1. For hard-SVM, there is perfectly separable data in the example. The hyperplane found by the hard-SVM will correctly classify all points without any misclassifications.
   For soft-SVM, choosing C is important. When the dataset is linearly separable, setting C to a high value will result in a similar solution to hard-SVM, pushing the solution to correctly classify all points. However, it isn't guaranteed that a value of C will always result in the same w* as the hard-SVM for every possible linearly separable dataset because the soft-SVM's optimization problem includes an additional term $C\sum\xi_i$ which might lead to a different margin if C is not large enough.
   While in many cases high values of C in the soft-SVM can approximate the hard-SVM solution closely, the statement that there exists a fixed C that makes sure the soft-SVM and hard-SVM solutions are always identical for every linearly separable dataset is false. In an example of n=2, d=1, with points (a,1) and (-a, -1), the hard-SVM solution would be a vector w such that the hyperplane w*x+b=0 separates the points with the maximal margin. For soft-SVM with a large C, we expect a similar separating hyperplane. However, for smaller values of C, the soft-SVM might choose a smaller margin if it reduces the objective function due to a less severe penalty on the slack variables. The soft-SVM solution might not have a maximal margin, resulting in a different w than the hard SVM solution.

2. a) Gaussian Kernel for one-dimensional inputs $x, x' \in \mathbb{R}$

$K(x, x') = \exp\left(\frac{-(x-x')^2}{2\sigma^2}\right)$ $\alpha = \frac{1}{\sqrt{2\sigma^2}}$

Taylor Series expansion of exponential func:

$$e^{-u} = \sum_{n=0}^{\infty} \frac{(-u)^n}{n!}$$

Substituting u with $\alpha^2(x-x')^2 \rightarrow$ series expansion for the Gaussian Kernel

Mapping function $\phi(x)$ in space $\mathcal{H}$: $\phi(x) = (1, \alpha x, \alpha^2 x^2, \alpha^3 x^3, \dots)$

Each term $(\alpha x)^n$ corresponds to the term in the taylor series expansion of the Gaussian Kernel

b) multi dimensional inputs $x, x' \in \mathbb{R}^D$ with $\|x\|^2 = \|x'\|^2 = 1$

Gaussian Kernel:

$K(x, x') = \exp\left(\frac{-\|x - x'\|^2}{2\sigma^2}\right)$

$\|x - x'\|^2 = 2 - 2x^T x' \leftarrow \|x\|^2 = \|x'\|^2 = 1$

using $\alpha = \frac{1}{\sqrt{2\sigma^2}}$ & Taylor expansion:

$$K(x, x') = \sum_{n=0}^{\infty} \frac{1}{n!}\left(-\alpha^2(2 - 2x^T x')\right)^n$$

Mapping Func $\phi(x)$ in space $\mathcal{H}$:

$\phi(x) = (1, \alpha^2(2-2x^T x), \alpha^4(2-2x^T x)^2, \alpha^6(2-2x^T x)^3, \dots)$

Each term $(\alpha^2(2-2x^T x))^n$ corresponds to the termin Taylor series expansion of the Gaussian kernel, reflecting the kernel's generalization to higher dimensions While keeping the norm of x & x' $= 1$.

In both cases, the mapping func $\phi(x)$ translates the input data into an infinite-dimensional space where the dot product corresponds to the Gaussian Kernel function.

3. a) $f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$ $\lambda > 0,\ X \in \mathbb{R}^{n \times d},\ w \in \mathbb{R}^d,\ y \in \mathbb{R}^n$

$\nabla_w f(w) = X^T(Xw - y) + \lambda w = 0 \rightarrow X^T X w + \lambda I w = X^T y$

$w = (X^T X + \lambda I)^{-1} X^T y$

$X^T X + \lambda I$ is invertible for $\lambda > 0$ bc if $X = U\Sigma V^T$ is the SVD of X, then $X^T X = V\Sigma^T \Sigma V^T$, & all eigenvalues of $X^T X$ are nonnegative. Adding λI, where I = Identity matrix & $\lambda > 0$, increases each eigenvalue by λ, making sure that all eigenvalues are positive which then makes sure that the matrix is invertible

b) rewriting the normal equation:

$X^T X w + \lambda I w = X^T y$

$\lambda w = X^T y - X^T X w$

$w = \frac{1}{\lambda} X^T y - \frac{1}{\lambda} X^T X w$

we can express w as $w = X^T \alpha$

where $\alpha \in \mathbb{R}^n$ is a vector of coefficients, substituting back into eq, we get:

$\alpha = \frac{1}{\lambda} y - \frac{1}{\lambda} X X^T \alpha$

c) saying that w is in the span of the data means that w can be expressed as a linear combinaton of the columns of X, which are the features of the training data. Since $w = X^T \alpha$, this is the case

d) Substituting $w = X^T \alpha$ into the normal equation & solving for α we get

$\alpha = (\lambda I + X X^T)^{-1} y$

$X X^T$ = Gram (kernel) matrix for the standard vector dot product which is a $n \times n$ matrix

e) predicted values on the training points: $Xw = X X^T \alpha$

substituting α from d): $Xw = X X^T (\lambda I + X X^T)^{-1} y$

f) for a test sample x not in the training set, $w^T x = \alpha^T X x$

since α can be expressed using kernel matrix & target values, we get

$w^T x = y^T (\lambda I + X X^T)^{-1} X x$

g) In Kernelized Ridge regression, the prediction for a new sample x uses the kernel function $k(x, x') = x^T x'$ to compute the inner products btwn x & the training samples within the kernel matrix $K = X X^T$.

The predicted value is given by:

$w^T x = y^T (\lambda I + K)^{-1} k(X, x)$

- $(\lambda I + K)^{-1}$ is the inverse of the kernel matrix regularized w/ λ
- $k(X, x)$ is the vector of kernel evaluations btwn x & training samples

4. a) $\alpha_i = \frac{1}{2}\log\left(\frac{1-\epsilon_i}{\epsilon_i}\right)$ ϵ_i = weighted training error of classifier f_i

misclassified = D2, 9, 10, 12, 13, 14

$\epsilon_1 = \frac{6}{16}$

$\alpha_1 = \frac{1}{2}\log\left(\frac{1-6/16}{6/16}\right) = 0.2554$

b)

error of first decision stump f_1: $\epsilon_1 = 6/16$

weight of f_1: $\alpha_1 = \frac{1}{2}\log\left(\frac{1-6/16}{6/16}\right)$

updates:

incorrectly classified instances: $W_{new,incorrect} = \frac{1}{16}e^{\alpha_1}$

correctly classified instances: $W_{new,correct} = \frac{1}{16}e^{-\alpha_1}$

Normalize weights:

- compute sum of all updated weights
- divide each weight by sum to normalize, ensuring that the total weight accross all instances equals 1

c) Technically the iterations might contince but its better to stop boosting when a weak classifier achieves a 0 error rate on weighted training data. This prevents overfitting & maintains a balance btwn bias & variance, which is needed for good generalization

Description of Classifiers

Random Forests are an ensemble learning method that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. It achieves better performance and generalization by averaging multiple deep decision trees, trained on different parts of the same training set, which reduces the variance.

SVMs are a set of supervised learning methods used for classification, regression, and outliers detection. The Gaussian kernel, also known as the Radial Basis Function (RBF), is used in SVM to transform the input space into a higher-dimensional space where a hyperplane can be used to separate classes. The RBF kernel is effective in cases where the relationship between class labels and attributes is nonlinear.

Training Methodology

For both classifiers, the training methodology involves:

- Loading the dataset in LibSVM format using load_svmlight_file
- Converting sparse matrices to dense arrays
- Splitting the training data further into training and validation sets using train_test_split
- Employing RandomizedSearchCV to optimize hyperparameters through a specified number of iterations over cross-validation, which ensures robust evaluation of each parameter combination
- Evaluating the final model on a separate test set to gauge generalization performance

List of Hyperparameters Tuned

Random Forest

- n_estimators: Number of trees in the forest
    - Default = 100
    - Tested = [100, 300]
- bootstrap: Whether bootstrap samples are used when building trees.
    - Default = True
    - Tested = [True, False]
- max_depth: Maximum depth of the tree.
    - Default = None, meaning nodes are expanded until all leaves are pure.
    - Tested = [None, 20]
- min_samples_leaf: Minimum number of samples required to be at a leaf node.
    - Default = 1
    - Tested = [1, 4]

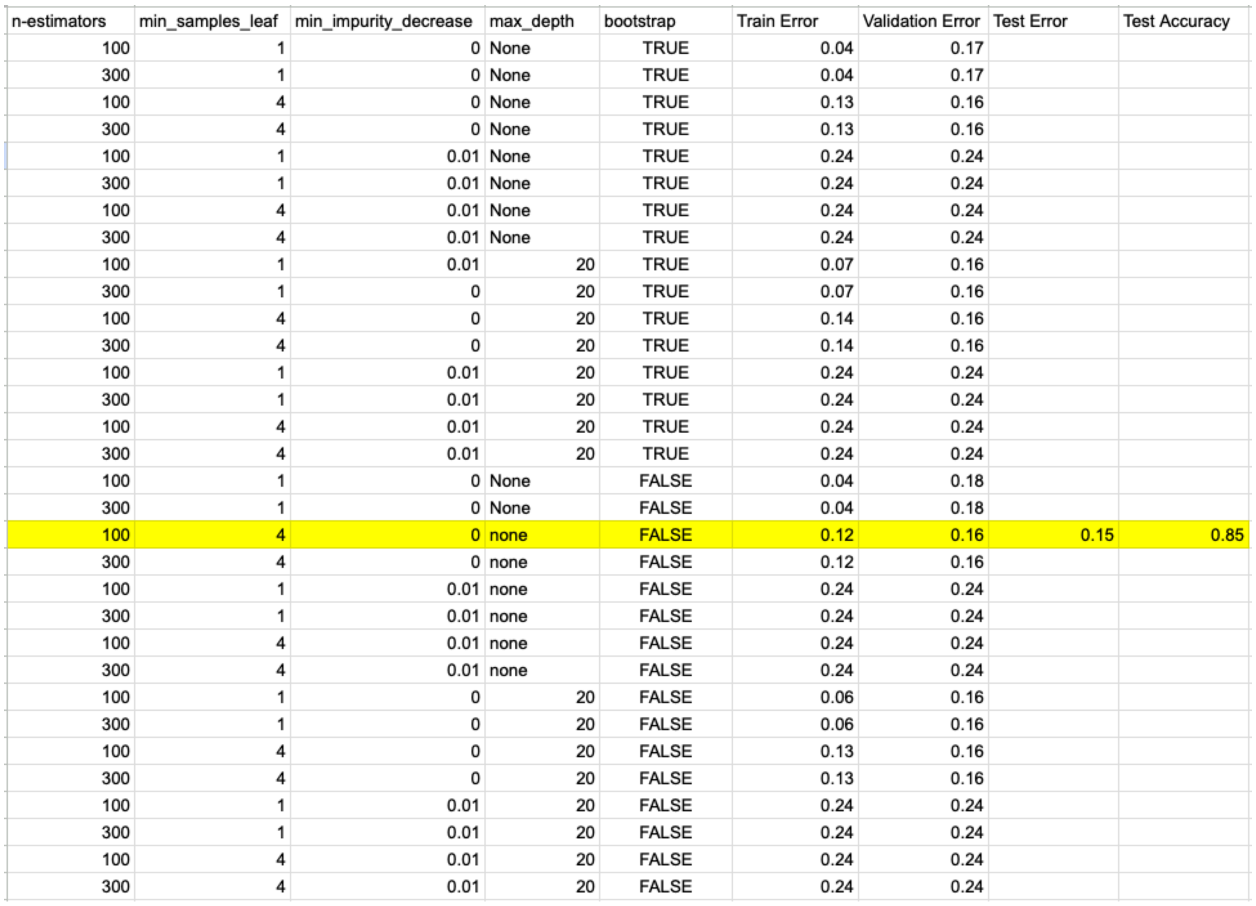
- min_impurity_decrease: A node will be split if this split induces a decrease of the impurity greater than or equal to this value
    - Default = 0.0
    - Tested = [0.0, 0.01]

SVM

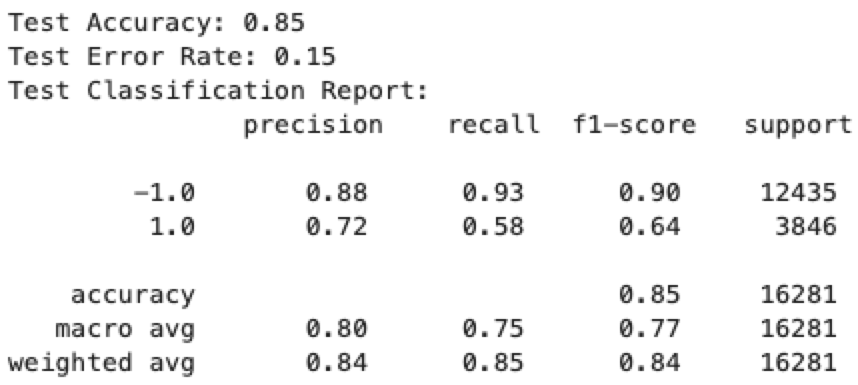
- C: Regularization parameter. The strength of the regularization is inversely proportional to C.
    - Default = 1.0
    - Tested = [1, 10]
- gamma: Kernel coefficient for ‘rbf’.
    - Default is ‘scale’ which uses 1 / (n_features * X.var()) as the value of gamma
    - Tested = [0.1, 0.01]
- kernel: Specifies the kernel type to be used in the algorithm.
    - Tested = 'rbf' (Gaussian)

Error Rates and Accuracy

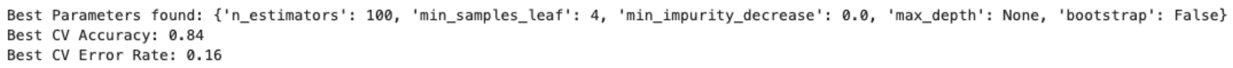
Random Forest



n-estimators	min_samples_leaf	min_impurity_decrease	max_depth	bootstrap	Train Error	Validation Error	Test Error	Test Accuracy
100	1	0	None	TRUE	0.04	0.17		
300	1	0	None	TRUE	0.04	0.17		
100	4	0	None	TRUE	0.13	0.16		
300	4	0	None	TRUE	0.13	0.16		
100	1	0.01	None	TRUE	0.24	0.24		
300	1	0.01	None	TRUE	0.24	0.24		
100	4	0.01	None	TRUE	0.24	0.24		
300	4	0.01	None	TRUE	0.24	0.24		
100	1	0.01	20	TRUE	0.07	0.16		
300	1	0	20	TRUE	0.07	0.16		
100	4	0	20	TRUE	0.14	0.16		
300	4	0	20	TRUE	0.14	0.16		
100	1	0.01	20	TRUE	0.24	0.24		
300	1	0.01	20	TRUE	0.24	0.24		
100	4	0.01	20	TRUE	0.24	0.24		
300	4	0.01	20	TRUE	0.24	0.24		
100	1	0	None	FALSE	0.04	0.18		
300	1	0	None	FALSE	0.04	0.18		
100	4	0	none	FALSE	0.12	0.16	0.15	0.85
300	4	0	none	FALSE	0.12	0.16		
100	1	0.01	none	FALSE	0.24	0.24		
300	1	0.01	none	FALSE	0.24	0.24		
100	4	0.01	none	FALSE	0.24	0.24		
300	4	0.01	none	FALSE	0.24	0.24		
100	1	0	20	FALSE	0.06	0.16		
300	1	0	20	FALSE	0.06	0.16		
100	4	0	20	FALSE	0.13	0.16		
300	4	0	20	FALSE	0.13	0.16		
100	1	0.01	20	FALSE	0.24	0.24		
300	1	0.01	20	FALSE	0.24	0.24		
100	4	0.01	20	FALSE	0.24	0.24		
300	4	0.01	20	FALSE	0.24	0.24		

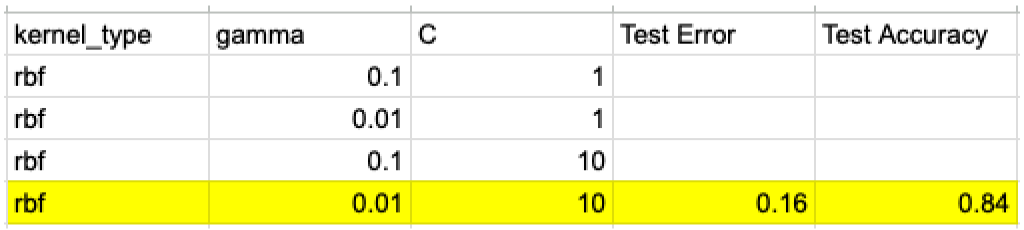


```
Test Accuracy: 0.85
Test Error Rate: 0.15
Test Classification Report:
              precision    recall  f1-score   support

        -1.0       0.88      0.93      0.90     12435
         1.0       0.72      0.58      0.64      3846

    accuracy                           0.85     16281
   macro avg       0.80      0.75      0.77     16281
weighted avg       0.84      0.85      0.84     16281
```

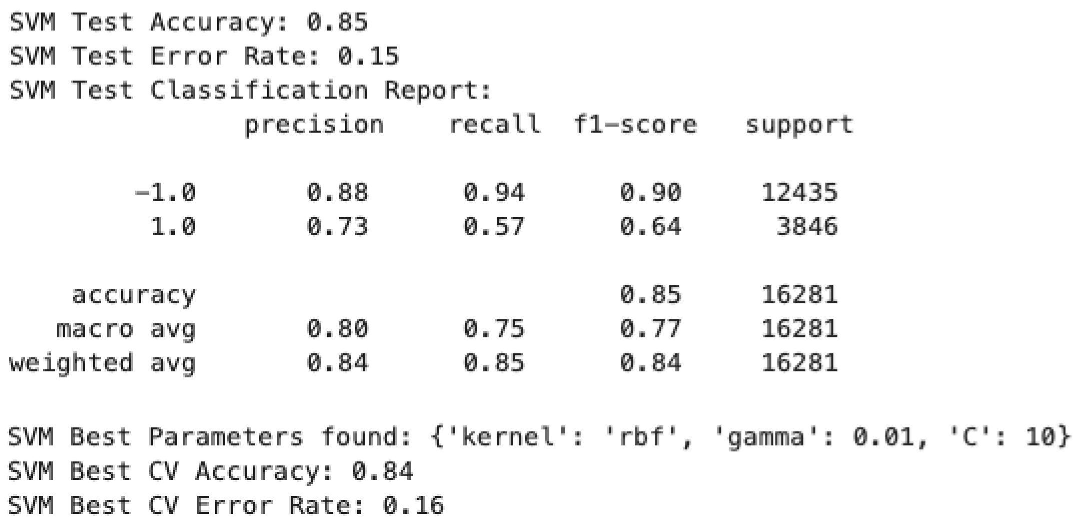


```
Best Parameters found: {'n_estimators': 100, 'min_samples_leaf': 4, 'min_impurity_decrease': 0.0, 'max_depth': None, 'bootstrap': False}
Best CV Accuracy: 0.84
Best CV Error Rate: 0.16
```

SVM



kernel_type	gamma	C	Test Error	Test Accuracy
rbf	0.1	1		
rbf	0.01	1		
rbf	0.1	10		
rbf	0.01	10	0.16	0.84



```
SVM Test Accuracy: 0.85
SVM Test Error Rate: 0.15
SVM Test Classification Report:
              precision    recall  f1-score   support

        -1.0       0.88      0.94      0.90     12435
         1.0       0.73      0.57      0.64      3846

    accuracy                           0.85     16281
   macro avg       0.80      0.75      0.77     16281
weighted avg       0.84      0.85      0.84     16281

SVM Best Parameters found: {'kernel': 'rbf', 'gamma': 0.01, 'C': 10}
SVM Best CV Accuracy: 0.84
SVM Best CV Error Rate: 0.16
```