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.

2 SVM

varying C on rbf:

C	Fit Time	Score
0.1	331.12s	0.877185
1	160.40s	0.891601
10	148.25s	0.906661

varying kernel with C: 10

Kernel	Fit Time	Score
'rbf'	148.25s	0.906661
'linear'	463.95s	0.904653
'poly'	274.41s	0.832592
'sigmoid'	393.91s	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

n-estimators	min-samples-split	max-depth	Fit Time	Test Score
50	2	10	22.33s	0.924643
50	2	15	30.71s	0.932408
50	2	20	36.04s	0.933984
50	5	10	22.69s	0.924575
50	5	15	30.91s	0.932860
50	5	20	35.91s	0.934388
50	10	10	22.65s	0.924864
50	10	15	31.00s	0.932552
50	10	20	35.57s	0.933110
100	2	10	44.06s	0.924883
100	2	15	62.45s	0.933225
100	2	20	71.87s	0.934695
100	5	10	46.42s	0.925114
100	5	15	62.23s	0.933292
100	5	20	71.88s	0.935205
100	10	10	45.57s	0.925162
100	10	15	64.83s	0.933273
100	10	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, min-samples-split: 5, max-depth: 20** 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: 0.939068
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