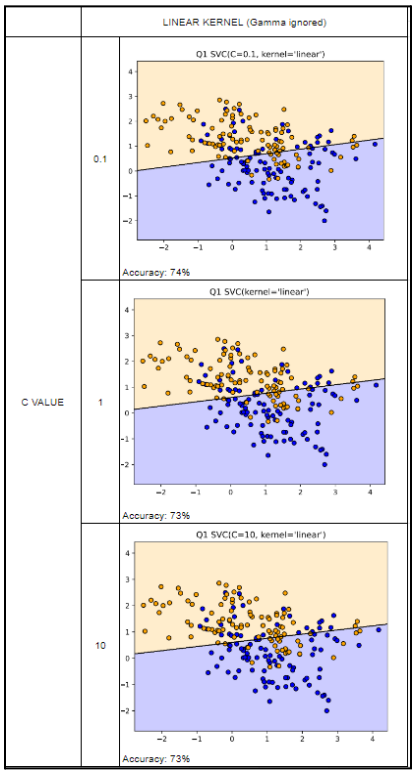
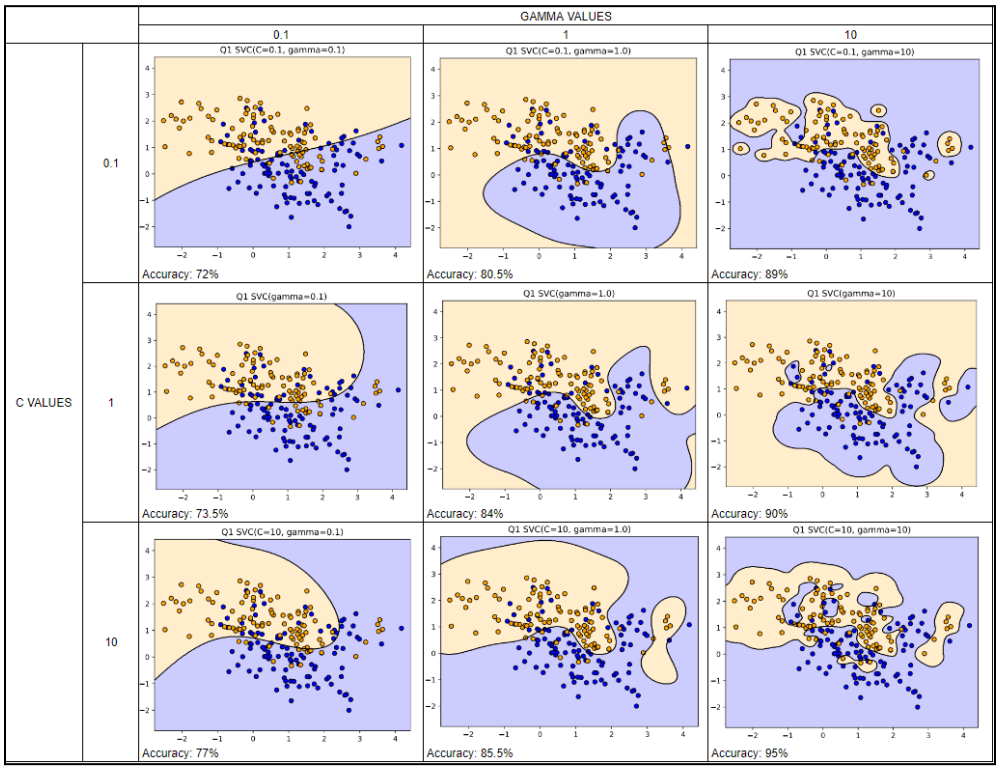
Conor Gagliardi, Machine Learning, Assignment 3 Writeup

Q1)

Linear Kernel



Radial Basis Function Kernel



A) Effect of changing the kernel parameter on class scores and decision boundaries.

- The linear kernel produces a linear boundary in the input feature space. Using a different kernel parameter such as the Radial Basis Function produces a boundary that is linear in an expanded feature space (increasing dimensions). When a linear boundary from the enlarged feature space is translated to the input space, it appears as a nonlinear boundary, allowing for better class separation, which typically results in higher accuracy numbers for class scores, given that the inputs are not discretely linearly separable in a 2d grid.

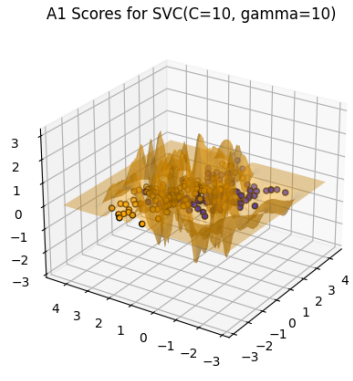
B) Effect of changing the kernel, C value, and gamma values on classification speed and why.

- Changing the kernel value affected the total duration depending on the C value. When the C value was 1, the duration for the linear kernel was the fastest. However, with C values of 0.1 or 10, the linear kernel had the highest durations compared to RBF kernel durations with the same C value. As previously mentioned, changing the C value had the greatest effect on total duration when it was changed from 1 in either direction (0.1 or 10 value) with the linear kernel, and with the rbf kernel, a higher C value was associated with lower durations. Alternatively, higher gamma values were associated with higher duration times. These results are because as the C value lowered, the margin of the SVM widened accounting for more correct points providing weights but more points overall to calculate. With a higher C value, the margin thinned giving more significant weight to incorrect points. The increasing gamma value caused the gaussian curves of the scores to narrow, revealing more areas of

varied class estimates (points having more “weight”) and requiring more calculations to predict the class boundaries.

C) Effect of changing C value and gamma value on decision boundaries and classification accuracy, individually and together.

- By increasing the cost (C value), the margin for support vectors is lowered, so with the low C value, more correctly classified points would fall into the margin calculation. This has a greater effect in the RBF kernel than in the linear kernel, which is "rigid" and not sensitive to changes in C values. By increasing the gamma values, the peaks of the RBF kernel are narrowed (greater dropoff with distance from points), and as gamma drops, the peaks are lower and wider. With a higher and narrower gaussian peak associated with a high gamma value, lower C values allow the greatest amount of "regulation," smoothing the boundaries and causing less overfitting. As seen in the 3D graph with a C value of 10 and a gamma value of 10, a higher C value makes the "neighborhood" of points for the kernel smaller. Paired with narrow peaked kernels, the combination will cause overfitting, similar to the nearest neighbors classifier with a low set number of neighbors, such as 1.

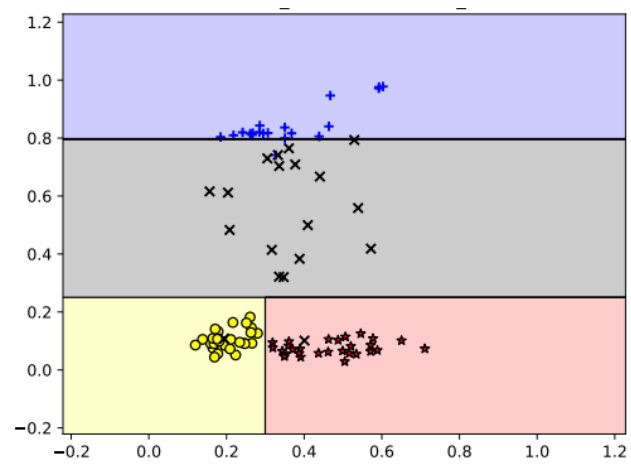
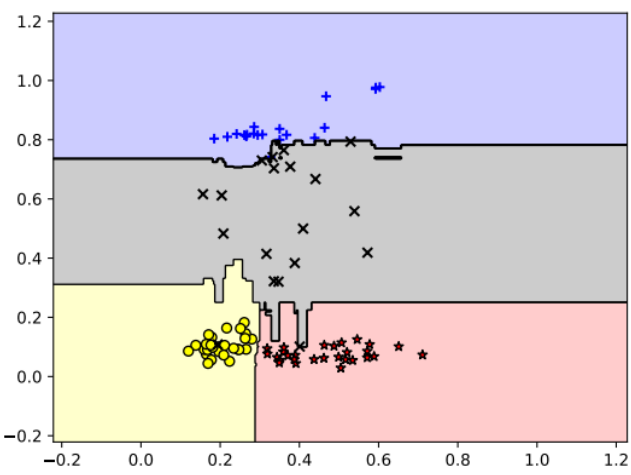


D) Identified decision boundaries that may be overfit or underfit and why.

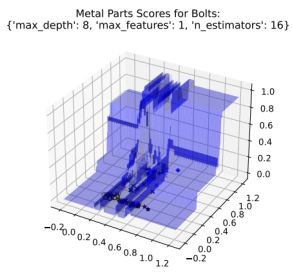
- The linear boundaries and 0.1 C and 0.1 gamma-valued boundaries may be underfit for not following the data shape. In the linear case, it is because a linear boundary is not optimal for data which is not easily separable in two dimensions. With the low C value and low gamma value, the margin of the SVM is wide and the peaks of the kernel are low, almost appearing linear in 2 dimensions and overgeneralizing on the data’s shape. All charts with gamma values of 10 may be overfit based on the nature of the peaks being narrow and individual points having too much “weight”. Even with a lower C value, the boundaries are tight around the class differences and do not generalize on the data like charts with gamma values of 1 or 0.1.

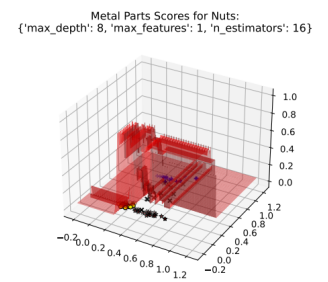
Q2)

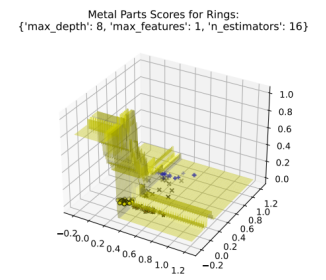
Decision Boundary Comparisons

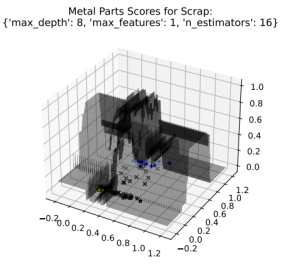
Single Decision Tree	Best Performing Random Forest
Q2 DTree RandomForestClassifier(max_features=None, n_estimators=1, random_state=0)	Q2 RF{'max_depth': 8, 'max_features': 1, 'n_estimators': 16}
	
Accuracy:	

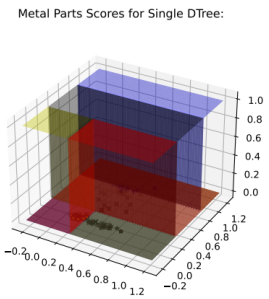
3D plots of the Single Decision Tree (bottom), and Best Performing Random Forest (Top)











A) The purpose of the RANDOM_SEED parameter

- The purpose of the random seed parameter is to keep the “randomness” of the tree consistent between runs of the program. Setting the random_state to a constant ensures the same outcomes will occur so performance can be accurately measured.

B) The difference in accuracy and speed between the single decision tree and the best performing random forests identified by grid search.

- The single decision tree's accuracy is lower than that of the best performing random forest, but its speed is faster. This is because the single decision tree performs a single split between data areas, whereas the random forest performs multiple decision splits on multiple trees giving more context for data generalization and form.

C) Cross-validation definition, and how it is used in scikit-learn grid search. Specifically how training data is used.

- Cross validation randomly divides a training set into subgroups and compares their training results to those of an untested subset to test performance accuracy. In scikit-learn grid search, subsets of training data are used with cross-validation to tune and evaluate hyperparameter values, selecting the best performing values.

D) Comment on decision boundaries of the best performing model, why they are shaped the way they are and presence of over/under-fitting. Then comment on class scoring functions produced for single decision tree and best-performing forest

- The decision boundaries of the best performing model are rigid. They are shaped as they are because of the nature of linear partitions in the layers of the decision trees. The model appears to be overfit based on the decision boundary's structure and its accuracy score. The shape of the class scoring functions for the single decision tree is binary. The score is 1 if it falls within the boundaries of a specified class, and 0 otherwise. The best performing random forest has scores between 0 and 1 and has prominent peaks and rough slopes of scores decreasing to 0 based on normalizing a process of “voting” from each of the different trees.

Q3

SVM Grid Search

Parameters:

```
C_values = [0.0001, 0.001, 0.01, 0.1, 1, 10]
kernels = ['linear', 'rbf', 'sigmoid', 'poly']
gamma_values = [0.0001, 0.001, 0.01, 0.1, 1, 10]
degree_values = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] #for poly kernel

parameter_values = [
    {"C": C_values, "kernel": [kernels[0]]},
    {"C": C_values, "kernel": [kernels[1], kernels[2]], "gamma": gamma_values},
    {"C": C_values, "kernel": [kernels[3]], "gamma": gamma_values, "degree": degree_values}
]
```

Parameter Reasoning:

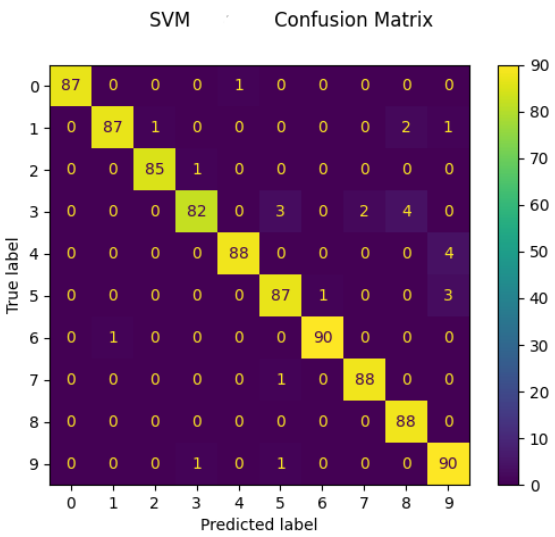
C_Values I used a range from 0.0001 to 10 which covers a wide margin for support vectors to a narrow margin. I left the max value to 10 and minimum to 0.0001 because further values did not produce any significant difference in results of any kernel.

Gamma_Values I used a range from 0.0001 to 10 which covers low peak, wide gaussian peaks to high peak, narrow width gaussian peaks. Like with C_Values points beyond 0.0001 and 10 did not produce any significant difference in results.

Kernels I used the two kernels presented to us, as well as two others including sigmoid and poly to test if they would produce higher results. The poly kernel produced the best results until I included the C_Value of 0.0001 which caused rbf to have the best result of the bunch.

Degree_values used specifically for the poly kernel which produced the best outcome with degree 2, having an accuracy of 95.2% +/- 3.341%

Confusion Matrix:



Report_Metrics:

```
Metrics for SVM CLF
Running GridSearchCV(estimator=SVC(), n_jobs=-1,
             param_grid=[{'C': [0.0001, 0.001, 0.01, 0.1, 1, 10],
                          'kernel': ['linear']},
                         {'C': [0.0001, 0.001, 0.01, 0.1, 1, 10],
                          'gamma': [0.0001, 0.001, 0.01, 0.1, 1, 10],
                          'kernel': ['rbf', 'sigmoid']},
                         {'C': [0.0001, 0.001, 0.01, 0.1, 1, 10],
                          'degree': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                          'gamma': [0.0001, 0.001, 0.01, 0.1, 1, 10],
                          'kernel': ['poly']}],
             refit=<function refit_accuracy at 0x7f9134509dc0>,
             scoring=['accuracy']):
              precision    recall  f1-score   support

           0      1.000     0.989     0.994        88
           1      0.989     0.956     0.972        91
           2      0.988     0.988     0.988        86
           3      0.976     0.901     0.937        91
           4      0.989     0.957     0.972        92
           5      0.946     0.956     0.951        91
           6      0.989     0.989     0.989        91
           7      0.978     0.989     0.983        89
           8      0.936     1.000     0.967        88
           9      0.918     0.978     0.947        92

    accuracy                          0.970       899
   macro avg      0.971     0.970     0.970       899
weighted avg      0.971     0.970     0.970       899
```

Worst And Best Performing Model:

Excluding poly and sigmoid, the lowest performance was the rbf kernel with a C and Gamma value of 10 with 10.1% accuracy +/- 0.241%. The best performing model had the parameters with a C value of 10, gamma value of 0.001, and used the RBF kernel which resulted in a final accuracy of 97% against testing data.

RF Grid Search

Parameters:

```
nTrees = [2, 4, 8, 16, 32, 64, 128, 256]
maxDepth = [1, 2, 4, 8, 16, 32, 64, 128]
nFeatures = ['sqrt', 'log2', 1, 10, 100]

parameter_values = [
    {"n_estimators": [1], "max_depth": [None], "max_features": [None] },
    {"n_estimators": nTrees, "max_depth": maxDepth, "max_features": nFeatures }
]
```

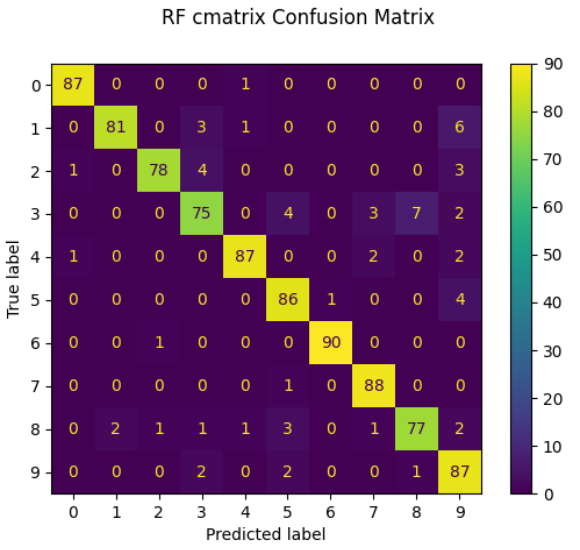
Parameter Reasoning:

nTrees I used 2 trees, up to 256 with any further not producing significant results and increasing run time substantially. 256 as a maximum was sufficient to split up the pixel matrix, diversifying the “voting” for predictions and any further did not produce any favorable results.

maxDepth A depth of 128 was a sufficient maximum depth any further did not improve results. This is likely due to the amount of splits performed for each tree and containing the optimal tree size.

nFeatures 1, 10, and 100 chosen to vary the features each tree would split on. Sqrt and log2 were shown in documentation, so I implemented them as well to observe their results. Neither proved to give better performance however.

Confusion Matrix:



Report_Metrics:

```
Metrics for Random Forest CLF
Running GridSearchCV(estimator=RandomForestClassifier(random_state=0), n_jobs=-1,
             param_grid=[{'max_depth': [None], 'max_features': [None],
                          'n_estimators': [1]},
                         {'max_depth': [1, 2, 4, 8, 16, 32, 64, 128],
                          'max_features': ['sqrt', 'log2', 1, 10, 100],
                          'n_estimators': [2, 4, 8, 16, 32, 64, 128, 256]}],
             refit=<function refit_accuracy at 0x7f9134509dc0>,
             scoring=['accuracy']):
              precision    recall  f1-score   support

           0      0.978     0.989     0.983        88
           1      0.976     0.890     0.931        91
           2      0.975     0.907     0.940        86
           3      0.882     0.824     0.852        91
           4      0.967     0.946     0.956        92
           5      0.896     0.945     0.920        91
           6      0.989     0.989     0.989        91
           7      0.936     0.989     0.962        89
           8      0.906     0.875     0.890        88
           9      0.821     0.946     0.879        92

    accuracy                          0.930       899
   macro avg      0.933     0.930     0.930       899
weighted avg      0.932     0.930     0.930       899
```

Worst and Best Performing Model:

The worst performing model had a max_depth of 1, max_features of 1, and n_estimators of 2. The resulting accuracy was 21.8% +/- 3.498%. The best performing model had the parameters max_depth of 16, max_features of 10, and n_estimators of 256, which resulted in a 93% accuracy against test data.

Analysis:

Depending on the kernel, the grid searches' outcomes for the SVM differed. I discovered that only the change in degree, which enhanced the poly kernel's outcomes as it decreased, has an impact on the poly kernel; c and gamma values have no bearing. With higher C values and lower gamma

values, the sigmoid kernel performed at its best. With C values greater than or equal to 0.01, the linear kernel performed optimally, with scores declining as C values decreased. The RBF kernel had varying performances and tended to perform better with higher C values paired with lower gamma values.

When splitting on 10 max features (digits 0-9) and 256 trees, the max depth parameter for the RF model was best with values of 16, 32, 64, and 128. This shows that no more substantial splits could be made on the trees after 16. Using a max feature split of sqrt, which took the square root of the number of features in the data set, performed slightly worse by 1%. The feature set's log2 split approach was the third-best method. Even worse performances were achieved with fewer depths, fewer trees, or arbitrary feature numbers like 1 or 100.

The best-performing SVM parameters met my expectations because nearest neighbors and RBF classification are similar, but the linear kernel's performance subverted my expectations, because I did not expect linearly separating the matrix would be able to reliably estimate 10 different classes to greater than 90% accuracy. I had expected the Random Forest classifier to perform better than the SVM in comparison, but the tests revealed that it had 4% less accuracy. I can see why the parameters were chosen after seeing the results. Up to a limit dependent on the number of pixels in a specific matrix, the depth of the trees beyond a certain point did not create further splits, and the more trees would provide a more accurate representation of the data. Based on the amount of classes to predict, I had anticipated that 10 features would produce the best results.