Umair Afzal
HW3
CS589

Collaborators:
Debankita,
Bill

1. SVM [50 points]

1.1

a	1.1.a) Plot 2. The $f(x)$ of quadratic kernel is $\Sigma_i \alpha_i (x_i x + (x_i x)^2) + b$. Given this, we can say that $f(x)=0$ is the decision boundry. We can see that the function is second order, as it is hyperbolic curve.
b	1.1.b) Plot 3. The $f(x)$ is $\Sigma_i \alpha_i \exp(-\gamma \lvert x_i - x \rvert^2) + b$. We can see that γ is large, we have more support vectors
c	1.1.c) Plot 1. Incorporting the previous part of this question, we have $\Sigma_i \alpha_i \exp(-\gamma \lvert x_i - x \rvert^2) + b$. It is true that if γ is larger and the distance from x to x_i is less then the kernel is very small. And due to this the classification is very difficult. With a greater γ it will be very difficult to classify circle points. So, it is Plot 1 as we have $-\frac{1}{5}$.

1.2

1.2) $cx_1^2 + dx_2^2 - 2acx_1 - 2bdx_2 + (a^2c + b^2d - 1) = 0$ [expanded]
weight $\Rightarrow (2ac, 2bd, c, d, 0)$. It meets at $a^2 + b^2 - r^2$
We can see that the elliptical boundry is linear in this space and is linearly separable.

$$c(x_1 - a)^2 + d(x_2 - b)^2 = 1$$
$$c(x_1^2 - 2ax_1 + a^2) + d(x_2^2 - 2bx_2 + b^2) - 1 = 0$$
$$cx_1^2 - 2acx_1 + a^2c + dx_2^2 - 2bdx_2 + b^2d - 1 = 0$$

1.3

Training Step Output (showing model converging)

Training step 1000: LearningRate[0.0000632], Objective[73641.9835398]
Training step 2000: LearningRate[0.0000447], Objective[49400.5337402]
Training step 3000: LearningRate[0.0000365], Objective[37345.4942462]
Training step 4000: LearningRate[0.0000316], Objective[29763.1245302]
Training step 5000: LearningRate[0.0000283], Objective[23757.8297916]
Training step 6000: LearningRate[0.0000258], Objective[18722.7528173]
Training step 7000: LearningRate[0.0000239], Objective[14606.4549457]
Training step 8000: LearningRate[0.0000224], Objective[11241.6021197]
Training step 9000: LearningRate[0.0000211], Objective[8557.8506020]
Training step 10000: LearningRate[0.0000200], Objective[6445.9347769]

	F1 score	Precision	Recall
Training set	0.908571428571428 7	0.919075144508670 6	0.898305084745762 8
Testing set	0.831168831168831 2	0.761904761904761 9	0.914285714285714 3

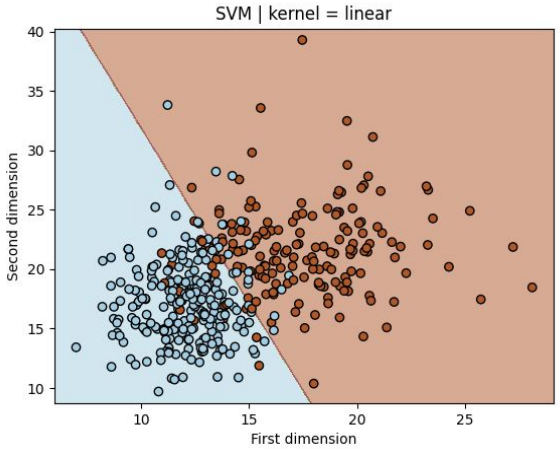
1.4

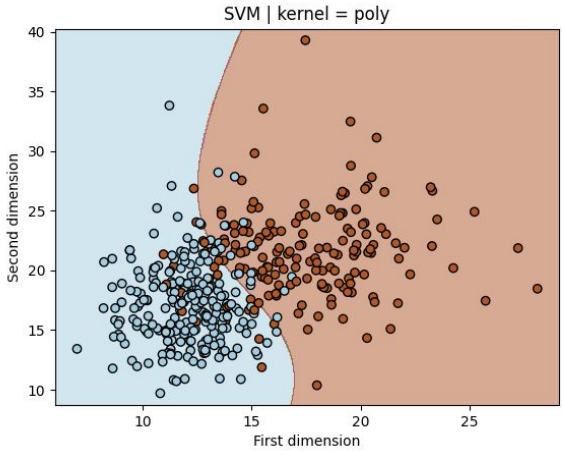
Note: These results are after I trained the model on all of the features.

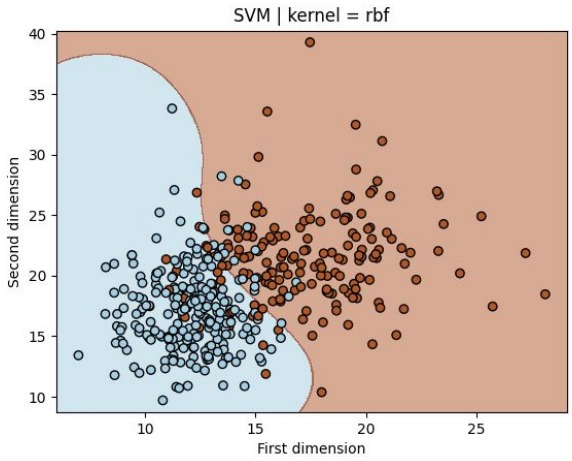
	F1 score	Precision	Recall
Linear train	0.963172804532577 8	0.965909090909090 9	0.96045197740113
Linear test	0.897435897435897 4	0.813953488372093	1.0
Poly train	0.880733944954128 5	0.96	0.813559322033898 4
Poly test	0.882352941176470 6	0.909090909090909 1	0.857142857142857 1
RBF train	0.882882882882882 9	0.942307692307692 3	0.830508474576271 2
RBF test	0.888888888888889	0.864864864864864	0.914285714285714

		9	3

1.5 PLOTS


SVM | kernel = linear
Second dimension
First dimension
10
15
20
25
30
35
40

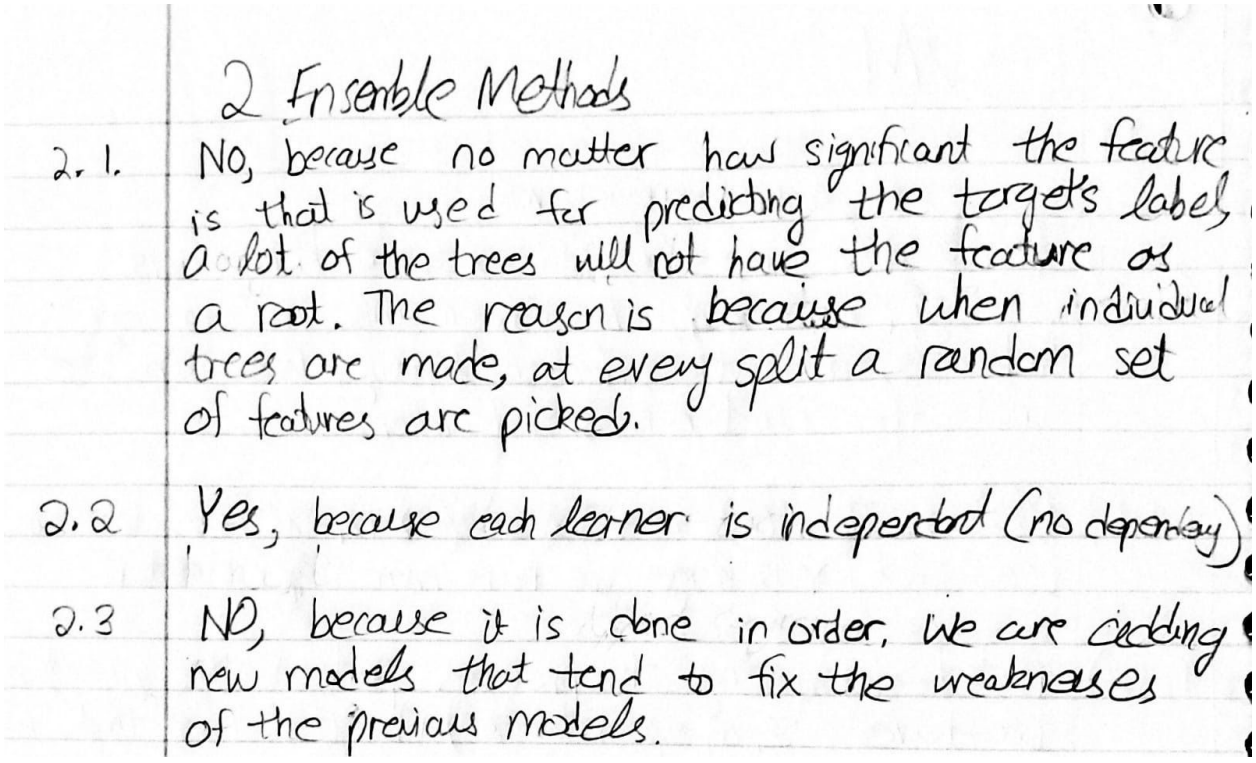

SVM | kernel = poly
Second dimension
First dimension
10
15
20
25
30
35
40


SVM | kernel = rbf
Second dimension
First dimension
10
15
20
25
30
35
40

-

2. Ensemble Methods [40 points]

2.1 2.2 2.3



2 Ensemble Methods

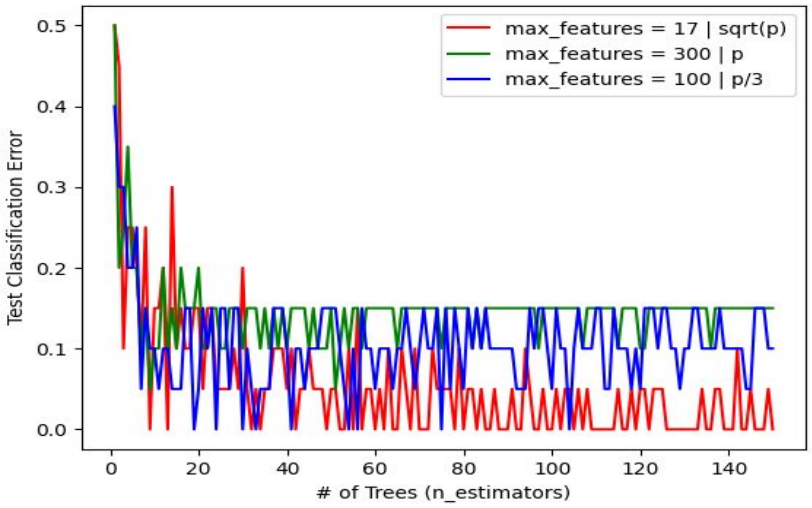
2.1. NO, because no matter how significant the feature is that is used for predicting the target's label, a lot of the trees will not have the feature as a root. The reason is because when individual trees are made, at every split a random set of features are picked.

2.2 Yes, because each learner is independent (no dependency)

2.3 NO, because it is done in order. We are adding new models that tend to fix the weaknesses of the previous models.

2.4

Figure can also be found in Figures folder -> “ensemble_randomforest_Q2.4_plot”

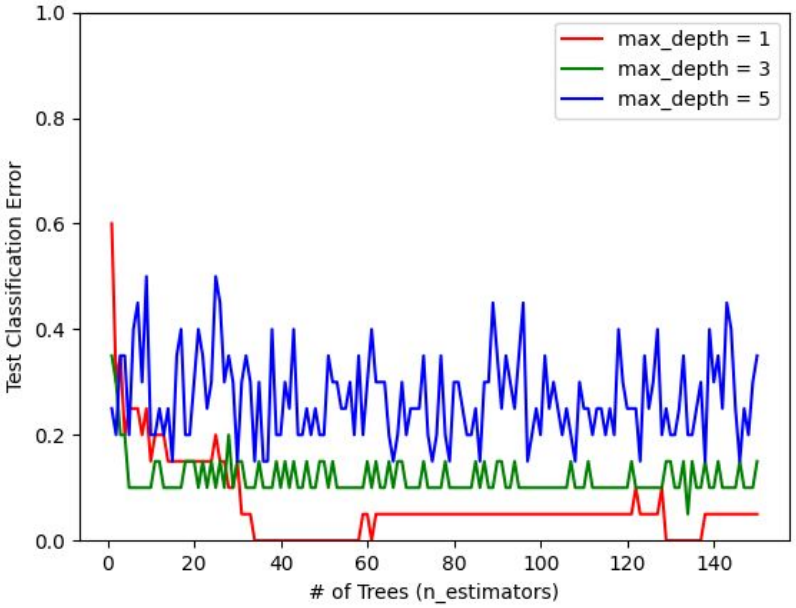


2.5

Basically, greater the “n_estimators”, we get better performance. Looking at the graph, it is overfitting. When “max_features” is more, the model splits on less significant features as well.

2.6

Figure can also be found in Figures folder -> “ensemble_adaboost_Q2.6_plot”



2.7

From the plot, Greater the max_depth, greater the test classification error.
The models can become more skilled if we increase the “max_depth” and the ensemble will perform better.

3. Stack different models [10 points]

f1 score = 0.9662019994769978

I decided to use Random Forest, SVC, Gaussian Naive Bayes, KNeighbors, and Decision Tree as estimators (Base Models) to my Stacking Classifier. Previously I only used KNeighbors and was getting an f1 score of 0.93. But, after adding other model classifiers (stacking models), I was able to achieve an f1 score >= 0.95. Also, the final estimator used by default was LogisticRegression and the cross-validation by default that was used was 5. My stacking model is trained on the predictions made by the models I mentioned above.

Sources:

https://scikit-learn.org/stable/modules/svm.html
https://towardsdatascience.com/linear-svm-classifier-step-by-step-theoretical-explanation-with-python-implementation-69767437e0e6
https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html
https://maviccprp.github.io/a-support-vector-machine-in-just-a-few-lines-of-python-code/