William Huang
CS 6140
Summer 2025

Predicting Breast Tumor Malignancy with Supervised Learning Models

Introduction:

Breast cancer is one of the most prevalent and life-threatening diseases that affect women worldwide. Obtaining an early and accurate diagnosis is critical to improving treatment outcomes and survival rates. Recently, machine learning has become a powerful tool for medical diagnostics because of its ability to detect complex patterns in clinical data that are not easily seen by people.

Prior work on the Breast Cancer Wisconsin Diagnostic (WDBC) dataset focused on developing robust linear separation methods for classification. One notable approach was the Multisurface Method-Tree (MSM-T), introduced by K. P. Bennett in 1992. This method utilized robust linear programming to create a decision tree that created separating planes between classes even with noise and outliers. This study demonstrates that the WDBC dataset can be linearly separable. Other notable approaches include a regularized General Linear Model (GLM) and a Support Vector Machine (SVM) with radial basis function (RBF) kernel, which were designed by Sidey-Gibbons and Sidey-Gibbons in 2019. Both models achieved high performance with accuracy ranging from 94% to 96%. In particular, the SVM with the RBF kernel achieved the highest accuracy by itself (96%). This study shows that other models like the SVM with the RBF kernel will be extremely successful on this dataset.

This project applies supervised learning techniques to predict breast tumor malignancy from quantitative features in medical images. Using the WDBC, I developed four models capable of accurately distinguishing between malignant and benign tumors. The Diagnostic Wisconsin Breast Cancer dataset contains 569 digitized images of breast tissue samples and 30 continuous numerical features. The four models utilize the following supervised learning algorithms: decision trees, logistic regression, random forest classification, and SVM. My ultimate goal in creating these models is to support physicians in making informed decisions.

Methods:

For each of the supervised learning algorithms, I made a manual version which was implemented by hand and a library version which was implemented using the corresponding module from the sklearn library. The decision tree and logistic regression algorithms serve as baselines to compare to the random forest classification and SVM algorithms. All algorithms were evaluated using multiple performance metrics: accuracy, recall, precision. Additionally, confusion matrices were made for all versions of each algorithm to compare the true positive, true negative, false positive, and false negative cases.

Results:

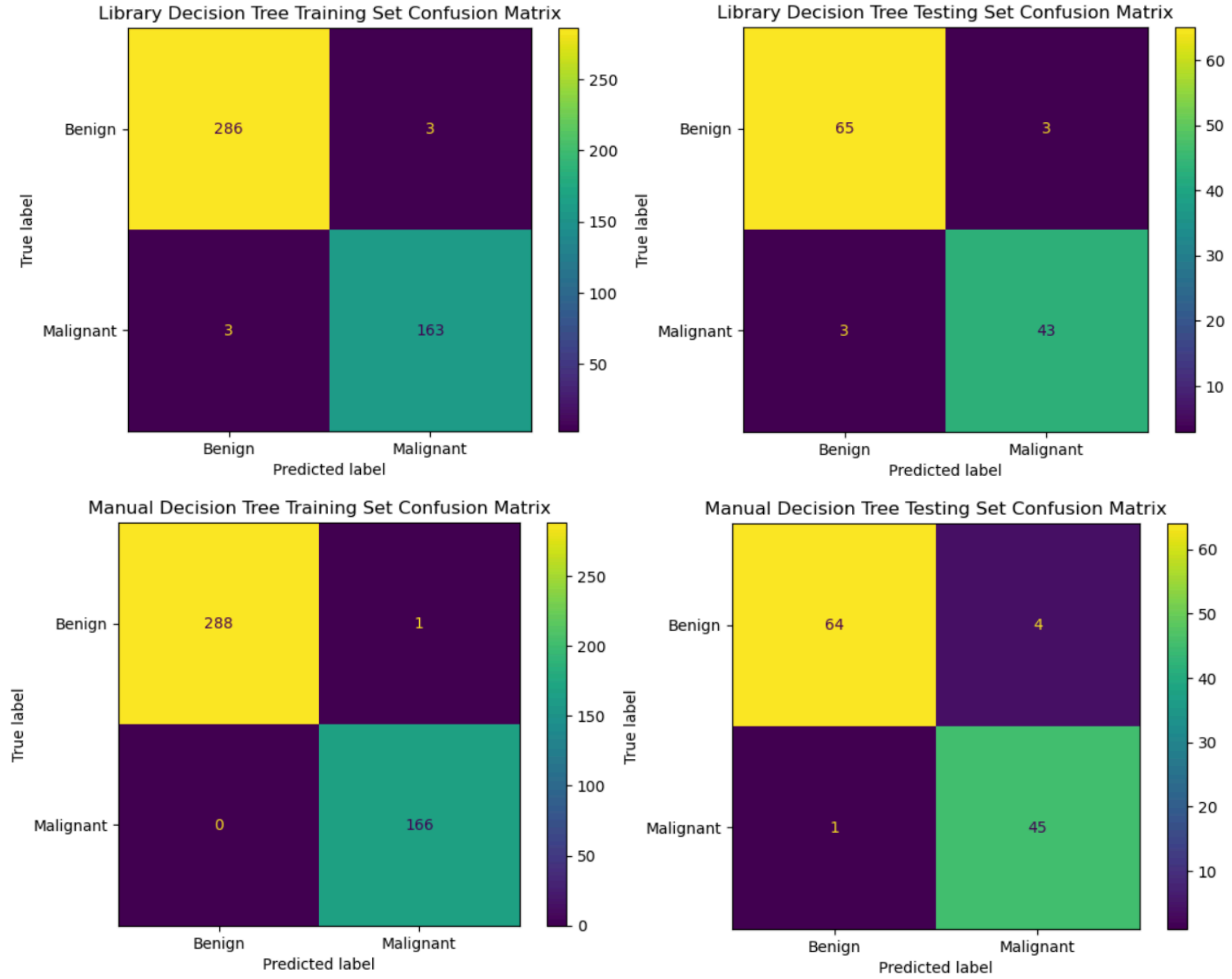
Decision Tree:

To evaluate the performance of the decision tree algorithm, I compared the library implementation with a manual version using accuracy, recall, and precision. Table 1 and Figure 1 present the results of this comparison.

Table 1: Decision Tree Performance Metrics:

	Accuracy	Recall	Precision
Library Training Set	0.9912	0.9765	1
Library Testing Set	0.9474	0.9524	0.9091
Manual Training Set	0.9956	1	0.9884
Manual Testing Set	0.9386	1	0.8571

Figure 1: Decision Tree Confusion Matrices (Library and Manual versions)



Library Decision Tree Training Set Confusion Matrix

Library Decision Tree Testing Set Confusion Matrix

Manual Decision Tree Training Set Confusion Matrix

Manual Decision Tree Testing Set Confusion Matrix

From Table 1 and Figure 1, we can see that both implementations of the Decision Tree classifiers perform very well on the training set based on the nearly perfect accuracy, recall, and precision metrics. This suggests that the model closely fits the training data. In the testing set, both implementations of the Decision Tree classifiers performed slightly worse but still performed pretty well. There is a small drop in the testing set accuracy and larger drop in the testing set precision scores, indicating that the models are producing more false positives. However, the recall remains really high for both implementations on the testing set and is really effective at identifying true positive cases, making it useful for applications where missing a malignant tumor is more critical than identifying false positives.

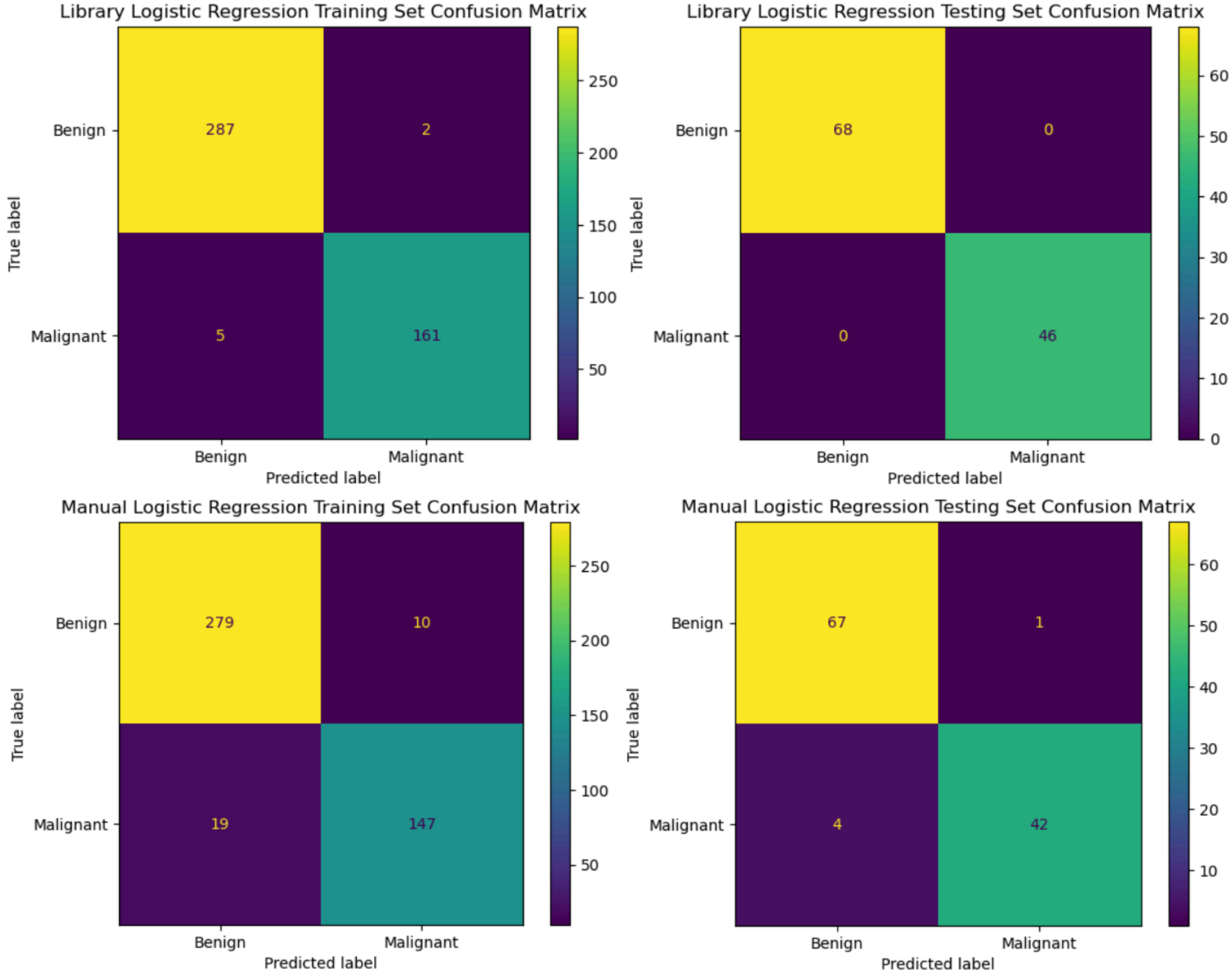
Logistic Regression:

To evaluate the performance of the logistic regression algorithm, I compared the library implementation with a manual version using accuracy, recall, and precision. Table 2 and Figure 2 present the results of this comparison.

Table 2: Logistic Regression Performance Metrics:

	Accuracy	Recall	Precision
Library Training Set	0.9846	0.9706	0.9880
Library Testing Set	1	1	1
Manual Training Set	0.9385	0.8882	0.9437
Manual Testing Set	0.9561	0.9286	0.9512

Figure 2: Logistic Regression Confusion Matrices (Library and Manual versions)



Library Logistic Regression Training Set Confusion Matrix

Library Logistic Regression Testing Set Confusion Matrix

Manual Logistic Regression Training Set Confusion Matrix

Manual Logistic Regression Testing Set Confusion Matrix

From Table 2 and Figure 2, we can see that both implementations of the Logistic Regression classifiers perform well on the training set. This suggests that the model closely fits the training data. In the testing set, both implementations of the Decision Tree classifiers performed better than they did on the training data. The library implementation achieved perfect scores across all metrics while the manual implementation performed strongly across all metrics. These strong results indicate that the model generalizes well and does not overfit the data. From these observations, logistic regression, which models linear relationships between features and outcomes, works well for this classification problem. Additionally, these results indicate that the breast cancer dataset is likely linearly separable.

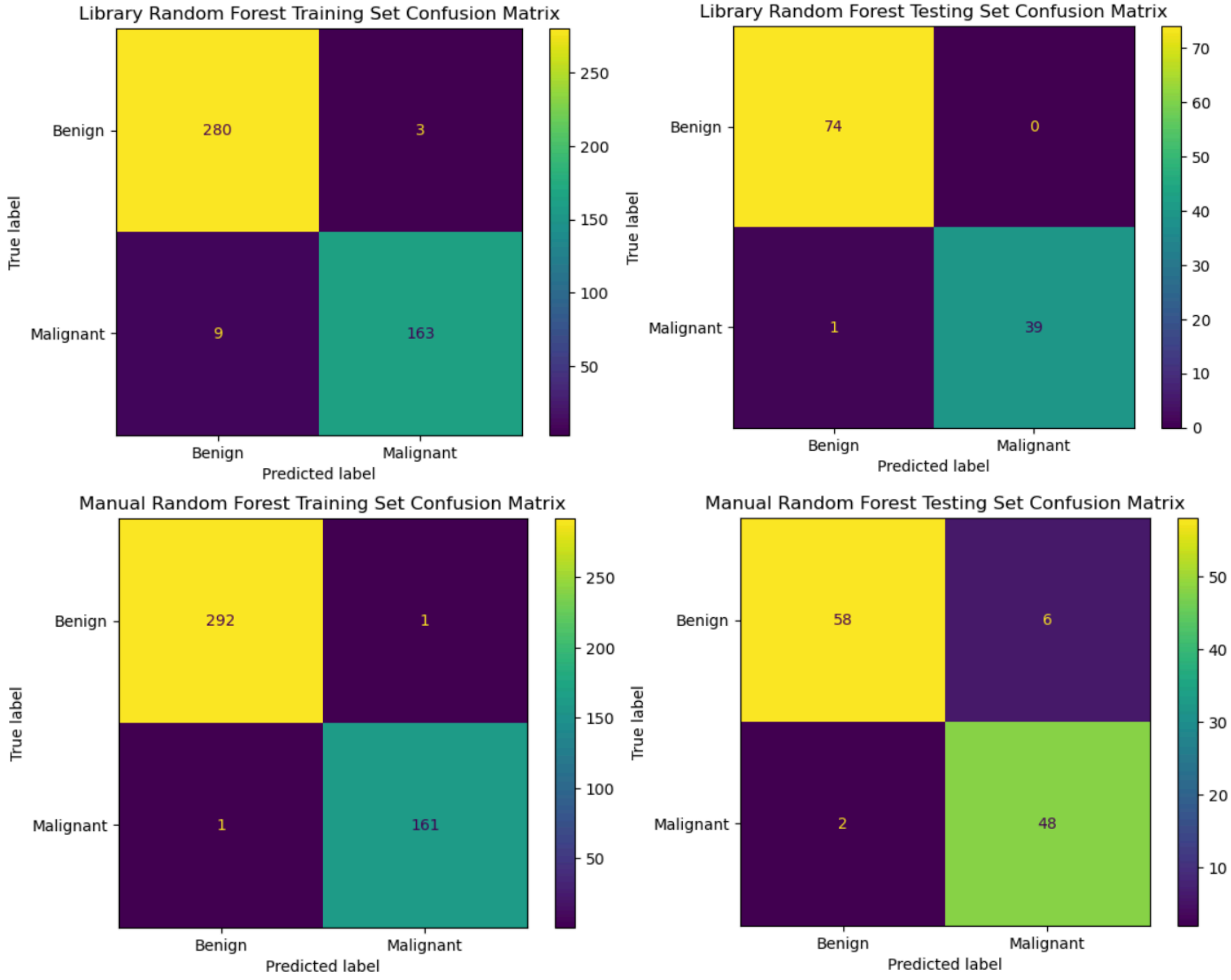
Random Forest:

To evaluate the performance of the random forest algorithm, I compared the library implementation with a manual version using accuracy, recall, and precision. Table 3 and Figure 3 present the results of this comparison.

Table 3: Random Forest Performance Metrics:

	Accuracy	Recall	Precision
Library Training Set	0.9736	0.9477	0.9819
Library Testing Set	0.9912	0.9750	1.0000
Manual Training Set	0.9956	0.9938	0.9938
Manual Testing Set	0.9298	0.9600	0.8889

Figure 3: Random Forest Confusion Matrices (Library and Manual versions)



From Table 3 and Figure 3, we can see that both implementations of the Random Forest classifiers perform similarly to the Decision Tree classifiers. Both implementations performed well on the training set, suggesting that the model closely fits the training data. In the testing set, both implementations of the Random Forest classifiers perform slightly worse but still have strong performances. The small decrease in accuracy and the larger decrease in precision for the manual implementation suggests that the model is producing more false positives. However, both implementations have high recall scores on the testing sets, which is critical to breast cancer detection.

I kept the hyperparameters (number of trees, max depth of each tree, minimum samples required to split node) of both implementations the same to have a more fair comparison between the two. In the manual implementation, I found that if I made the number of trees in the forest too high, the algorithm took a long time to produce results. Thus, I set the number of trees in the forest to be 10 and prevent long run times. I also found that as I increased the maximum depth of each tree, the model started to overfit more because the trees became overly complex after training. Hence, I set the maximum depth of each tree to be 5 to prevent the model from overfitting. Additionally, I found that as I decreased the minimum samples required to split, the model started to overfit more because the trees split a lot and became overly complex after training. So, I set the minimum samples required to split to be 5 to prevent the model from overfitting. In the library implementation, I changed the hyperparameters from the default values to match those of the manual implementation. The maximum tree depth was set to 5 instead of no limit, the number of trees in the forest was limited to 10 instead of the default of 100, and the minimum samples required to split a node was set to 5 instead of the default of 2. When experimenting with those 3 hyperparameters, I found similar results to the manual implementation, thus I wanted to keep the values of the hyperparameters consistent between the two implementations.

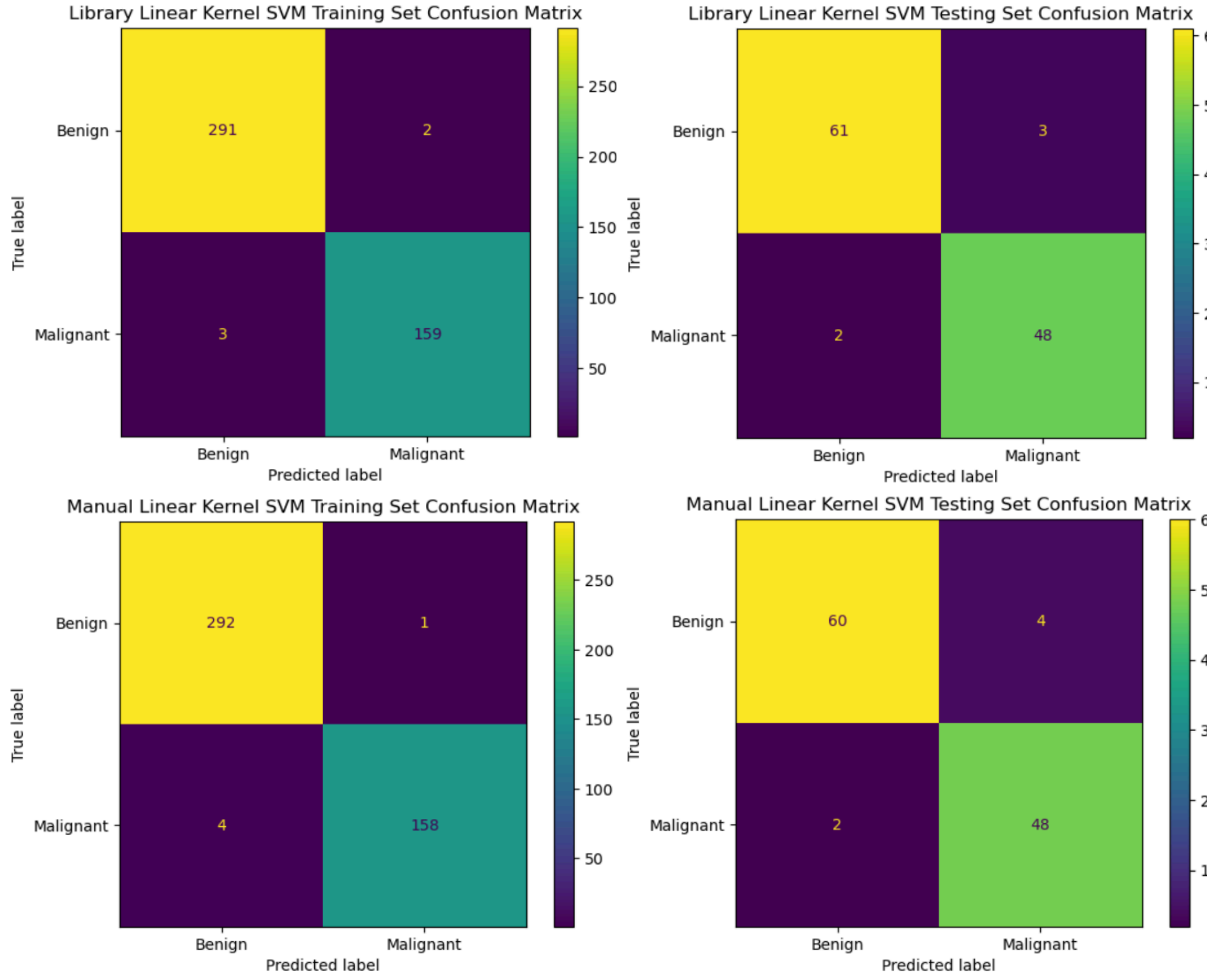
SVM Linear Kernel:

To evaluate the performance of the SVM algorithm, I compared the library implementation with a manual version using accuracy, recall, and precision. Tables 4-7 and Figures 4-7 present the results of this comparison.

Table 4: SVM Linear Kernel Performance Metrics

	Accuracy	Recall	Precision
Library Training Set	0.9890	0.9815	0.9876
Library Testing Set	0.9561	0.9600	0.9412
Manual Training Set	0.9890	0.9753	0.9937
Manual Testing Set	0.9474	0.9600	0.9231

Figure 4: SVM Linear Kernel Confusion Matrices (Library and Manual versions)



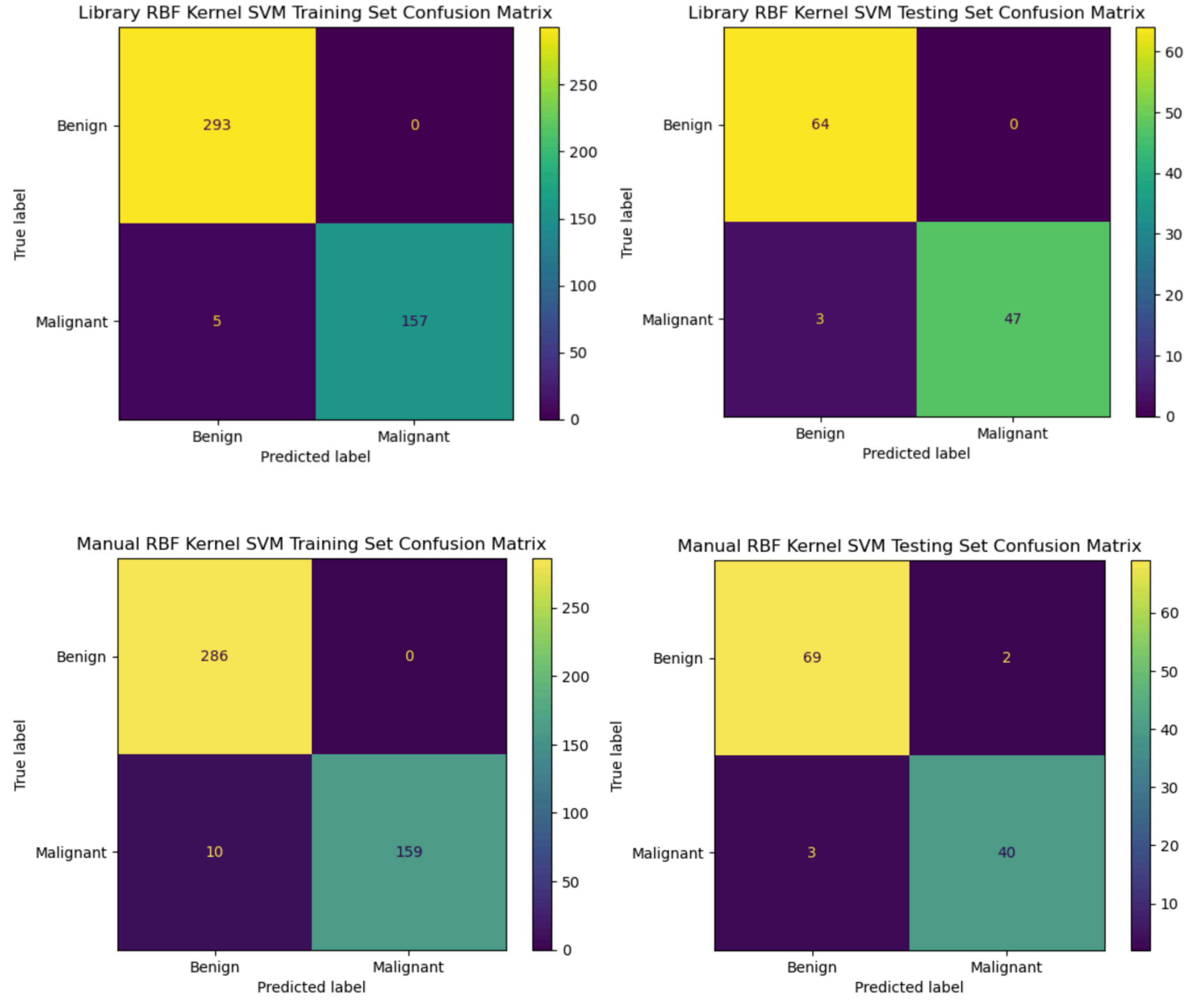
From Table 4 and Figure 4, we can see that both implementations of the Linear SVM classifiers perform well on the training set. This suggests that the model closely fits the training data. The two implementations of the Linear SVM classifier also performed comparably well on the testing set, indicating that the model generalizes well. The recall for both the training and testing sets in both implementations also show the model's reliability to identify most malignant cases, which is critical for cancer detection. The results imply that the data is likely linearly separable, making the Linear SVM model effective for this classification task.

SVM RBF Kernel:

Table 5: SVM RBF Kernel Performance Metrics

	Accuracy	Recall	Precision
Library Training Set	0.9890	0.9691	1
Library Testing Set	0.9737	0.9400	1
Manual Training Set	0.9780	0.9408	1
Manual Testing Set	0.9561	0.9302	0.9524

Figure 5: SVM RBF Kernel Confusion Matrices (Library and Manual versions)


Library RBF Kernel SVM Training Set Confusion Matrix

Library RBF Kernel SVM Testing Set Confusion Matrix

Manual RBF Kernel SVM Training Set Confusion Matrix

Manual RBF Kernel SVM Testing Set Confusion Matrix

From Table 5 and Figure 5, we can see that both implementations of the RBF SVM classifiers perform really well on the training set. This suggests that the model closely fits the training data. The two implementations of the RBF SVM classifier also performed comparably well on the testing set, indicating that the model generalizes well. The recall for both the training and testing sets in both implementations also show the model's reliability to identify most malignant cases, which is critical for cancer detection. From these observations, the RBF SVM classifier tells us that the breast cancer data is separable by carefully tuned non-linear boundaries.

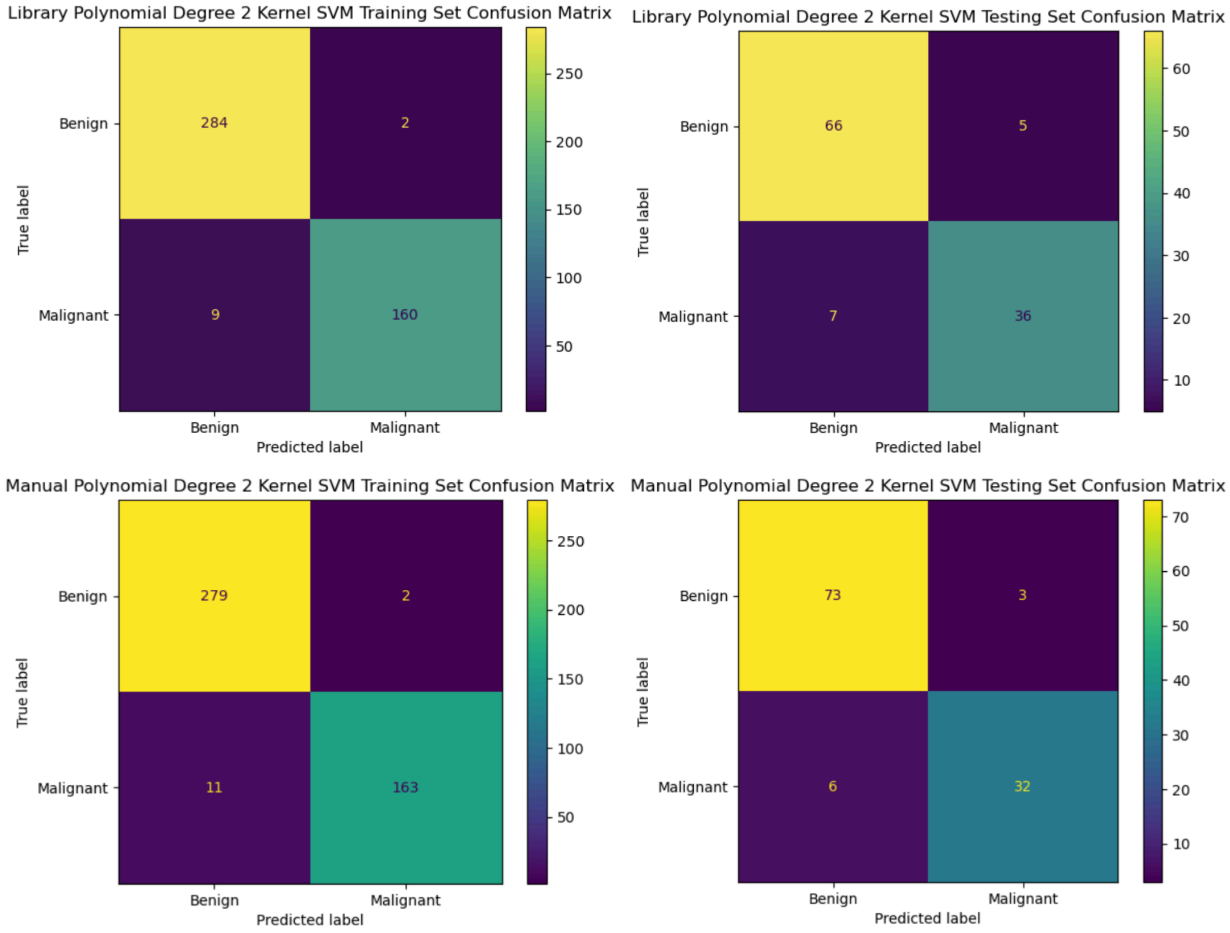
In the sklearn library implementation, I kept the gamma hyperparameter at the default value of 1 / (n_features * X.var()). However, when experimenting with the gamma value, I found that the best results were produced when gamma was near 0.1. When gamma was greater than 0.1, the model overfitted because the kernel became too narrow, allowing the model to draw extremely tight decision boundaries that do not generalize to unseen data. When gamma was smaller than 0.1, the model underfitted because the kernel became too wide, making smooth decision boundaries that cannot adapt to new data. In the manual implementation, I set the gamma hyperparameter to be 5. As I experimented with the gamma value, I found that the best results were produced when gamma was at 5 and the model progressively performed worse as gamma got further away from 5. Again, the model overfitted when gamma was larger than 5 and underfitted when gamma was smaller than 5. The reason why the gamma value in the manual implementation is much larger is because the RBF kernel is represented as the following: $e^{-|X-y|^2/\gamma^2}$, where gamma is in the denominator here. Hence, a larger gamma value in the manual implementation means that the model creates smaller, tighter boundaries.

SVM Polynomial Degree 2 Kernel:

Table 6: SVM Polynomial Degree 2 Kernel Performance Metrics

	Accuracy	Recall	Precision
Library Training Set	0.9758	0.9467	0.9877
Library Testing Set	0.8947	0.8372	0.8780
Manual Training Set	0.9714	0.9368	0.9879
Manual Testing Set	0.9211	0.8421	0.9143

Figure 6: SVM Polynomial Degree 2 Kernel Confusion Matrices (Library and Manual versions)



From Table 6 and Figure 6, we can see that both implementations of the Polynomial SVM classifiers perform decently on the training set, indicating that the model closely fits the training data. Both implementations of the Polynomial SVM classifiers perform the second worst in all metrics (accuracy, recall, precision) on the testing set than all other models. The lower overall recall scores from the testing set indicates that the Polynomial SVM classifiers will miss malignant cases during prediction. From these observations, the model does a decent job not overfitting and can still generalize unseen data if tuned correctly, but it does not outperform any other model during prediction. This tells me that the breast cancer dataset is not likely to be polynomial in structure. Nonetheless, the model shows consistent and balanced results in both implementations, suggesting that it is a viable but not optimal choice for this classification task.

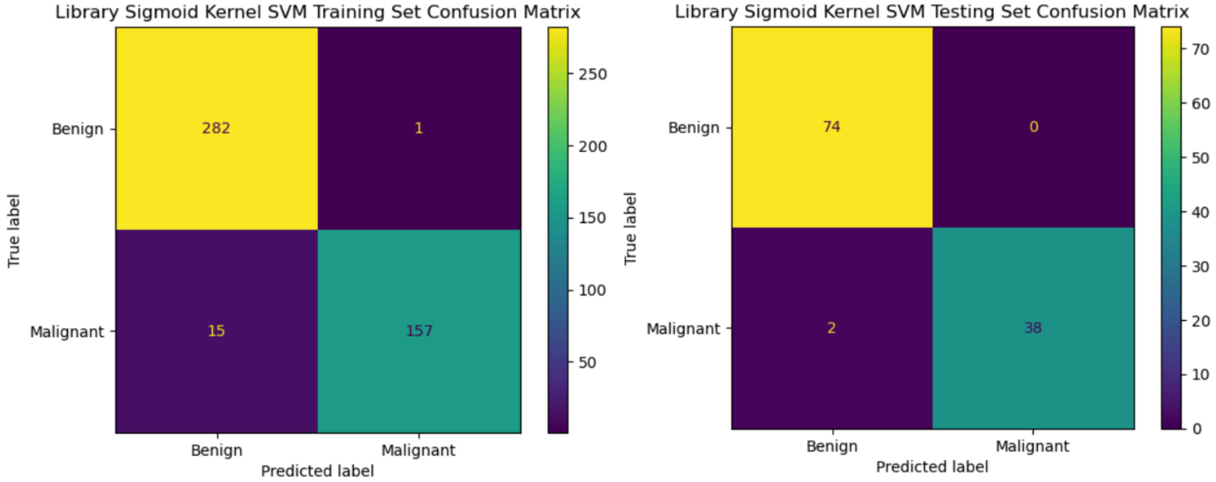
In the sklearn library implementation, I found that the model produced the best results when the alpha hyperparameter is at 0.2 and the coefficient hyperparameter is at 0 for second degree polynomials. When the alpha hyperparameter is larger than 0.2, the decision boundary curve overfits the training data by completely following the shape of the training data. If the alpha hyperparameter is smaller than 0.2, the decision boundary curve underfits the training data and cannot predict unseen data properly. Similar to the library implementation, the manual implementation produced the best results when the alpha hyperparameter is 0.35 and the coefficient hyperparameter is at 0 for second degree polynomials. As I experimented with the alpha value, I found that the best results were produced when gamma was at 0.35 and the model progressively performed worse as gamma got further away from 0.35. Again, the model overfitted when alpha was larger than 0.35 and underfitted when alpha was smaller than 0.35.

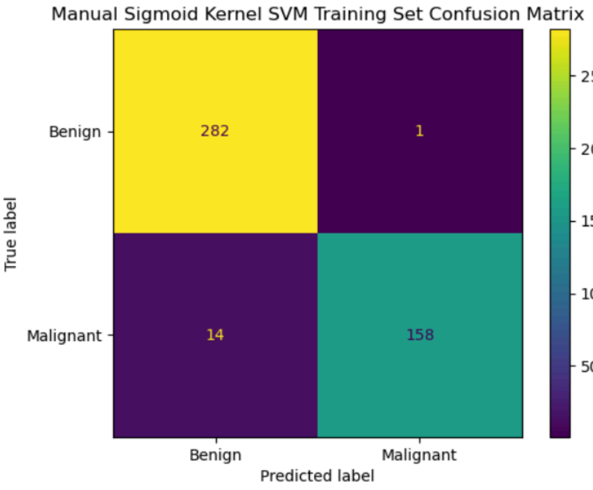
SVM Sigmoid Kernel:

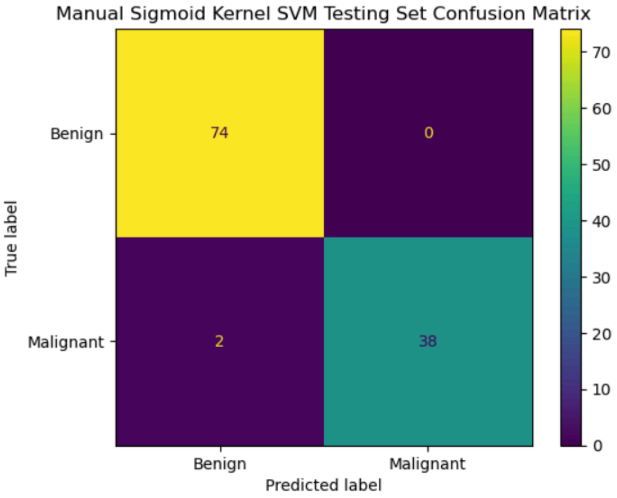
Table 7: SVM Sigmoid Kernel Performance Metrics

	Accuracy	Recall	Precision
Library Training Set	0.9648	0.9128	0.9937
Library Testing Set	0.9825	0.9500	1.0000
Manual Training Set	0.9670	0.9186	0.9937
Manual Testing Set	0.9825	0.9500	1.0000

Figure 7: SVM Sigmoid Kernel Confusion Matrices (Library and Manual versions)







From Table 7 and Figure 7, we can see that the library implementation of the Sigmoid SVM classifier performed decently in both the training and testing sets but had a lower recall value than most of the other models. The manual implementation performed decently in both the training and testing sets, but not nearly as good as the linear or RBF SVM models. From these observations, the model does a decent job not overfitting the training data and is still able to generalize unseen data, but it does not outperform any other model during prediction. This tells me that the breast cancer dataset is not as likely to be sigmoid in structure. Nonetheless, the model shows consistent and balanced results in both implementations, suggesting that it is a viable but not optimal choice for this classification task.

Both implementations were able to produce the best results when the alpha hyperparameter is 0.01 and the coefficient hyperparameter is at 0. When the alpha hyperparameter is much larger than 0.01, the decision boundary curve overfits the training data by completely following the shape of the training data. If the alpha hyperparameter is much smaller than 0.01, the decision boundary curve underfits the training data and cannot predict unseen data properly. The low alpha hyperparameter suggests that the best performing sigmoid kernel is close to a linear model, which means that the data is likely to be linearly separable.

Conclusions:

Overall, most models achieve high performance with proper hyperparameter tuning, but the differences in accuracy, recall, and precision highlight differences on how each model handles complexity and generalization. The Logistic Regression model stands out for its stable and consistently high performance across both the library and manual implementations. In particular, the library implementation achieves perfect scores on the testing set, indicating that the data is likely linearly separable. The Logistic Regression model has fewer hyperparameters, simple assumptions, and performs well, so it is a strong baseline model for this classification task.

The Decision Tree model performed well overall with good metrics in accuracy and recall, but slightly lower precision on the testing set in both implementations. This means that the Decision Tree model identified slightly more false positives than the Logistic Regression model. The Random Forest model also performed well overall with good metrics in the training set and testing set. The manual implementation of the Random Forest model had a lower precision of 0.8889, meaning that the model identified slightly more false positives than the Logistic Regression model. To ensure a fair comparison, I used the same hyperparameters for both the manual and library implementations: 10 trees, a maximum depth of 5, and a minimum of 5 samples required to split a node. These values were chosen to reduce overfitting and keep run times reasonable, based on observations from experimenting with the manual implementation. The Random Forest model performed similarly to the Decision Tree model because the Random Forest algorithm utilizes decision trees to predict labels on the testing set. In the Random Forest algorithm, the model first creates multiple decision trees that each use a random part of the data, making each tree unique to avoid overfitting. When looking at new data, each tree makes a prediction based on what it learned from its random section of the data. Afterwards, the algorithm combines the results of each tree and produces the final classification based on majority vote.

The Support Vector Machines (SVM) with linear kernel performed similarly to the Logistic Regression model and performed very well with high accuracy, recall, and precision across both implementations. This observation further supports the idea that the breast cancer dataset is likely to be linearly separable. Additionally, the SVM with RBF kernel shows an equally strong performance and excels in precision (often scoring 1.0). With a gamma hyperparameter that allows for smaller, tighter boundaries, the SVM with RBF kernel is a robust model for predicting breast cancer across datasets. Moreover, the SVM with sigmoid kernel was not the strongest model overall, but the manual implementation of SVM with sigmoid kernel had one of the highest testing accuracy (0.9825) compared to the manual implementation of other models. This makes the SVM with sigmoid kernel model a good predictor for other breast cancer datasets. The small alpha hyperparameter for the sigmoid kernel does suggest that data is likely to be linearly separable. Lastly, the SVM with polynomial kernel (degree 2) model had the weakest performance among all of the models despite optimizing the hyperparameters, especially on the testing sets where recall and precision dropped significantly. This indicates that the SVM with polynomial kernel did not generalize as well to the testing data and that the breast cancer dataset is less likely to be polynomial shaped.

Hence, the Logistic Regression, the SVM with linear kernel, and the SVM with RBF kernel models showed the best overall performance. The SVM with sigmoid kernel, Decision Tree, and Random Forest models all performed well and are suitable to predict breast cancer diagnoses from other data sets. Meanwhile, the SVM with polynomial kernel model showed the worst overall performance.

References:

1. *Decision Tree Construction Via Linear Programming*. K. P. Bennett. Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992
2. *Machine learning in medicine: a practical introduction*. Sidey-Gibbons, J., Sidey-Gibbons, C. BMC Med Res Methodol 19, 64 (2019). https://doi.org/10.1186/s12874-019-0681-4