). The difference in SVR is that we want to find a function, $f(x)$, with at most ε -deviation from target y .

$$\begin{aligned} \min \quad & \frac{1}{2} \|\mathbf{w}\|^2 \\ s.t. \quad & y_i - \mathbf{w}_1 \cdot \mathbf{x}_i - b \leq \varepsilon; \\ & \mathbf{w}_1 \cdot \mathbf{x}_i + b - y_i \leq \varepsilon; \end{aligned}$$



6.4 LGR

$$P(y = 1 | x) = \pi(x) = \frac{1}{1 + e^{-g(x)}} \quad P(y = 0 | x) = 1 - P(y = 1 | x) = 1 - \frac{1}{1 + e^{-g(x)}} = \frac{1}{1 + e^{g(x)}}$$

Thus the odds of experiencing an event is:

$$\frac{P(y = 1 | x)}{P(y = 0 | x)} = \frac{p}{1 - p} = e^{g(x)}$$

Use log function to both sides:

$$\ln\left(\frac{p}{1 - p}\right) = g(x) = w_0 + w_1 x_1 + \dots + w_n x_n$$

Finally, we can use the Maximum Likelihood Estimation to solve this problem and the Likelihood function is as follows:

$$L(w) = \prod_{i=1}^m (\pi(x_i))^{y_i} (1 - \pi(x_i))^{1 - y_i}$$

7. References:

1. Introduction to SVM (Three levels to understand SVM)
http://blog.csdn.net/v_july_v/article/details/7624837
2. SVM Kernel
<https://www.cs.cmu.edu/~ggordon/SVMs/new-svms-and-kernels.pdf>
<https://people.cs.pitt.edu/~milos/courses/cs3750-Fall2007/lectures/class-kernels.pdf>
http://www.doc.ic.ac.uk/~mpd37/teaching/2014/ml_tutorials/2014-02-26-kernels.pdf
3. Book: Kernel Methods for Pattern Analysis:
<http://read.pudn.com/downloads167/ebook/769401/Kernel%20Methods%20for%20Pattern%20Analysis.pdf>
4. Zhihu Questions
<https://www.zhihu.com/question/24627666>
<https://www.zhihu.com/question/35602879>
5. Quora Questions
<https://www.quora.com/What-are-Kernels-in-Machine-Learning-and-SVM>

6. SVR vs LR

http://cs.adelaide.edu.au/~chhshen/teaching/ML_SVR.pdf

<https://www.zhihu.com/question/21704547>

7. Linear Regression in Matrix Form:

<http://www.stat.purdue.edu/~jennings/stat514/stat512notes/topic3.pdf>

8. Logistic Regression Loss Function:

<http://www.robots.ox.ac.uk/~az/lectures/ml/2011/lect4.pdf>

<http://blog.csdn.net/ariessurfer/article/details/41310525>

Part 2: Experimental Tutorial

1. Experiment Environment

1.1 Python 3.5.2 (<https://www.python.org/>)

Justification for Python: Compared with Python, Java is too verbose and stubborn. R stores every variable into memory and can't handle relatively big data. Abundance of Python packages is also another bonus.

1.2 scikit-learn package (<http://scikit-learn.org/stable/index.html>)

scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. (<https://en.wikipedia.org/wiki/Scikit-learn>)

2. Data Set for Classification

2.1 Source Link

<http://archive.ics.uci.edu/ml/datasets/Credit+Approval>

2.2 Data Set Information

This file concerns credit card applications. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data.

This dataset is interesting because there is a good mix of attributes -- continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values.

2.3 Attribute Information

A1: b, a.

A2: continuous.

A3: continuous.

A4: u, y, l, t.

A5: g, p, gg.

A6: c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff.

A7: v, h, bb, j, n, z, dd, ff, o.

A8: continuous.

A9: t, f.

A10: t, f.

A11: continuous.

A12: t, f.

A13: g, p, s.

A14: continuous.

A15: continuous.

A16: +, - (class attribute)

2.4 Understanding Data Set

A16 is the dependent variable (response variable)

A1 – A15 are the independent variables (predict variables)

In order to simplify the data clean process, we discard the categorical variables within A1 - A15. In other words, I discard A1, A4, A5, A6, A7, A9, A10, A12, A13.

****Alternatively, if we want to take categorical variables into consideration, we can adopt the one hot encoding (<https://en.wikipedia.org/wiki/One-hot>).**

3. Logistic Regression vs SVM

3.1 Data Pre-clean

Open file “crx.data.csv” with Excel, and delete the categorical independent variables. Then replace + by 1, replace – by 0.

	A	B	C	D	E	F	G
1	30.83	0	1.25	1	202	0	1
2	58.67	4.46	3.04	6	43	560	1
3	24.5	0.5	1.5	0	280	824	1
4	27.83	1.54	3.75	5	100	3	1
5	20.17	5.625	1.71	0	120	0	1
6	32.08	4	2.5	0	360	0	1
7	33.17	1.04	6.5	0	164	31285	1
8	22.92	11.585	0.04	0	80	1349	1
9	54.42	0.5	3.96	0	180	314	1
10	42.5	4.915	3.165	0	52	1442	1
11	22.08	0.83	2.165	0	128	0	1
12	29.92	1.835	4.335	0	260	200	1
13	38.25	6	1	0	0	0	1
14	48.08	6.04	0.04	0	0	2690	1

3.2 Data Import to Python and Process Missing values

```
import csv
import numpy as np
import sklearn.linear_model as sklr
from sklearn.cross_validation import train_test_split
from sklearn import metrics

with open('/home/shuai/Documents/courses/cs634/crx.data-v1.1.csv') as f:
    reader = csv.reader(f)
    my_list = list(reader)
f.closed
print("The number of observations is:", len(my_list))

#Delete records that have missing attributes '?'
#Use descending order to avoid the change of item index
for i in range(len(my_list) - 1, -1, -1):
    for j in range(len(my_list[1])):
        if my_list[i][j] == '?':
            del my_list[i]
```



```
print("After deleting observations containing missing values, the
number of observations is:", len(my_list))
```

The output is:

```
Python Console
from sklearn import svm
with open('/home/shuai/Documents/courses/cs634/crx.data-v1.1.csv') as f:
    reader = csv.reader(f)
    my_list = list(reader)
f.closed
print("The number of observations is:", len(my_list))
#Delete records that have missing attributes '?'
# Use descending order to avoid the change of item index
for i in range(len(my_list) - 1, -1, -1):
    for j in range(len(my_list[i])):
        if my_list[i][j] == '?':
            del my_list[i]
print("After deleting observations containing missing values, the total number of observations is:", len(my_list))

The number of observations is: 690
After deleting observations containing missing values, the total number of observations is: 666

>>>
```

3.3 Split Train and Test Data

```
#Split training data and test data
#Turn string to float type
my_list2 = [list(map(float, my_list[i])) for i in
range(len(my_list))]
data1 = np.array(my_list2)
X = data1[:, :-1]
Y = data1[:, -1:]
Y = Y.astype(int)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.3, random_state=0)
print("Training Data:", X_train.shape[0], "\nTest Data",
X_test.shape[0])
```

The output is:

```
Python Console
print("After deleting observations containing missing values, the total number of observations is:", len(my_list))

The number of observations is: 690
After deleting observations containing missing values, the total number of observations is: 666
>>> #Split training data and test data
#Turn string to float type
my_list2 = [list(map(float, my_list[i])) for i in range(len(my_list))]
data1 = np.array(my_list2)
X = data1[:, :-1]
Y = data1[:, -1:]
Y = Y.astype(int)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=0)
print("Training Data:", X_train.shape[0], "\nTest Data", X_test.shape[0])

Training Data: 466
Test Data 200

>>>
```

```
#Model Training

lr = sklearn.LinearRegression()

lr.fit(X_train, Y_train)

#Predict class labels for the test set

predicted = lr.predict(X_test)

predicted1 = np.ravel(predicted)

predicted2 = np.array([round(predicted1[i]) for i in
range(len(predicted1))])

print("The predicted values are:", predicted2)

print("The Accuracy Score of Logistic Regression is:",
metrics.accuracy_score(Y_test, predicted2))
```

The output is:

```
Python Console
print("The predicted values are:", predicted2)
print("The Accuracy Score of Logistic Regression is:", metrics.accuracy_score(Y_test, predicted2))
```

The predicted values are: [1. 0. 1. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]

The Accuracy Score of Logistic Regression is: 0.75

And we can tell the accuracy score for LR is 0.75

3.5 SVM with scikit learn

The data clean and train-test data split are the same with Logistic Regression.

```
#Model Training
clf = svm.SVC()
Y_train2 = np.ravel(Y_train)
clf.fit(X_train, Y_train2)

#Predict class labels for the test set
predicted = clf.predict(X_test)
predicted1 = np.ravel(predicted)
predicted2 = np.array([round(predicted1[i]) for i in
range(len(predicted1))])
print("The predicted values are:", predicted2)
```

```
print("The Accuracy Score of SVM is:", metrics.accuracy_score(Y_test,
predicted2))
```

The output is:

[illegible]

3.6 Analysis of Logistic Regression and SVM

Based on the results, Logistic Regression outperforms SVM method.

4. Data Set for Regression

4.1 Source Link

<http://archive.ics.uci.edu/ml/datasets/BlogFeedback>

4.2 Data Set Information

This data originates from blog posts. The raw HTML-documents of the blog posts were crawled and processed. The prediction task associated with the data is the prediction of the number of comments in the upcoming 24 hours. In order to simulate this situation, we choose a basetime (in the past) and select the blog posts that were published at most 72 hours before the selected base date/time. Then, we calculate all the features of the selected blog posts from the information that was available at the basetime, therefore each instance corresponds to a blog post. The target is the number of comments that the blog post received in the next 24 hours relative to the basetime.

In the train data, the basetimes were in the years 2010 and 2011. In the test data the basetimes were in February and March 2012. This simulates the real-world situation in which training data from the past is available to predict events in the future.

The train data was generated from different basetimes that may temporally overlap. Therefore, if you simply split the train into disjoint partitions, the underlying time intervals may overlap. Therefore, the you should use the provided, temporally disjoint train and test splits in order to ensure that the evaluation is fair.

4.3 Attribute Information

1...50: Average, standard deviation, min, max and median of the Attributes

51...60 for the source of the current blog post With source we mean the blog on which the post appeared. For example, myblog.blog.org would be the source of the post myblog.blog.org/post_2010_09_10

51: Total number of comments before basetime 52: Number of comments in the last 24 hours before the basetime

53: Let T1 denote the datetime 48 hours before basetime, Let T2 denote the datetime 24 hours before basetime. This attribute is the number of comments in the time period between T1 and T2

54: Number of comments in the first 24 hours after the publication of the blog post, but before basetime

55: The difference of Attribute 52 and Attribute 53

56...60: The same features as the attributes 51...55, but features 56...60 refer to the number of links (trackbacks), while features 51...55 refer to the number of comments.

61: The length of time between the publication of the blog post and basetime

62: The length of the blog post

63...262: The 200 bag of words features for 200 frequent words of the text of the blog post

263...269: binary indicator features (0 or 1) for the weekday (Monday...Sunday) of the basetime

270...276: binary indicator features (0 or 1) for the weekday (Monday...Sunday) of the date of publication of the blog post

277: Number of parent pages: we consider a blog post P as a parent of blog post B, if B is a reply (trackback) to blog post P.

278...280: Minimum, maximum, average number of comments that the parents received

281: The target: the number of comments in the next 24 hours (relative to basetime)

5. Linear Regression vs SVR

5.1 Data Import to Python

```
import csv
import random
import numpy as np
import sklearn.linear_model as skl
from sklearn.metrics import mean_squared_error
from sklearn.svm import SVR

with
open('/home/shuai/Documents/courses/cs634/BlogFeedback/blogData_train
.csv') as f:
    reader = csv.reader(f)
    my_list = list(reader)
f.closed
```

```
print("The number of observations is:", len(my_list))
```

The output is

```
Python Console
PyDev console: starting.
Python 3.5.2 (default, Jul 5 2016, 12:43:10)
[GCC 5.4.0 20160609] on linux
>>> import csv
import random
import numpy as np
import sklearn.linear_model as skl
from sklearn.metrics import mean_squared_error
from sklearn.svm import SVR
with open('/home/shuai/Documents/courses/cs634/BlogFeedback/blogData_train.csv') as f:
    reader = csv.reader(f)
    my_list = list(reader)
f.closed
print("The number of observations is:", len(my_list))
The number of observations is: 52397
```

5.2 Preparing Train and Test Data

Due to the training data has large number of observations. I randomly select 30% of data to train.

```
#Random select 30% data
my_list2 = []
for i in range(len(my_list)):
    if random.random() < 0.3:
        my_list2.append(my_list[i])
del my_list
```

```
print("After Random selection, the total number of samples is:",
len(my_list2))
```

The output is:

```
Python Console
with open('/home/shuai/Documents/courses/cs634/BlogFeedback/blogData_train.csv') as f:
    reader = csv.reader(f)
    my_list = list(reader)
f.closed
print("The number of observations is:", len(my_list))
The number of observations is: 52397
>>> #Random select 30% data
my_list2 = []
for i in range(len(my_list)):
    if random.random() < 0.3:
        my_list2.append(my_list[i])
del my_list
print("After Random selection, the total number of samples is:", len(my_list2))
After Random selection, the total number of samples is: 15626
```

Next prepare the train and test data:

```
#Training Data
```

```

#Turn string to float type
my_list3 = [list(map(float, my_list2[i])) for i in
range(len(my_list2))]

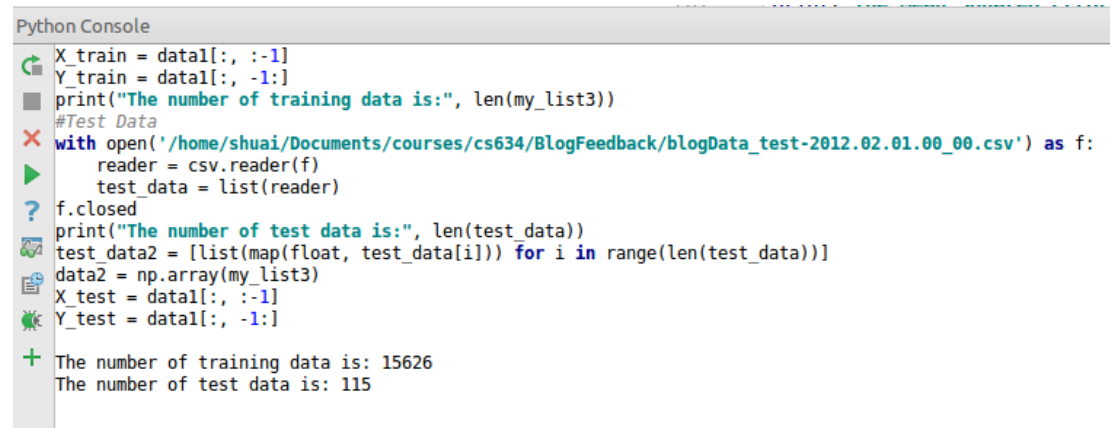
data1 = np.array(my_list3)
X_train = data1[:, :-1]
Y_train = data1[:, -1:]
print("The number of test data is:", len(my_list3))

#Test Data
with
open('/home/shuai/Documents/courses/cs634/BlogFeedback/blogData_test-
2012.02.01.00_00.csv') as f:
    reader = csv.reader(f)
    test_data = list(reader)
f.closed
print("The number of test data is:", len(test_data))

test_data2 = [list(map(float, test_data[i])) for i in
range(len(test_data))]
data2 = np.array(my_list3)
X_test = data1[:, :-1]
Y_test = data1[:, -1:]

```

The output is:



```

Python Console
X_train = data1[:, :-1]
Y_train = data1[:, -1:]
print("The number of training data is:", len(my_list3))
#Test Data
with open('/home/shuai/Documents/courses/cs634/BlogFeedback/blogData_test-2012.02.01.00_00.csv') as f:
    reader = csv.reader(f)
    test_data = list(reader)
f.closed
print("The number of test data is:", len(test_data))
test_data2 = [list(map(float, test_data[i])) for i in range(len(test_data))]
data2 = np.array(my_list3)
X_test = data1[:, :-1]
Y_test = data1[:, -1:]
+ The number of training data is: 15626
  The number of test data is: 115

```

5.3 SVR model with scikit learn

```

#Model Training
clf = SVR(C=1.0, epsilon=0.2)
Y_train2 = np.ravel(Y_train)
clf.fit(X_train, Y_train2)

#Model Prediction

```

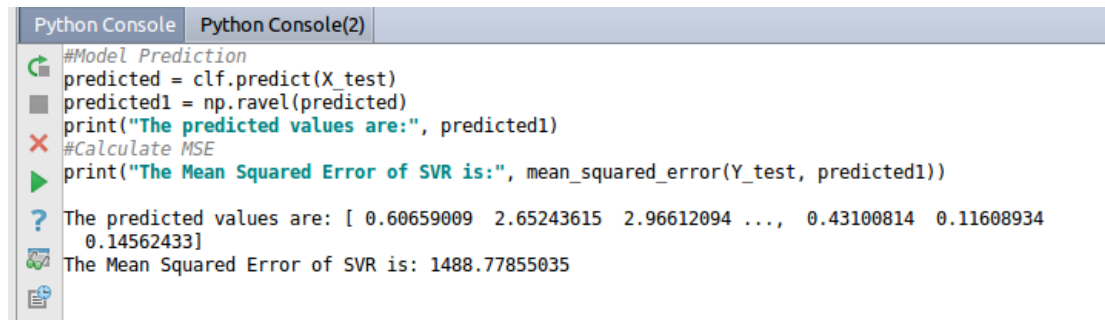
```

predicted = clf.predict(X_test)
predicted1 = np.ravel(predicted)
print("The predicted values are:", predicted1)

#Calculate MSE
print("The Mean Squared Error of SVR is:", mean_squared_error(Y_test,
predicted1))

```

And the output is:



```

Python Console Python Console(2)
#Model Prediction
predicted = clf.predict(X_test)
predicted1 = np.ravel(predicted)
print("The predicted values are:", predicted1)
#Calculate MSE
print("The Mean Squared Error of SVR is:", mean_squared_error(Y_test, predicted1))
The predicted values are: [ 0.60659009  2.65243615  2.96612094 ...,  0.43100814  0.11608934
 0.14562433]
The Mean Squared Error of SVR is: 1488.77855035

```

5.4 Linear Regression with scikit learn

```

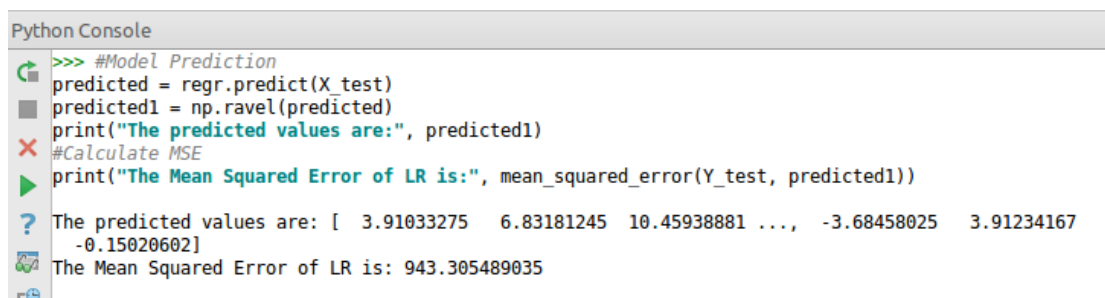
from sklearn import linear_model
#Model Training
regr = linear_model.LinearRegression()
Y_train2 = np.ravel(Y_train)
regr.fit(X_train, Y_train2)

#Model Prediction
predicted = regr.predict(X_test)
predicted1 = np.ravel(predicted)
print("The predicted values are:", predicted1)

#Calculate MSE
print("The Mean Squared Error of SVR is:", mean_squared_error(Y_test,
predicted1))

```

The output is:



```

Python Console
>>> #Model Prediction
predicted = regr.predict(X_test)
predicted1 = np.ravel(predicted)
print("The predicted values are:", predicted1)
#Calculate MSE
print("The Mean Squared Error of LR is:", mean_squared_error(Y_test, predicted1))
The predicted values are: [ 3.91033275  6.83181245 10.45938881 ..., -3.68458025  3.91234167
-0.15020602]
The Mean Squared Error of LR is: 943.305489035

```

5.5 Analysis of Linear Regression and SVR

Linear Regression outperforms SVR in this data set. My most interesting finding is that it takes much more time to run SVR compared with LR when the data is large.

6. Limitation

- This experiment discards the categorical independent variables in SVM/LGR experiment for simplification. One hot coding for categorical would be better.
- In the SVM/LGR experiment, I delete the observations which contain missing values. There are also some other better ways to process missing values.
- In order to simplify the experiment process, I use almost all the default parameters in the methods. It would improve the performance by modifying some parameters in the methods, especially for SVM/SVR.