COMP7703

Assignment Report

Georgia Wyldbore
43235779

Due 2pm Friday 24th May 2024

Word count: 4393

Table of Contents

1 Introduction

1.1 Overview and objectives

This report aims to identify and solve a classification problem on the datasets provided from Loesche, Bundgaard and Barker's (2000) paper, which investigated variation in the body and wing sizes of two species of fruit fly in response to temperature, as a proxy for measuring the variation of size at different latitudes. Their overall finding was that temperature has a significant impact on the size of an individual fruit fly, with those flies in hotter climates being smaller in size.
The classification problem selected was to predict the temperature a fly developed at, given relevant body measurements and data. This was decided upon after exploratory analysis was performed on the data, as will be explained in section 2.2 of this report. While intuitively, predicting temperature seems as though it would be better suited to regression models rather than classification, the temperature labels from the original data fall into three discrete categories. As such, classification felt more appropriate for this situation. Python was used to create all of the models and plots.

Ultimately, two main objectives were identified for this report:

- Objective 1: Given the study showed temperature impacts size, reverse this and determine whether temperature can be accurately predicted from the measurements of a fruit fly.
- Objective 2: Determine the impact varying the number of features and observations has on the performance of model prediction.

1.2 Understanding the Data

Three datasets were provided for use in this report. These were .csv files titled Thorax_&_wing_traits, Wing_traits_&_asymmetry, and Wing_asymmetry, along with the identifier of the study. For simplicity, these files will be referred to as 'traits', 'trait asymmetry' and 'asymmetry' respectively. While it is not necessary to understand what each of the variables included in the data represents biologically, it is useful to have some understanding of the data – especially where the feature name may not

be immediately descriptive. This allows removal of features which may not provide any benefit to the predictive models, reducing dimensionality and computational time, and allowing validation that data is of the type expected.

Each of the three files included the variables Species, Population, Latitude, Longitude, Year_start, Year_end, Temperature, Vial, Replicate, and Sex. *Traits* then included data on measurements, including Thorax_length, l2, l3p, l3d, lpd, l3, w1, w2, w3, wing_loading - an explanation of these measurements can be seen in Table 1.

The *trait asymmetry* file included measurements of wing area, shape, and vein length ratios, and asymmetry data on those three traits, and the *asymmetry* file included the asymmetry data for the wing-related distances from the thorax and wing traits file. Asymmetry data was measured as the variance between the measurements from the two wings of an individual fly. All of the data were in relation to experimental, lab-reared populations, not the individuals caught in the wild.

Feature Name	**Explanation –** see Figure 1 for points
Thorax_length	Length from anterior margin of the thorax to the posterior tip of the scutellum
l2	Distance between points 1 and 3
l3p	Distance between points 1 and 4
l3d	Distance between points 4 and 5
lpd	Not clear in study – from peer discussion, appears to be l3p + l3d
l3	Distance between points 1 and 5
w1	Distance between midpoint of points 2 and 3, and point 6
w2	Distance between points 2 and 6
w3	Distance between points 3 and 6
wing_loading	ratio of l3 / Thorax_length
Wing_area	(w1 * l3) / 2
Wing_shape	l3 / w1
Wing_vein	l3d / l3p

Table 1. Explanation of wing trait measurements

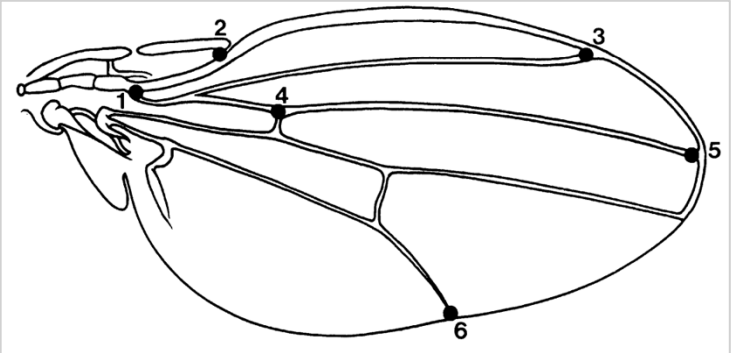


Figure 1. Diagram taken from Loesche, Bundgaard and Barker (2000) showing six points on the wing relevant to the measurements recorded in the data.

2 Data Preprocessing

2.1 Quality checks and initial transformations

Each of the three datasets was loaded into a pandas dataframe, and the basic information was printed as a starting point. Note: larger versions of all figures can be found in the Appendix, as can the code used to process the data.

Wing Traits

```
RangeIndex: 1731 entries, 0 to 1730
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Species         1731 non-null   object
 1   Population      1731 non-null   object
 2   Latitude        1731 non-null   float64
 3   Longitude       1731 non-null   float64
 4   Year_start      1731 non-null   int64
 5   Year_end        1731 non-null   int64
 6   Temperature     1731 non-null   int64
 7   Vial            1731 non-null   int64
 8   Replicate       1731 non-null   int64
 9   Sex             1731 non-null   object
 10  Thorax_length   1731 non-null   object
 11  l2              1731 non-null   float64
 12  l3p             1731 non-null   float64
 13  l3d             1731 non-null   float64
 14  lpd             1731 non-null   float64
 15  l3              1731 non-null   float64
 16  w1              1731 non-null   float64
 17  w2              1731 non-null   float64
 18  w3              1731 non-null   float64
 19  wing_loading    1731 non-null   object
```

Trait Asymmetry

```
RangeIndex: 1731 entries, 0 to 1730
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Species               1731 non-null   object
 1   Population            1731 non-null   object
 2   Latitude              1731 non-null   float64
 3   Longitude             1731 non-null   float64
 4   Year_start            1731 non-null   int64
 5   Year_end              1731 non-null   int64
 6   Temperature           1731 non-null   int64
 7   Vial                  1731 non-null   int64
 8   Replicate             1731 non-null   int64
 9   Sex                   1731 non-null   object
 10  Wing_area             1730 non-null   float64
 11  Wing_shape            1712 non-null   float64
 12  Wing_vein             1725 non-null   float64
 13  Asymmetry_wing_area   1705 non-null   float64
 14  Asymmetry_wing_shape  1705 non-null   float64
 15  Asymmetry_wing_vein   1717 non-null   float64
```

Asymmetry

```
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Species         1727 non-null   object
 1   Population      1727 non-null   object
 2   Latitude        1727 non-null   float64
 3   Longitude       1727 non-null   float64
 4   Year_start      1727 non-null   int64
 5   Year_end        1727 non-null   int64
 6   Temperature     1727 non-null   int64
 7   Vial            1727 non-null   int64
 8   Replicate       1727 non-null   int64
 9   Sex             1727 non-null   object
 10  Asymmetry_l2    1721 non-null   float64
 11  Asymmetry_l3p   1726 non-null   float64
 12  Asymmetry_l3d   1718 non-null   float64
 13  Asymmetry_lpd   1717 non-null   float64
 14  Asymmetry_l3    1717 non-null   float64
 15  Asymmetry_w1    1712 non-null   float64
 16  Asymmetry_w2    1715 non-null   float64
 17  Asymmetry_w3    1713 non-null   float64
```

Table 2. Basic information on each dataset, including feature name, number of non-null entries, and data type.

It was noted that there were the same number of observations in *traits* and *trait asymmetry,* with a similar number in *asymmetry*, but that any feature which included asymmetry data tended to have fewer non-null values. Given the context of the data is measurements of individuals in an experiment, along with the similar number of

observations, the question was posed whether the datasets could be combined, and the assumption was made that the three files do refer to the same individuals.

Trait asymmetry identified species in a different format to the other two files, by omitting an underscore in the name. Once this was altered, the three files contained the same sets of unique values in each of the first ten features, i.e. Species through Sex.

The combination of Species, Population, Temperature, Vial, Replicate and Sex provided a unique key for any given individual. This unique combination was used to create a new variable – 'Fly_ID' – which was assigned to each observation in all three sets. This was then used to combine the three datasets into one large dataset, containing 35 features and 1731 observations.

```
RangeIndex: 1731 entries, 0 to 1730
Data columns (total 35 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Fly_ID                1731 non-null   int64
 1   Species               1731 non-null   object
 2   Population            1731 non-null   object
 3   Latitude              1731 non-null   float64
 4   Longitude             1731 non-null   float64
 5   Year_start            1731 non-null   int64
 6   Year_end              1731 non-null   int64
 7   Temperature           1731 non-null   int64
 8   Vial                  1731 non-null   int64
 9   Replicate             1731 non-null   int64
 10  Sex                   1731 non-null   object
 11  Thorax_length         1731 non-null   object
 12  l2                    1731 non-null   float64
 13  l3p                   1731 non-null   float64
 14  l3d                   1731 non-null   float64
 15  lpd                   1731 non-null   float64
 16  l3                    1731 non-null   float64
 17  w1                    1731 non-null   float64
 18  w2                    1731 non-null   float64
 19  w3                    1731 non-null   float64
 20  wing_loading          1731 non-null   object
 21  Wing_area             1730 non-null   float64
 22  Wing_shape            1712 non-null   float64
 23  Wing_vein             1725 non-null   float64
 24  Asymmetry_l2          1721 non-null   float64
 25  Asymmetry_l3p         1726 non-null   float64
 26  Asymmetry_l3d         1718 non-null   float64
 27  Asymmetry_lpd         1717 non-null   float64
 28  Asymmetry_l3          1717 non-null   float64
 29  Asymmetry_w1          1712 non-null   float64
 30  Asymmetry_w2          1715 non-null   float64
 31  Asymmetry_w3          1713 non-null   float64
 32  Asymmetry_wing_area   1705 non-null   float64
 33  Asymmetry_wing_shape  1705 non-null   float64
 34  Asymmetry_wing_vein   1717 non-null   float64
dtypes: float64(24), int64(6), object(5)
```

Figure 2. Information on the newly combined dataset.

An abnormality was noticed in the Thorax_length and wing_loading features; both of these were expected to be numerical values, but were being interpreted as objects. Some investigation discovered that some entries for these features contained only '.' rather than nulls or numerical data. These were replaced with 'NaN' and the features were then converted to floats.

Additionally, investigating the summary statistics of each feature revealed that several values contained zeroes, so the assumption was made that this represented

null data, rather than an individual having a wing measurement of 0. These were also replaced with ‘NaN’. The summary statistics also made clear that Year_start and Year_end both contained only one value throughout.

Several of the features, while useful in combining the data, are not useful to analysis. Latitude and Longitude were only used to identify the location of the Population, so can be removed. Year_start and Year_end, as mentioned, contain only one value throughout, so provide no insight. Vial and Replicate are experimental identifiers only – while they would be useful if a particular vial or replicate had issues experimentally, affecting the data, the assumption is made here that this is not the case, and so these are not useful. Finally, Fly_ID was artificially manufactured, so provides nothing useful for analysis.

Removing these leaves the data with 28 features for analysis. Sex, Population and Species were then numerically encoded to allow them to be used in analysis.

A heatmap was created to help identify a potential aim for the report, and can be seen in Figure 3 below.

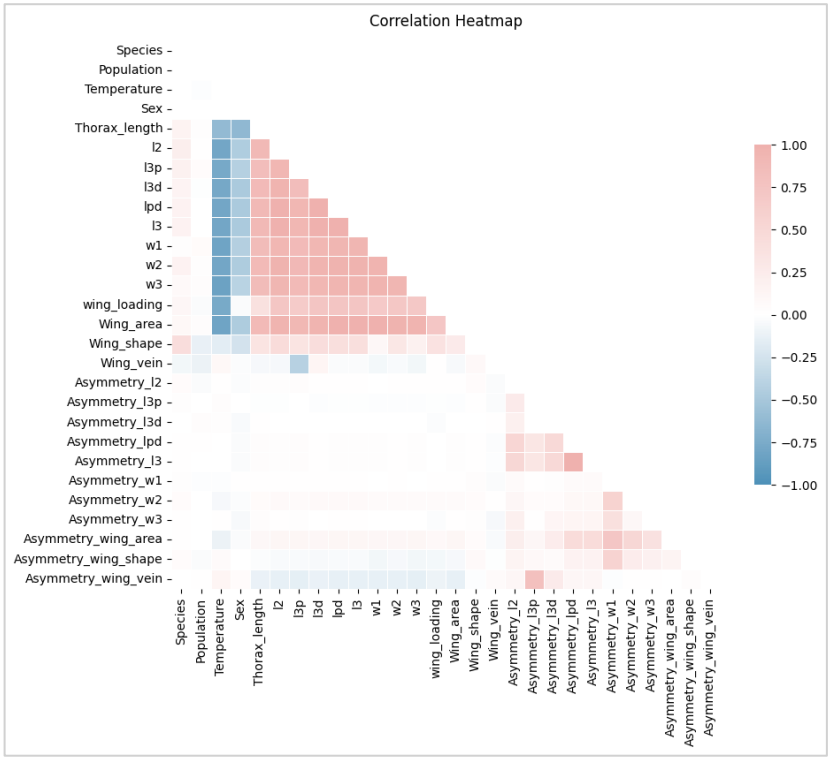


Figure 3. Correlation heatmap for all remaining features in the combined dataset.

From this, it is clear there is a strong negative correlation between temperature and the wing trait measurements, a moderate negative correlation between sex and

those measurements, and a strong positive correlation between all of the wing trait measurements. Some of the asymmetry measurement data also correlates with itself, but very little with anything else. The strength of the association between temperature and the trait measurements makes temperature an ideal candidate for the predictions in this report. As mentioned in the introduction, it feels intuitive that this should provide a regression problem, but as temperature is measured as discrete values of 20, 25 and 30, classification feels more appropriate.

2.2 Features versus observations

At this point in the analysis, the issue was raised of how to handle null data. The majority of the null values existed in the asymmetry-related features, with between 17 and 57 null observations in each of these features.

The potential options were to remove the asymmetry features altogether, and impute the few remaining null values, or to remove all of the observations which had null values. This decision raised the question of what the impact would actually be of this decision on the performance of the models used.

Given the so-called ‘curse of dimensionality’ in machine learning, it would be expected that having a larger number of features would decrease the performance of a given model, especially in the case of distance-based models such as K-Nearest Neighbours or Decision Trees. More features also come with the risk of creating less generalisable models, as they may fit to random noise rather than underlying patterns, and of increasing computation time. It is also intuitive that if a model can see more datapoints on a given number of features, it should be able to refine further and so make more accurate predictions – notwithstanding overfitting.

The issues with dimensionality tend to occur when the number of features approaches, or is greater than, the number of observations. In this situation, there are at most 28 features, and between 1445 and 1731 observations, depending on null removal. As such, dimensionality is unlikely to have any real impact on the models here. However, removing the observations containing null values makes a loosely rounded change of approximately 20% in the volume of the data, which is a reasonably significant amount, and so could be expected to impact model performance in some way.

As a way to investigate this issue, the decision was made to split the data into three versions. One version with more features, where all the null data was removed; one with more observations, where the asymmetry features were removed and the few missing values imputed; and one third 'base' version to act as a control, with the same number of observations as the first, and features as the second. The expectation is that an increased number of features will have minimal effect, but an increased number of observations will improve performance slightly.

	Base	More Features	More Observations
No. of features	17	28	17
No. of observations	1445	1445	1731

Table 3. Number of features and observations in each of the new data subsets.

The models used are a K-Nearest Neighbours classifier, a Random Forest – which is an ensemble method of using Decision Trees, and a Support Vector Classifier. For each of these, the optimal hyperparameters for each of the data versions will be found, and the performance will be evaluated between them. These were chosen as they each have a differing degree of susceptibility to increased dimensionality – broadly speaking, KNNs decrease in performance and are at risk of overfitting with increased dimensionality, Random Forests are generally quite robust to overfitting despite dimensionality, and Support Vector Machines tend to increase performance with more data and without great influence from dimensionality, though this depends on the kernel.

After splitting into these three versions of the data, each of them was randomly shuffled – using a randomstate of 42 – and put through a train-test split of 70-30, again using a randomstate, this time of 10. These randomstate values were arbitrary, but recorded here for reproducibility.
Each of the input features of the split data was then normalised, as all three models chosen for classification perform better on normalised data. This normalisation was performed after the split to prevent information leakage, and better evaluate the performance of each model on unseen data.

3 K-Nearest Neighbours Classifier

A K-nearest neightbours (KNN) model is a non-parametric clustering model which implements one of the simplest principles in supervised learning; if the test datapoint x_a is close to the training datapoint x_i, then the prediction $y(x_a)$ should be close to y_i (Lindholm et al., 2022). KNN classifiers do this by finding the distance between the test datapoint and the k-nearest training datapoints, and using a majority vote to output a predicted value for y.

Generally, very low values of k will lead to overfitting in the model. However, making k too high will lead to the model ignoring any true patterns in the data. As such, k is a hyperparameter which needs to be optimised for the model to perform at its best. A common way of performing this optimisation is to perform cross-validation, which is how the best value of k will be chosen for each of these datasets. This involves performing repeated train-test splits on the training data itself, and allows the model to be more reasonably evaluated in terms of how it performs on unseen data, as we are effectively evaluating the model on many different test datasets by taking the average score for each split.

3.1 Model optimisation

The score used to evaluate each model created here was the accuracy, or 1 – misclassification rate. According to Lindholm et al. (2022), misclassification is the default choice for evaluating classification models, and the two are directly proportional.

A value of 5 was arbitrarily chosen for the number of cross-fold validations of each model, as this is the default setting in scikit-learn's cross validation package. The values of k were a range between 1 and 1000. While 1000 is much larger than would be expected to be a reasonable choice, it represented the majority of the size of the training data for the more observations subset, allowing visualisation of exactly what happens to the accuracy across almost all values of k possible.

As discussed in 2.2, normalised data was used to train and test this model. This is because, by virtue of being a distance-based model, having features on varied scales may lead the model to view particular features to be more or less important for the prediction only because of a greater absolute distance, rather than because of any patterns in the data. However, a pipeline had to be constructed which

performed the normalisation after the cross-fold validation had split the training data, in order to better generalise the performance to unseen data. This was done using the scikit-learn Pipeline package, and can be seen in Figure 4.

```
from sklearn.pipeline import Pipeline

for k in k_values:

    # Define the pipeline
    pipeline = Pipeline([
        ('scaler', StandardScaler()),  # Step 1: Normalize the data
        ('knn', KNeighborsClassifier(n_neighbors=k))  # Step 2: kNN classifier
    ])

    # Perform cross-validation
```

Figure 4. Code used to construct normalisation and model pipeline.

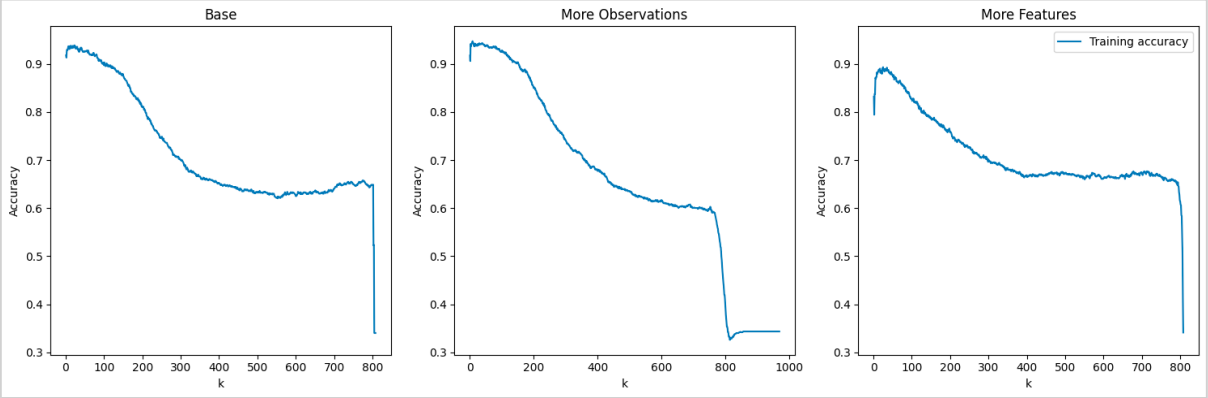


Figure 5. Plots of k against accuracy for each data subset.

	Base	More observations	More features
Max accuracy	0.9386	0.9472	0.8932
Value of k	23	9	25

Table 4. Maximum cross-fold accuracy, and corresponding value of k, for each subset.

All three models were able to achieve an average accuracy on cross-validation above 0.89, with both subsets that have less features performing at around 0.94. This slightly worse performance in the higher feature subset is expected, given than KNN models tend to struggle with higher dimensionality.

The base subset appears to increase in accuracy slightly towards the upper limit of its k values, which may suggest slight overfitting, while having more features or observations appears to allow those data subsets to avoid this overfitting. Despite this apparent decline in performance, even at values where k is approaching 800 these models are able to perform at around a 0.6 accuracy, which is almost twice that of a simple random classifier. At approximately k=800 – and above in the model where this is possible – the models each effectively do function as a random classifier, which is the expected behaviour when considering all the possible datapoints as neighbours.

3.2 Optimised performance

To make a final assessment of each model's optimal k value, each subset was run with its holdout test data. The accuracies for the cross-fold validation and the test data can be seen in Figures 6 and 7.

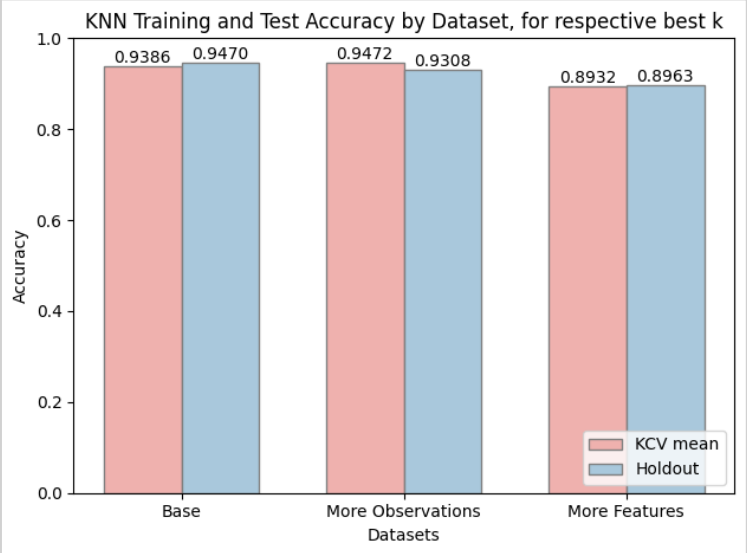


Figure 6. Accuracies from the cross-fold validation training, and the holdout test data.

It can be seen that the 5-fold cross-validation accuracy gave a very close prediction of the performance on unseen data, with at most around a 1.6% difference between the two values in the subset with more features. While the maximum test set accuracies were obtained at slightly different values of k to the training data (8, 11 and 22 respectively), models can of course not be optimised to unseen training data,

and these maximum test accuracies were within a maximum margin of around 3% - a very small difference when considering the models are already performing at around 90% accuracy and above. Figure 7 demonstrates the overfitting that KNN is at risk of with higher dimensionality, as expected.

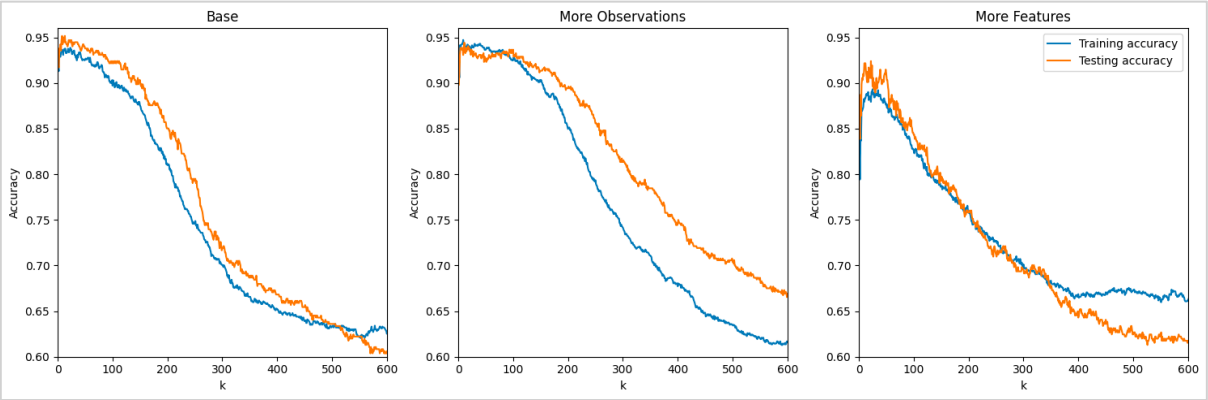


Figure 7. Plots of k against accuracy for each data subset as in Figure 4, with test accuracy overlaid.

Note: scale is different to Figure 4, to better see the variation between train and test accuracy.

The consistently higher performance of the test data than the training in the two lower-dimensional models is interesting, and the reason for this is unclear. It may simply be an artefact of the particular train-test splits that were performed – both prior to testing and within the k-fold validation.

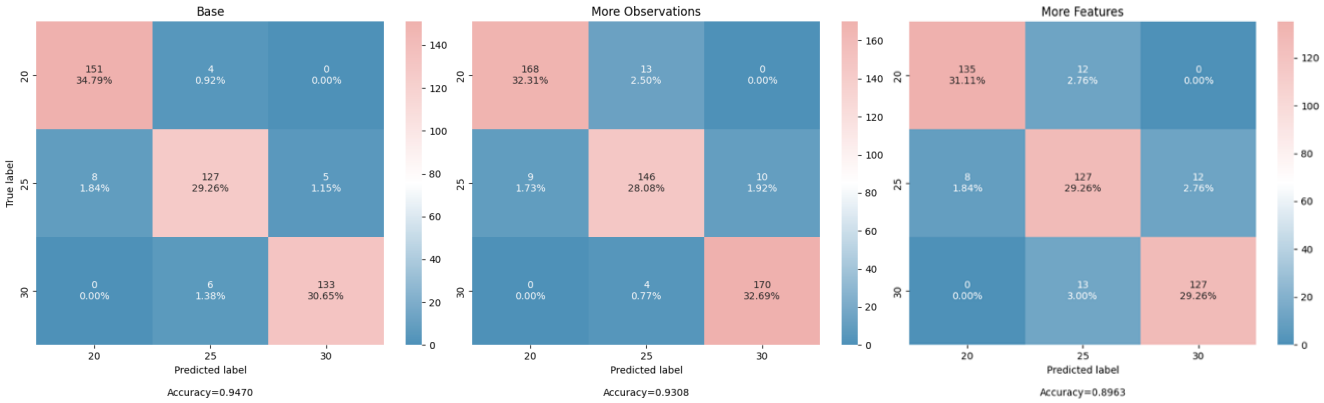


Figure 8. Confusion Matrices for each data subset.

Figures 8 and 9 further show the accuracy of the specific predictions made by the models using the holdout test data. One notable feature of these is that the class of

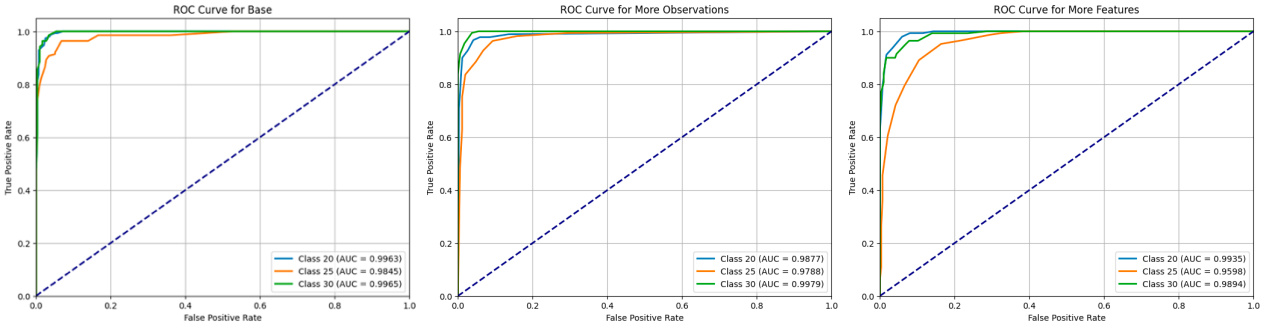


Figure 9. ROC curves (using one versus all) for each data subset.

25 degrees is the most often misclassified, having a lower precision and recall than the two other classes. Given the features have a strong and at least somewhat linear correlation to temperature, as shown by the heatmap in Figure 3, it holds that the middle class would be most often misclassified as it is close to both other classes.

The features of the observations in the temperature class 30 are likely much closer to the features of those in the class 25 than they are to class 20. This is shown clearly in the confusion matrix, where values with an actual label of 20 are never predicted as 30, and vice versa. Including more features also seems to have emphasised this error, with that data subset having the smallest AUC value for the class 25.

It is worth noting here that at this point in the analysis originally the graphs displayed a very odd pattern in the test and train accuracies, and one of the models had chosen a value of 43 for its best k. Upon discussion and investigation into these patterns, it became apparent that there had been some slight data leakage by using pre-normalised data in the cross-validation. The normalisation had also been done using a MinMaxScaler, rather than a StandardScaler. GPT3.5 suggests that the standardisation is more appropriate for KNN classifiers than normalisation (i.e., standard scaler is better than minmax scaler), but various machine learning websites seem to disagree amongst themselves which is better. The standardised data was used here as it performed better than the normalised, and comparing the two broadened the scope of the report too far. The original graphs can be found in the appendix.

4 Random Forest Classifier

A random forest is a bagging technique which uses decision trees as its base model. Bagging is an ensemble method which creates a group of similar but slightly different base models by randomly sampling overlapping subsets of the training data; this sampling is known as bootstrapping. Averaging the group of base models gives a model with lower variance than any single model and, since the base models tend to have relatively low bias, the averaged model will also have low bias.
Because of the random sampling of each bootstrap, roughly one third of the datapoints in any individual model in the ensemble will not have been used for training, so can act as a test point for that model. The misclassification rate calculated using this 'out-of-bag' test set is referred to as the out-of-bag (OOB) error. Averaging the OOB error for each ensemble member provides a reasonably reliable approximation of the expected error of the model on unseen data, and so provides an alternative to k-fold cross-validation (Lindholm et al., 2022).
Random forests take bagging one step further, taking a random subset of features to consider at each split of the decision tree. This affects the potential for one variable to dominate the early binary splits, which may improve overall accuracy.

As with the KNN model above, the accuracy will be used as a measure rather than the error, out of preference.

4.1 Model optimisation

Random forests have several parameters which can be varied, including the number of ensemble members (B), the number of features used at each split (q), the depth of each tree (d), and the splitting criterion at each node.
Lindholm et al. (2022) suggest that B can be made as large as desired without risk of overfitting, and the only reason to choose a small value is to reduce computational time. As such, B=50 was used as a reasonably large value without large computational times. Splitting criterion was chosen as the gini index, as this favours node purity more than the misclassification rate, and a comparison was not performed to reduce scope.

Both q and d seemed interesting hyperparameters to optimise, and so both were compared against the OOB score as a measure of generalisability to unseen data. d q were both varied between 1 and the maximum number of features possible for that subset, as this allowed for visualisation more easily. Normalisation is performed on the training data before bootstrapping, as this ensures each model interprets features in the same way, making the averages more accurate.

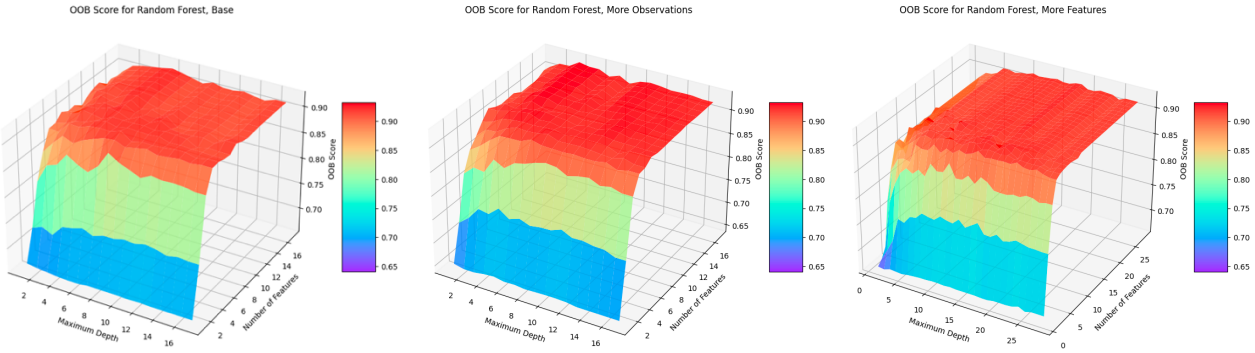


Figure 10. Plots of maximum tree depth vs number of features considered at each split vs OOB score, for each data subset.

	Base	More observations	More features
Max OOB	0.9228	0.9323	0.9248
Depth	9	15	9
No. of features	6	4	10

Table 5. Maximum OOB score achieved, and corresponding values of depth and number of features considered at each split.

It is evident from Figure 10 that the OOB score quickly plateaus at a relative maximum for both a small number of features at each split, and the depth of each tree. Even when only considering one feature per split, the OOB score does not fall below 0.62, which is nearly double the expected performance of a random classifier. Once again, having more observations results in a higher expected performance, but having more features results in a very small expected increase over the base subset. Having more observations found its optimal score at a higher depth than the other two subsets, which may be indicative that the extra observations allowed more patterns to be observed in the data, or of overfitting. However, given the robustness of ensemble methods to overfitting, this seems unlikely.

4.2 Optimised performance

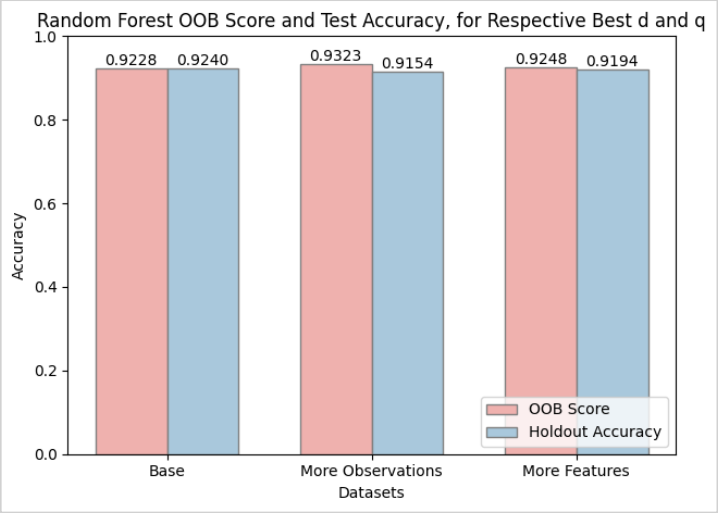


Figure 11. OOB scores versus accuracies of the holdout test data.

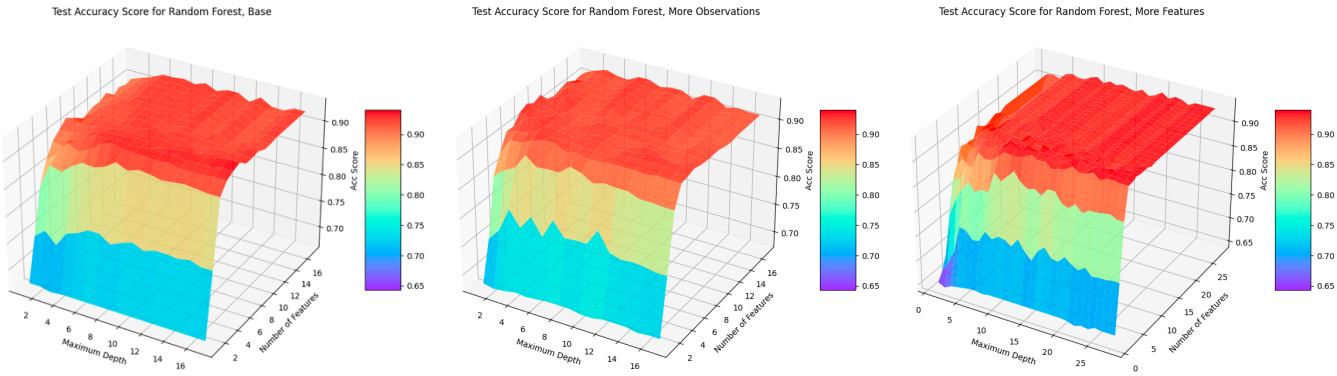


Figure 12. Plots of maximum tree depth vs number of features considered at each split vs test accuracy score, for each data subset.

The accuracy of the test data set on each version of this model followed the OOB scores fairly closely, once again suggesting that this was a good measure of how well the model would perform on unseen data. The maximum accuracies which were achieved for the base, more observations and more features subsets were 0.9355 at d=6, q=16; 0.9308 at d=10, q=3; and 0.9401 at d=5, q=10 respectively. These values are all within a small margin of the accuracy at the optimal d and q as determined by OOB score. As this maximum accuracy would fluctuate with differing test sets this further shows that the values selected by the tuning of the OOB score are likely to be reasonable.

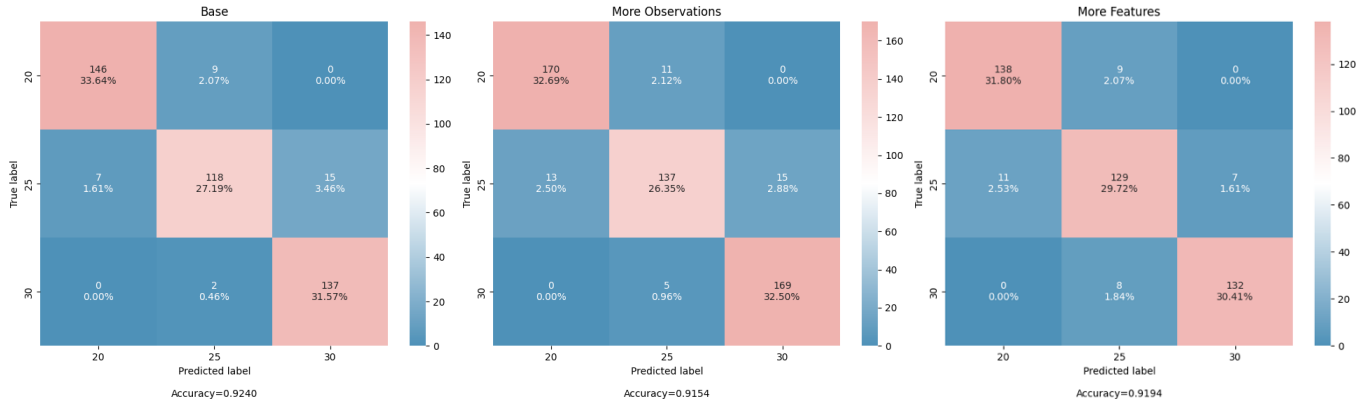


Figure 13. Confusion Matrices for each data subset, using respective best d and q.

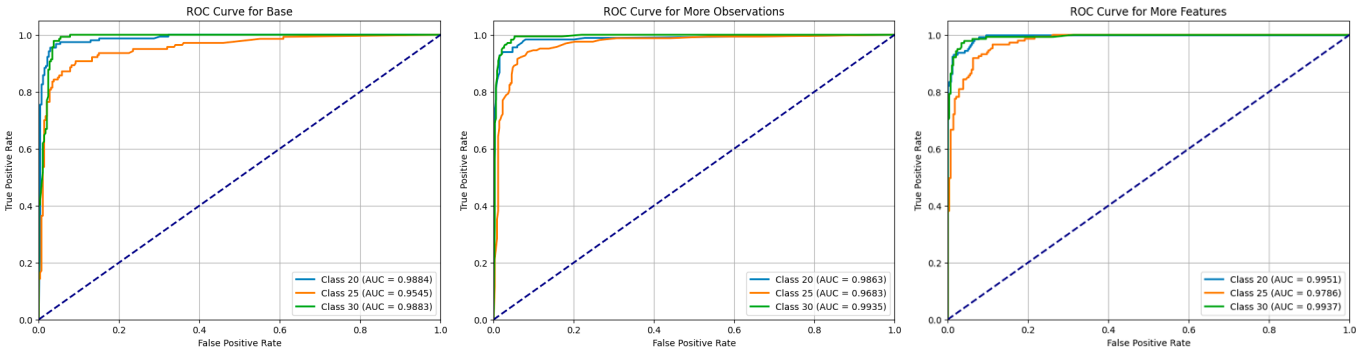


Figure 14. ROC curves (one vs. rest) for each data subset, using respective best d and q.

As with the KNN models, the confusion matrices and ROC curves in Figures 13 and 14 show that the majority of the misclassifications occur in relation to the middle temperature class, and all misclassifications happen only with a neighbouring class. That is, once again there are no instances of class 20 being misclassified as 30, or vice versa, and each of those classes has an AUC of close to 1.

5 Support Vector Classifier

A support vector machine is a nonparametric supervised model which aims to find a hyperplane in n-dimensional space which separates datapoints into classes by the largest margin. The points closest to the hyperplane are referred to as the support vectors, and a kernel function helps to calculate the distance between points, allowing the maximisation of the margins between the support vectors and the hyperplane in high dimensional spaces. While the trained model ends up relying on

only a small number of points, all training points are needed to create that trained model. Support vector classifiers, designed for binary classification, generally classify points as either inside or outside the decision boundary; for multi-class predictions this becomes more complex, and loss functions which use the margin become more complex. To tackle this, the data can be binarized into one-versus-one or one-versus-rest, with multiple binary classifiers trained. Here, one-versus-one was used, which takes majority vote from each classifier trained on a pair of classes, using margins to break ties.

Support vector classifiers also make use of a regularisation parameter, which penalises points inside the margin. This controls the trade-off between maximising the margin and minimising the classification error, and is a hyperparameter, as is the choice of kernel function.

5.1 Model optimisation

The hyperparameter chosen to optimise the SVC is the value of C. Higher values will cause the support vectors to be closer to the hyperplane, and classification errors will be minimised – though this may lead to overfitting. The choice of kernel can also be important, as a linear kernel is only suitable if the data is linearly separable. The scikit-learn documentation suggests that a linear kernel is often a good baseline choice. While the default setting provided in their tool is a radial basis function kernel, this requires tuning of an additional hyperparameter. Because the other models used in this report have assumed a degree of linearity, a linear kernel was used here as well.

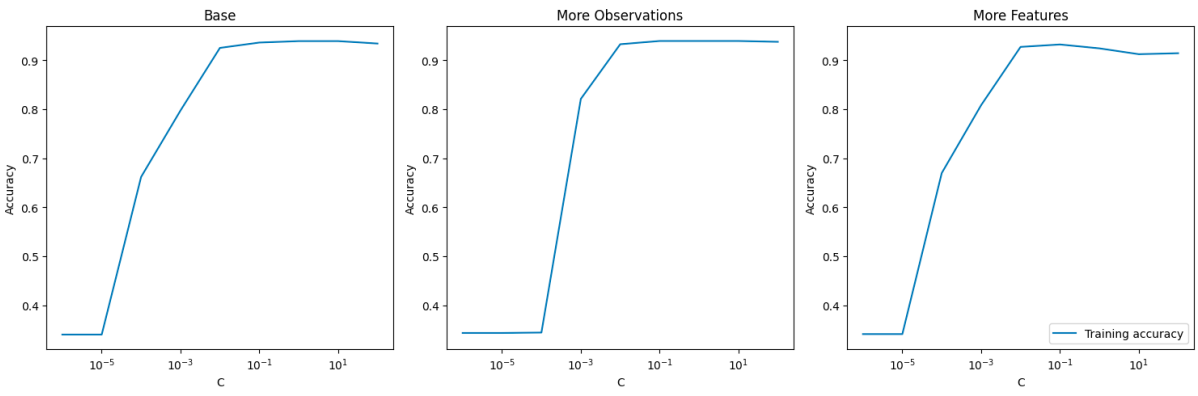


Figure 15. 5-fold cross validation accuracy on training data for each data subset, across values of C.

	Base	More observations	More features
Max accuracy	0.9386	0.9389	0.9318
Value of C	1	0.1	0.1

Table 6. Maximum cross-fold accuracy, and corresponding value of k, for each subset.

All three data subsets display accuracy of a random classifier at very small values of C, and plateau to a high accuracy at around -1 to 0 on a log10 scale. Having more observations allowed the accuracy to climb more quickly with increasing C, and this group did achieve the highest training accuracy by a very small margin. This is in line with expectations, as having more observations tends to allow SVCs to decrease their risk of overfitting, though at the trade off of higher computational time.

5.2 Optimised performance

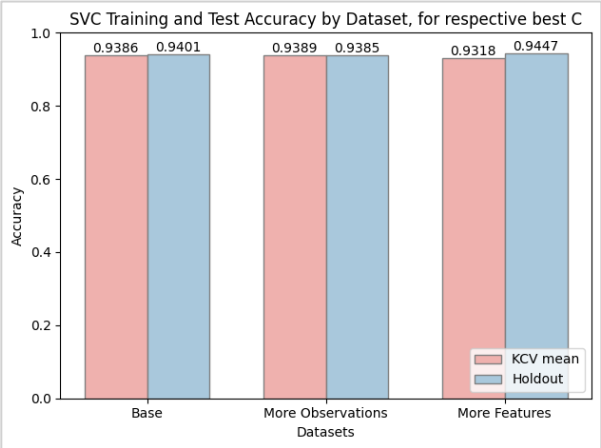


Figure 16. 5-fold cross validation training accuracy against test accuracy for respective best values of C.

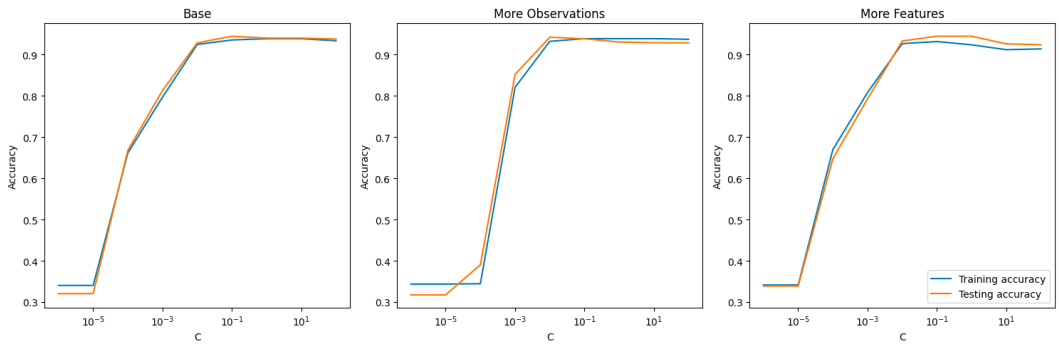


Figure 17. Training and test accuracy for varied values of C.

The test accuracy closely follows the training accuracy curve, for each model, though at small values of C a lower number of features causes a decreased test performance, and at high values a higher number of features appears to improve this performance slightly.

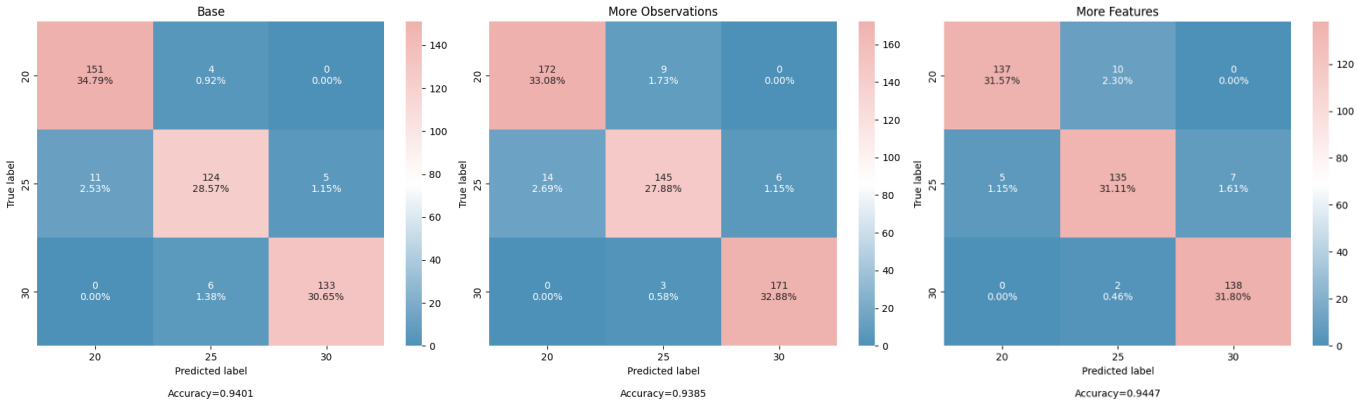


Figure 18. Confusion Matrices for each data subset, using respective best C.

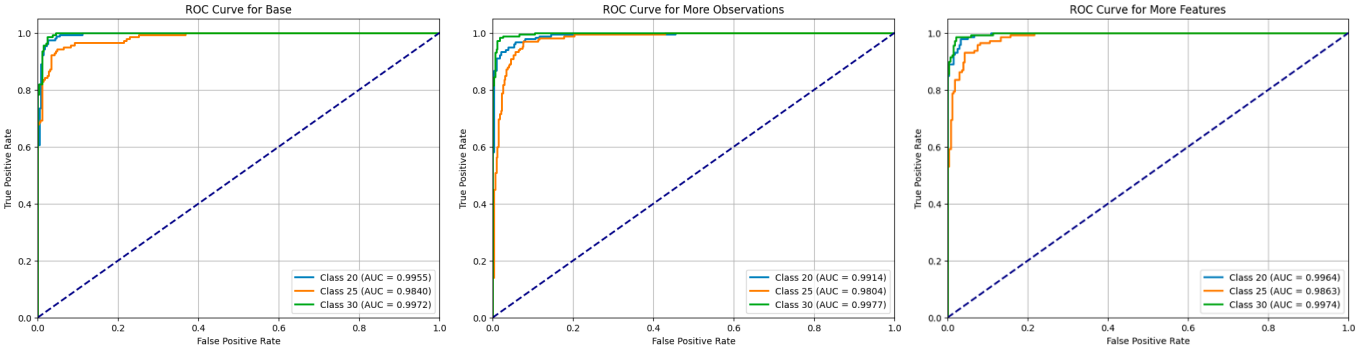


Figure 19. ROC curves (one vs. rest) for each data subset, using respective best C

As with the previous two models, the AUC values are very close to 1, with the middle class displaying the lowest recall and precision due to misclassifications only occurring within neighbouring classes. This difference is, however, still extremely small, and the overall accuracy is excellent.

6 Discussion

Across the three subsets of the data, the model which achieved the highest training accuracy was the set with more observations. For both the KNN and random forest, the base model achieved the highest accuracy at the optimised parameters, and the set with more features performed the best on the optimised SVC.
These results are consistent with what was expected from the models – seeing more data without the noise of excess dimensionality resulted in more accurate classifications. However, the largest difference in test accuracy between any of the models was around 0.05 – an almost negligible difference when considering the performance is consistently sitting at approximately 0.9 and above. In all three cases, the class of 25 – which was the middle class of the three – displayed the most inaccuracy. No class was ever misclassified as anything but the class beside it, but as 25 had two neighbouring classes rather than 1, it was more frequently misclassified. This suggests that even in multi-dimensional space, there was significant overlap between this class and the others, but the classes 20 and 30 were easily separable.

While some degree of linearity was assumed in the data for all three of the models, it is likely that this was not the case. With at least 17 dimensions, linearly separating the classes is difficult, and more so as the number of features increases. Had the scope and length of this report allowed, it would have been interesting to reduce the dimensionality using a technique such as principal component analysis and compare the performance of models even further.

Two objectives were outlined in the introduction of this report.

- Objective 1: Given the study showed temperature impacts size, reverse this and determine whether temperature can be accurately predicted from the measurements of a fruit fly.
- Objective 2: Determine the impact varying the number of features and observations has on the performance of model prediction.

From the results gathered for the three models, it can be seen that temperature can very accurately be predicted from the measurements of a fruit fly, regardless of model or increased dimensionality.

The impact that varying the observations and features has is that in general, increasing the number of observations and decreasing the number of features will provide the optimum expected performance of a model. However, it was also shown that the variation in this context was extremely small, and for this relatively low dimensionality the number of features can be selected quite liberally without extreme risk of decreasing performance.

7 References

Lindholm, A., Wahlström, N., Lindsten, F., & Schön, T. (2022). *Machine learning: A First Course for Engineers and Scientists*. Cambridge: Cambridge University Press.

Loeschke, V., Bundgaard, J., & Barker, J.S.F. (2000). Variation in body size and life history traits in Drosophila aldrichi and D. buzzatii from a latitudinal cline in eastern Australia. *Heredity, 85*(5), 423-433. https://doi.org/10.1046/j.1365-2540.2000.00766.x

GPT 3.5 and 4o were consulted for discussion, ideas, and code help in the formation of this report.

8 Appendix

All code used to create the models and graphs in this report can be found at https://github.com/gwyldbore/COMP7703_report

Images from Table 2.

```
RangeIndex: 1731 entries, 0 to 1730
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Species        1731 non-null   object
 1   Population     1731 non-null   object
 2   Latitude       1731 non-null   float64
 3   Longitude      1731 non-null   float64
 4   Year_start     1731 non-null   int64
 5   Year_end       1731 non-null   int64
 6   Temperature    1731 non-null   int64
 7   Vial           1731 non-null   int64
 8   Replicate      1731 non-null   int64
 9   Sex            1731 non-null   object
 10  Thorax_length  1731 non-null   object
 11  l2             1731 non-null   float64
 12  l3p            1731 non-null   float64
 13  l3d            1731 non-null   float64
 14  lpd            1731 non-null   float64
 15  l3             1731 non-null   float64
 16  w1             1731 non-null   float64
 17  w2             1731 non-null   float64
 18  w3             1731 non-null   float64
 19  wing_loading   1731 non-null   object
```

```
RangeIndex: 1731 entries, 0 to 1730
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Species               1731 non-null   object
 1   Population            1731 non-null   object
 2   Latitude              1731 non-null   float64
 3   Longitude             1731 non-null   float64
 4   Year_start            1731 non-null   int64
 5   Year_end              1731 non-null   int64
 6   Temperature           1731 non-null   int64
 7   Vial                  1731 non-null   int64
 8   Replicate             1731 non-null   int64
 9   Sex                   1731 non-null   object
 10  Wing_area             1730 non-null   float64
 11  Wing_shape            1712 non-null   float64
 12  Wing_vein             1725 non-null   float64
 13  Asymmetry_wing_area   1705 non-null   float64
 14  Asymmetry_wing_shape  1705 non-null   float64
 15  Asymmetry_wing_vein   1717 non-null   float64
```

```
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Species         1727 non-null   object
 1   Population      1727 non-null   object
 2   Latitude        1727 non-null   float64
 3   Longitude       1727 non-null   float64
 4   Year_start      1727 non-null   int64
 5   Year_end        1727 non-null   int64
 6   Temperature     1727 non-null   int64
 7   Vial            1727 non-null   int64
 8   Replicate       1727 non-null   int64
 9   Sex             1727 non-null   object
 10  Asymmetry_l2    1721 non-null   float64
 11  Asymmetry_l3p   1726 non-null   float64
 12  Asymmetry_l3d   1718 non-null   float64
 13  Asymmetry_lpd   1717 non-null   float64
 14  Asymmetry_l3    1717 non-null   float64
 15  Asymmetry_w1    1712 non-null   float64
 16  Asymmetry_w2    1715 non-null   float64
 17  Asymmetry_w3    1713 non-null   float64
```

Original KNN figures from incorrectly normalised data

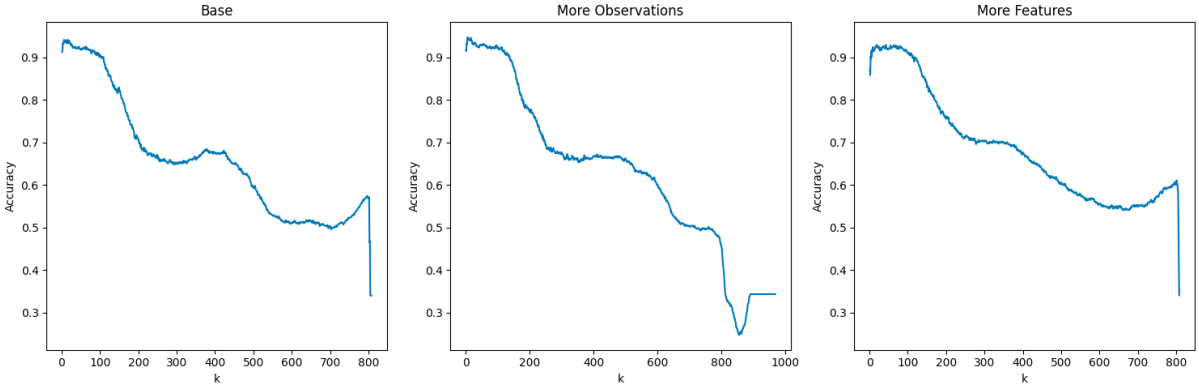


Figure x. Plots of k against accuracy for each data subset.

	Base	More observations	More features
Max accuracy	0.9416	0.9480	0.9298
Value of k	6	7	43

Table x. Maximum cross-fold accuracy, and corresponding value of k, for each subset.

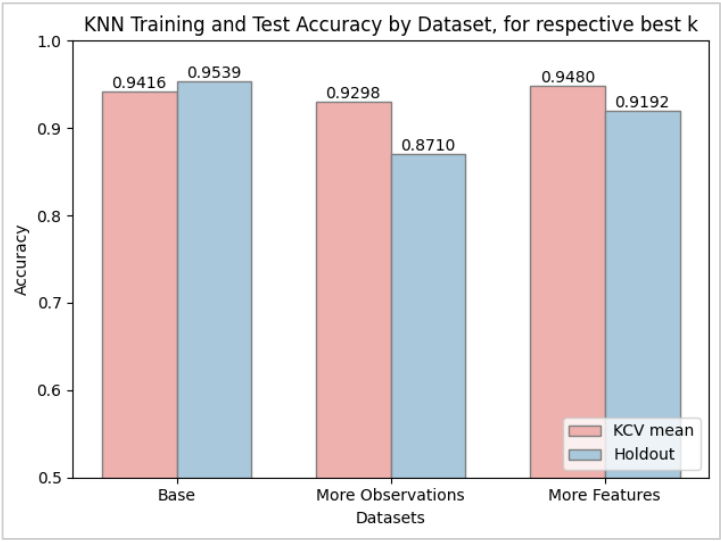


Figure x. Accuracies from the cross-fold validation training, and the holdout test data.

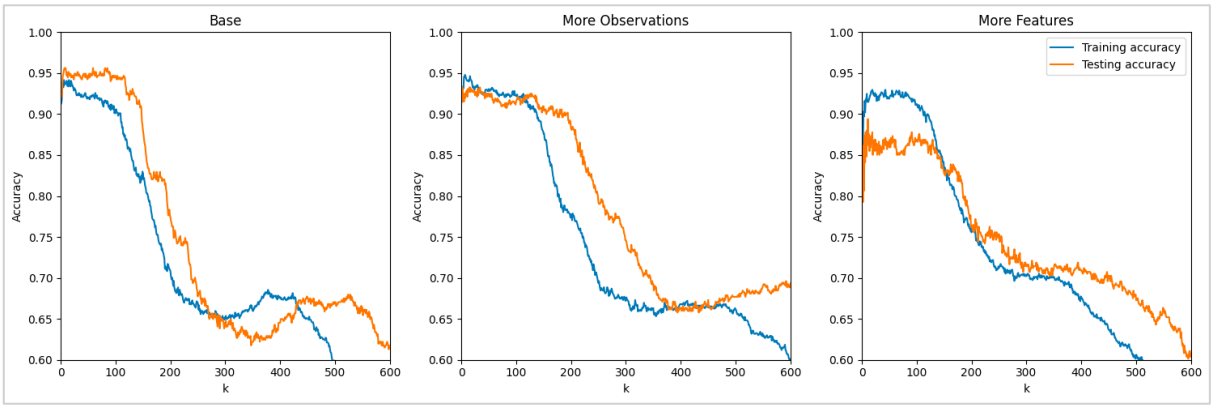
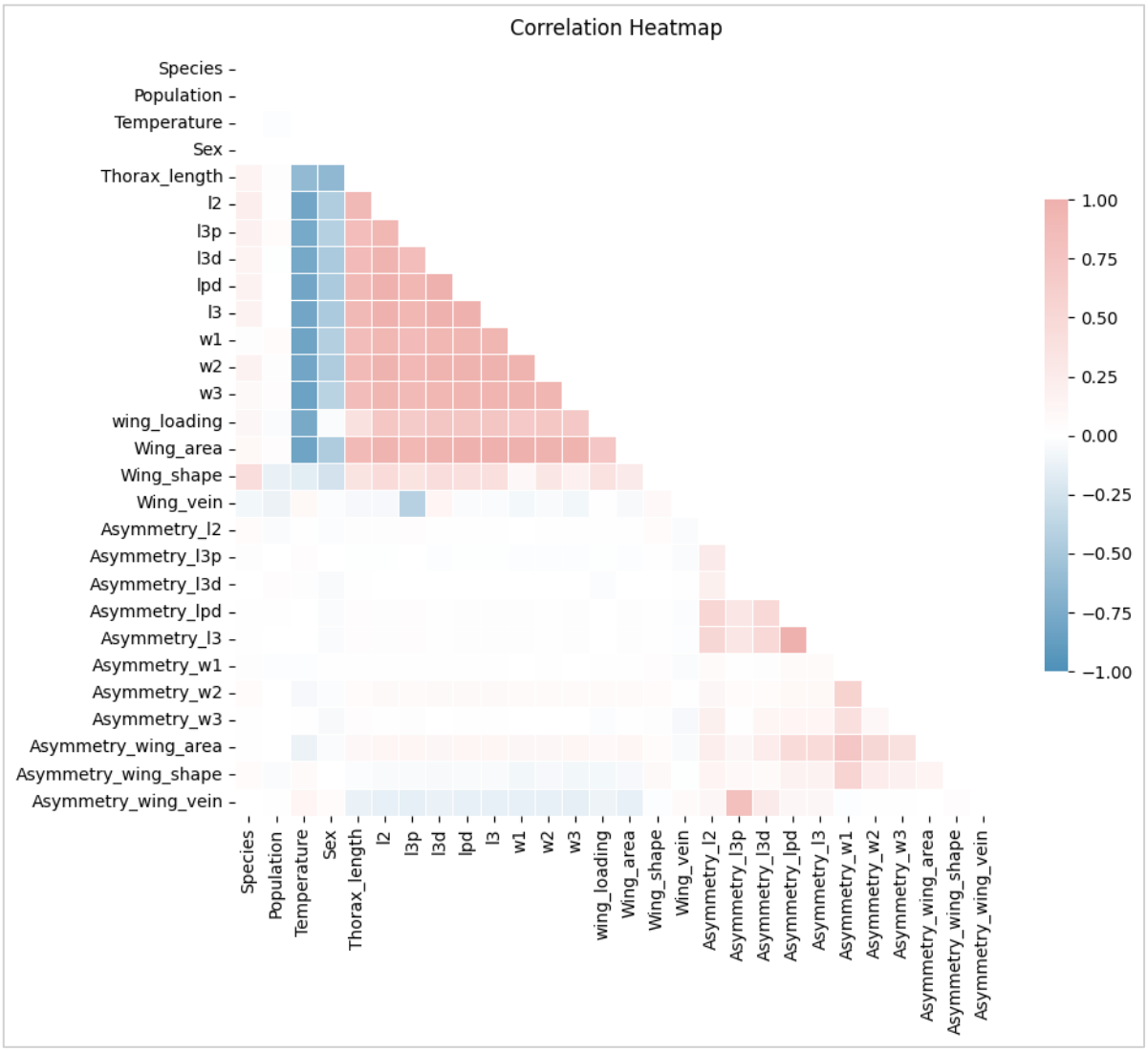


Figure x. Plots of k against accuracy for each data subset as in Figure 4, with test accuracy overlaid.

Figure 3


Correlation Heatmap
Species
Population
Temperature
Sex
Thorax_length
l2
l3p
l3d
lpd
l3
w1
w2
w3
wing_loading
Wing_area
Wing_shape
Wing_vein
Asymmetry_l2
Asymmetry_l3p
Asymmetry_l3d
Asymmetry_lpd
Asymmetry_l3
Asymmetry_w1
Asymmetry_w2
Asymmetry_w3
Asymmetry_wing_area
Asymmetry_wing_shape
Asymmetry_wing_vein
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Plots from Figures 5 and 7

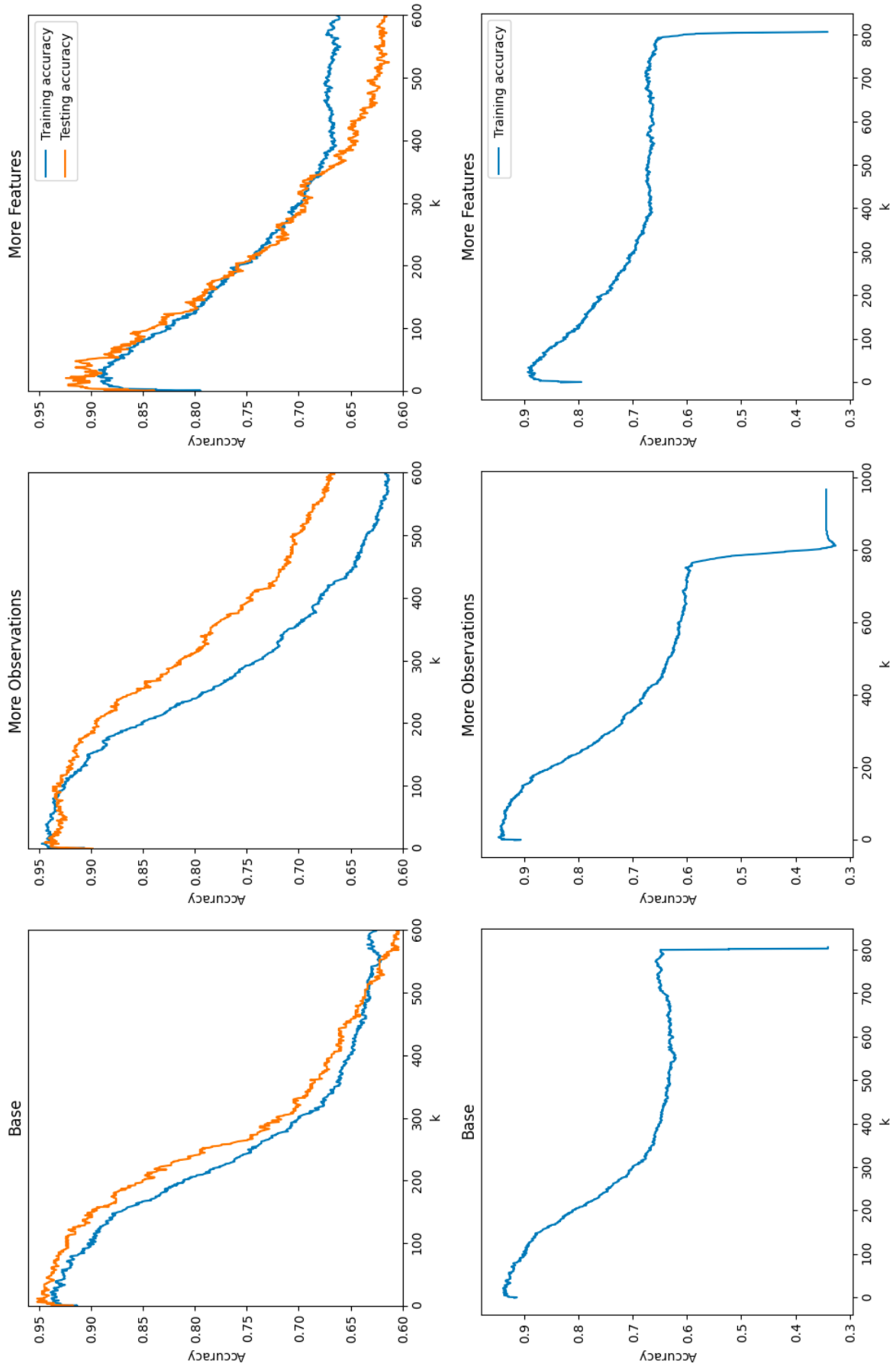


Figure 6

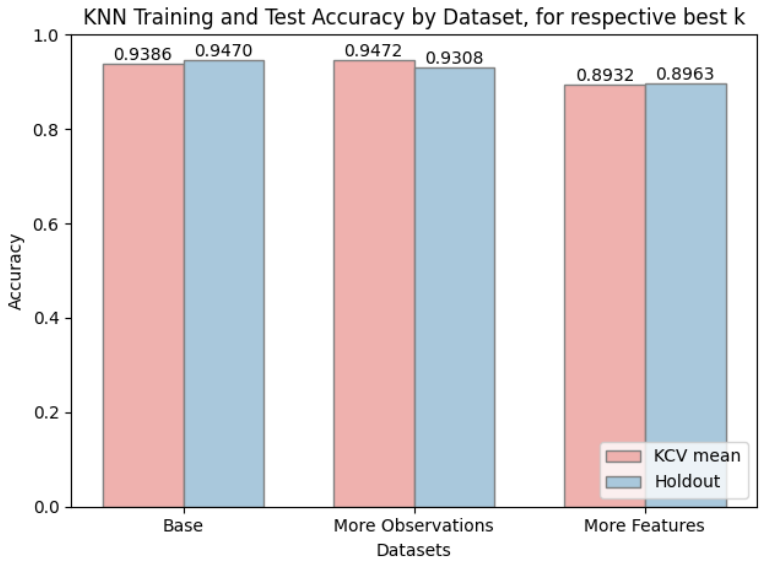


Figure 11

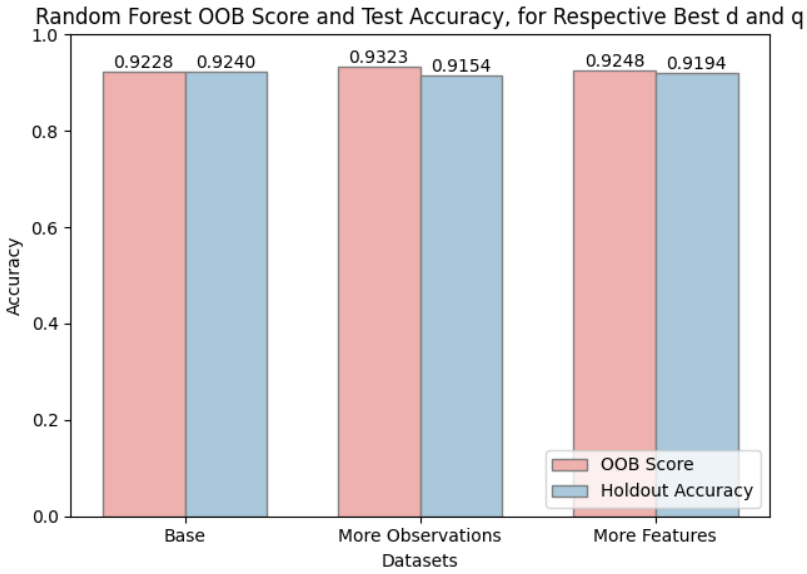
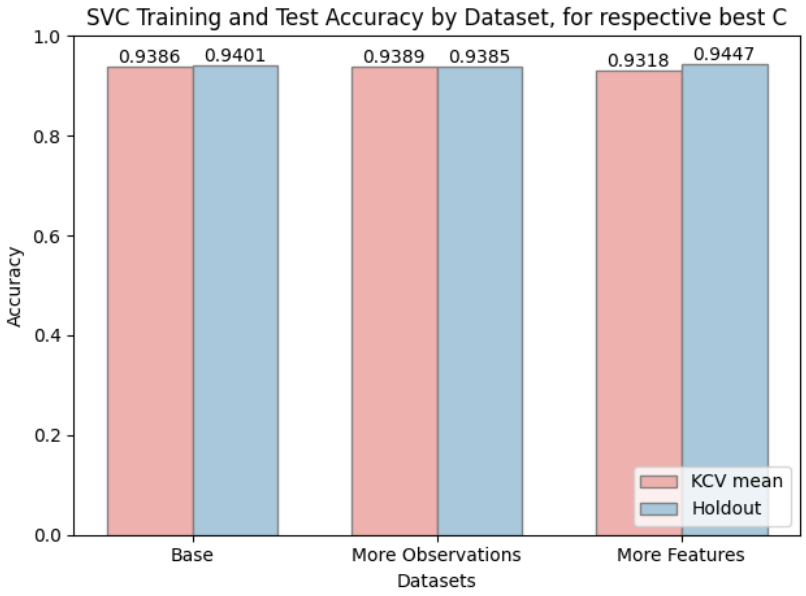
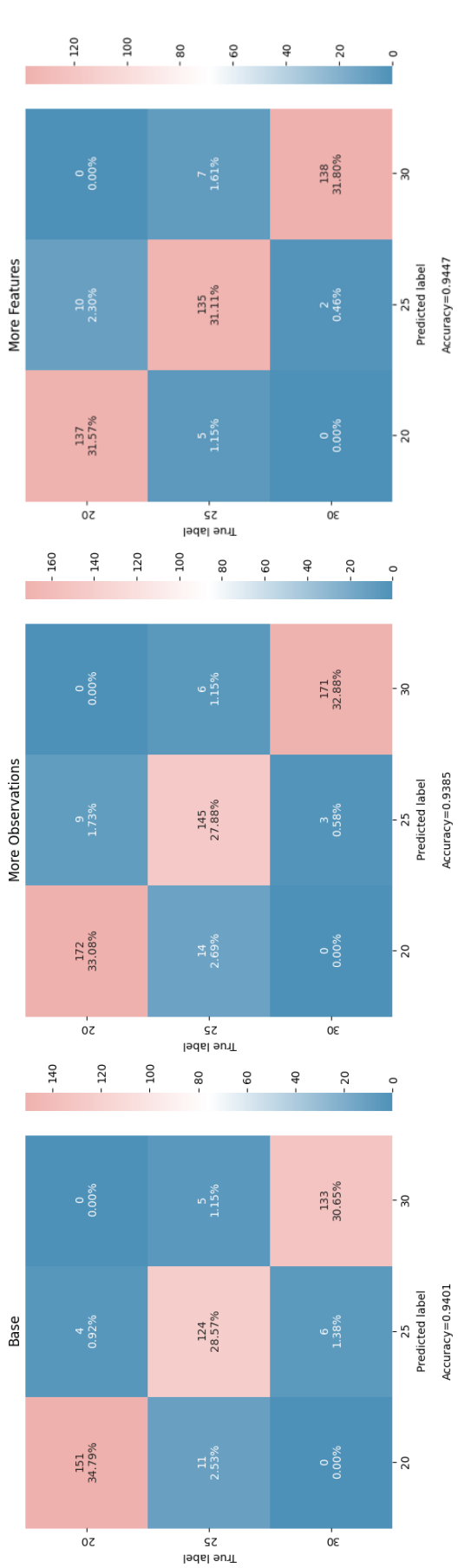
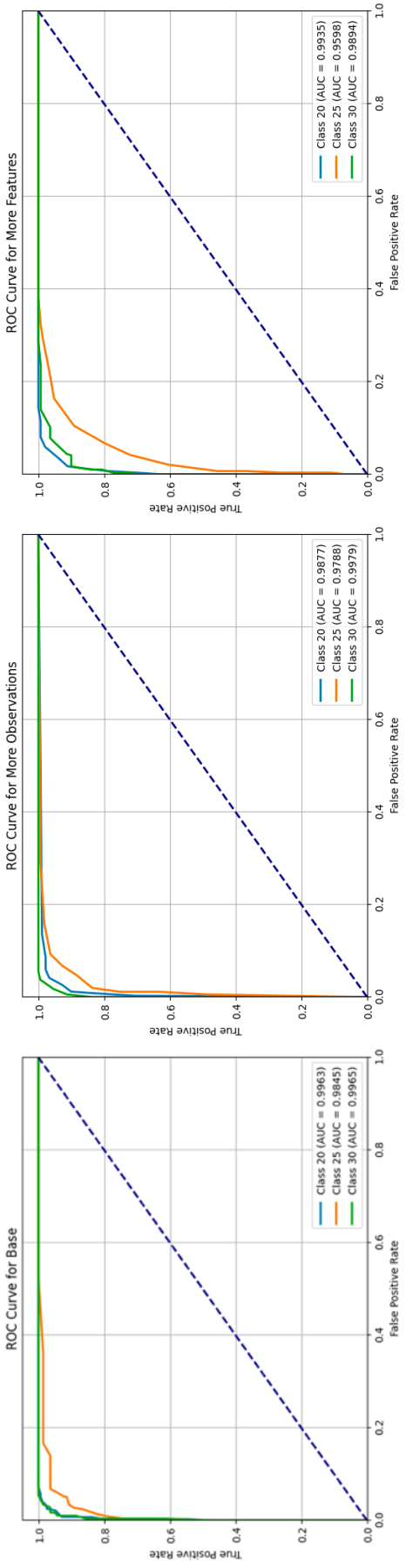


Figure 16

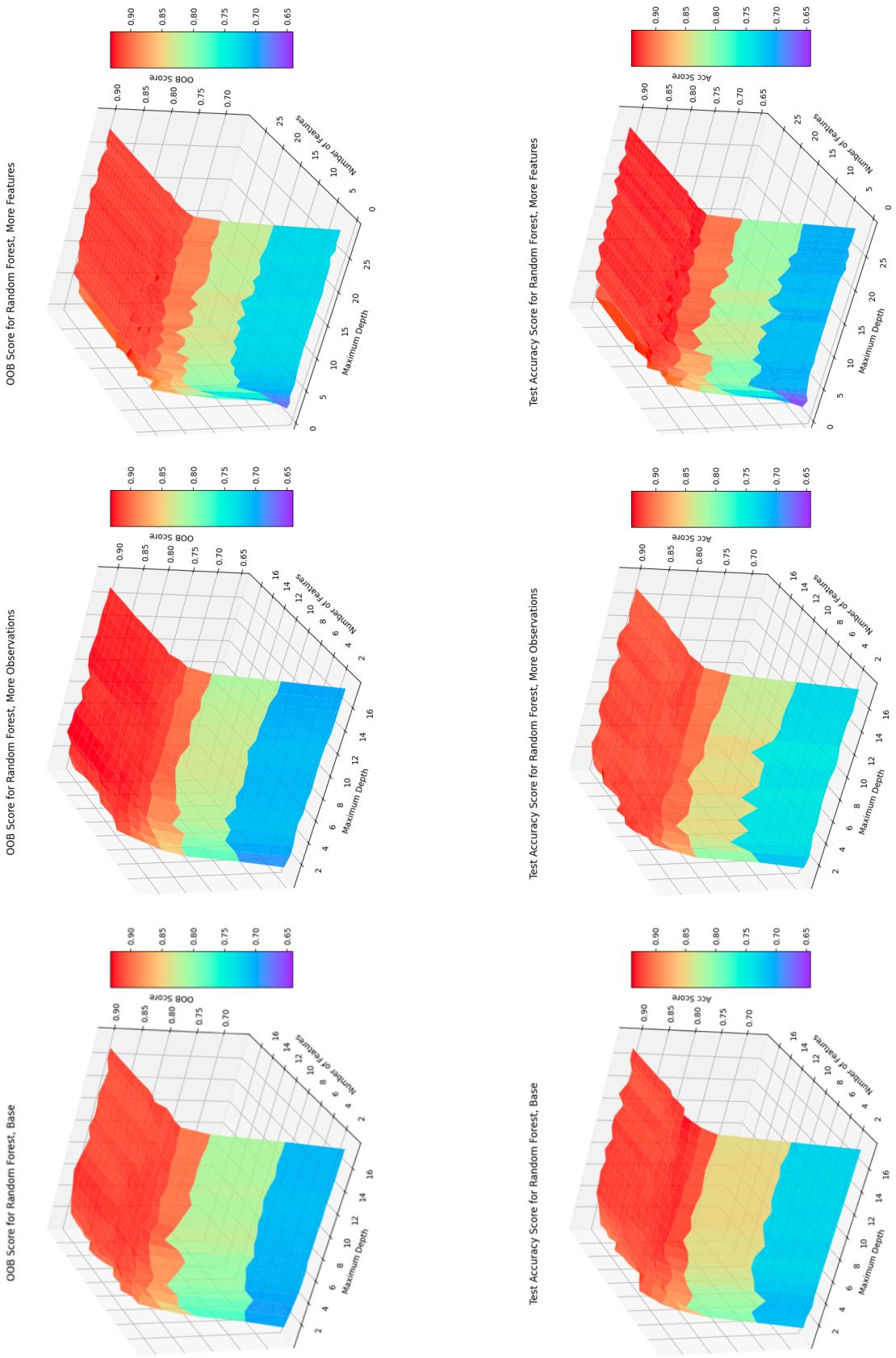


Figures 8 and 9

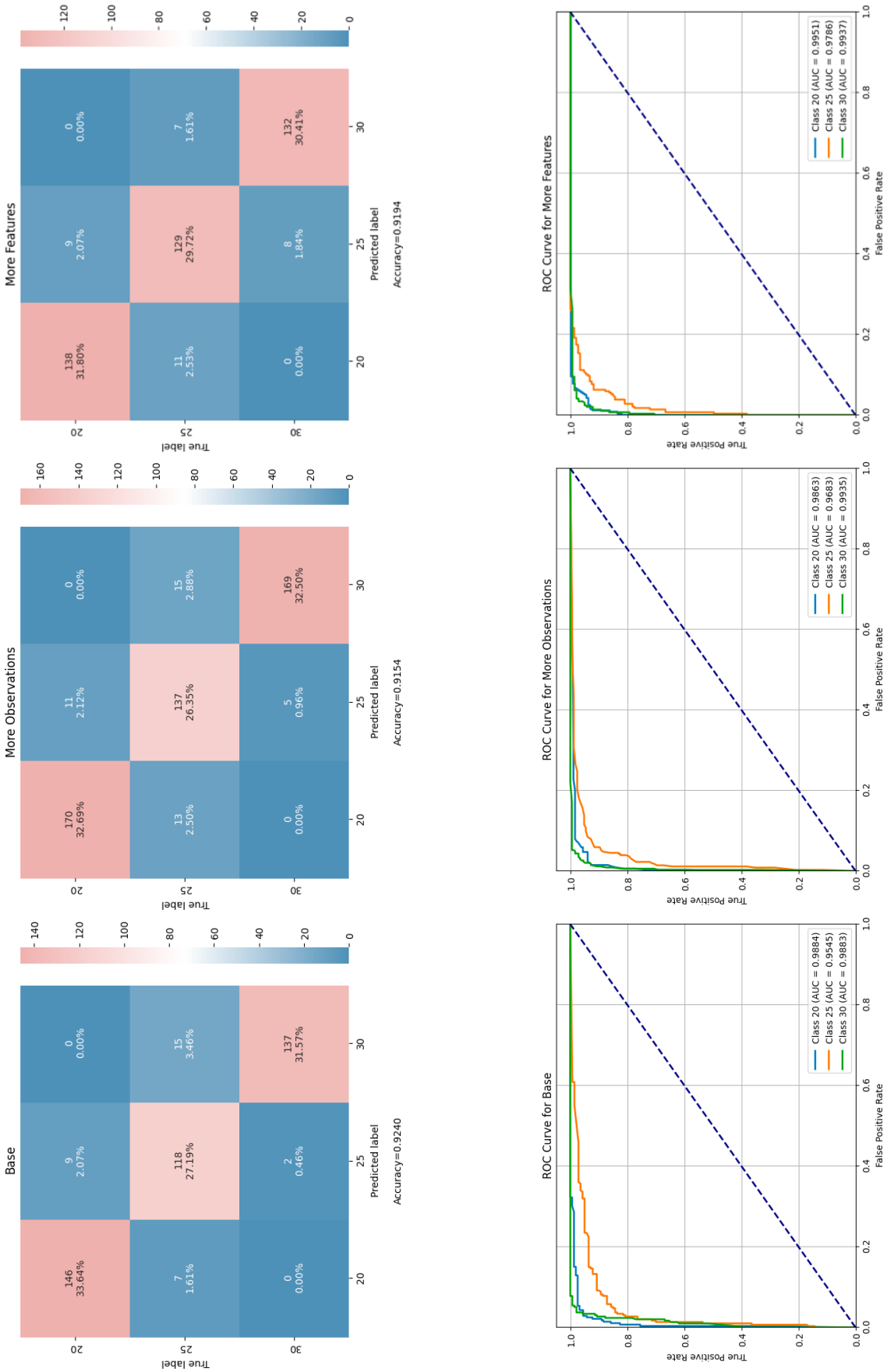




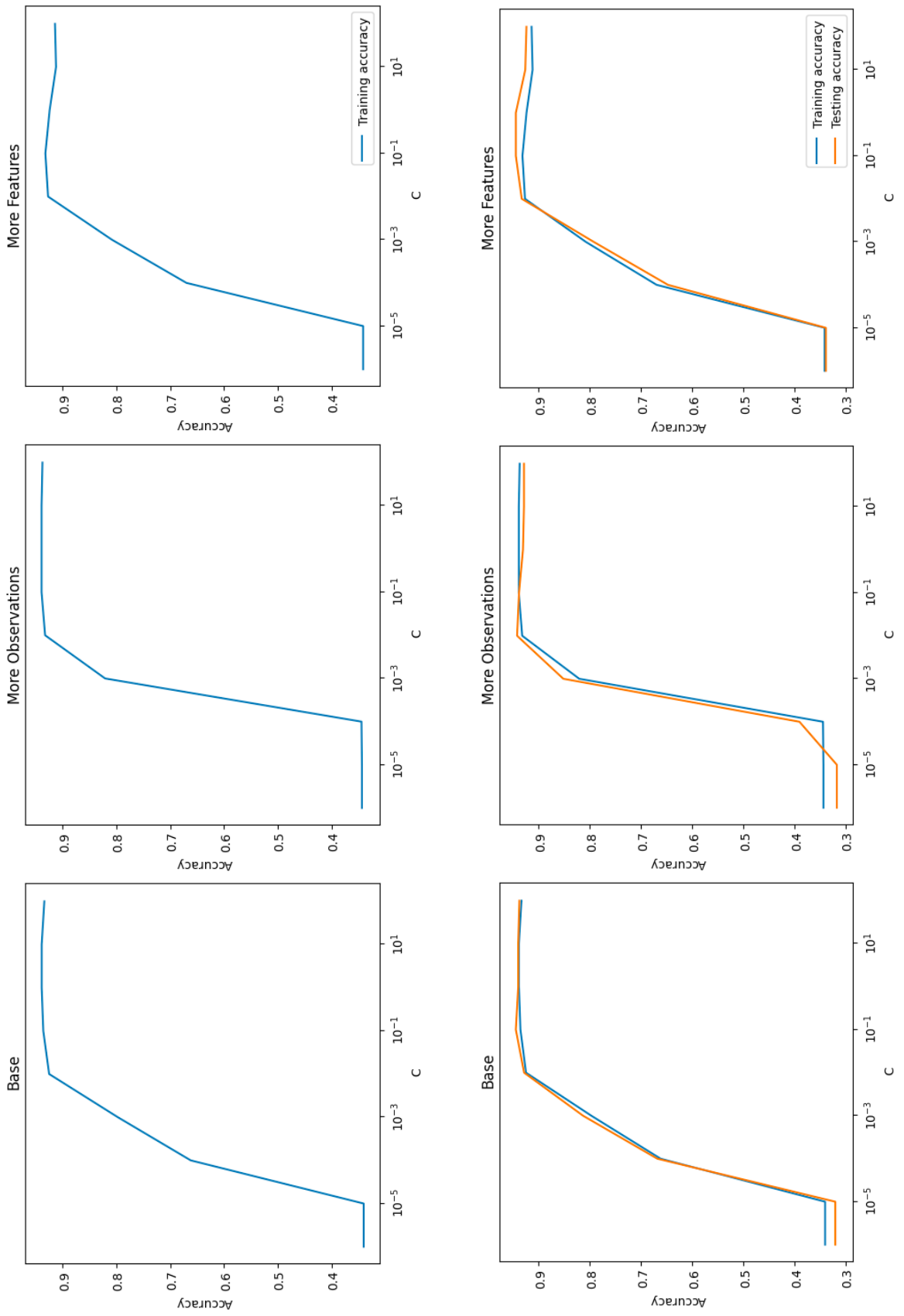
Figures 10 and 12



Figures 13 and 14



Figures 15 and 17



Figures 18 and 19