HOMEWORK 6

Madhushree Nijagal
9084490524

Instructions: You can choose any programming language as long as you implement the algorithm from scratch. Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file to Canvas. Please check Piazza for updates about the homework.

1 Kernel SVM [30pts]

Consider the following kernel function defined over $\mathbf{z}, \mathbf{z}' \in Z$, where Z is some set:

$$\kappa(\mathbf{z}, \mathbf{z}') = \begin{cases} 1 & \text{if } \mathbf{z} = \mathbf{z}', \\ 0 & \text{otherwise.} \end{cases}$$

1. (10 pts) Prove that for any positive integer m, any $\mathbf{z}_1, \ldots, \mathbf{z}_m \in Z$, the $m \times m$ kernel matrix $\mathbf{K} = [\mathbf{K}_{ij}]$ is positive semi-definite, where $\mathbf{K}_{ij} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$ for $i, j = 1 \ldots m$. (Let us assume that for $i \neq j$, we have $\mathbf{z}_i \neq \mathbf{z}_j$) Hint: An $m \times m$ matrix K is positive semi-definite if $\forall \mathbf{u} \in \mathbb{R}^d : \mathbf{u}^\top \mathbf{K} \mathbf{u} \geq 0$.

   We are given that for any $m > 0, z_1 \ldots . z_m \in Z$. The kernel matrix is given as $K = [K_{ij}]$ and $i \neq j$ for $z_i \neq z_j$. Since $\kappa(z_1, z_1) = \kappa(z_2, z_2) = 1$ and $\kappa(z_i, z_j) = 0$ when $i \neq j$ we have $K = I_m$ (An identity matrix). Consider the following matrix where m = 3, we have will have K as below:

   $$K_{3X3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

   We are given that K is positive semi-definite if $\forall u \in \mathbb{R}^d : u^\top K u \geq 0$. Since we know that $K = I_m$, we have $u^\top K = u^\top I = u^\top$. Therefore we have $u^\top K u = u^\top u = ||u||^2$ which is always $>= 0$. Thus we can say that $\forall u \in \mathbb{R}^d : u^\top K u \geq 0$ and therefore the kernel matrix K is positive semi-definite.

2. (10 pts) Given a training set $(\mathbf{z}_1, y_1), \ldots, (\mathbf{z}_n, y_n)$ with binary labels, the dual SVM problem with the above kernel κ will have parameters $a_1, \ldots, a_n, b \in \mathbb{R}$. (Let us assume that for $i \neq j$, we have $\mathbf{z}_i \neq \mathbf{z}_j$) The predictor for input $\mathbf{z}$ takes the form

   $$f(\mathbf{z}) = \sum_{i=1}^{n} a_i y_i \kappa(\mathbf{z}_i, z) + b \, .$$

   Recall that the label prediction is $\text{sgn}(f(\mathbf{z}))$. Prove that there exist $a_1, \ldots, a_n, b$ such that f correctly separates the training set. In other words, κ induces a feature space rich enough such that in it, any training set can be linearly separated.

   We know that the predictor takes the form

   $$f(\mathbf{z}) = \sum_{i=1}^{n} a_i y_i \kappa(\mathbf{z}_i, z) + b \, .$$

   We are getting z from the training set, but there will be only one instance where $z_i = z$. The definition of the kernel is given as:

   $$\kappa(\mathbf{z}, \mathbf{z}') = \begin{cases} 1 & \text{if } \mathbf{z} = \mathbf{z}', \\ 0 & \text{otherwise.} \end{cases}$$

.

From the above we can get $\kappa(z_i, z) = 1$ when $z_i = z$. Thus the predictor can be written for any instance as:

$$f(\mathbf{z}_i) = a_i y_i + b \ .$$

We know that $y_i = \pm 1$ that is the label prediction is $\text{sgn}(f(\mathbf{z})) = sgn(y_i)$. We have $f(z_i) = a_i + b$ for positive instance and $f(z_i) = -a_i + b$ for negative instance. If we intuitively pick $a_i = 1$ and b = 0, then $f(z_i) = 1$ for positive sample and $f(z_i) = -1$ for negative sample. The condition $\text{sgn}(f(\mathbf{z})) = sgn(y_i)$ is true when $a > 0 \forall a \in \{a_1, a_2.., a_n\}$. Hence whenever this is true, kernel κ induces a feature space such that any training set can be linearly separated.

3. (10 pts) How does that f predict input $\mathbf{z}$ that is not in the training set?
Again we know that the predictor takes the form

$$f(\mathbf{z}) = \sum_{i=1}^{n} a_i y_i \kappa(\mathbf{z}_i, z) + b \ .$$

and the label prediction is $\text{sgn}(f(\mathbf{z}))$. The label for input z needs to be predicted. The definition of the kernel is given as:

$$\kappa(\mathbf{z}, \mathbf{z}') = \begin{cases} 1 & \text{if } \mathbf{z} = \mathbf{z}', \\ 0 & \text{otherwise.} \end{cases}$$

.

Since $k(z, z') = 1$ only when $z \neq z'$. We are also given that z is not a part of the training set. Therefore $k(z_i, z) = 0$. We can express the predictor as:

$$f(\mathbf{z}) = \sum_{i=1}^{n} a_i y_i * 0 + b = b \ .$$

Therefore $sgn(f(z)) = sgn(b)$. That is the label of input z is 1 if $sgn(b) > 0$ and is -1 otherwise.

Comment: One useful property of kernel functions is that the input space Z does not need to be a vector space; in other words, $\mathbf{z}$ does not need to be a feature vector. For all we know, Z can be all the turkeys in the world. As long as we can compute $\kappa(\mathbf{z}, \mathbf{z}')$, kernel SVM works on turkeys.

2 Chow-Liu Algorithm [30 pts]

Suppose we wish to construct a directed graphical model for 3 features X, Y, and Z using the Chow-Liu algorithm. We are given data from 100 independent experiments where each feature is binary and takes value T or F. Below is a table summarizing the observations of the experiment:

X	Y	Z	Count
T	T	T	36
T	T	F	4
T	F	T	2
T	F	F	8
F	T	T	9
F	T	F	1
F	F	T	8
F	F	F	32

1. Compute the mutual information $I(X, Y)$ based on the frequencies observed in the data. (5 pts)

The mutual information can be represented in terms of entropy as:

$I(X,Y) = H(Y) - H(Y|X)$

According to the chain rule of entropy of 2 random variables:

$H(Y|X) = H(X,Y) - H(X)$

We can get $I(X,Y) = H(Y) + H(X) - H(X,Y)$

From the observations we can calculate:

$H(X) = -\frac{50}{100}log_2\frac{50}{100} - \frac{50}{100}log_2\frac{50}{100} = 1$

$H(Y) = -\frac{50}{100}log_2\frac{50}{100} - \frac{50}{100}log_2\frac{50}{100} = 1$

To calculate the value of H(X, Y), we create the joint distribution table as below:

X/Y	T	F
T	40/100	10/100
F	10/100	40/100

From the above table, we can calculate the joint entropy of X and Y as:

H(X, Y) = $-\frac{40}{100}log_2\frac{40}{100} - \frac{10}{100}log_2\frac{10}{100} - \frac{10}{100}log_2\frac{10}{100} - \frac{40}{100}log_2\frac{40}{100} = 1.7219$

We can calculate I(X, Y) = 1 + 1 - 1.7219 = 0.2781

2. Compute the mutual information $I(X,Z)$ based on the frequencies observed in the data. (5 pts)

Similarly to calculate $I(X,Z) = H(Z) - H(Z|X)$

$H(Z|X) = H(X,Z) - H(X)$

We can calculate $I(X,Z) = H(Z) + H(X) - H(X,Z)$

We know from the part 1, that the value of H(X) = 1

H(Z) = $-\frac{55}{100}log_2\frac{55}{100} - \frac{45}{100}log_2\frac{45}{100}$ = 0.9927

To calculate the joint entropy X and Z we draw up the joint distribution table as shown below:

X/Z	T	F
T	38/100	12/100
F	17/100	33/100

H(X, Z) = $-\frac{38}{100}log_2\frac{38}{100} - \frac{12}{100}log_2\frac{12}{100} - \frac{17}{100}log_2\frac{17}{100} - \frac{33}{100}log_2\frac{33}{100} = 1.8599$

Therefore, I(X, Z) = 1 + 0.9927 - 1.8599 = 0.1328

3. Compute the mutual information $I(Z,Y)$ based on the frequencies observed in the data. (5 pts)

To calculate $I(Z,Y) = H(Y) - H(Y|Z)$

$H(Y|Z) = H(Z,Y) - H(Z)$

Therefore, $I(Z,Y) = H(Y) + H(Z) - H(Z,Y)$

We know from the part 1 and part 2, that the value of H(Y) = 1 and H(Z) = 0.9927

Drawing out the joint distribution table to calculate joint entropy H(Z, Y) is as shown below:

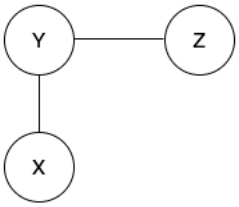
Z/Y	T	F
T	45/100	10/100
F	5/100	40/100

H(Z, Y) = $-\frac{45}{100}log_2\frac{45}{100} - \frac{10}{100}log_2\frac{10}{100} - \frac{5}{100}log_2\frac{5}{100} - \frac{40}{100}log_2\frac{40}{100} = 1.5954$

Therefore, I(Z, Y) = 1 + 0.9927 - 1.5954 = 0.3973

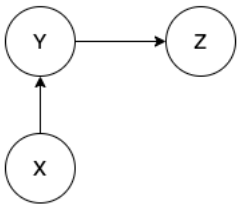
4. Which undirected edges will be selected by the Chow-Liu algorithm as the maximum spanning tree? (5 pts)

   We are given 3 features X, Y and Z. The information gain between them as calculated above are: I(X, Y) = 0.2782; I(X, Z) = 0.1328 and I(Z, Y) = 0.3973. The Chow-Liu algorithm adds edges between vertices Y and Z first as their information gain is the highest and then adds another edge between X and Y and does not add any more edges as the maximum weight spanning tree is complete. The tree looks like below:



5. Root your tree at node X, and assign directions to the selected edges. (10 pts)

   If we pick X as the root, we need to making all edges directed away from the root. Therefore the tree becomes as shown below:



3 Game of Classifiers [60pts]

3.1 Implementation

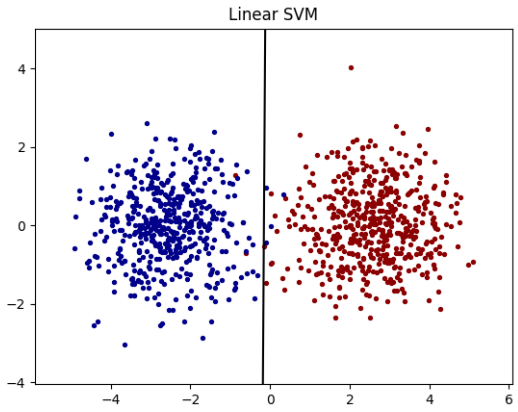
Implement the following models in the choice of your programming language. Include slack variables in SVM implementation if needed. You can use autograd features of PyTorch, TensorFlow, etc., or derive gradients on your own (but do not use inbuilt models for SVM, Kernel SVM, and Logistic Regression from libraries).

- Implement Linear SVM (without kernels).
- Implement Kernel SVM, with options for linear, rbf, and polynomial kernels. You should keep the kernel parameters tunable (e.g., do not fix the degree of polynomial kernels but keep it as a variable and play with different values of it.) Is Linear SVM a special case of Kernel SVMs?
- Implement Logistic Regression with and without kernels (use same kernels as above).

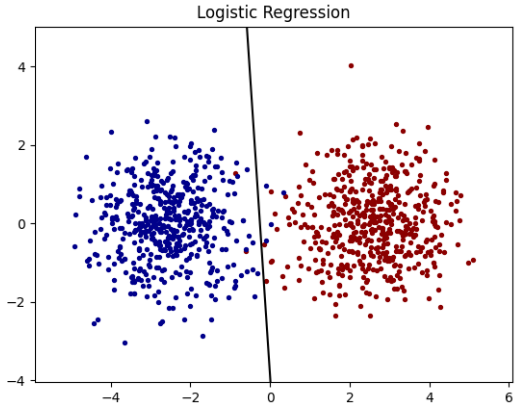
3.2 Synthetic Dataset-1 (20 pts)

Generate a 2-D dataset as follows: Let $\mu = 2.5$ and $\mathbf{I}_2$ be the 2×2 identity matrix. Generate points for the positive and negative classes, respectively from $\mathcal{N}([\mu, 0], \mathbf{I}_2)$, and $\mathcal{N}([-\mu, 0], \mathbf{I}_2)$. For each class, generate 750 points (1500 in total). Randomly create train, validation, and test splits of 1000, 250, and 250 points, respectively. Do the following with this dataset:

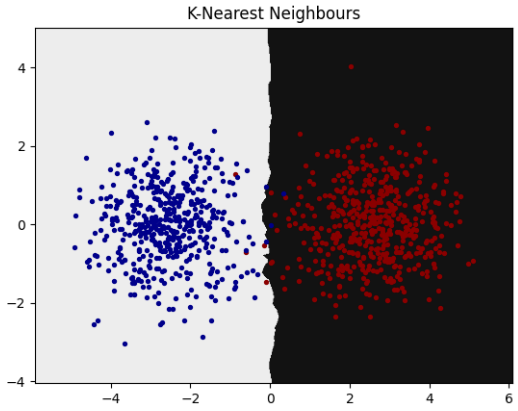
1. (5 pts) Train your Linear SVM, Logistic Regression models and report decision boundaries and test accuracies.
   Linear SVM Accuracy: 99.6%



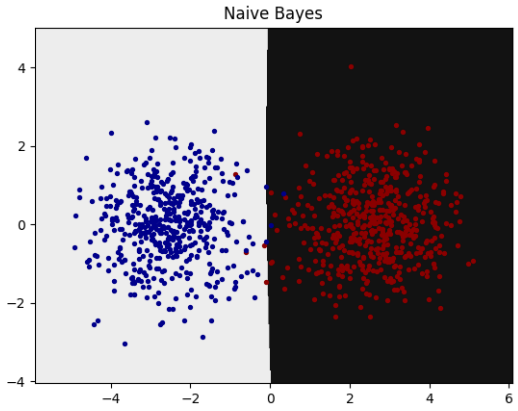
   Logistic Regression Accuracy: 98.8%
   Parameters: Learning Rate = 0.01; Epochs = 1500; Batch Size = 30



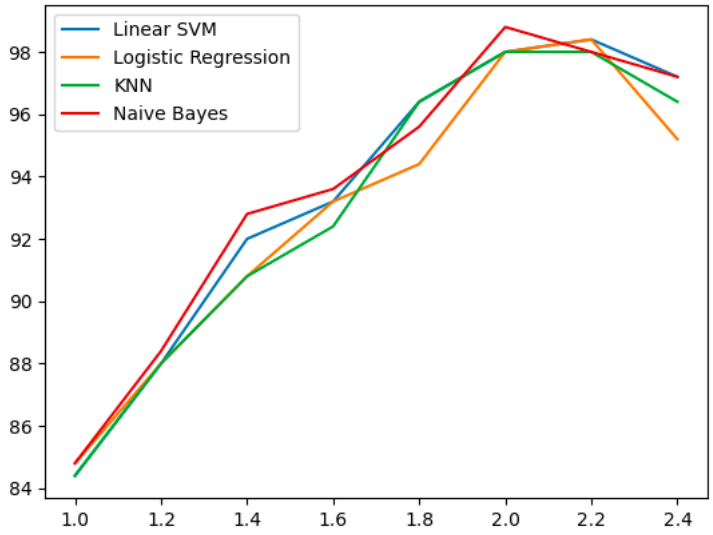
2. (5 pts) Show the decision boundaries with k-NN and Naive Bayes Classifiers. (You can use library implementations or implement from scratch. Figure out the hyper-parameters using the validation set.)
   kNN Classifier Accuracy: 99.6%
   Parameters: Number of neighbours = 15



Naive Bayes Accuracy: 99.6%



3. (5 pts) Repeat the process by varying μ from 1.0 to 2.4 with a step size of 0.2 for each value of μ to obtain test accuracies of the models and plot (μ on x-axis and test accuracy on y-axis). (You will have a curve for each of the 4-classifiers mentioned above.)
The learning curve is as below after varying μ from 1.0 to 2.4 with a step size of 0.2:
The mean (μ) is on the x-axis and test accuracy is on the y-axis



Test Accuracy(y-axis) vs Mean(x-axis)

4. (5 pts) What are your conclusions from this exercise?
   All 4 classifiers have performed more or less equally well on the 2D Gaussian dataset. Among the linear classifiers: Linear SVM and Logistic Regression, the former performs comparatively better as SVM tries to maximise the margin between positive and negative datapoints. It is also evident in the learning curve. The KNN and Naive Bayes also perform equally well. When the value of μ is varied, all of the classifiers have low accuracy when $\mu = 1.0$ and are almost equal. As the value increases the distributions separate and the non linear classifiers perform better compared to the linear classifiers.

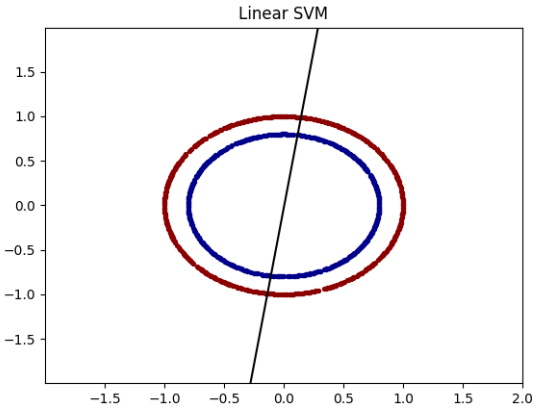
3.3 Synthetic Dataset-2 (20 pts)

Generate 1500 data points from the 2-D circles dataset of sklearn:

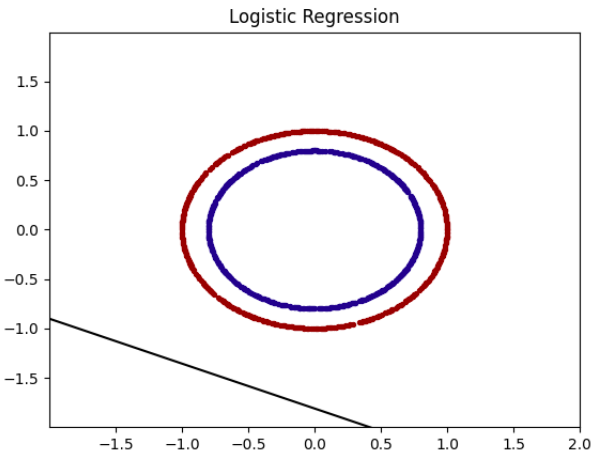
```
sklearn.datasets.make_circles
```

Randomly create train, validation, and test splits of 1000, 250, and 250 points, respectively. Evaluate the above classifiers in this setting.

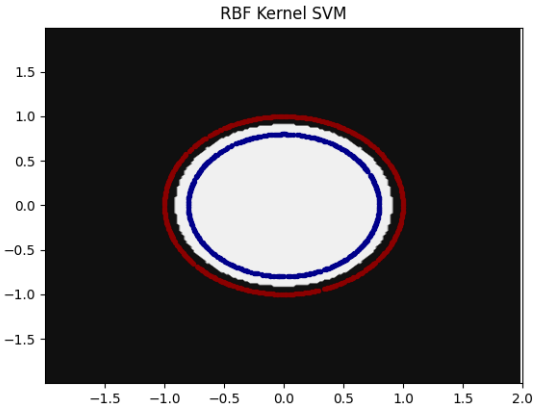
1. (5 pts) Show decision boundaries for Linear SVM and Logistic Regression classifiers.
   Linear SVM Accuracy: 49.2%



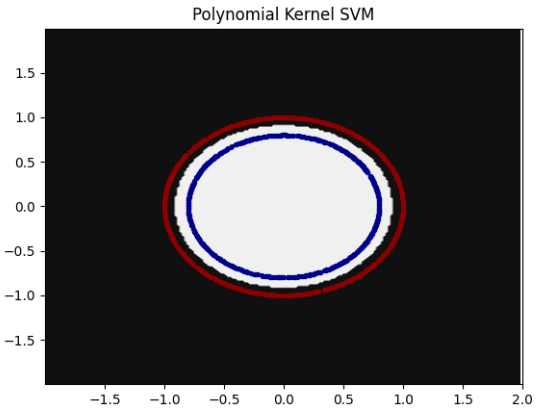
Logistic Regression Accuracy: 54.2%
Parameters: Learning Rate = 0.01; Epochs = 1500; Batch Size = 30



2. (5 pts) Show decision boundaries for Kernel SVM and Kernel Logistic Regression (use rbf, polynomial kernels). Try different values of hyperparameters, and report results with whichever works best.
RBF Kernel SVM Accuracy: 100%
Parameters: Sigma = 5

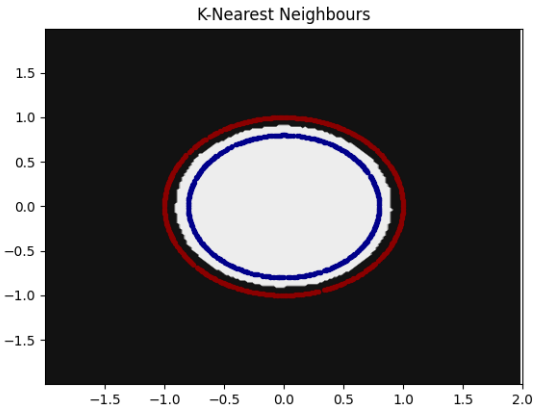


Polynomial Kernel SVM Accuracy: 100%
Parameters: Degree = 3

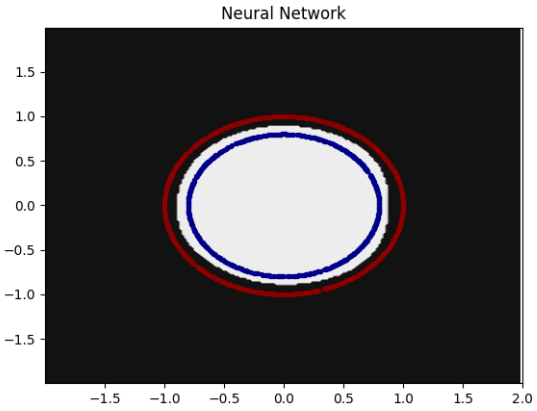


3. (5 pts) Train Neural Network from HW4, and k-NN classifiers on this dataset and show decision boundaries. (You can use library implementation for these classifiers).

kNN Classifier Accuracy: 100%
Parameters: Number of neighbours = 15



Neural Network: 100%
Parameters: Epochs = 100; Batch Size = 20; Learning Rate = 0.3; Layers = 2; Hidden Layer = 64



4. (5 pts) What are your conclusions from this exercise?
   Since the dataset is not linearly separable, the linear models: Linear SVM, Logistic Regression perform poorly. They have an accuracy of 49.2% and 52.4% respectively and are not able to separate out the data as expected. The non-linear models: KNN, NaiveBayes perform accurately achieving a 100% accuracy. Adding kernels (RBF and Polynomial) to the SVM classifier makes it capable of creating a decision boundary and achieves 100% accuracy in both cases as well.

3.4 Evaluation on Real Dataset (20 pts)

Let's put all this to some real use. For this problem, use the Wisconsin Breast Cancer dataset. You can download it from the sklearn library:

```
sklearn.datasets.load_breast_cancer
```

1. (10 pts) Do all the points of Section 3.3 in this dataset. Since these are high-dimensional data, you do not have to show the decision boundaries. Report test accuracies for these classifiers and discuss your findings.

   Linear SVM accuracy: 94.73%

   Polynomial Kernel SVM accuracy: 96.49%
   Parameters: Degree = 3

RBF Kernel SVM accuracy: 94.73%
Parameters: Sigma = 5

Logistic Regression accuracy : 98.24%
Parameters: Learning Rate = 0.01; Epochs = 1500; Batch Size = 30

kNN Classifier accuracy: 94.73%
Parameters: Neighbours = 15

Neural Network accuracy: 98.24%
Parameters: Epochs = 100; Batch Size = 20; Learning Rate = 0.3; Layers = 2; Hidden Layer = 64

All of the models perform fairly well, including the linear models: Linear SVM and Logistic Regression with accuracies of 94.73% and 98.24% respectively. The non-linear models are consistent with their performance over this dataset with a high accuracy. The polynomial kernel SVM performs better than RBF kernel SVM and Linear SVM with an accuracy of 96.49% on the real dataset and perform equally well as KNN and Neural Network. The accuracy of the linear models haven't improved as much. These accuracies prove that this model is fairly well separable in the d-dimensional space.

2. (10 pts) In addition, you also want to figure out the important features which determine the class. Which regularization will you use for this? Upgrade your SVM, Kernel SVM implementation to include this regularization. Discuss the important features that you obtain by running your regularized SVM on this dataset. (You might need to normalize this dataset before training any classifier.)
In order to figure out the important features which determine the class, L1 regularization can be used because of its property to force the coefficients of the features that are close to zero to zero, effectively removing the features that are not providing much information for classification. After implementing the L1 regularization on the SVM classifiers and correlating the features with target variable, the following 14 out of 30 features were selected as important features:
'mean concavity', 'mean concave points', 'mean fractal dimension', 'radius error', 'texture error', 'smoothness error', 'compactness error', 'fractal dimension error', 'worst radius', 'worst texture', 'worst smoothness', 'worst concavity', 'worst concave points', 'worst symmetry'