

User Guide

Problem statement:

The text file sample_data.txt contains a dataset with a response column, y, and 14 columns of features, X1-X14.

Create a model for predicting values of y for new values of X1-X14 that can be accessible as a web service e.g. through HTTP GET or POST requests.

Introduction:

I built two web applications, including a simple version with 'GET' request and a user friendly website version with 'POST' request. Both applications are using the same model, an SVM model.

API cookbook:

Environment:

OSX or Linux

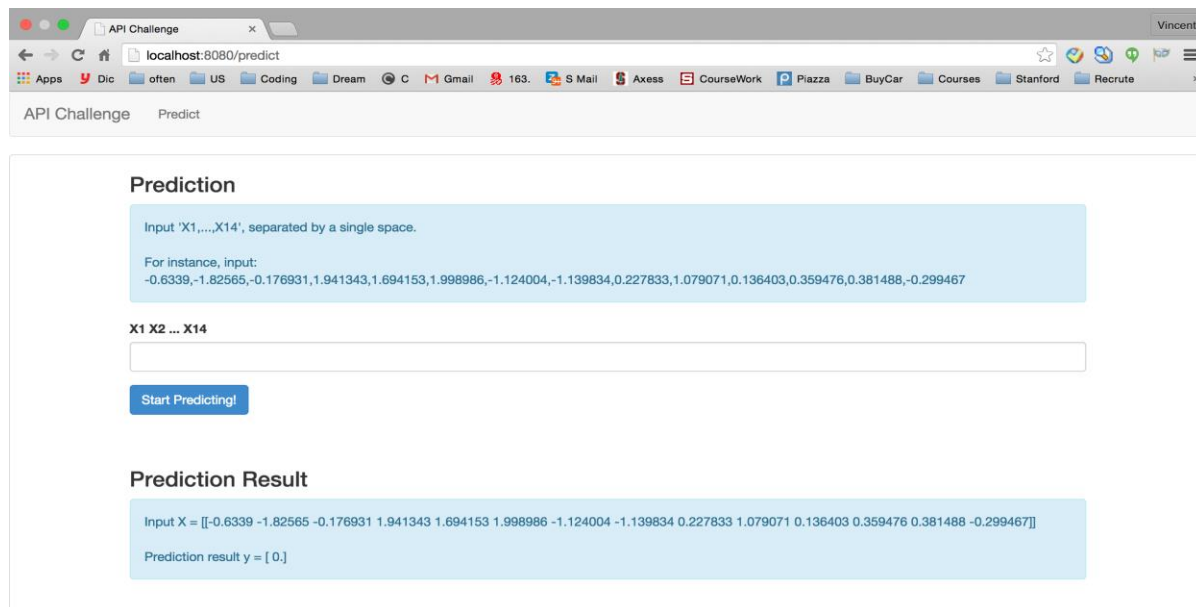
Python (≥ 2.6 or ≥ 3.3),

NumPy ($\geq 1.6.1$),

SciPy (≥ 0.9).

Part 1 Api-post: using post to serve this model

1. Set up the web service. Open terminal, go to directory 'api-post', and run API_Challenge.py.
\$ cd api-post/
\$ python API_Challenge.py
2. Keep API_Challenge.py running, and then open a web browser (i.e Chrome), go to this url:
<http://localhost:8080/predict>
3. You should see a web page like this:

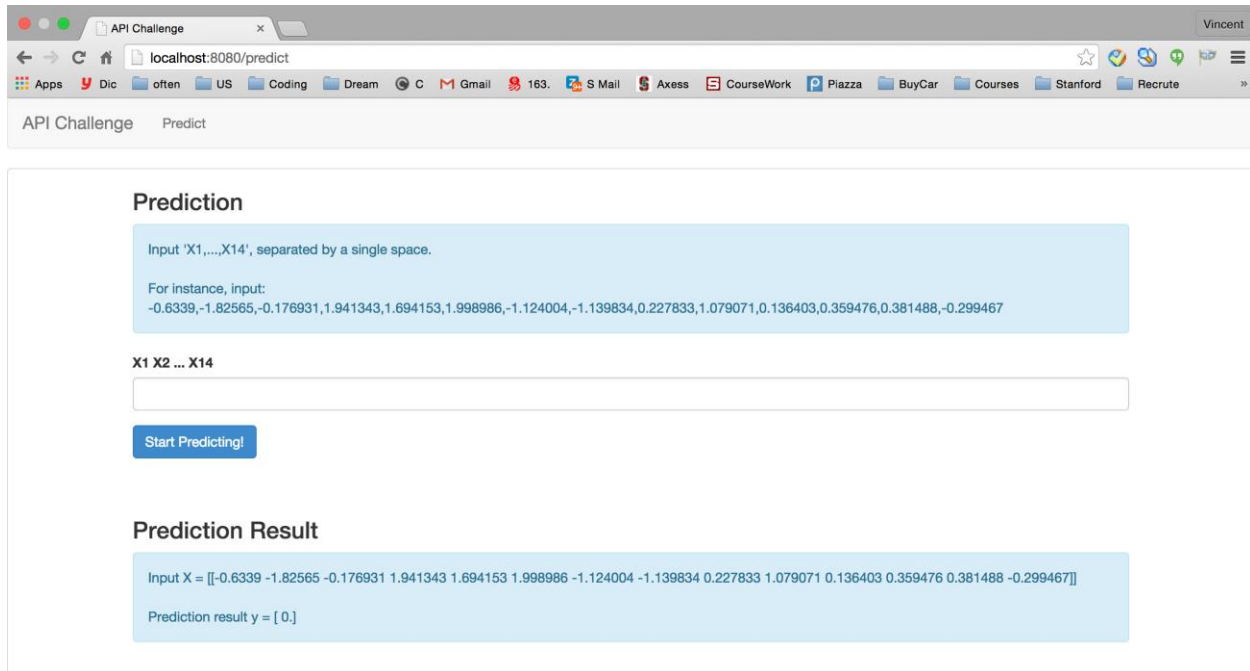


4. Input 'X1,...,X14', separated by a single ','.

For instance, input:

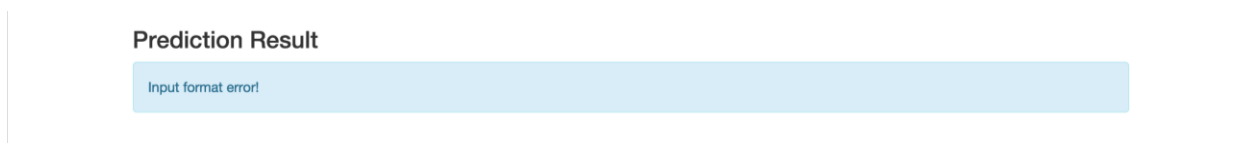
-0.6339,-1.82565,-0.176931,1.941343,1.694153,1.998986,-1.124004,-1.139834,0.227833,1.079071,0.136403,0.359476,0.381488,-0.299467

5. Click 'Start Prediction', then it should return this page:



The screenshot shows a web browser window titled 'API Challenge' with the address bar at 'localhost:8080/predict'. The page has a header with 'API Challenge' and 'Predict' links. The main content area is titled 'Prediction' and contains a light blue box with instructions: 'Input 'X1,...,X14', separated by a single space. For instance, input: -0.6339,-1.82565,-0.176931,1.941343,1.694153,1.998986,-1.124004,-1.139834,0.227833,1.079071,0.136403,0.359476,0.381488,-0.299467'. Below this is a text input field labeled 'X1 X2 ... X14' and a blue button labeled 'Start Predicting!'. The 'Prediction Result' section shows a light blue box with the output: 'Input X = [-0.6339 -1.82565 -0.176931 1.941343 1.694153 1.998986 -1.124004 -1.139834 0.227833 1.079071 0.136403 0.359476 0.381488 -0.299467]' and 'Prediction result y = [0.]'.

6. If you input X in a wrong format, it will return an warning 'Input format error!'. You can only get the prediction by input X with exactly correct format.



The screenshot shows the 'Prediction Result' section of the web interface. A light blue box displays the message 'Input format error!'.

Part 2 Api-get: using get to serve this model

1. Set up the web service. Open terminal, go to directory 'api-get', and run API_Challenge.py.

```
$ cd api-post/
```

```
$ python API_Challenge.py
```

2. Since this is a simple version of api implemented with GET request, you can just use an url to send the request and get the result. Keep API_Challenge.py running, and then open a web browser (i.e Chrome), go to this url:

http://localhost:8080/predict?X_input={your input X}

X_input represents X1...X14, separated by a single ','.

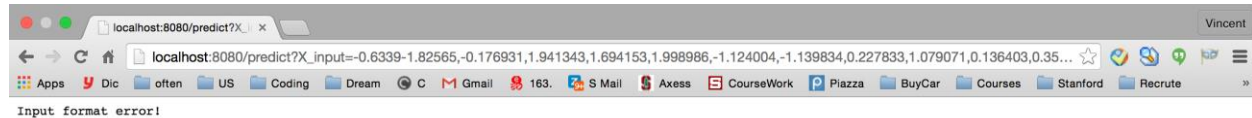
For instance, url:

http://localhost:8080/predict?X_input=-0.6339,-1.82565,-0.176931,1.941343,1.694153,1.998986,-1.124004,-1.139834,0.227833,1.079071,0.136403,0.359476,0.381488,-0.299467

Then, you should get the prediction result:



4. If you input X in a wrong format, it will return an warning 'Input format error!'. You can only get the



prediction by input X with an exactly correct format.

Part 3 Model performance:

Since this is an API challenge, not a model performance challenge, I didn't spend much time on model selection and optimization. If it was required, I could further optimize his model.

We have 10000 samples with 14 features and we want to classify them to two classes. Although, in general, with a low dimensional feature space and relatively larger sample set, SVM might have a better performance on this kind of two-class classification problem, I still implemented many other models. I also implemented logistic regression, k nearest neighbor, principal component analysis. After all, SVM has a better performance, with a polynomial kernel, degree 5, and uniformed sample weight 1.

You can go to file 'model-generate', and run 'model_generate.py'.
The performance for this model is:

Confusion matrix:

		Prediction	
		0	1
Real	0	1819	52
	1	121	8

Over all accuracy: 91.35 %

Fasle positive rate: 2.77926242651 %

Fasle negative rate: 93.7984496124 %

Over all error rate: 8.65 %

The false nagetive rate is too high, but the overall accuracy is reasonable. Although there are still many ways to come up with a better model, without the detail of this problem, it's hard to assume what kind of trade off we should make to further optimize this model. In this 'API' challenge, we just focus more on API.

If there were more time, I could further optimize his model.