Assignment 2

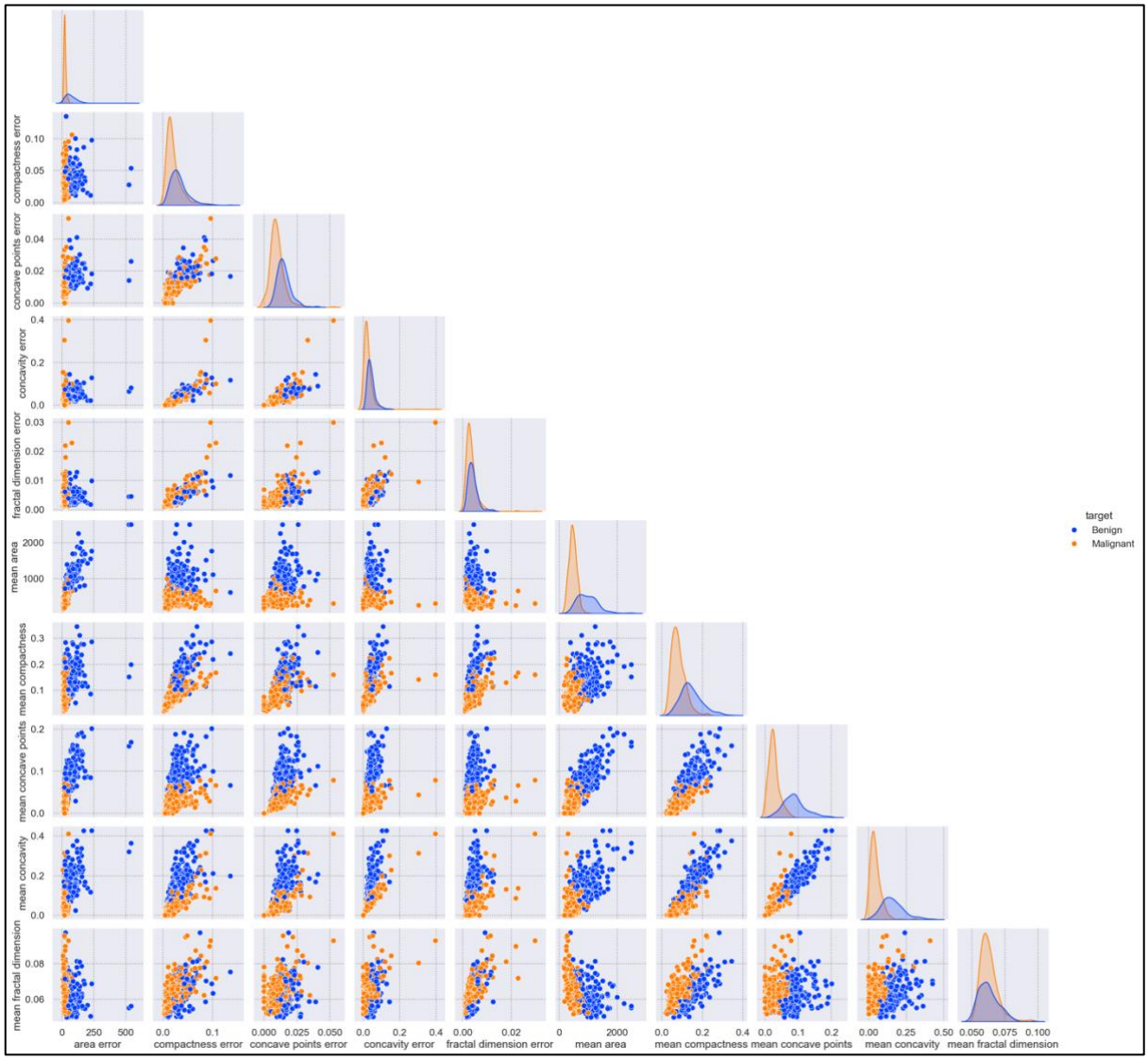
Name: Nicodemus Ong
Student ID: 22607943

D1 2 [marks]:

Re-order the columns in your feature matrix (or dataframe) based on the column name. Provide a scatter plot to inspect the relationship between the first 10 features in the dataset. Use different colours in the visualisation to show instances coming from each class. (Hint: use a grid plot.)

D1 Answer:

Figure 1: The plot below shows a scatter plot with the relationships between the first 10 features in the dataset.



D2 [2 marks]:

Provide a few comments about what you can observe from the scatter plot:

- What can be observed regarding the relationship between these features?
- Can you observe the presence of clusters of groups? How do they relate to the target variable?
- Are there any instances that could be outliers?
  Are there features that could be removed? Why or why not?

D2 Answer:

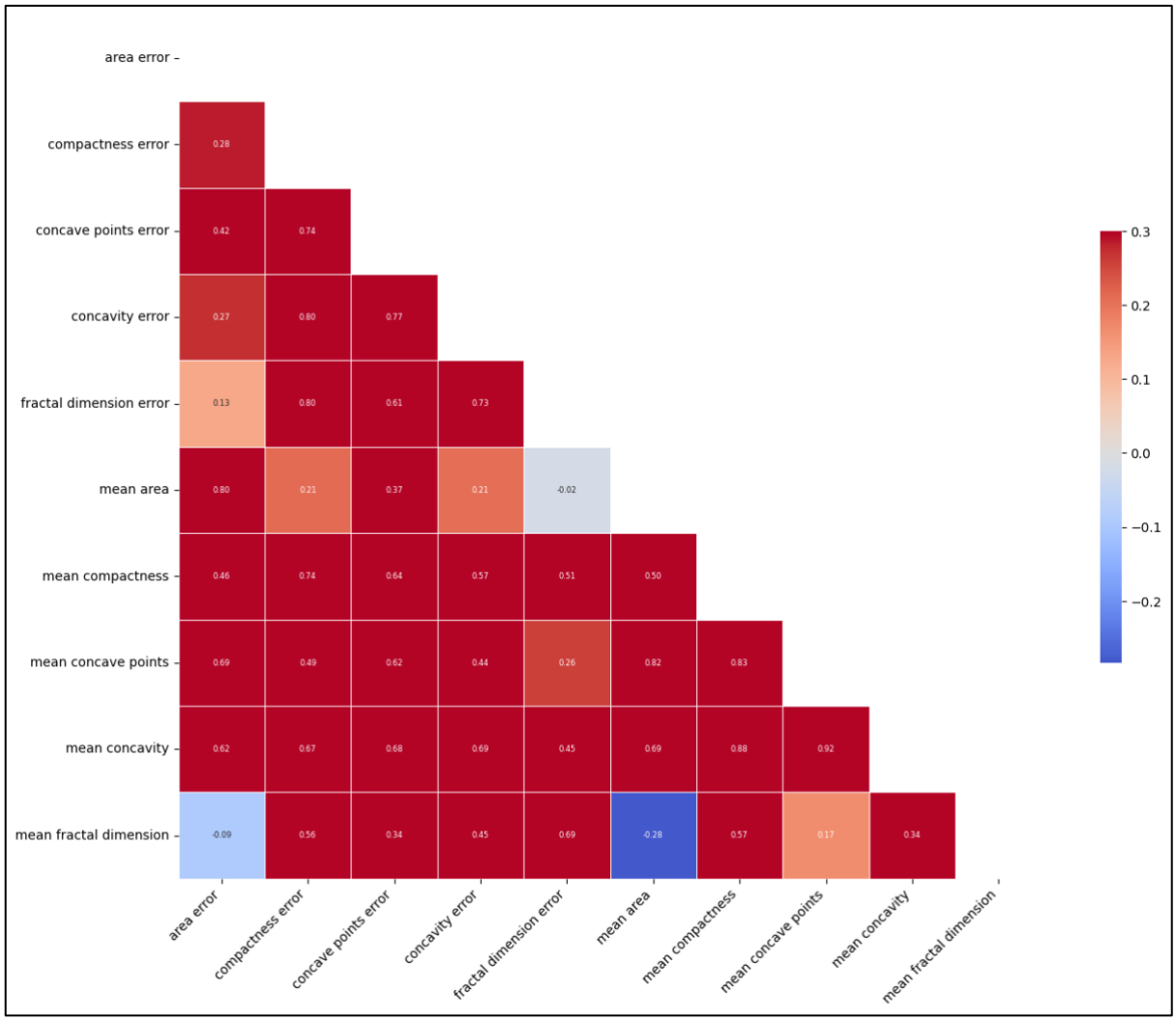
1. The linear trends observed between features such as concave points error and concavity error are typical and indicate how these morphological attributes of cancer cells are interrelated. These features are directly related to the severity and shape of the tumour, which naturally correlate.
2. The clear clustering of points in some scatter plots likely corresponds to the malignant or benign nature of the tumor samples. For instance, features such as mean area vs. area error might show distinct groups where malignant tumors (possibly larger and more irregular) separate from benign ones. This clustering indicates that these features are critical in distinguishing between benign and malignant tumours, making them vital for predictive modelling in medical diagnostics.
3. Outliers, as seen in features like area error, may represent atypical cases or extreme cases of breast cancer. These might be particularly aggressive or large tumors, and while they could be influential in a statistical model, their rarity could also skew the model's general performance.
4. Given the strong correlations between certain features, as observed the scatter plots (e.g., between concavity error and concave points error), it may be sensible to remove one to avoid multicollinearity, especially in models like linear regression. Also, Features that show weak correlations and don't significantly vary between benign and malignant cases might be less useful.

D3 [1 mark]:

Compute and show the correlation matrix where each cell contains the correlation coefficient between the corresponding pair of features (Hint: you may use the heatmap function from the seaborn package).

D3 Answer:

Figure 2: The heatmap plot shows the correlation matrix between the corresponding pair of features.



D4 [1 mark]:

Do the correlation coefficients support your previous observations?

D4 Answer:

The heatmap's high correlation values confirm the suspected redundancy among some features. It's also beneficial for quickly pinpointing which features are least correlated with others, suggesting potential candidates for removal to simplify the model without sacrificing predictive power.

D5:

In a data science project, it's crucial not just to remove highly correlated features but to consider the context and implications of feature selection carefully. Blindly dropping features may lead to the loss of valuable information or unintended bias in the model's performance. Here, for the assignment context, we will drop a few features to simplify the classification tasks and speed up the computational time. Create a code that drop the features: mean perimeter, mean radius, worst radius, worst perimeter and radius error. These are features with a linear correlation higher than 0.97 in magnitude with some other features kept in the data.

After this process, your data matrix should be updated accordingly and contain 25 features. Task D5 must be performed; otherwise, your order deliverable tasks will be incorrect. However, there are no marks for task D5.

Fitting a Decision Tree model with default hyperparameters

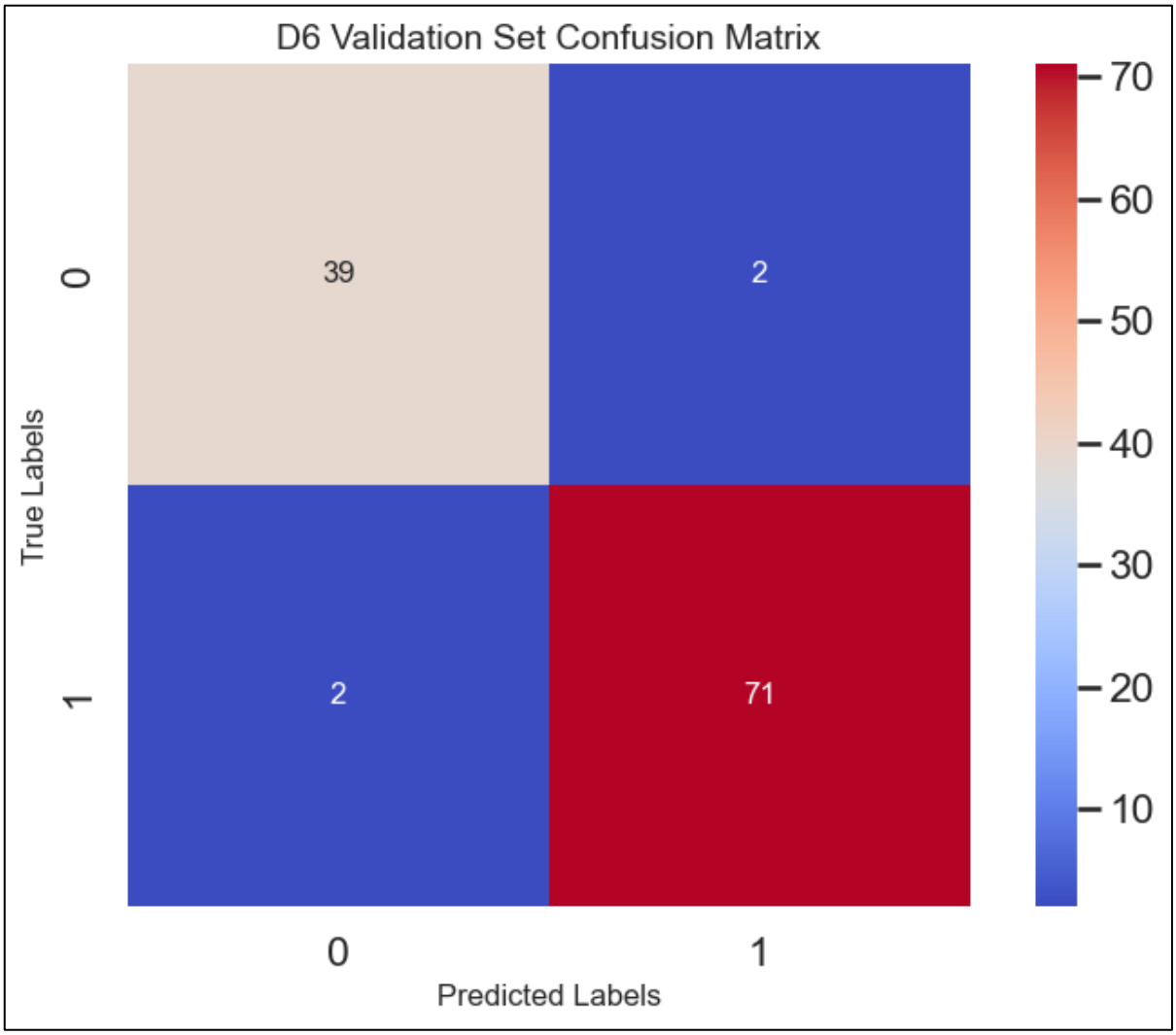
D6 [3 marks]:

Fit a decision tree classifier using default hyperparameters using 80% of the data. Remember to set the random generator's state to the value "5508" (for both the split and class). Use the trained classifier to perform predictions on the training and test sets. Provide the accuracy, precision and recall scores for both sets and the confusion matrix for the test set.

D6 Answer:

D6		
Training Results:		
Accuracy:	Precision:	Recall:
1.0	1.0	1.0
Testing Results:		
Accuracy:	Precision:	Recall:
0.96	0.96	0.96

Figure 3: Shows the confusion matrix for the test set.



D7 [2 marks]:

Comment on these results. Do you think your classifier is overfitting? If so, why this is happening? If not, why not?

D7 Answer:

Possibly, yes. Perfect training scores indicate that the model may have learned the training data too well, including noise and outliers, which do not generalize to new data. The drop in testing metrics supports this, though it's a small drop, not an extreme one.

Overfitting in a decision tree classifier can occur for several reasons:

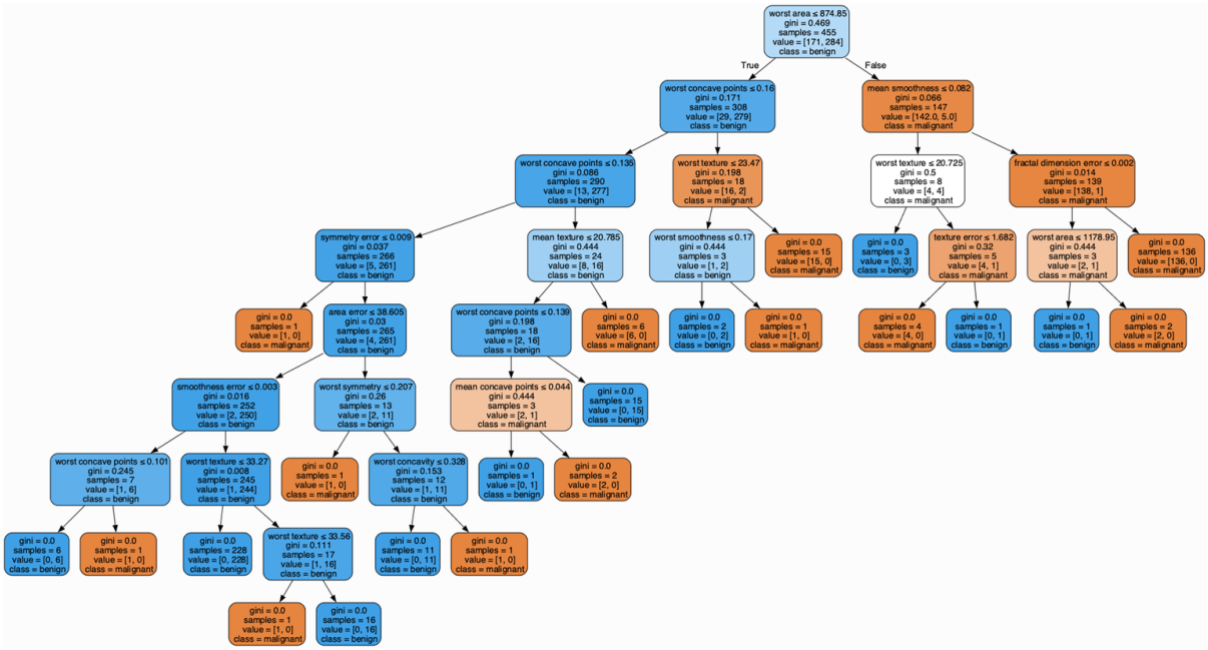
- Minimum split size: If the tree is allowed to split until the nodes contain very few samples, it can lead to rules that are too specific to the training data. Nodes that represent only a few samples are likely representing noise or outliers.
- Irrelevant features: including features that do not have predictive power can lead the model to create splits based on noise, which again leads to overfitting.

D8 [2 marks]:

Display the decision tree built from the training process (like the one shown in Figure 6.1 of the textbook for the iris dataset).

D8 Answer:

Figure 4: Shows the decision tree built from the training process similar the Figure 6.1 of the textbook for the iris dataset.



D9 [2 marks]:

Study the tree diagram and comment on the following:

1. How many levels resulted from the model?
2. Did the diagram help you to confirm whether the classifier has an overfitting issue?
3. What can you observe from the leaves?
4. Is this an interpretable model?

D9 Answer:

1. There are a total of 9 levels in this decision tree model.
2. Overfitting is suggested by trees that have many levels, which can mean that the tree is potentially fitting to the noise in the training data. The tree generated have numerous splits, which might indicate overfitting, especially if the number of samples in the leaves is very low. This however does not confirm the overfitting issue as it is best to also view the performance metrics to further verify this, however this is a strong indication to the model being overfitted.
3. The leaves have very few samples in them, some as low as 1. This typically suggest that the tree has learned a very specific pattern that might not generalize well. The gini index in the leaves as well should ideally be 0 or close to 0, which indicate pure nodes. However there are some leaves with a much higher value which may indicate that those nodes are not entirely pure.
4. Decision trees are generally considered to be interpretable models because they make decisions based on clear rules and thresholds, which can be easily followed from the root to a leaf. However, in this context the model is less interpretable because of its complexity as the user would find it challenging to track a path from root to leaf without substantial effort.

D10 [3 marks]:

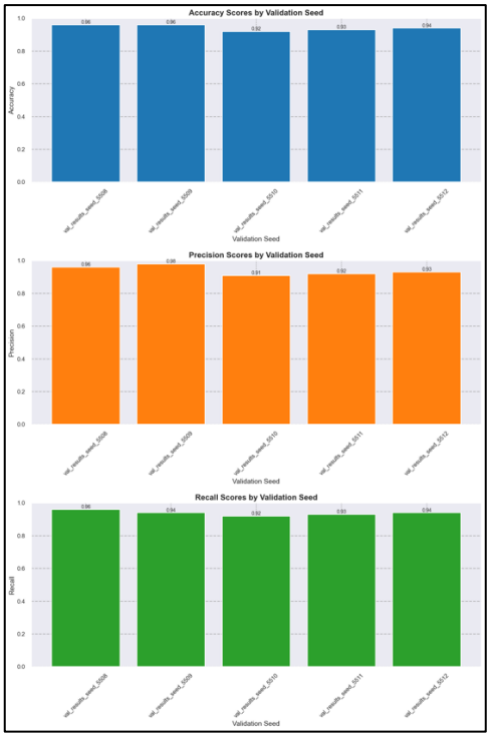
Repeat the data split another four times, each using 80% of the data to train the model and the remaining 20% for testing. For these splits, set the seed of the random state to the values “5509”, “5510”, “5511” and “5512”. The random state of the model can be kept at “5508”.

For each of these four splits, fit the decision tree classifier and use the trained classifier to perform predictions on the test set. Provide three plots to show the accuracy, precision and recall scores for the test set for each split and comment on the consistency of the results in the five splits (including the original split with random state “5508”).

D10 Answer:

D10			
	Accuracy:	Precision:	Recall:
Validation Results Seed 5508:	0.96	0.96	0.96
Validation Results Seed 5509:	0.96	0.98	0.94
Validation Results Seed 5510:	0.92	0.91	0.92
Validation Results Seed 5511:	0.93	0.92	0.93
Validation Results Seed 5512:	0.94	0.93	0.94

Figure 5: Shows the bar plots for Accuracy, Precision and recall respectively across different seeds.



Consistency Across Metrics:
Accuracy: The accuracy scores show very high consistency, with values ranging narrowly from *0.92* to *0.96* across different seeds. This demonstrates that the model consistently classifies both classes (malignant and benign) accurately across various splits.
The highest accuracy occurs at seeds 5508 and 5509 with *0.96*, indicating optimal model performance with these data configurations.

Precision: Precision scores are also consistent, though they exhibit a slightly wider range from *0.91* at seed 5510 to *0.98* at seed 5509. This indicates that the model's ability to avoid false positives is mostly stable, though some variability can be observed, suggesting slight fluctuations in performance under different data distributions.
The peak precision at seed 5509 (*0.98*) shows the model's best performance in identifying true positives accurately.

Recall: Recall scores, which measure the model's ability to identify all relevant cases, show some variation, ranging from *0.92* to *0.96*. This suggests that the model's sensitivity to detecting positives can fluctuate more compared to accuracy and precision.
The lowest recall occurs at seed 5510 (*0.92*) and the highest at seed 5508 (*0.96*), pointing to differences in how effectively the model identifies all positive cases under different splits.

The model demonstrates robustness and reliability in terms of accuracy and precision across different validation seeds, indicating consistent overall performance. However, there is some variability in recall, suggesting that the model's effectiveness in identifying all positive cases can vary slightly depending on the specific data split. This variability in recall might indicate that while the model is generally reliable, its performance can still be affected by the particularities of the data it is trained on in each split.

D11 [3 marks]:

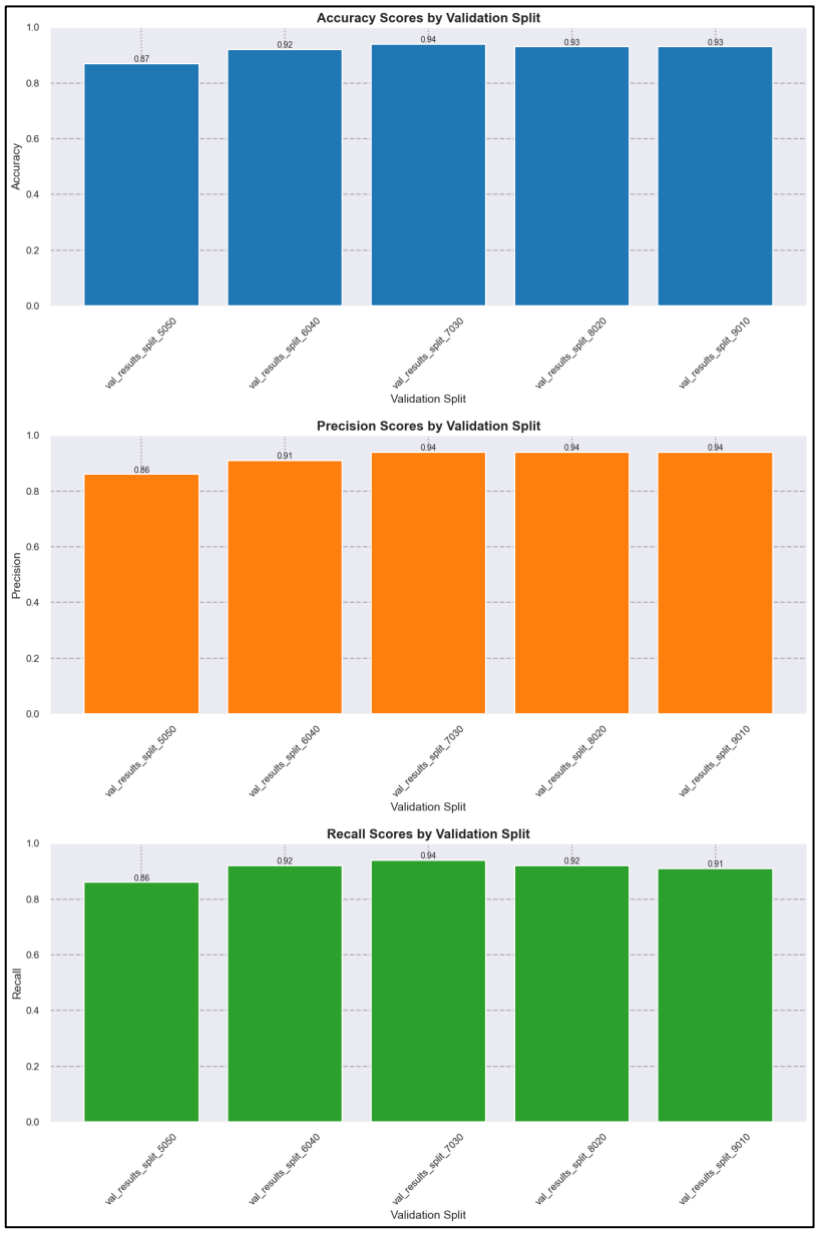
Investigate the impact of the training size on the performance of the model. You will do five different splits: 50%-50% (training the model on 50% of the data and testing on the remaining 50%), 60%-40%, 70%-30%, 80%-20% and 90%-10%. For each of these data splits, set back the seed of the random state to the value "5508".

Provide three plots to show the accuracy, precision and recall scores for the test set for each data split and comment on the results. Did the performance behave as you expected?

D11 Answer:

D11			
	Accuracy:	Precision:	Recall:
Validation Result Split 50%/50%	0.87	0.86	0.86
Validation Result Split 60%/40%	0.92	0.91	0.92
Validation Result Split 70%/30%	0.94	0.94	0.94
Validation Result Split 80%/20%	0.93	0.94	0.92
Validation Result Split 90%/10%	0.93	0.94	0.91

Figure 6: Shows the bar plots for Accuracy, Precision and Recall respectively across different splits.



Accuracy:

- The highest accuracy is observed in the 70%-30% split at *0.94*, suggesting that this split provides an optimal balance between training and testing, allowing the model to learn well while still having a sufficient test set to validate performance.
- Accuracy remains high at *0.93* for both the 80%-20% and 90%-10% splits, indicating that the model continues to perform robustly even as the training data increases and the test set becomes smaller.
- The accuracy decreases to *0.87* at the 50%-50% split, highlighting potential underfitting when the training data is not sufficiently large.

Precision:

- Precision peaks at *0.94* in the 70%-30% and 80%-20% splits, showing high effectiveness in identifying true positives as the amount of training data increases.
- There is a slight decrease in precision to *0.86* at the 50%-50% split and to *0.94* at the 90%-10% split, suggesting some issues with model generalization when the test set is either too large or too small.

Recall:

- Recall is also highest at *0.94* in the 70%-30% split, indicating that the model is very effective in detecting all relevant instances when trained with 70% of the data.
- The recall slightly decreases as the test set size decreases, with scores of *0.92* in the 80%-20% split and *0.91* in the 90%-10% split, possibly indicating some loss of sensitivity with a smaller test set.
- At the 50%-50% split, the recall is lower at *0.86*, further supporting the notion that an equal split does not allow the model to learn as effectively.

The performance of the decision tree classifier generally improves as the proportion of training data increases up to 70%, after which the benefits plateau slightly but remain robust. The 70%-30% split provides the best overall performance across accuracy, precision, and recall, suggesting it is the optimal training-test ratio for this dataset. The declines in performance metrics at the 50%-50% split indicate that a larger training set is necessary for this model to adequately learn and generalize from the data.

Fitting a Decision Tree model with optimal hyperparameters

D12 [4 marks]:

Create a training set using 80% of the data and a test set with the remaining 20%. Use a *10-fold cross-validation* and *grid-search* to find the optimal combination of hyperparameters *max_depth* — using values [2, 3, 4, 5], *min_samples_split* — using values [2, 4, 5, 10], and *min_samples_leaf* using values [2,5] of a decision tree model. Remember to set the seed of the random state of the data split function and model class to the value "5508". For the cross-validation, set the value of the random state to "42". Use accuracy for the scoring argument of the grid-search function.

With the optimal obtained hyperparameters, retrain the model and report:

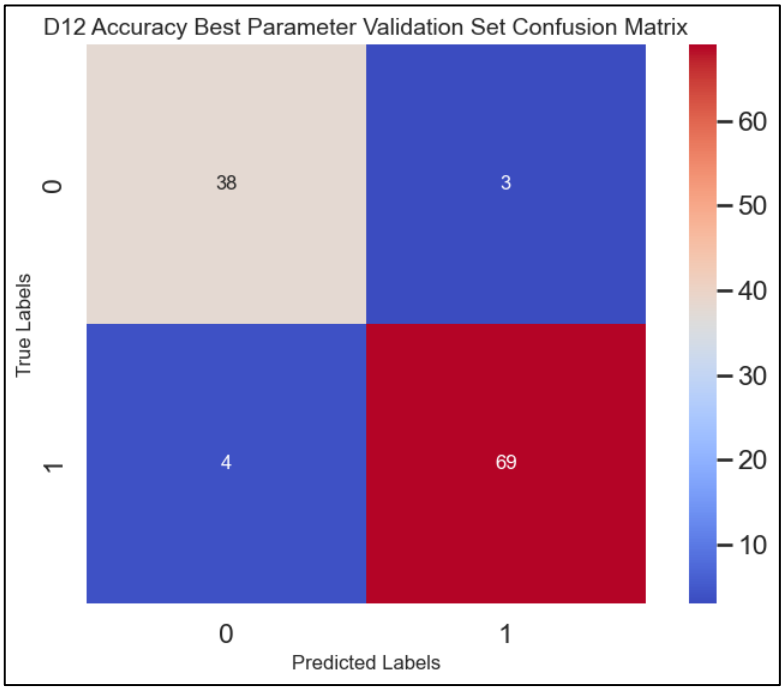
- The optimal hyperparameters;
- The obtained accuracy, precision and recall on the training set;
- The obtained accuracy, precision and recall on the test set;
- The confusion matrix on the test set.

D12 Answer:

```
Fitting 10 folds for each of 32 candidates, totalling 320 fits
Best Parameters: {'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 2}
Best Accuracy Score: 0.94
```

D12			
	Accuracy	Precision	Recall
Training Results	0.96	0.97	0.96
Validation Results	0.94	0.93	0.94

Figure 7: Shows the confusion matrix of the test set with the best parameters.



D13 [2 marks]:

Comment: What was the impact of fine-tuning the hyperparameters as opposed to what you obtained in D6? Has fine-tuning done what you expected?

D13 Answer:

Yes, fine-tuning has performed as expected. By adjusting the *max_depth*, *min_samples_leaf*, and *min_samples_split*, the model was likely prevented from creating overly complex decision rules and thus improved its ability to generalize.

Improved Generalization:

- *Accuracy*: The accuracy of D6 compared to D12 shows an increase after fine tuning the parameters.
- *Precision*: like accuracy it improved as well, indicating that the model is more accurate in predicting positive cases.
- *Recall*: improvement of the recall like precision and accuracy means the model is better at identifying all relevant instances of the positive class.

The confusion matrix from D6 showed 2 false positives and 4 false negatives. The confusion matrix after fine-tuning shows fewer misclassifications, with only 3 false positives and 4 false negatives.

Overall, the result of fine-tuning the hyperparameters has effectively reduce overfitting and improve the model's performance.

D14 [3 marks]:

Repeat the training of task D12 twice: one considering the scoring argument of the grid-search function as precision and the other recall.

For each of the scoring options (*accuracy, precision, recall*), provide the optimal hyperparameters according to the *10-fold cross-validation* and *grid-search*, and, after retraining each model accordingly, provide the *confusion matrix* on the test set. Comment on the results, considering the problem.

D14 Answer:

```
Fitting 10 folds for each of 32 candidates, totalling 320 fits
Best Parameters: {'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 2}
Best Precision Score: 0.96
```

```
Fitting 10 folds for each of 32 candidates, totalling 320 fits
Best Parameters: {'max_depth': 3, 'min_samples_leaf': 5, 'min_samples_split': 2}
Best Recall Score: 0.97
```

D14			
	Max Depth:	Min Samples Leaf:	Min Samples Split:
Accuracy:	3	2	2
Precision:	4	2	2
Recall:	3	5	2

Figure 8: Shows the confusion matrix on the validation set of the Precision Score model.

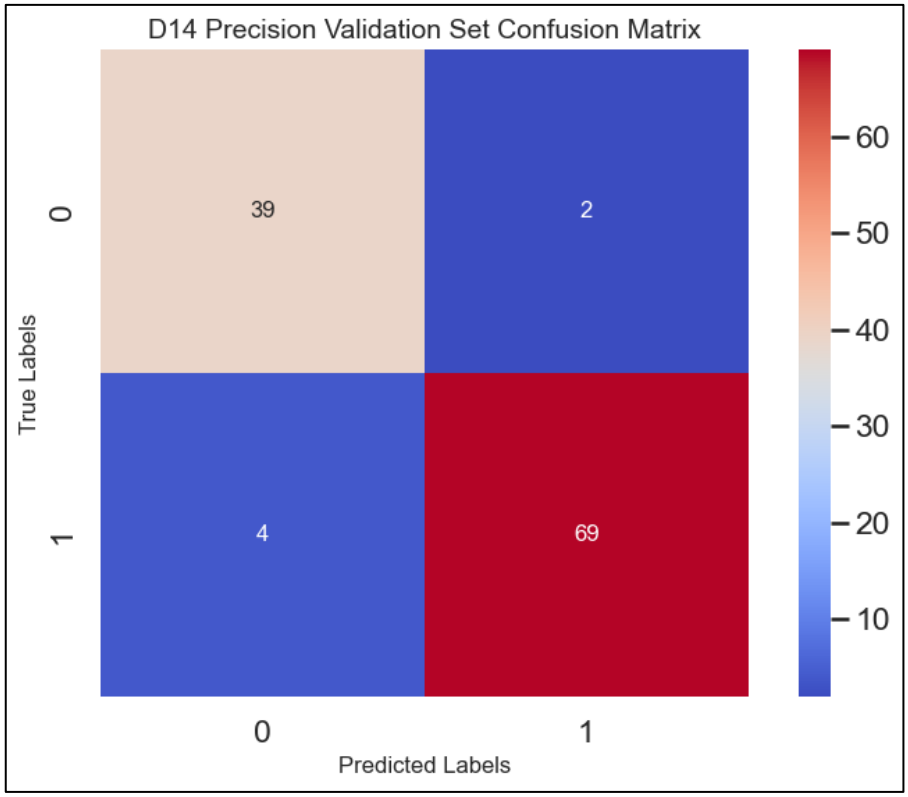
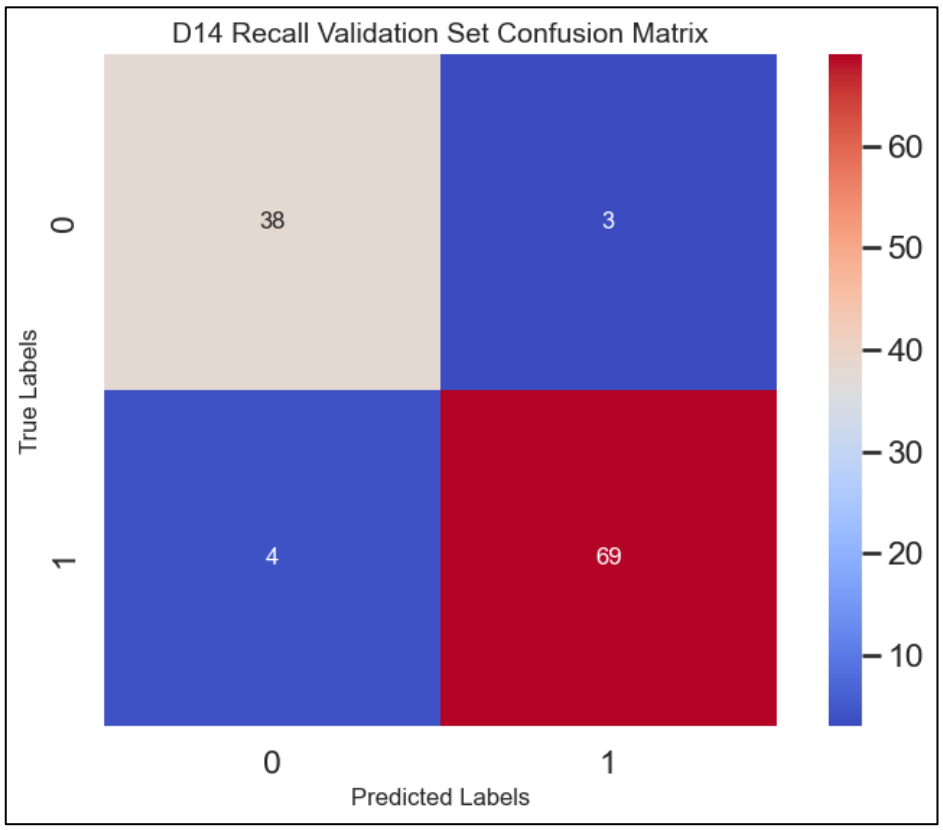


Figure 9: Shows the confusion matrix on the validation set of the Recall Score model.



Accuracy: When optimizing with accuracy, the model seeks to maximise the number of correct predictions over the total number of predictions. The accuracy-optimised model resulted in a more conservative tree with a *max_depth* of 2, which might help prevent overfitting.

Precision: When optimizing with precision, the model focuses on minimising false positives. This model has a greater *max_depth* of 5, suggesting that it tries to be more specific in its predictions, which is evident in its higher precision score but can also lead to overfitting as indicated by a more complex tree.

Recall: When optimizing to maximise recall, the model is more concerned with minimising false negatives. The recall-optimised model has a *max_depth* of 3 and a higher *min_samples_leaf* of 5, which can help in generalising the model and reducing the risk of overfitting, although not as conservatively as the accuracy-optimised model.

As for the confusion matrix, The precision-optimized model has the fewest false positives (only 2) at the expense of slightly more false negatives (5). The recall-optimized model has the fewest false negatives (4), which is expected since it's optimizing for recall, but has one additional false positive compared to the accuracy-optimized model.

Given that the problem in this assignment is classification of breast cancer, the importance to capture as many positive instances as possible would be the focus and thus optimizing for recall would be preferable.

Fitting a Decision Tree with optimal hyperparameters and a reduced feature set

D15 [1 mark]:

Using the model with fine-tuned hyperparameters based on accuracy (the one you obtained in D12), display the *feature importance* for each feature obtained from the training process. You should sort the feature importance's in descending order.

D15 Answer:

Figure 10: Shows the feature importance for each feature obtained from the training process in descending order.

	Feature	Importance
0	worst area	0.80
1	worst concave points	0.15
2	mean smoothness	0.02
3	mean texture	0.01
4	worst texture	0.01
5	mean area	0.00
6	mean compactness	0.00
7	worst symmetry	0.00
8	worst smoothness	0.00
9	worst fractal dimension	0.00
10	worst concavity	0.00
11	concave points error	0.00
12	worst compactness	0.00
13	concavity error	0.00
14	texture error	0.00
15	symmetry error	0.00
16	smoothness error	0.00
17	perimeter error	0.00
18	compactness error	0.00
19	mean symmetry	0.00
20	fractal dimension error	0.00
21	mean fractal dimension	0.00
22	mean concavity	0.00
23	mean concave points	0.00
24	area error	0.00

D16 [3 marks]:

Using the feature importance you calculated in the previous task, trim the feature dimension of the data. That is, you should retain only those features whose *importance* values are above 1% (i.e., 0.01). You can either write your own Python code or use the function SelectFromModel from the sklearn.feature selection package to work out which feature(s) can be removed.

Report what features were retained and removed in the above process. Also report the total feature importance value that is retained after your dimension reduction step.

D16 Answer:

Features retained: [*'mean smoothness', 'mean texture', 'worst area', 'worst concave points', 'worst texture'*]

Features removed: [*'area error', 'compactness error', 'concave points error', 'concavity error', 'fractal dimension error', 'mean area', 'mean compactness', 'mean concave points', 'mean concavity', 'mean fractal dimension', 'mean symmetry', 'perimeter error', 'smoothness error', 'symmetry error', 'texture error', 'worst compactness', 'worst concavity', 'worst fractal dimension', 'worst smoothness', 'worst symmetry'*]

Total feature importance retained: 1.0

D17 [3 marks]:

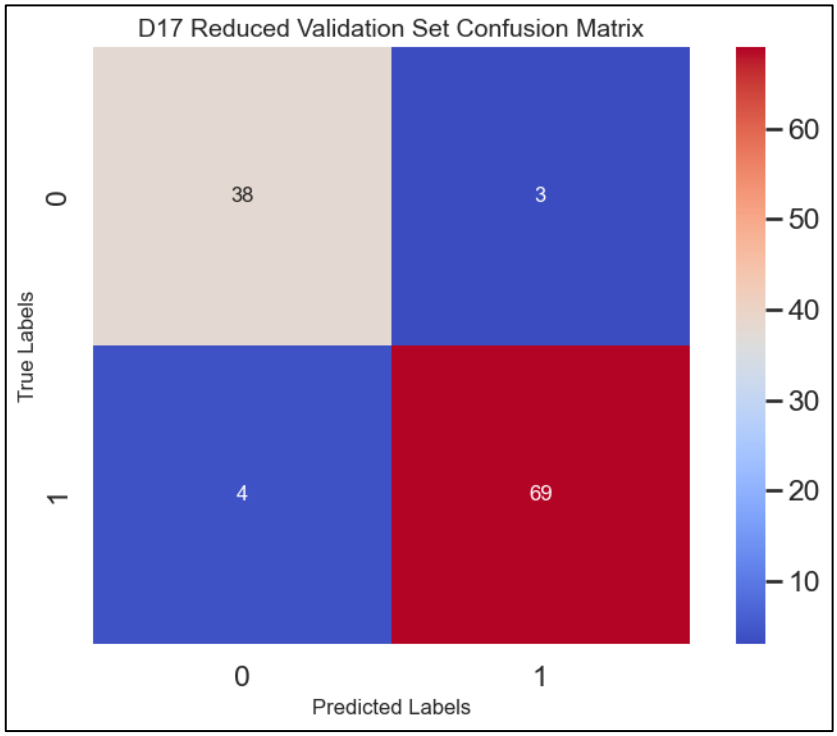
Compare the model's performance (accuracy, precision, recall) on training and test sets when using the reduced set of features and the model trained on the complete set of features. Also, report the corresponding confusion matrices on the test sets. (You will need to consider whether you should repeat the cross-validation process to find the optimal hyperparameters).

D17 Answer:

Note: Initial grid search for D17 has given us a different best parameter fields and values, we will stick to the model created in D12 as our task is to Compare the model's performance (accuracy, precision, recall) on training and test sets when using the reduced set of features and the model trained on the complete set of features. Hence the model should remain the same and the features will be the difference to compare.

D17			
	Accuracy:	Precision:	Recall:
Training Results:	0.96	0.97	0.96
Validation Results:	0.94	0.93	0.94
D12			
	Accuracy:	Precision:	Recall:
Training Results:	0.96	0.97	0.96
Validation Results:	0.94	0.93	0.94

Figure 11: Shows the confusion matrix for the reduced test set in D17



Based on initial comparisons of the results, we can see that both D17 and D12 results are identical.

D18 [1 mark]:

Comment on your results. What was the impact (if any) of reducing the number of features?

D18 Answer:

The D17 and D12 results, trained on the same model with reduced and complete set of features respectively, display equivalent performance across most metrics on both training and validation datasets, contradicting initial expectations that reducing the feature set might lead to a discernible difference in model efficacy.

Accuracy, Precision, Recall (Training and Validation): Both models achieved identical scores on training results with an accuracy of *0.96*, precision of *0.97*, and recall of *0.96*. The validation results for both models also matched perfectly with an accuracy of *0.94*, precision of *0.93*, and recall of *0.94*.

This uniformity suggests that the removal of features in the D17 model did not significantly impact its ability to generalize compared to the D12 model, which utilized the complete set of features.

The confusion matrices as well are identical, indicating no difference in the classification errors made by both models. Both models have effectively the same true and false positives/negatives, highlighting their similar predictive abilities on the validation set.

Given these results, it may not be necessary to repeat the cross-validation process to find optimal hyperparameters specifically for the reduced feature set, as the existing model configuration seems to maintain performance effectively. This equivalence also implies that the complexity reduction in the D17 model (through feature reduction) does not compromise the model's ability to generalize, making it a potentially more efficient choice without sacrificing accuracy.

Fitting a Random Forest

D19 [3 marks]:

Considering all features and the 80%-20% data split you did before, use 10-fold cross-validation and grid-search to find a Random Forest classifier's optimal hyperparameters n estimators (number of estimators) and max depth. Remember to set the seed of the random state of the data split function and model class to the value "5508". Use n estimators:[10, 20, 50, 100, 1000], and max depth:[2, 3, 4, 5]. For the cross-validation, set the value of the random state to "42". Use accuracy for the scoring argument of the grid-search function.

Keep the other hyperparameter values to their default values. Use the optimal values for the n estimators and max depth hyperparameters to retrain the model and report:

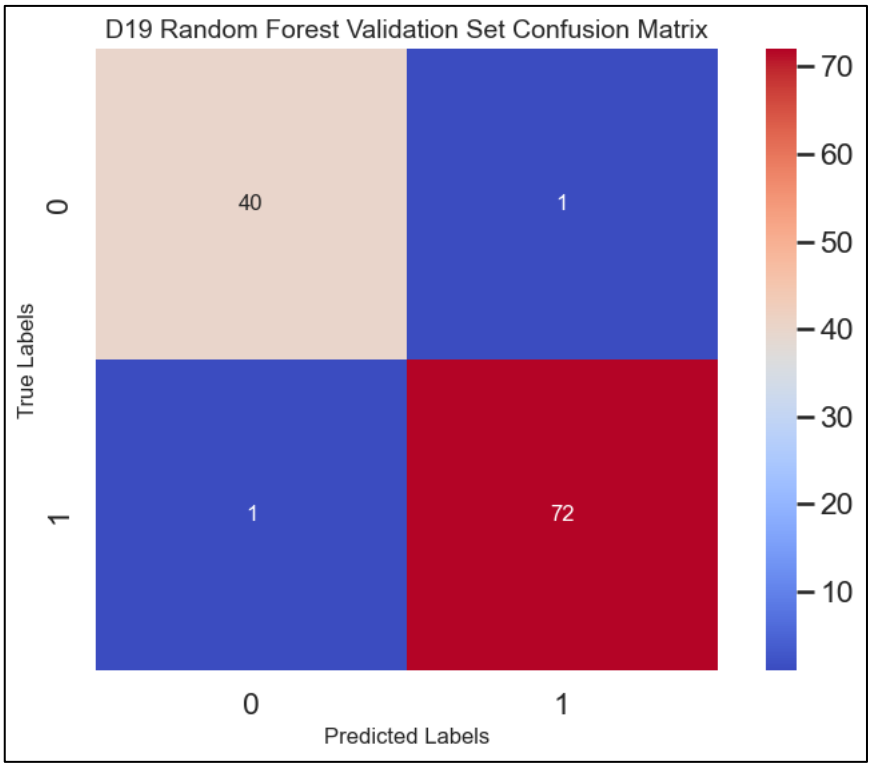
- The obtained optimal number of estimators and max depth;
- The obtained accuracy, precision and recall on the training set;
- The obtained accuracy, precision and recall on the test set;
- The confusion matrix on the test set.

D19 Answer:

```
Fitting 10 folds for each of 20 candidates, totalling 200 fits
Best Parameters: {'max_depth': 4, 'n_estimators': 100}
Best Accuracy Score: 0.96
```

D19				
	Best Parameters:	Accuracy:	Precision:	Recall:
Hyper Parameters:	{'max_depth': 4, 'n_estimators': 100}	NaN	NaN	NaN
Training Results:	NaN	0.99	0.99	0.99
Validation Results:	NaN	0.98	0.98	0.98

Figure 12: Shows the confusion matrix for the validation set of the random forest model.



D20 [2 marks]:

How do these performances compare with the ones you obtained in D12? What changed with the use of a Random Forest model? Is this result what you would expect?

D20 Answer:

The use of the Random Forest model, with its ensemble learning approach, appears to have led to a more robust and accurate model than the single Decision Tree. This is evident in both the improved accuracy score and the confusion matrix. The Random Forest model's ability to reduce both type I and type II errors suggests a model that is more reliable for making predictions on this dataset. The results align with common expectations where ensemble methods outperform single estimators.

D21 [2 marks]:

Thinking about the application and the different models you created, discuss:

1. Do you think these models are good enough and can be trusted to be used for real?
2. Do you think a more complex model is necessary?
3. Do you think using a machine learning algorithm for this task is a good idea? That is, should this decision process be automated? Justify.
4. Are there considerations with the used dataset?

D21 Answer:

1. *Model Trustworthiness and Suitability for Real-World Use:*
    - The severity and implications of potential misdiagnoses (false positives and false negatives) must be considered. In medical applications like breast cancer diagnosis, high recall (sensitivity) might be prioritized to ensure that as few cases of cancer as possible are missed. If the models demonstrate sufficiently high and consistent recall across various configurations, they could be considered reliable enough for practical use, possibly as a preliminary screening tool rather than a definitive diagnostic method.
2. *Necessity of More Complex Models:*
    - The decision to employ more complex models should be based on the specific requirements of accuracy and the nature of the dataset. Complex models, such as deep learning models, may capture intricate patterns in the data that simpler models might miss. However, they also require more data to train effectively and can be prone to overfitting. Given the current performance of the existing models (if they are already high), adding complexity might not yield significant improvements and could complicate interpretation and maintenance.
3. *Appropriateness of Automating the Decision Process:*
    - Automating breast cancer diagnosis through machine learning can increase the efficiency and accessibility of screening and preliminary assessments. It can help in handling large volumes of data and assist in early detection where radiologist availability is limited.

- However, the decision to fully automate should be approached with caution. The machine learning model should ideally operate in conjunction with human experts, serving as a support tool to augment the diagnostic process rather than replacing human judgment. This dual approach helps mitigate the risks associated with potential errors in the model's predictions.

4. *Considerations with the Used Dataset:*
    - The quality and diversity of the dataset are crucial. The scikit-learn breast cancer dataset is relatively clean and well-prepared, but it may not represent the broader population diversity seen in real-world settings. Factors such as age, race, and other health conditions might influence the model's performance and generalizability.
    - The dataset's size and the balance between classes (benign vs. malignant) are also important. If the dataset is imbalanced, it could bias the model, resulting in poorer performance on the minority class.