PROJECT REPORT

Pulsar Star Classification using Support Vector Machines (PSSVM)

METHODS USED FOR BUILDING SVM:

The Python package CVXOPT has been used to solve the dual optimization problem involved in the SVM.
The QP solver of CVXOPT solves the following problem:

$$minimize\ \frac{1}{2}x^T Px\ + q^T x$$

$$subject\ to\ \ Ax\ =\ b\ and\ Gx\ \leq\ h$$

$$where\ x, q, h\ are\ vectors\ and\ A, P, G, b\ are\ matrices$$

The SVM dual optimization problem is (applying a negative on first condition and on alpha more than 0)

$$min_\alpha\ -\sum_{i=1}^{n}\ \ \alpha_i\ +\ \frac{1}{2}\sum_{i=1}^{n}\ \ \sum_{j=1}^{n}\ \ \alpha_i\alpha_j y_i y_j x_i^T x_j$$

$$subject\ to\ \sum_{i=1}^{n}\ \ \alpha_i y_i\ =\ 0\ and\ \ \alpha_i \leq C\ and\ -\alpha_i \leq 0\ for\ all\ i$$

Define H as a matrix such that $H_{ij} = y_i y_j x_i^T x_j$. Let $\alpha\ =\ [\alpha_1\ \alpha_2 \ldots\ldots\ldots \alpha_n]^T$, $y\ =\ [y_1\ y_2 \ldots\ldots\ldots y_n]^T$, $1_n\ =\ [1\ 1 \ldots\ldots\ldots 1]_n^T$ and $k\ =\ [0\ 0\ ..\ 0\ C\ C\ \ldots C]^T$, where 0 and C are repeated n times. Also, let J be a 2n x n matrix such that the first n rows form the negative n x n identity matrix and the next n rows form the positive n x n identity matrix. That is,

M = [-1 0 … 0]
[0 -1 … 0]
.
.
[0 0 … -1]
[1 0 … 0]
[0 1 … 0]
.
.
[0 0 … 1]

Then $\alpha^T H\alpha\ =\ \sum_{i=1}^{n}\ \ \sum_{j=1}^{n}\ \ \alpha_i H_{ij}\alpha_j = \sum_{i=1}^{n}\ \ \sum_{j=1}^{n}\ \ \alpha_i\alpha_j y_i y_j x_i^T x_j$, $1_n^T\alpha\ = \sum_{i=1}^{n}\ \ \alpha_i$, $y^T\alpha\ =\ \sum_{i=1}^{n}\ \ \alpha_i y_i$, $M\alpha\ =\ [-\alpha_1 -\ \alpha_2 \ldots\ldots\ldots -\alpha_n\ \ \alpha_1\ \alpha_2 \ldots\ldots\ldots \alpha_n]^T$.
Substituting these in the SVM problem, we get

$$min_\alpha\ \ \frac{1}{2}\alpha^T H\alpha\ -1_n^T\alpha$$

$$subject\ to\ y^T\alpha = 0\ and\ M\alpha \leq k$$

We note that this is identical to what CVXOPT expects, if

$$P = H, q = -1_n, x = \alpha, b = 0, G = M, h = k\ .$$

This has been used in calling the solver in the code.

RESULTS:

Accuracy using our SVM and linear kernel:

```
75 support vectors out of 800 points
Accuracy:  0.97
```

Accuracy using our SVM and quadratic kernel:

```
66 support vectors out of 800 points
Accuracy:  0.965
```

Accuracy using our SVM and gaussian rbf kernel:

```
96 support vectors out of 800 points
Accuracy:  0.955
```

Accuracy using scikit-learn SVM and linear kernel:

```
Accuracy:  0.94
```

Accuracy using scikit-learn SVM and quadratic kernel:

```
Accuracy:  0.935
```

Accuracy using scikit-learn SVM and gaussian rbf kernel:

```
Accuracy:  0.95
```

Grid search table using our SVM and linear & quadratic kernels:

Kernel	Hyperparameters (C)	Accuracy
Linear	C=0.1	accuracy=0.955
Linear	C=1	accuracy=0.97
Linear	C=10	accuracy=0.975
Linear	C=100	accuracy=0.97
Linear	C=1000	accuracy=0.985
Quadratic	C=0.1	accuracy=0.965
Quadratic	C=1	accuracy=0.965
Quadratic	C=10	accuracy=0.965
Quadratic	C=100	accuracy=0.97
Quadratic	C=1000	accuracy=0.975

Grid search table using our SVM and gaussian rbf kernels:

Hyperparameters (C)	Hyperparameters (gamma)	Accuracy
C=0.1	gamma=0.1	accuracy=0.945
C=0.1	gamma=1	accuracy=0.945
C=0.1	gamma=10	accuracy=0.885
C=0.1	gamma=100	accuracy=0.875
C=0.1	gamma=1000	accuracy=0.875
C=1	gamma=0.1	accuracy=0.955
C=1	gamma=1	accuracy=0.965
C=1	gamma=10	accuracy=0.955
C=1	gamma=100	accuracy=0.885
C=1	gamma=1000	accuracy=0.875
C=10	gamma=0.1	accuracy=0.965
C=10	gamma=1	accuracy=0.965
C=10	gamma=10	accuracy=0.945
C=10	gamma=100	accuracy=0.885
C=10	gamma=1000	accuracy=0.875
C=100	gamma=0.1	accuracy=0.975
C=100	gamma=1	accuracy=0.965
C=100	gamma=10	accuracy=0.925
C=100	gamma=100	accuracy=0.885
C=100	gamma=1000	accuracy=0.875
C=1000	gamma=0.1	accuracy=0.965
C=1000	gamma=1	accuracy=0.965
C=1000	gamma=10	accuracy=0.905
C=1000	gamma=100	accuracy=0.885
C=1000	gamma=1000	accuracy=0.875

Grid search table using scikit-learn SVM & GridSearch and gaussian rbf kernels:

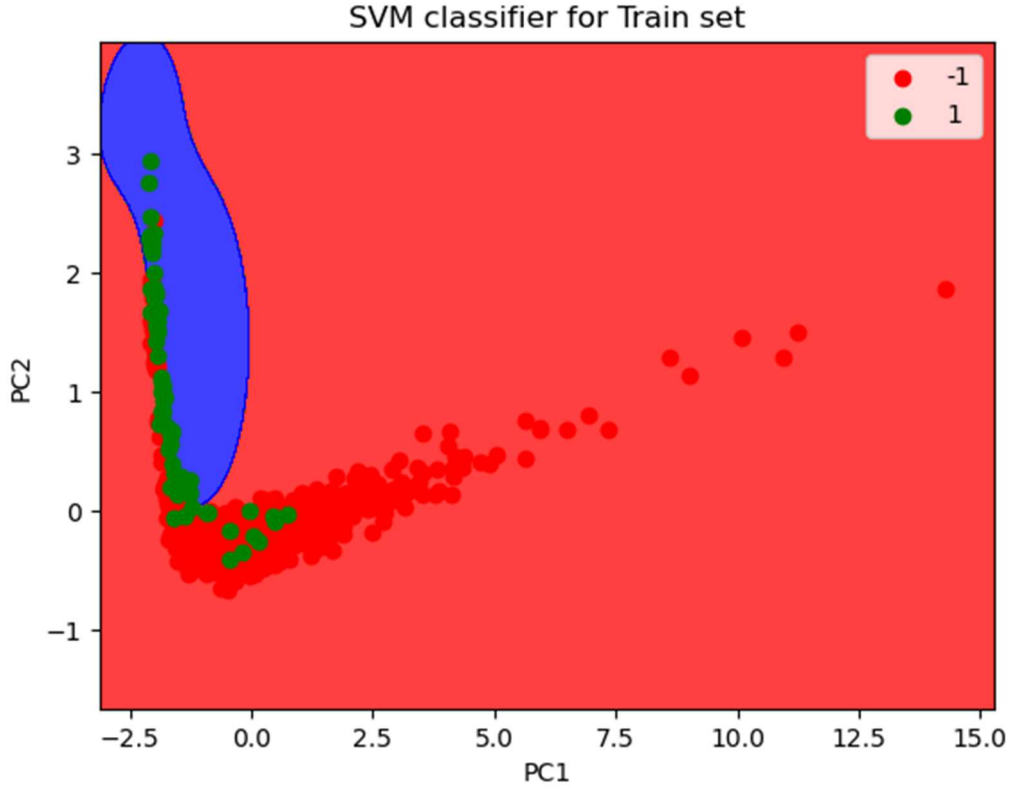
```
Best params: 0.970000 using {'C': 10, 'gamma': 0.1}
```

	param_C	param_gamma	mean_test_score
0	0.1	0.1	0.94875
1	0.1	1	0.96000
2	0.1	10	0.90750
3	0.1	100	0.90750
4	0.1	1000	0.90750
5	1	0.1	0.96500
6	1	1	0.96875
7	1	10	0.96125
8	1	100	0.91125
9	1	1000	0.90750
10	10	0.1	0.97000
11	10	1	0.96750
12	10	10	0.96125
13	10	100	0.91625
14	10	1000	0.90750
15	100	0.1	0.97000
16	100	1	0.96625
17	100	10	0.94750
18	100	100	0.91625
19	100	1000	0.90750
20	1000	0.1	0.96750
21	1000	1	0.96625
22	1000	10	0.93875
23	1000	100	0.91625
24	1000	1000	0.90750

Calculate accuracy using our SVM and gaussian rbf with best hyper-parameters(c=10 an d gamma=1), after applying PCA:

```
193 support vectors out of 800 points
Accuracy:  0.96
```

Visualization based on above best hyper-parameters for Train set:



Visualization based on above best hyper-parameters for Test set: