For all models I used the default parameters except for the ones stated in the text.

1   Vanilla Neural Network

Hidden Layer Size	Fit Time	Score
(10,)	99.88s	0.927113
(20,)	100.26s	0.931188
(30,)	74.61s	0.933052
(40,)	376.35s	0.935090
(50,)	163.39s	0.933744
(40, 40)	86.75s	0.936723
(40, 40, 40)	163.92s	0.934984
(40, 40, 40, 40)	249.26s	0.932158

Using (40,40) and varying the activation function:

Activation	Fit Time	Score
'relu'	86.75s	0.936723
'logistic'	168.04s	0.930813
'tanh'	164.20s	0.937665
'identity'	41.14s	0.893321

The activation function 'tanh' gave the best results but takes twice the time 'relu' does.

Considering the time to train the model I choose a model with **hidden layer: (40,40), activation: 'relu'**.

The number of hidden layers helps a neural network to learn complex relationships, however too many hidden layers lead to overfitting which reduces performance. It seems like two hidden layers with 40 neurons lead to a model that understands the relationship of the features well. The activation functions lie close to each other, I believe the identity functions is the worst performing one as it is a linear function and the data is not linearly separable.

2   SVM

Except for the stated hyperparameters all others are the default from scikit-learn. varying C on rbf:

C: 0.1	fit-time: 331.12s	test-score: 0.877185
C: 1	fit-time: 160.40s	test-score: 0.891601
C: 10	fit-time: 148.25s	test-score: 0.906661

varying kernel with C: 10

kernel: 'rbf'	fit-time: 148.25s	test-score: 0.906661
kernel: 'linear'	fit-time: 463.95s	test-score: 0.904653
kernel: 'poly'	fit-time: 274.41s	test-score: 0.832592
kernel: 'sigmoid'	fit-time: 393.91s	test-score: 0.728335

The best parameters are: **'C': 10, 'kernel': 'rbf'** with a test score of 0.906661.

Regularization encourages the model to find a balance between complexity and accuracy which lead to a better performing model. I assume in this example rbf lead to the best results because it does a better job capturing the non-linear relationship between the features and the target. And also because 'rbf' is tuned by a single hyperparameter and I did not tune the other models hyperparameters.

3 Random Forest Classifier

Parameters			Fit Time	Test Score
n-estimators: 50	min-samples-split: 2	max-depth: 10	22.33s	0.924643
n-estimators: 50	min-samples-split: 2	max-depth: 15	30.71s	0.932408
n-estimators: 50	min-samples-split: 2	max-depth: 20	36.04s	0.933984
n-estimators: 50	min-samples-split: 5	max-depth: 10	22.69s	0.924575
n-estimators: 50	min-samples-split: 5	max-depth: 15	30.91s	0.932860
n-estimators: 50	min-samples-split: 5	max-depth: 20	35.91s	0.934388
n-estimators: 50	min-samples-split: 10	max-depth: 10	22.65s	0.924864
n-estimators: 50	min-samples-split: 10	max-depth: 15	31.00s	0.932552
n-estimators: 50	min-samples-split: 10	max-depth: 20	35.57s	0.933110
n-estimators: 100	min-samples-split: 2	max-depth: 10	44.06s	0.924883
n-estimators: 100	min-samples-split: 2	max-depth: 15	62.45s	0.933225
n-estimators: 100	min-samples-split: 2	max-depth: 20	71.87s	0.934695
n-estimators: 100	min-samples-split: 5	max-depth: 10	46.42s	0.925114
n-estimators: 100	min-samples-split: 5	max-depth: 15	62.23s	0.933292
n-estimators: 100	min-samples-split: 5	max-depth: 20	71.88s	0.935205
n-estimators: 100	min-samples-split: 10	max-depth: 10	45.57s	0.925162
n-estimators: 100	min-samples-split: 10	max-depth: 15	64.83s	0.933273
n-estimators: 100	min-samples-split: 10	max-depth: 20	71.10s	0.934484

The best performing model is: **'n-estimators': 100, 'min-samples-split': 5, 'max-depth': 20** with a fit time of 71.88s and a test score of 0.935205.

Considering the time to train the model a random forest with **n-estimators: 50, max-depth: 20 and min-samples-split: 5** is a good performing one with 35sec fit time and 0.934 accuracy.

As seen from the data above the number of estimators or the min-samples-split does not have a significant impact on the performance of the model. I believe max-depth has the biggest impact on the performance because it changes how complex of a pattern the model can learn.

4 Conclusion

VNN-hyperparameters: **{'hidden-layers': (40, 40), 'activation': 'relu'}** Accuracy: 0.939376
SVM-hyperparameters: **{'C': 10, kernel='rbf'}** Accuracy: 0.908468
RF hyperparameters: **{'n-estimators': 50, 'min-samples-split': 5, 'max-depth': 20 }** Accuracy: 0.935186
I believe SVM is the worst performing model because it struggles to capture the complexity and non-linear relationship of the features. It is also the slowest model to train.
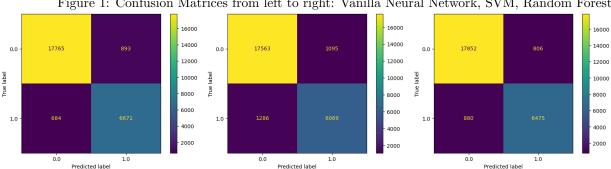
Figure 1: Confusion Matrices from left to right: Vanilla Neural Network, SVM, Random Forest