

Homework 1

Gnana Deep Pallela

011822369

Algorithm: Support Vector Machines (SVM)

Languages: Python 3.6.1, HTML

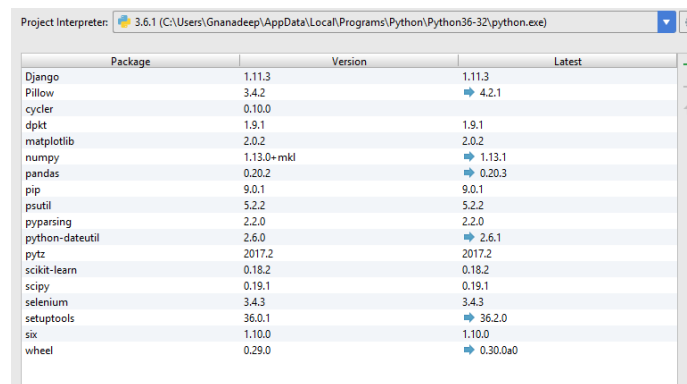
Web framework: Django 1.11.3

Platforms: Windows, Web

SVM:

It is an algorithm which classifies the data into different sets with respect to the hyper plane. In this implementation, the Iris data set is taken (From reference 1) and the data coordinates are plotted as a point in Two - Dimensional space.

The implementation of this algorithm is already available in the Scikit-learn Library. Therefore, this library is imported in the program. And, the other packages such as pip, Django, Pillow, dpkt, matplotlib, numpy, pandas, psutil, etc. are also installed.



The screenshot shows the 'Project Interpreter' window for Python 3.6.1. It displays a table of installed packages with their current versions and the latest available versions. Packages listed include Django, Pillow, cyciler, dpkt, matplotlib, numpy, pandas, pip, psutil, pyparsing, python-dateutil, pytz, scikit-learn, scipy, selenium, setuptools, six, and wheel.

Package	Version	Latest
Django	1.11.3	1.11.3
Pillow	3.4.2	4.2.1
cyciler	0.10.0	
dpkt	1.9.1	1.9.1
matplotlib	2.0.2	2.0.2
numpy	1.13.0+mkl	1.13.1
pandas	0.20.2	0.20.3
pip	9.0.1	9.0.1
psutil	5.2.2	5.2.2
pyparsing	2.2.0	2.2.0
python-dateutil	2.6.0	2.6.1
pytz	2017.2	2017.2
scikit-learn	0.18.2	0.18.2
scipy	0.19.1	0.19.1
selenium	3.4.3	3.4.3
setuptools	36.0.1	36.2.0
six	1.10.0	1.10.0
wheel	0.29.0	0.30.0a0

Test Data:

The test data is same for both the implementations

	Sepal.Length	Sepal.Width	Species
1	5.1	3.5	0
2	4.9	3	0
3	4.7	3.2	0
4	4.6	3.1	0
5	5	3.6	0
6	5.4	3.9	0

7	4.6	3.4	0
8	5	3.4	0
9	4.4	2.9	0
10	4.9	3.1	0
11	5.4	3.7	0
12	4.8	3.4	0
13	4.8	3	0
14	4.3	3	0
15	5.8	4	0
16	5.7	4.4	0
17	5.4	3.9	0
18	5.1	3.5	0
19	5.7	3.8	0
20	5.1	3.8	0
21	5.4	3.4	0
22	5.1	3.7	0
23	4.6	3.6	0
24	5.1	3.3	0
25	4.8	3.4	0
26	5	3	0
27	5	3.4	0
28	5.2	3.5	0
29	5.2	3.4	0
30	4.7	3.2	0
31	4.8	3.1	0
32	5.4	3.4	0
33	5.2	4.1	0
34	5.5	4.2	0
35	4.9	3.1	0
36	5	3.2	0
37	5.5	3.5	0
38	4.9	3.6	0
39	4.4	3	0
40	5.1	3.4	0
41	5	3.5	0
42	4.5	2.3	0
43	4.4	3.2	0
44	5	3.5	0
45	5.1	3.8	0

46	4.8	3	0
47	5.1	3.8	0
48	4.6	3.2	0
49	5.3	3.7	0
50	5	3.3	0
51	7	3.2	1
52	6.4	3.2	1
53	6.9	3.1	1
54	5.5	2.3	1
55	6.5	2.8	1
56	5.7	2.8	1
57	6.3	3.3	1
58	4.9	2.4	1
59	6.6	2.9	1
60	5.2	2.7	1
61	5	2	1
62	5.9	3	1
63	6	2.2	1
64	6.1	2.9	1
65	5.6	2.9	1
66	6.7	3.1	1
67	5.6	3	1
68	5.8	2.7	1
69	6.2	2.2	1
70	5.6	2.5	1
71	5.9	3.2	1
72	6.1	2.8	1
73	6.3	2.5	1
74	6.1	2.8	1
75	6.4	2.9	1
76	6.6	3	1
77	6.8	2.8	1
78	6.7	3	1
79	6	2.9	1
80	5.7	2.6	1
81	5.5	2.4	1
82	5.5	2.4	1
83	5.8	2.7	1
84	6	2.7	1

85	5.4	3	1
86	6	3.4	1
87	6.7	3.1	1
88	6.3	2.3	1
89	5.6	3	1
90	5.5	2.5	1
91	5.5	2.6	1
92	6.1	3	1
93	5.8	2.6	1
94	5	2.3	1
95	5.6	2.7	1
96	5.7	3	1
97	5.7	2.9	1
98	6.2	2.9	1
99	5.1	2.5	1
100	5.7	2.8	1
101	6.3	3.3	2
102	5.8	2.7	2
103	7.1	3	2
104	6.3	2.9	2
105	6.5	3	2
106	7.6	3	2
107	4.9	2.5	2
108	7.3	2.9	2
109	6.7	2.5	2
110	7.2	3.6	2
111	6.5	3.2	2
112	6.4	2.7	2
113	6.8	3	2
114	5.7	2.5	2
115	5.8	2.8	2
116	6.4	3.2	2
117	6.5	3	2
118	7.7	3.8	2
119	7.7	2.6	2
120	6	2.2	2
121	6.9	3.2	2
122	5.6	2.8	2
123	7.7	2.8	2

124	6.3	2.7	2
125	6.7	3.3	2
126	7.2	3.2	2
127	6.2	2.8	2
128	6.1	3	2
129	6.4	2.8	2
130	7.2	3	2
131	7.4	2.8	2
132	7.9	3.8	2
133	6.4	2.8	2
134	6.3	2.8	2
135	6.1	2.6	2
136	7.7	3	2
137	6.3	3.4	2
138	6.4	3.1	2
139	6	3	2
140	6.9	3.1	2
141	6.7	3.1	2
142	6.9	3.1	2
143	5.8	2.7	2
144	6.8	3.2	2
145	6.7	3.3	2
146	6.7	3	2
147	6.3	2.5	2
148	6.5	3	2
149	6.2	3.4	2
150	5.9	3	2

Windows implementation:

1. To implement the algorithm on the windows platform, the Pycharm editor is used to write and execute the program. And, the Windows PowerShell can also be used to compile the program.
2. Initially, the 'iris.csv' file is read from the specified location and the data is separated by using the delimiter ",". And, the data type is set as 'float64' format.

```
df = pd.read_csv(location, sep=",", error_bad_lines=False, index_col=False,
dtype='float64')
```

3. In this program, the X_data contains data in 2 columns (i.e. Sepal length and the Sepal width) whereas the Y_target contains the data of species which is taken as 0,1 and 2. (i.e. Setosa, Versicolor and Virginica.

```
X_data = np.array(df.ix[0:150,1:3])
Y_target = np.array(df.ix[0:150,5:])
```

4. In the next step, the data is split into training data and the test data. This is done using the train_test_split function imported from the library.

```
X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_target,
                                                    test_size=validation_size, random_state=seed)
```

5. The SVM classification object is created using the svc algorithm by selecting the desired kernel (i.e. linear, rbf or poly), c and gamma values.

```
clf=svm.SVC(kernel='poly', verbose=True, gamma='auto', C=1.0)
```

6. Then, the data is scaled and the training sets are taken to check the score. If the score is one, then it is the perfect prediction. And, if the score is zero, then the data has no linear relationship.

```
m=clf.fit(X_train,Y_train.ravel())
predictions = clf.predict(X_test)
print(clf.score(X_test, Y_test))
```

7. The mean square error is also calculated to check the accuracy of the prediction. In the further implementation, the mesh is created to plot the decision surface using the contour function.

```
np.mean((clf.predict(X_test)-Y_test)**2)
plt.contourf(xx, yy, Z, color='red', alpha=0.8)
```

8. Finally, the data is plotted in the 2D plane using the scatter function with Sepal_length as X -axis and Sepal_width as Y-axis. And, the figure is displayed.

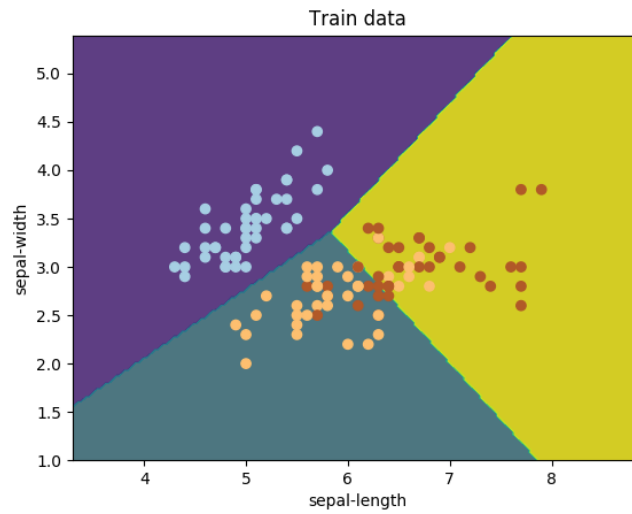
```
plt.scatter(X_train[:, 0], X_train[:, 1], c=Y_train, cmap=plt.cm.Paired)
plt.xlabel('sepal-length')
plt.ylabel('sepal-width')
plt.title("Train data")
plt.show()
```

9. The CPU utilization, Memory utilization and the running time of the program are calculated by importing the libraries such as psutil, os and sys. And, printing the results on the console or the PowerShell.

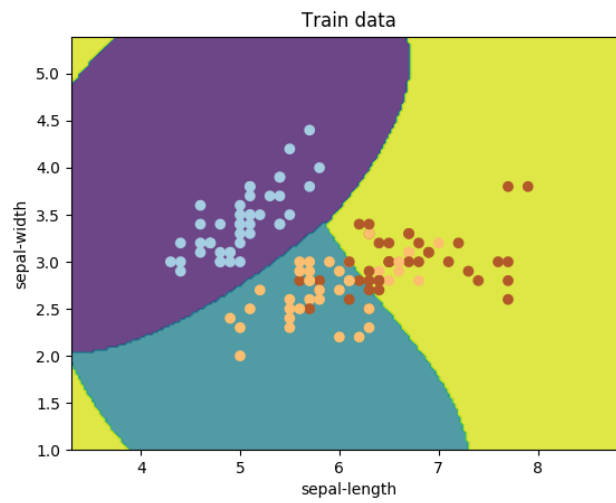
```
start = timeit.default_timer()
stop = timeit.default_timer()
psutil.cpu_percent()
process = psutil.Process(os.getpid())
```

```
mem = process.memory_percent()
```

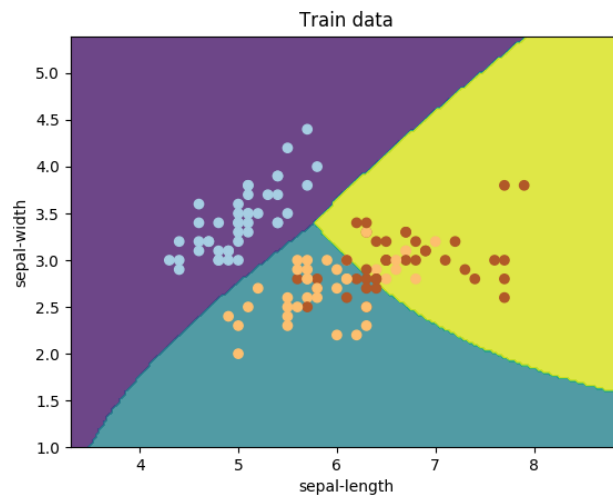
Results:



Linear plot



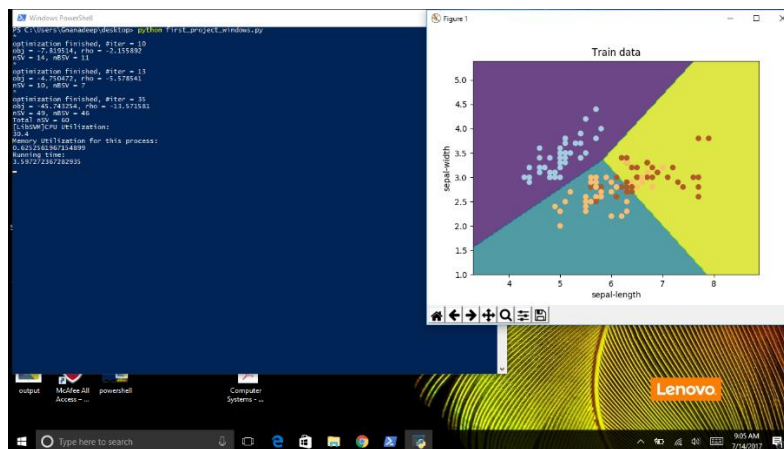
RBF plot



POLY plot

Performance Evaluation:

When the command 'python first_project_windows.py' was entered on the PowerShell the results are displayed.



CPU Utilization:

32.4

Memory Utilization for this process:

0.6277881173439448

Running time:

0.27450519915978444

2. Web implementation using Django framework:

Django is a high-level python web framework that encourages rapid development with ease and clean design. This web framework is best suitable for professionals with well-organized design.

Commands to run on the PowerShell:

1. To start a project

`django-admin startproject project_name`

2. To start the server

`python manage.py runserver`

3. To create an application

`python manage.py startapp app_name`

Steps to create New Project:

1. Create a project
2. Start an application
3. Write your templates
4. Define your views
5. Add pluggable modules
6. Create URL mapping
7. Test Application
8. Deploy Application

Description:

A simple web application is built through which a user can upload a .csv file and the plot is displayed on the new page. Hence, to build the web application Django framework is used which supports python script.

The basic outline of this web application is that it contains two web pages written in html. We have used bootstrap in both the pages to make the pages more responsive. When we use bootstrap, the web page gets the predefined stylesheets and the web pages can dynamically modify to fit in mobile devices. The first page is the loic page, which asks the user to submit the .csv file. And, the second page is to display the plot. To take the input from this page and display the plot on the page, we used html forms. But, we must define the three URL's in the urls.py file shown below in the figure, which was dynamically created when the application was started on the Django framework.

urls.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'firstapp'
]
```

Settings.py

As we require the .csv file to be uploaded, we use html form in the loic page. We are using the get method to accept the input request. Now our duty is to pass this .csv file to our main python script which resides in views.py and execute the code when the submit is clicked by the user.

[illegible]

```

s0="sepal-length"
n='\n'
sd=psutil.cpu_percent()
s1="CPU Utilization: "
s2=str(sd)
s3="Memory Utilization: "
s4=str(mem)
s5="Running time: "
s6=str(stop - start)

s=s1+s2+n+s3+s4+" "+s5+s6
plt.xlabel('sepal-length')
plt.title(s)
#plt.figure(figsize=(20,10))
canvas = FigureCanvas(plt.figure(1))
response = HttpResponse(content_type='image/png')
canvas.print_png(response)
return response

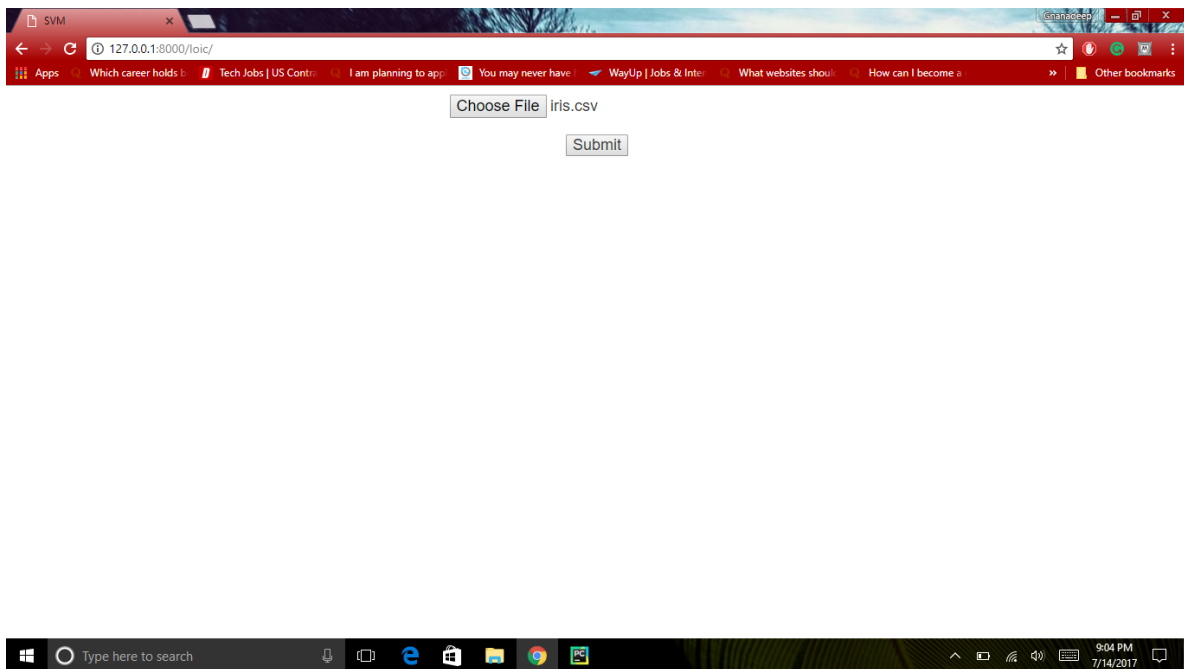
```

views.py

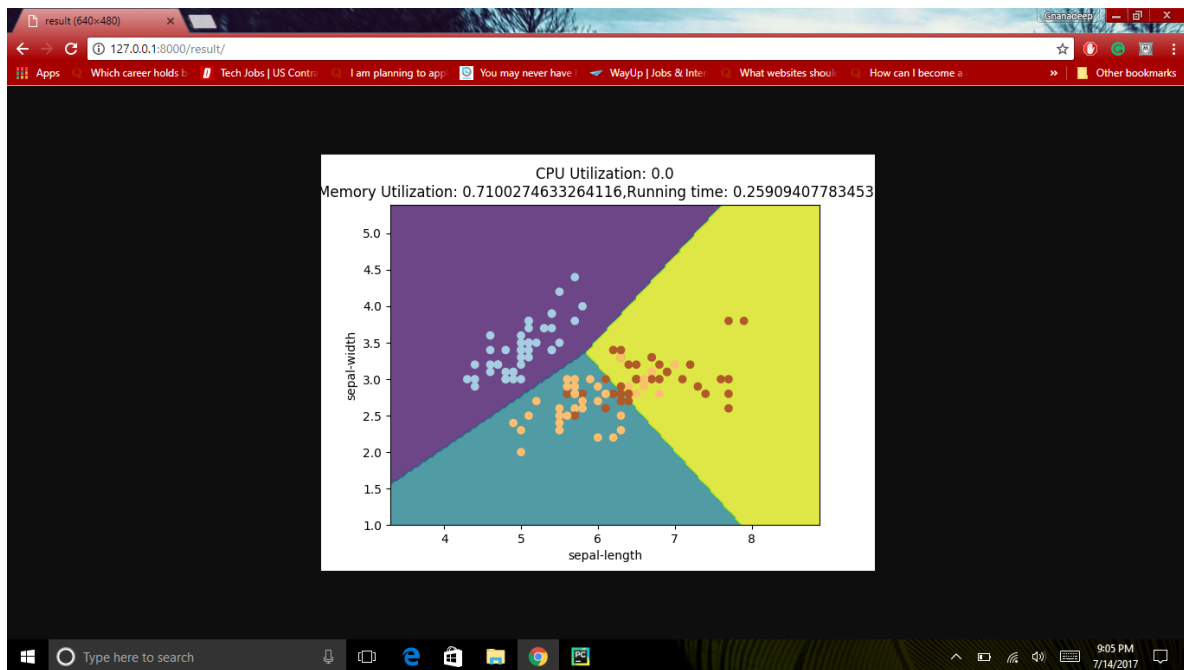
Finally, the plot is displayed on the new page when the response is returned which has the figure.

Results:

When the .csv file is uploaded and clicked the submit button. The plot is successfully displayed on the new page with the performance evaluation calculated above the figure.



loic.html



result.html

Performance Evaluation:

```
s0="sepal-length"
n="\n"
sd=psutil.cpu_percent()
s1="CPU Utilization: "
s2=str(sd)
s3="Memory Utilization: "
s4=str(mem)
s5="Running time: "
s6=str(stop - start)

s=s1+s2+n+s3+s4+", "+s5+s6
plt.xlabel('sepal-length')
plt.title(s)
```

```
CPU Utilization: 0.0
Memory Utilization: 0.7139054683396145, Running time: 0.24341743598857
```

References

1. <https://vincentarelbundock.github.io/Rdatasets/datasets.html>
2. <https://www.python.org/>
3. <https://www.analyticsvidhya.com/blog/2015/10/understaing-support-vector-machine-example-code/>
4. <http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy>
5. <https://pythonprogramming.net/support-vector-machine-svm-example-tutorial-scikit-learn-python/>
6. http://scikit-learn.org/stable/auto_examples/semi_supervised/plot_label_propagation_versus_svm_iris.html#sphx-glr-auto-examples-semi-supervised-plot-label-propagation-versus-svm-iris-py
7. <https://www.djangoproject.com/>
8. <https://docs.djangoproject.com/en/1.10/intro/tutorial01/#creating-the-polls-app>
9. <https://stackoverflow.com/questions/4996504/how-to-output-horizontal-barchart-to-django-site-page>
10. https://github.com/puneethreddy20/DDoS_detection
11. <https://stackoverflow.com/questions/5622976/how-do-you-calculate-program-run-time-in-python>