## AI in Climate Change: Interpretable Ice Sheet Modeling using Machine Learning

Serafina Slevin, Bing Xue, Bach Nguen, Nicholas Golledge

*Victoria University of Wellington | Te Herenga Waka*

### Introduction

Current climate simulations are in agreement with empirical data, both suggesting that global mean sea level (GMSL) is rising [1, 2]. Simulations also predict that the GMSL will continue to rise in the coming years and centuries [3]. Sea level rise (SLR) will affect many communities and economies around the world. Large areas of low-lying land would be affected by an increase in the local sea level, seeing many cities and economic centres displaced through flooding [5]. Understanding the mechanisms by which this is caused, the rate at which this occurs, and the uncertainties associated with these predictions is crucial for informed policy making.

The four major contributors to SLR are thermal expansion, the Greenland ice sheet, the Antarctic ice sheet, and other small glaciers around the world [3]. The Antarctic ice sheet (AIS) would have the largest contribution, raising GMSL by 55-60 metres if it melted in its entirety [4]. The Response of the ice sheet to changes in its surrounding system are delayed [6], meaning the effects of changes we set into motion now may take centuries to millennia to be fully realised. For this reason, it is essential that we work to understand the impact we are having on future generations through our actions in current times.

Currently, the most popular technique for making predictions about future ice melt and SLR is using large statistical models [7]. These models are designed to represent the physical processes within the Earth’s system, and the feedback between them as closely as possible. Although effective, these simulations can be prohibitively expensive computationally and time-consuming to run. There are scenarios where it would be beneficial to have a way to create predictions of the same style as the simulation, without running it.

A secondary limitation within the field is sparse data. The large timescales associated with the reactions of the Greenland and Antarctic ice sheets mean that although there has been considerable data collection in recent decades, it is not enough to capture the full behaviour of the sheets empirically [7]. Geological and sedimentary records can be used to infer information about the AIS and GIS in other periods, with the downside of introducing further uncertainty into the work [7].

### Overview

The focus of this work has been on using machine learning (ML) methods to mimic the behaviour of a climate simulation. The climate simulation data was provided by Nicholas Golledge at the Antarctic Research Centre (Te Puna Pātiotio). We have used data from the climate simulation to train various models, hoping to capture the underlying patterns. If the underlying patterns can be found and represented in some more simple form, this could be used to calculate predictions directly instead of running the entire simulation. Calculating predictions this way could prove to be effective at cutting down the time and compute power needed to produce a prediction. A decrease in time spent obtaining a prediction would mean a larger number of climate scenarios and their predictions could be analysed. This is the broader goal of the work, to find methods that are less time consuming and computationally expensive that produce the same (or similar) results as the climate simulations. What I have done so far is just the first exploratory steps towards this goal. Since it’s a new project, I have focused on completing a solid foundation of initial work. There are many more areas of this work that could and should be expanded on further, something which I have covered in great detail in section 7.

I first completed an extensive EDA, allowing myself and others in the future to understand the data we have to work with. I then moved on to casting a wide net. I used a variety of different model types to predict our target classes, with the goal of finding those with the most promise. After narrowing down my focus to random forest for the continuous targets and support vector machine for the categorical target, I looked at improving the performance of these. I carried out feature engineering and tuned the models to optimise their performance. I also looked at sequential modelling, using the predicted value of ice mask as a feature for predicting ice velocity and ice thickness. The final method I covered was genetic programming, hoping to find some more interpretable results.

### Exploring The Data

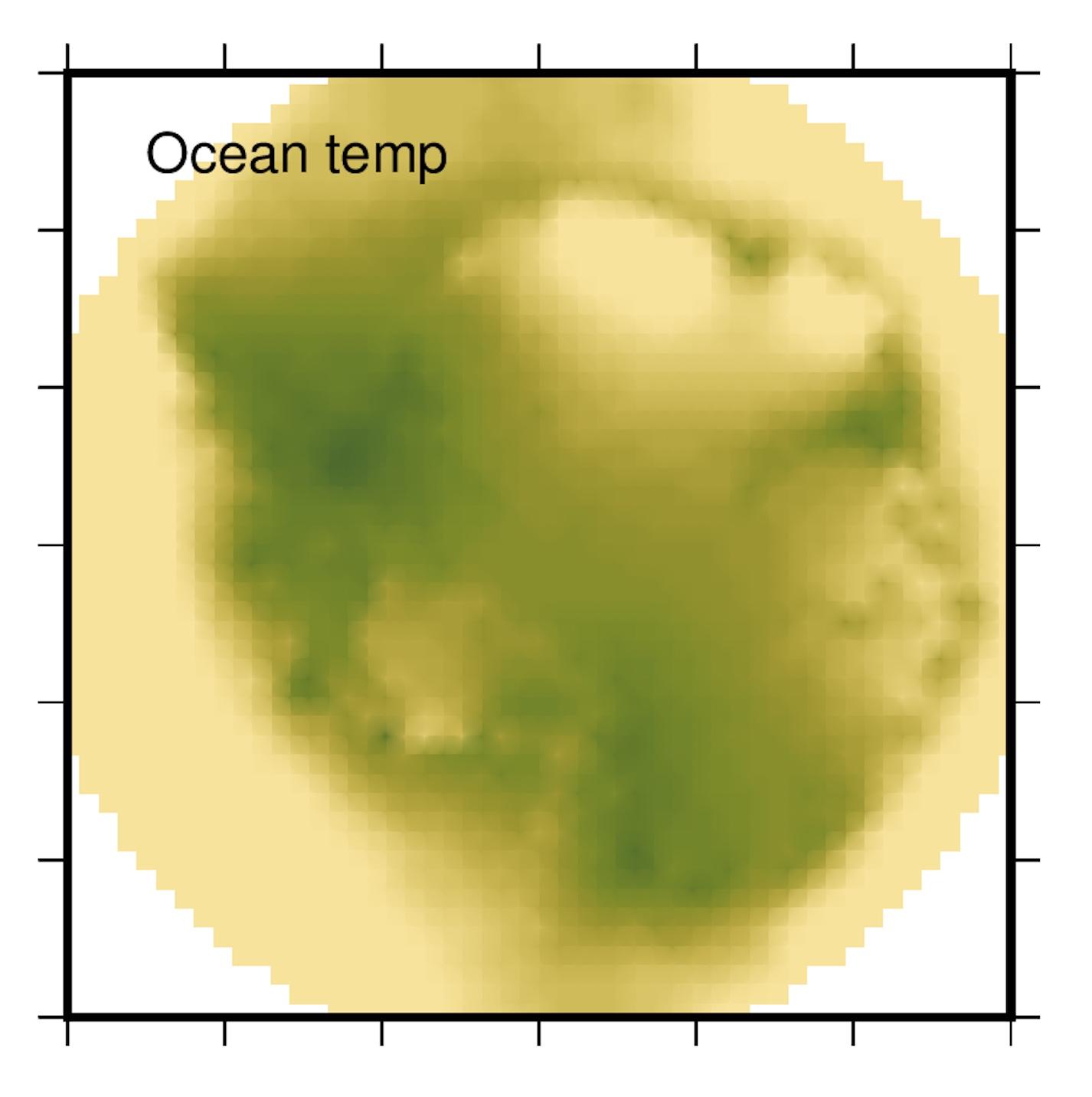
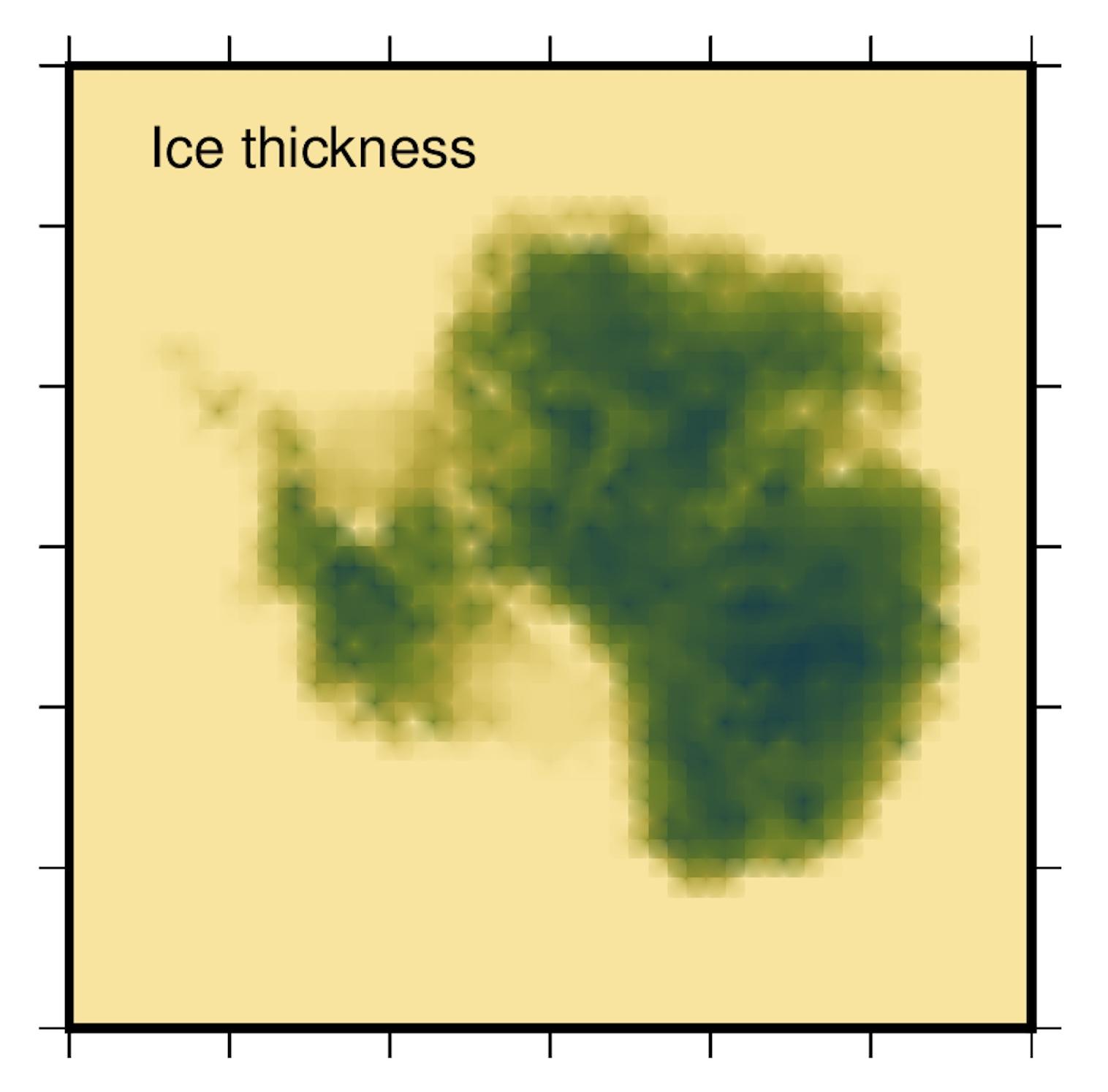
The first thing I would like to do is give a good understanding of the data and its characteristics. We have tabular data, with each instance representing a coordinate point on the planet. The points form a square shaped grid, centred about the Antarctic ice sheet. The data given to me (and included in the repository) has a grid of points like this for each year from 2015 to 2100. Currently I am only looking at the first time slice of 2015. The modelling I have done is only using the data from the first time slice, so I have not expanded beyond that. Looking forward, a goal would be to predict the output for multiple years and not just one. When that becomes a focus, the EDA would need to consider further years also.

In its raw form, each instance has 5 inputs/features and 3 outputs/targets. In the table below I have included each variable, along with a description of what that variable is and any general characteristics it has.

| Variable Name: | Target/Feature: | Characteristics/Comments: |
| --- | --- | --- |
| x-axis | feature | This is the point’s x-coordinate value. |
| y-axis | feature | This is the points y-coordinate value. The x and y coordinates are centred around a (0,0) point in the middle of the ice sheet. |
| precipitation | feature | This is a measure of water that has left the atmosphere and comes down to the surface in the form of rain, snow, sleet, or hail. |
| air\_temp | feature | This is the air temperature of the point. |
| ocean\_temp | feature | This is the ocean temperature of the point. Ocean temperatures are only available in a circle around the centre point of the grid. Where they are not available, there is a fill value in the data of 9.969... e+36. |
| ice\_thickness | target | This is the thickness of the ice at a given point. Where there is no ice - ie. in the open ocean, the thickness is 0. |
| ice\_velocity | target | This is the speed at which the ice is moving. Ice is dynamic and can move, particularly around the border of the ice sheet. Where there is no ice, we have a fill value of -1 or NaN. |
| Ice\_mask | target | This is a categorical variable with three integers of 2, 3, and 4, which represent grounded ice, floating ice, and open ocean, respectively. |

#### Preprocessing

Before completing any EDA, I chose to delete any rows with the fill value for ocean temperature. Since those fill values are only missing for a small area in each corner, I felt this was a reasonable choice. I chose to do this before any further EDA, as the EDA would be more valuable if carried out only on the data being given to the models. We can get a better idea of what this preprocessing step does by looking at a visualisation. Below is a comparison of the ocean temperature values to ice thickness values. We can see that cutting out points where there are no values for ocean temperature would only remove areas of open ocean. We are more concerned with those data points closer to or within the border of the ice sheet, so I feel that this is a reasonable decision. By reducing the number of data points, we also cut down the time needed for any further processing steps.



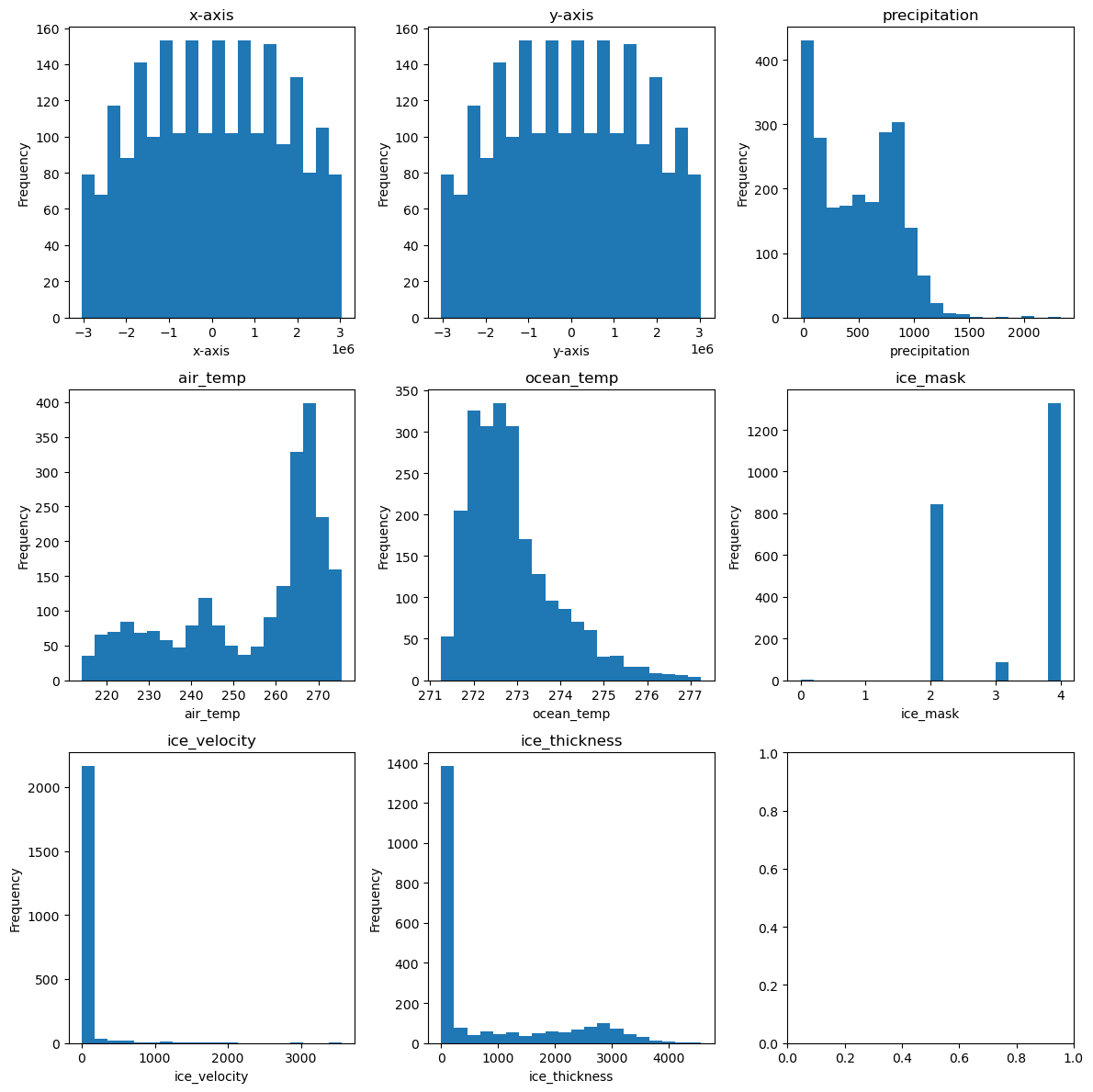
Images above provided by Nick Golledge.

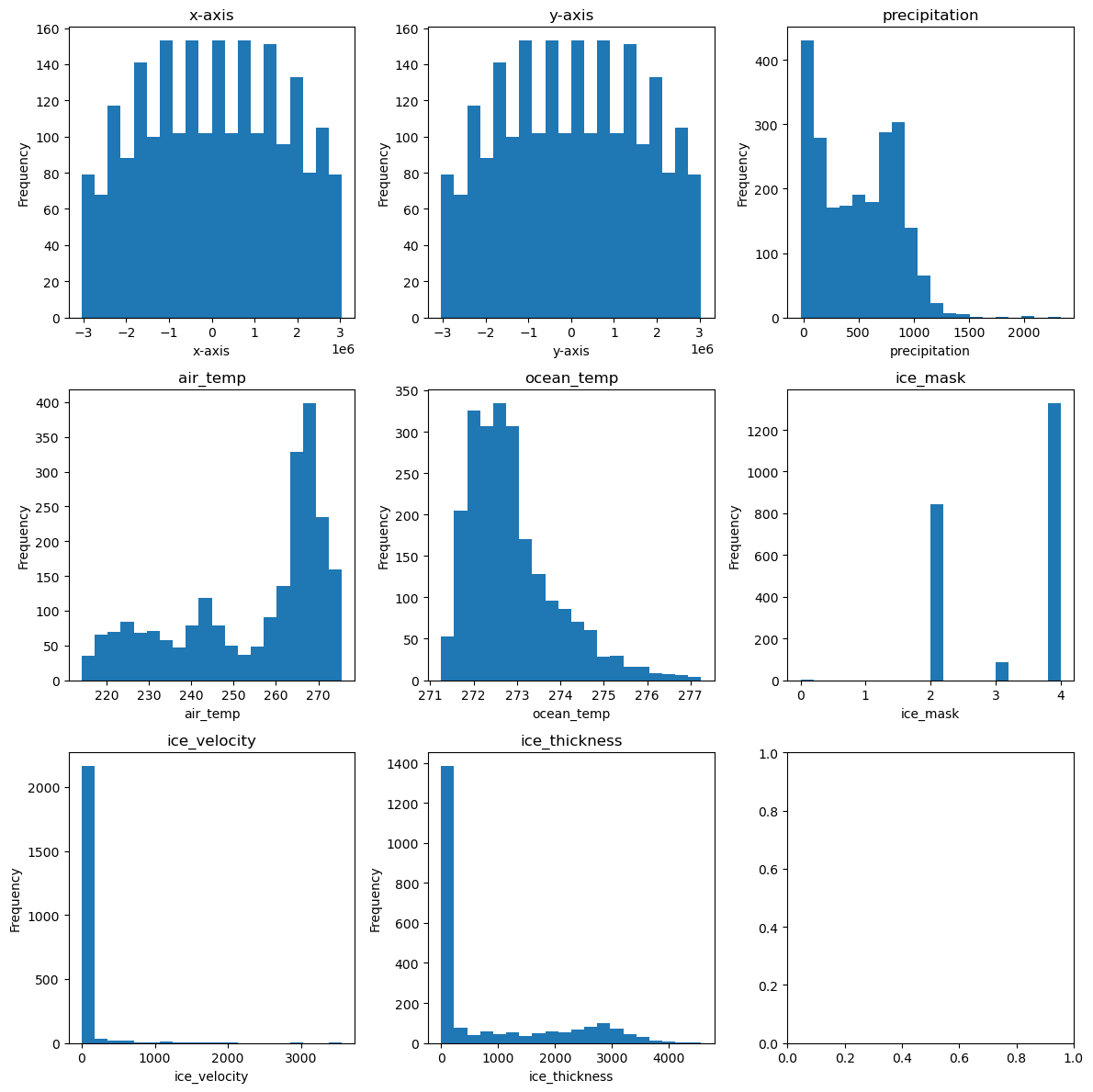
#### Basic Characteristics

Before we look further at the data, I will include here a basic overview table of the full dataset. I did not include the xy coordinate columns here, or the ice\_mask column, as I felt they would not be particularly relevant.

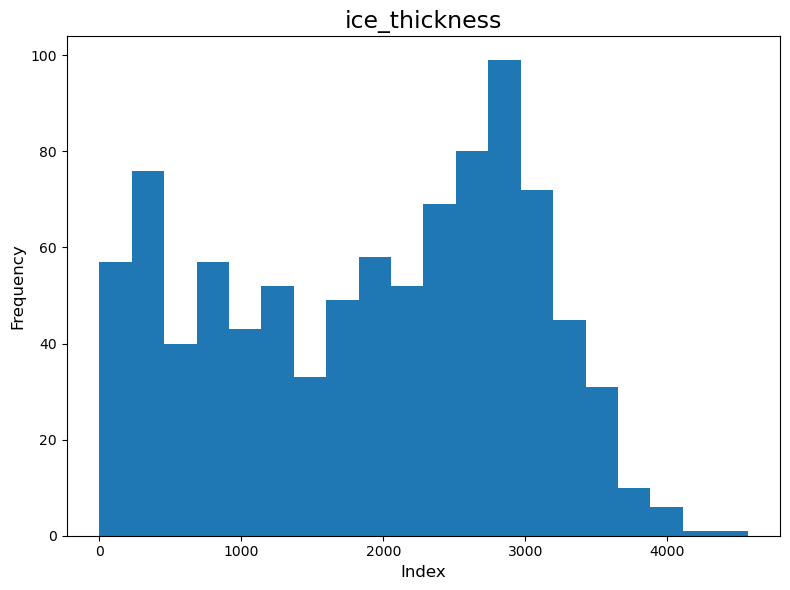
| precipitation air\_temp ocean\_temp ice\_velocity ice\_thickness  count 2257.000000 2257.000000 2257.000000 2257.000000 2257.000000  mean 492.847577 254.383848 272.871865 35.630226 785.948573  std 351.946762 17.431594 1.014827 201.272326 1165.304012  min -26.345100 214.175415 271.249695 -1.000000 0.000000  25% 148.012894 241.369202 272.128540 -1.000000 0.000000  50% 500.870361 263.044708 272.672302 -1.000000 0.000000  75% 792.505066 268.029938 273.321198 5.063869 1634.755371  max 2333.787109 275.543671 277.248840 3561.592041 4563.849609 |
| --- |

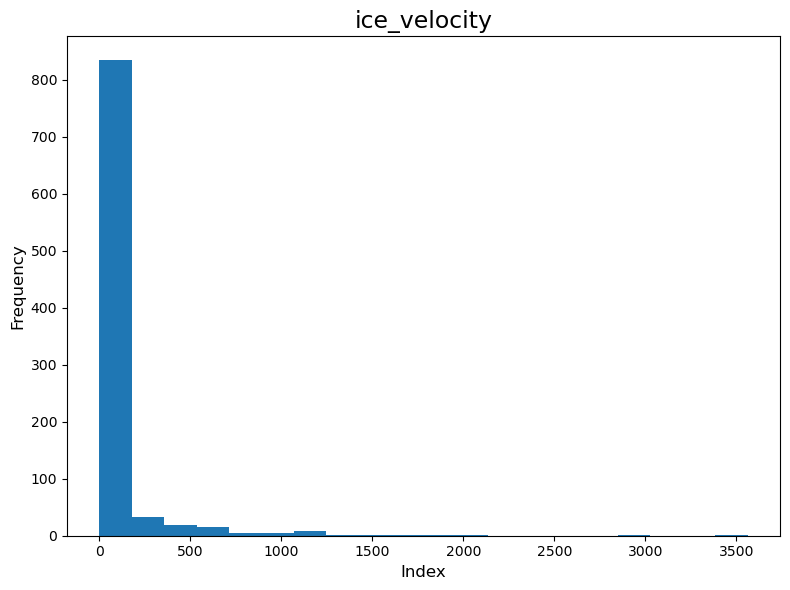
#### Distributions

The next thing to consider is the distribution of each variable. I have created a series of histograms, one for each variable. In the figure below, I notice a few things. The axis take on this rounded bell shape, which is due to the way I have ‘curved’ the edges of the instance space. Our categorical target ice\_mask is quite unbalanced, with only a small number of samples being in the floating ice category. This is something to consider in future. There is also a very small number of instances with ice\_mask = 0, which will need to be cleaned up. I also notice that the other two targets ice\_thickness and ice\_velocity are very skewed. 



When we remove instances with no ice, we can get a better sense of the distributions in the tail for ice\_velocity and ice\_thickness. I produced these graphs below by removing rows where ice\_velocity and ice\_mask had fill values of -1 and 0.0 respectively.





I notice that ice thickness has changed significantly, but ice velocity still has this skewed shape. It follows given the context of the data. Velocity refers to ice movement, which generally only happens at the very edge of the sheet. A large amount of ice is in the middle of the continent with little or no movement, so removing all the fill values of -1 does decrease the skewness towards 0 somewhat, but not entirely.

#### Scatter Plots

Scatter plots allow us to observe any correlations, clusters, or other patterns in the data. I plotted each of the continuous features against both continuous targets. I notice a quite strong negative correlation between ice thickness and air temperature. Ice velocity generally looks to be less strongly correlated with the features, with a large number of the points being to the far left of the graph. This makes sense as the fill value used here for ice velocity is -1, so many of the points there are areas of open ocean.

| Ice Velocity | Ice Thickness |
| --- | --- |
|  |  |
|  |  |
|  |  |

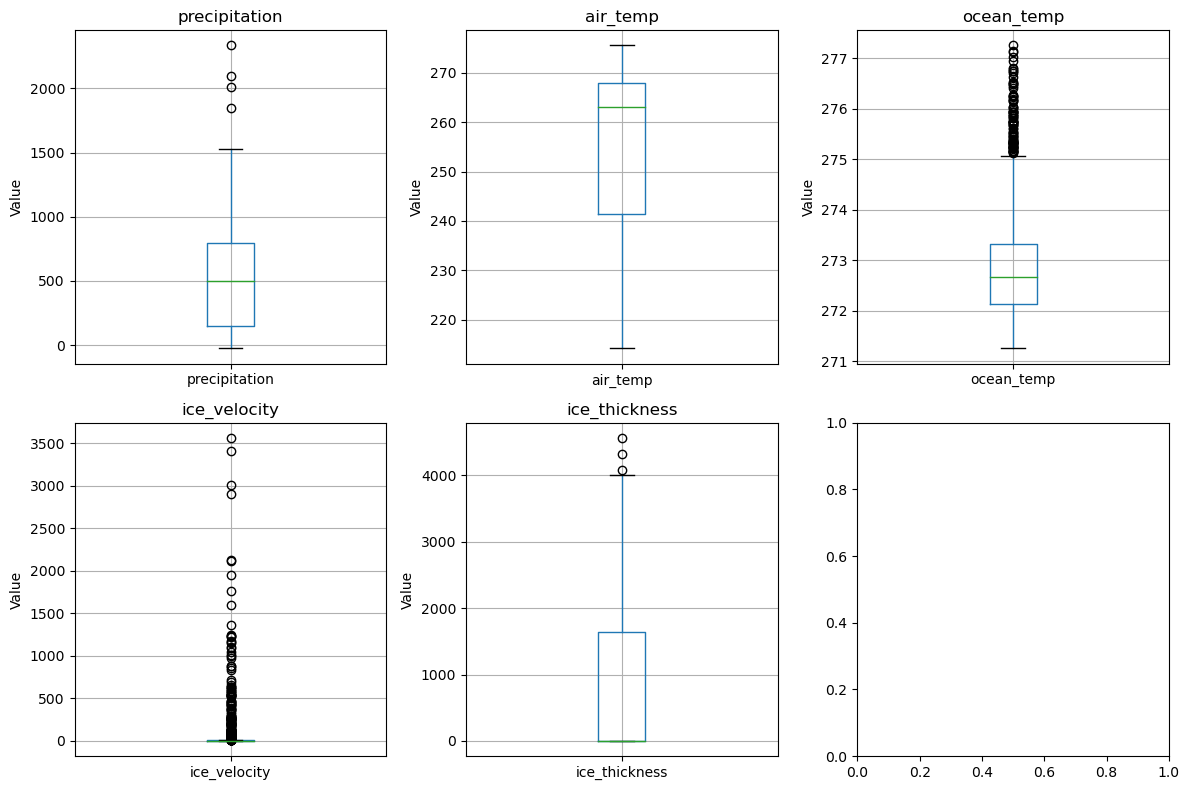
#### 

#### 

#### Outliers

I have created a series of box plots to examine any outliers in the data. I have left out the axis columns and categorical target ice\_mask. These have been generated from the ‘full’ data set, with only the corners where there were fill values for ocean\_temp being removed.

There are not too many outliers except for ocean temperature and ice velocity. I believe the large number of outliers and very small box (a line in this case!) in ice\_velocity tell us that the tail of the distribution is all or mostly being considered as outliers. This is a point to note, as those areas of higher velocity are the ones we are particularly interested in. We want to predict where the ice is moving, as this is a strong indicator for changes in the ice sheet.



#### Location/Mean

Because of the large amount of outliers in ocean temperature as well as the skewness in ice velocity and ice thickness, I decided that using a mean or weighted mean would not be an effective way to describe the location of each attribute. These metrics are sensitive to outliers, unlike the truncated or trimmed mean. I have left out the x and y axis, as well as ice mask from these metrics as I feel they are not important to include.

To calculate the trimmed mean, I discarded the top and bottom 10% of values (ie. if I had 100 values, the trimmed mean would be calculated with the middle 80). I chose 10% as it is a standard choice, and a higher 20% also. A larger amount of trimming can be useful in data that is very skewed, such as the ice velocity and ice thickness. I note that the mean changes very little between the three calculations for the four inputs.

| Attribute | Arithmetic Mean | Trimmed Mean (10%) | Trimmed Mean (20%) |
| --- | --- | --- | --- |
| Precipitation | 492.847577 | 479.180 | 482.701 |
| Air Temperature | 254.383848 | 256.235 | 258.268 |
| Ocean Temperature | 272.871865 | 272.748 | 272.68 |
| Ice Thickness | 785.948573 | 582.617 | 357.234 |
| Ice Velocity | 35.630226 | 2.237 | 0.429 |

#### Variability

To assess the variability in the data, I have calculated the range, variation, and standard deviation for the data. I have done this using both trimmed and untrimmed data. I used delta degrees of freedom = 1, for this, as this data represents a subset of the larger world. I could have used ddofm = 0 also, I am unsure which would be appropriate in this context.

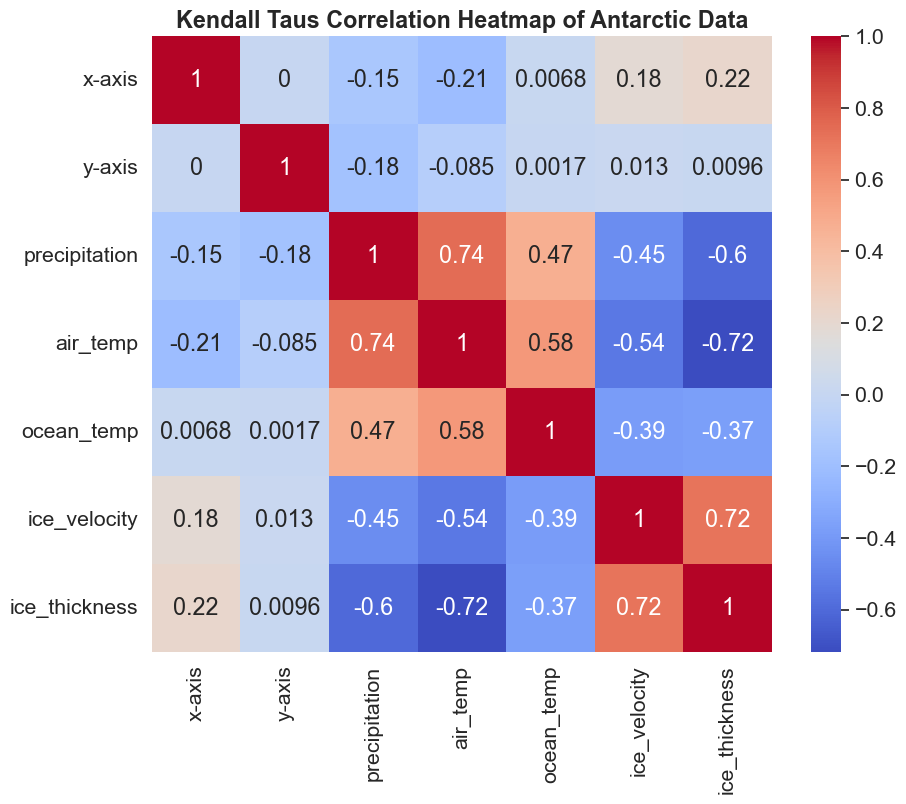
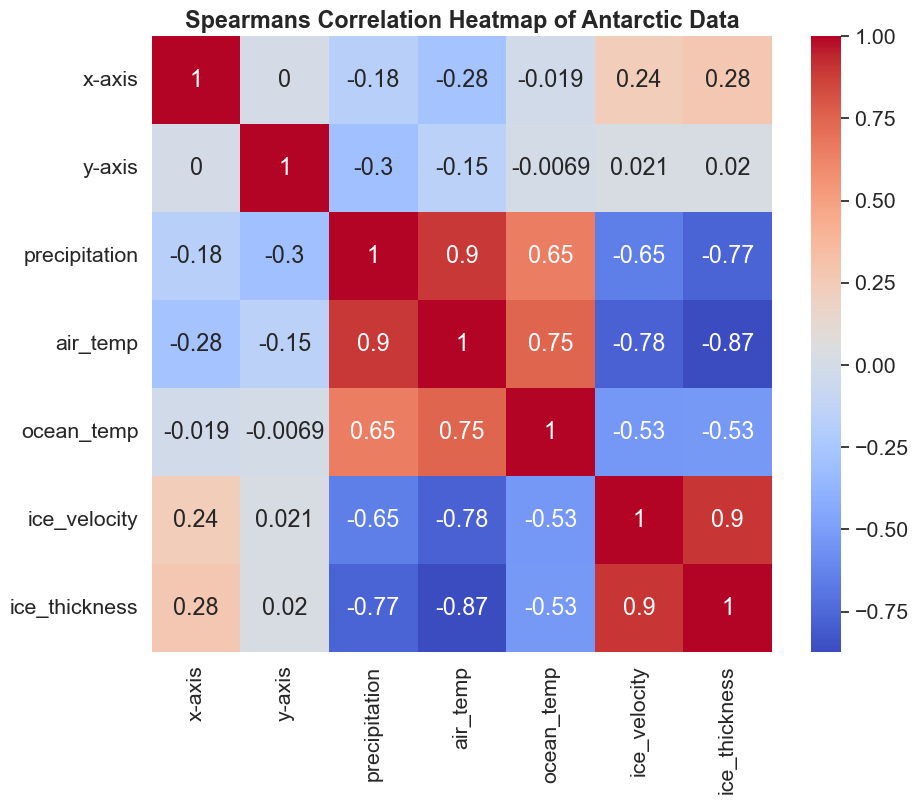
| Untrimmed Variability | | | |
| --- | --- | --- | --- |
| Attribute | Range | Variation | Standard Deviation |
| Precipitation | 2.360132e+03 | 1.238665e+05 | 3.519468e+02 |
| Air Temperature | 6.136826e+01 | 3.038605e+02 | 1.743159e+01 |
| Ocean Temperature | 5.999146e+00 | 1.029874e+00 | 1.014827e+00 |
| Ice Thickness | 4.563850e+03 | 1.357933e+06 | 1.165304e+03 |
| Ice Velocity | 3.562592e+03 | 4.051055e+04 | 2.012723e+02 |

| Trimmed Variability (10%) | | | |
| --- | --- | --- | --- |
| Attribute | Range | Variation | Standard Deviation |
| Precipitation | 8.871732e+02 | 8.167291e+04 | 2.857847e+02 |
| Air Temperature | 4.628976e+0 | 1.893425e+02 | 1.376018e+01 |
| Ocean Temperature | 2.533783e+00 | 3.933448e-01 | 6.271721e-01 |
| Ice Thickness | 2.847460e+03 | 8.378613e+05 | 9.153476e+02 |
| Ice Velocity | 3.009826e+01 | 3.682785e+01 | 6.068595e+00 |

#### Correlations

I have not included the categorical output of Ice Mask within these heatmaps, as it is not a *binary* categorical variable. I did not use Pearson's correlation, as this requires the assumption that the two data follow a normal distribution. I have instead prepared two heatmaps, one using Spearman's correlation, and one using Kendall’s Tau.

I notice that in both correlations, ice thickness and ice velocity are quite correlated. This could be a useful thing to remember when I look at this as a multimodal problem. In both correlations, the strongest indicator of the two targets is air temperature, closely followed by precipitation and finally by ocean temperature.



#### Grouped analysis for ice mask

Because of the categorical nature of this variable, I was unable to include ice mask in much of my previous EDA. I wanted to still do some further looking into this data, so I have included some basic statistics here.

As we saw earlier, there was a small number of outliers in this variable. I singled these out, and it turned out to be just two instances. I decided that it would be alright to delete them. This leaves us with three categories:

* 2 - grounded ice
* 3 - floating ice
* 4 - open ocean

| *Grounded Ice* |
| --- |
| precipitation air\_temp ocean\_temp ice\_velocity ice\_thickness  count 844.000000 844.000000 844.000000 844.000000 844.000000  mean 191.331420 234.940939 272.279628 48.032727 2055.660503  std 270.748128 11.786791 0.503733 227.533123 1011.773599  min -26.345100 214.175415 271.297577 0.000000 1.479372  25% 37.822948 224.678150 271.915398 2.196384 1231.778687  50% 90.779663 234.403671 272.174835 6.403733 2239.159546  75% 212.424507 244.352627 272.616768 19.404735 2874.668823  max 2333.787109 266.270020 274.844666 3006.586426 4563.849609 |

| *Floating Ice* |
| --- |
| precipitation air\_temp ocean\_temp ice\_velocity ice\_thickness  count 85.000000 85.000000 85.000000 85.000000 85.000000  mean 277.740004 247.108146 271.927351 484.774101 457.746579  std 257.746020 6.580246 0.542664 584.418766 351.509503  min 35.111038 239.800919 271.249695 3.151653 35.666943  25% 150.058350 242.287659 271.598999 156.430069 249.241241  50% 183.907974 244.056168 271.831787 296.031555 338.751648  75% 254.268616 250.595932 272.122681 574.336914 520.083252  max 1529.742310 263.922668 274.150208 3561.592041 2165.036377 |

| *Open Ocean* |
| --- |
| precipitation air\_temp ocean\_temp ice\_velocity ice\_thickness  count 1326.000000 1326.000000 1326.000000 1326.0 1326.0  mean 698.943443 267.215505 273.309355 -1.0 0.0  std 237.812288 4.146041 1.049344 0.0 0.0  min 72.892464 251.302094 271.427673 -1.0 0.0  25% 521.099731 264.815559 272.586060 -1.0 0.0  50% 746.732727 267.356384 273.034134 -1.0 0.0  75% 853.530899 270.134598 273.938400 -1.0 0.0  max 1849.135620 275.543671 277.248840 -1.0 0.0 |

### Casting a wide net

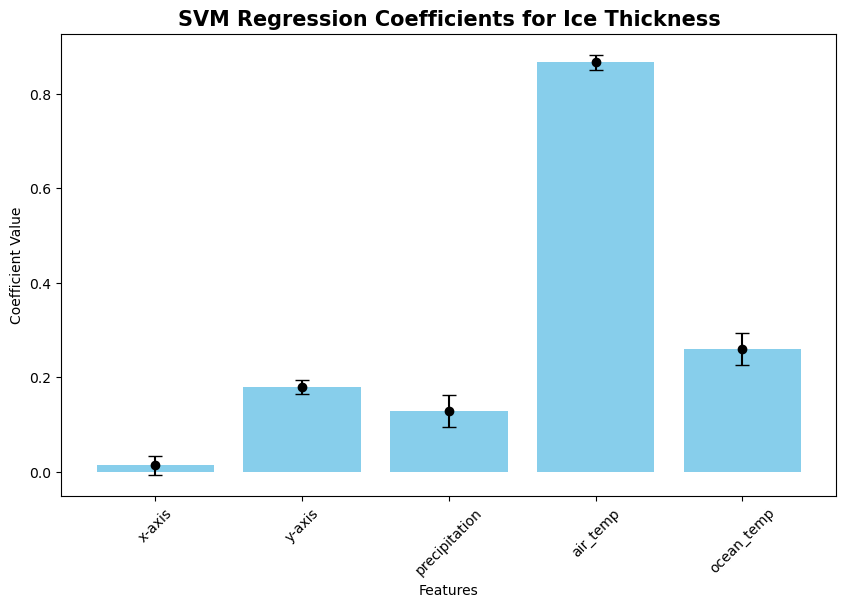
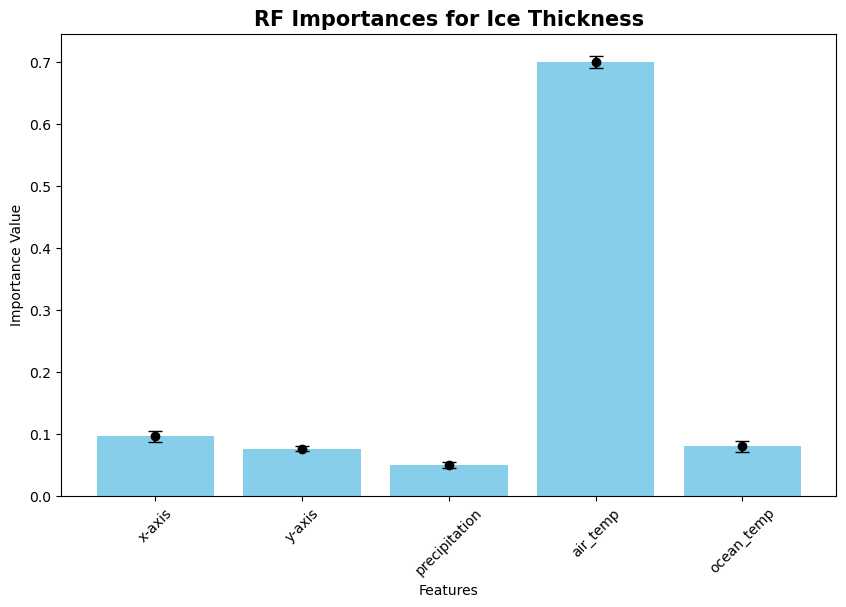
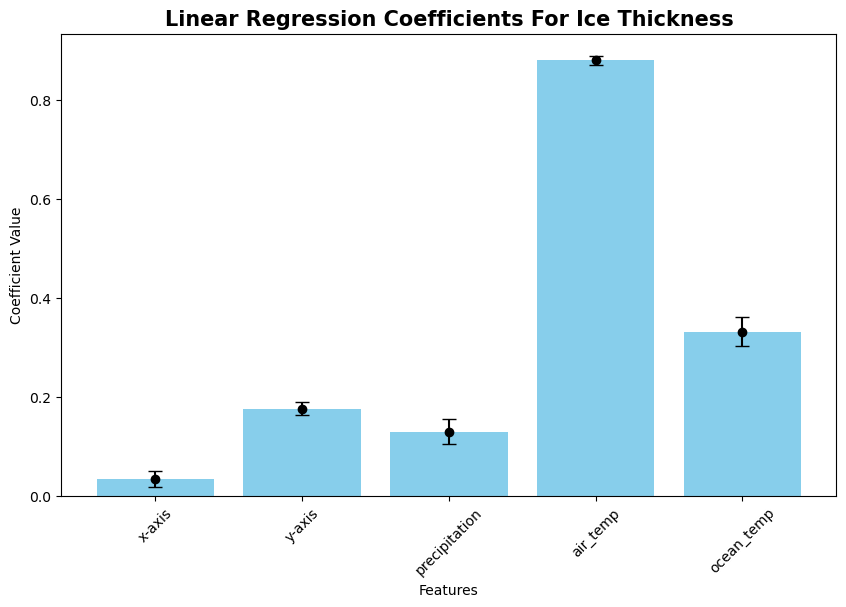
After this EDA, I experimented with a wide variety of models. I implemented each one using the standard settings in the scikit-learn library. I wanted to make sure I was putting further efforts into those with the most promise. For the targets ice thickness and ice velocity, these results are only looking at the area where there is ice - I removed all points with fill values. For each model, I ran it 30 times using 30 random seeds and splits of the data. The standard deviations in the tables below are reflective of the variation between these runs. The r-squared and RMSE values are the average from these runs.

#### Ice Thickness

The highest performing model for the ice thickness target was random forest. Using the r-squared as a measure, it outperformed Linear regression, as well as both implementations of SVM. Regarding this high performing model I notice that there is some drop in r-squared between the training and test sets. This could indicate some overfitting to the training data, and is something I investigate at a later stage.

|  | Training Results | | | | Test Results | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Average r^2 | r^2 Std.  Deviation | Average RMSE | RMSE Std. Deviation | Average r^2 | r^2  Std.  Deviation | Average RMSE | RMSE  Std.  Deviation |
| Linear Regression | 0.6711 | 0.0093 | 615.89 | 9.13 | 0.6716 | 0.0215 | 617.34 | 21.24 |
| Random Forest | 0.9781 | 0.0010 | 159.09 | 3.53 | 0.8459 | 0.0158 | 422.53 | 20.21 |
| Linear SVM Regression | 0.6687 | 0.0094 | 618.11 | 9.24 | 0.6687 | 0.0223 | 620.00 | 21.95 |
| Polynomial Kernel SVM | 0.7485 | 0.0060 | 538.61 | 8.20 | 0.7244 | 0.0184 | 565.64 | 22.02 |

Along with these r-squared and RMSE values in the table, I also produced some other results from these regressions. I wanted to understand how the different features were affecting the models. I did this to gain a more detailed understanding of what features were important in correctly predicting the targets. I graphed the coefficients of the linear regression and linear SVM, as well as the importances from the random forest. There was no real equivalent for polynomial SVM so I just left that one out. I still included the polynomial SVM in my table as I felt that if the problem were to be tackled with an SVM, it would probably not be a linear pattern. We can see that air temperature is consistently the most significant feature for ice thickness across the models. Again, there is not much deviation between runs with the small error bars indicating little change. These error bars are in agreement with the small deviations seen in r-squared and RMSE between runs on this data.

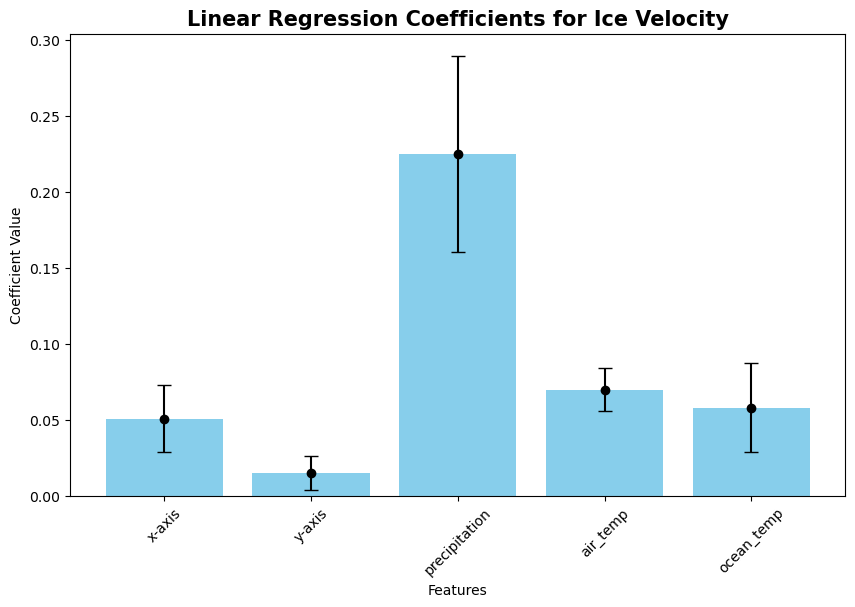
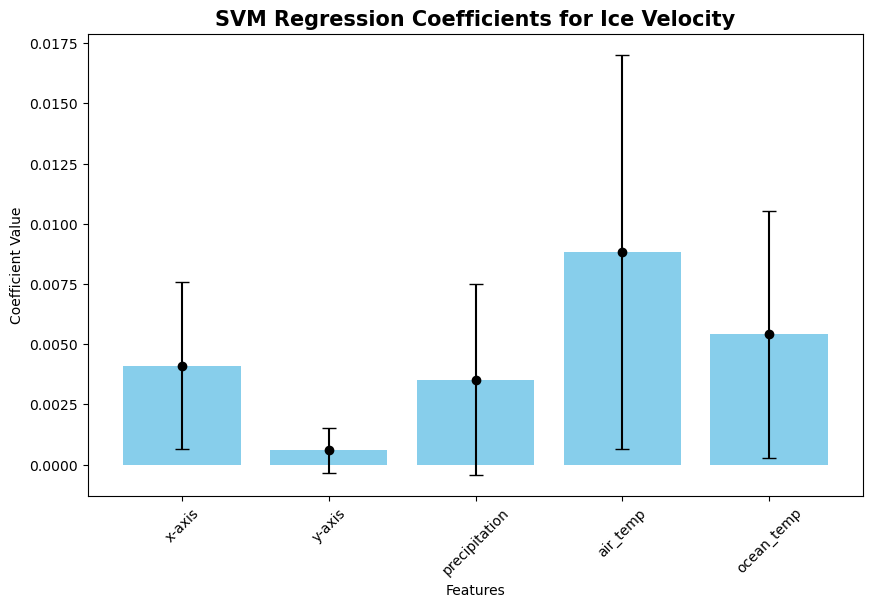


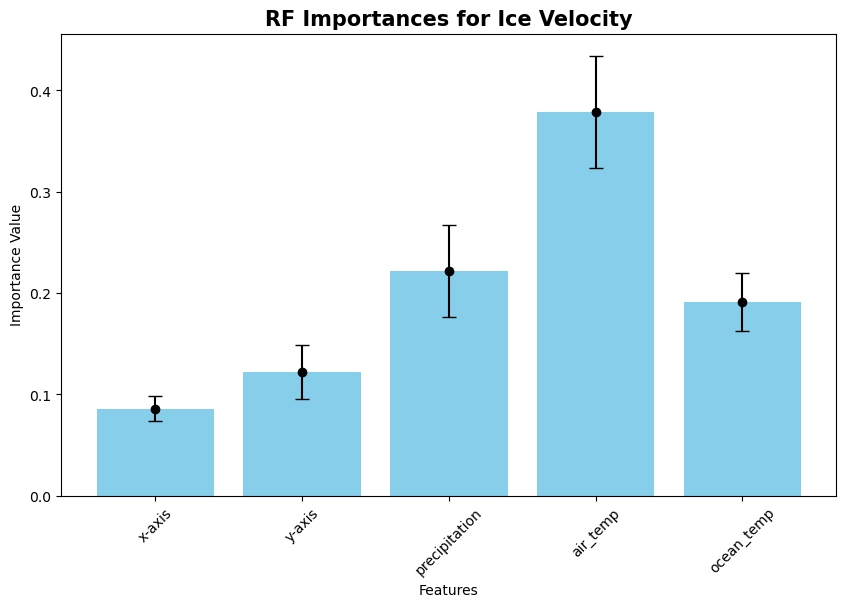
#### Ice Velocity

Moving onto ice velocity, we can see that the best performing model in the test set was linear regression. In saying that, the r-squared for the training set was still quite low at 0.17 (2d.p). Across all the models, we can see generally much higher deviations in r-squared when compared to ice thickness. This tells me that the performance of the models in predicting ice velocity is much more impacted by the particular data split. An interesting point is that the r-squared for random forest is quite high in the training set. In speaking with my supervisors, we felt that this was the model that required further investigation. Although the linear regressor performed the best, we felt that the underlying pattern of ice velocity was likely to be non-linear. We ultimately decided that it would be best to continue looking at random forest instead of linear regression for this target.

|  | Training Results | | | | Test Results | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Average r^2 | r^2 Std.  Deviation | Average RMSE | RMSE Std. Deviation | Average r^2 | r^2  Std.  Deviation | Average RMSE | RMSE  Std.  Deviation |
| Linear Regression | 0.1701 | 0.0249 | 277.41 | 21.73 | 0.1131 | 0.0735 | 282.74 | 49.17 |
| Random Forest | 0.8707 | 0.0136 | 109.24 | 8.75 | 0.0014 | 0.2240 | 295.44 | 42.79 |
| Linear SVM Regression | -0.7458 | 0.2235 | 399.88 | 13.80 | -0.8405 | 0.4562 | 398.45 | 36.79 |
| Polynomial Kernel SVM | 0.0837 | 0.1998 | 288.37 | 23.82 | -0.4455 | 0.4923 | 353.35 | 66.42 |

Looking at the coefficients, notice is how large the standard deviation bars are - particularly in the SVM regression. This makes sense, given the nature of SVM. In the linear regression, precipitation is the most significant feature for predicting ice velocity, whereas in the other two it is air temperature.





#### 

#### 

#### Ice Mask

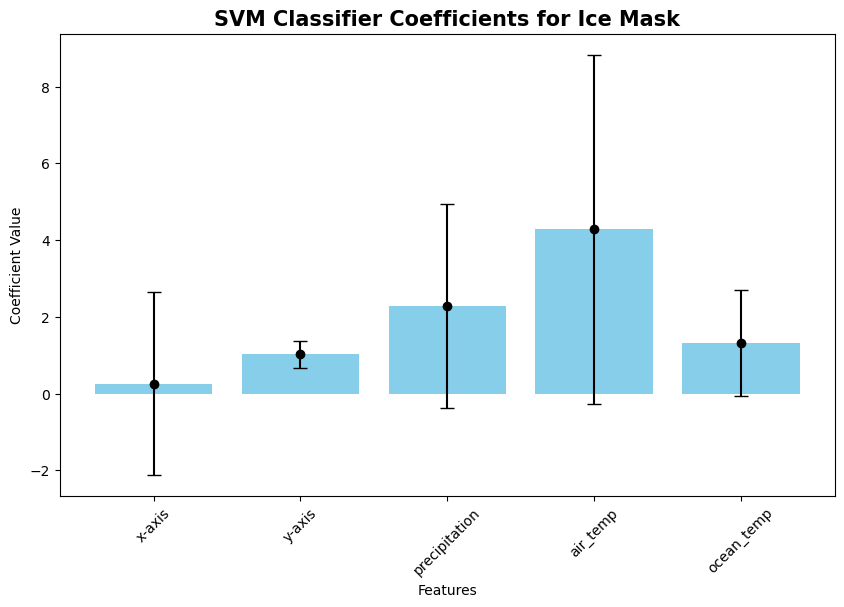
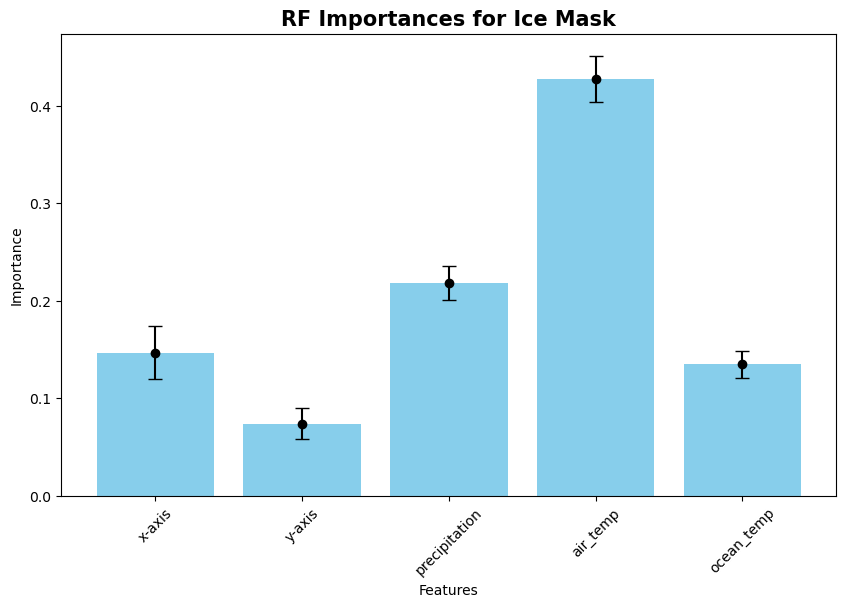
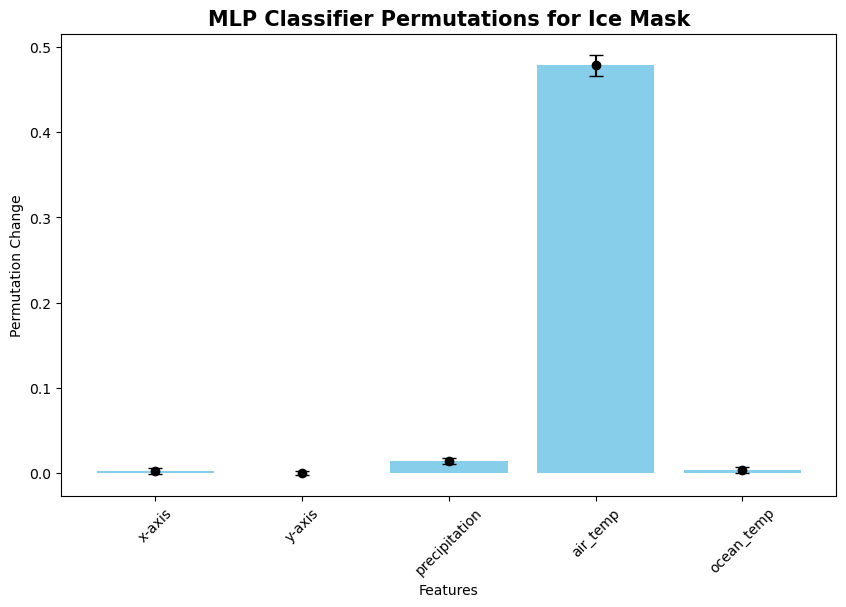
I'd like to start off this section by clarifying the metrics that I have used for ice mask. The values for the three categories are the class accuracies. They are calculated as the number of correct predictions for that category divided by the total number of predictions made for that category. The overall accuracy is not an average of these three values. This is not so obvious at this stage, but becomes clearer in the results later on in the report. Overall Accuracy measures the proportion of all correct predictions (both true positives and true negatives) out of all predictions made across all categories. It is calculated as the total number of correct predictions divided by the total number of predictions. This gives a single measure of how often the model is correct, regardless of class.

Looking at ice mask, we can see generally quite good performance across the classifiers. These implementations used mostly standard settings, with a few exceptions. I used a linear kernel for the SVM, again so that we could have a comparison of how each feature affected the predictions. I also included SVM with the parameter kernel = ‘poly’. This does not have an equivalent to coefficients or the like, so I have only included results from that model in the tables below. I felt it was important to include as I feel that given the context of the data, the pattern is probably nonlinear. I used the parameter class\_weight = ‘balanced’ in both the random forest classifier and the SVM classifier. I chose to do this, as floating ice is much less represented in the data. It has only 85 points in the floating ice category compared to grounded ice with 844 and open ocean with 1326. Floating ice is an important category to get right, as this is where we see the most change in the ice sheet. Areas around the edge where ice is extending over the water are the first to break off into the ocean as climate factors change. I would have implemented a control for the unbalanced nature of the data in the MLP classifier, but there was no real equivalent functionality in the method.

|  | Training Results (accuracies) | | | | Test Results (accuracies) | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Classifier | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy |
| Random Forest | 1.000 | 0.999 | 1.000 | 1.000 | 0.976 | 0.473 | 0.990 | 0.964 |
| SVM (linear kernel) | 0.644 | 0.918 | 0.987 | 0.823 | 0.644 | 0.860 | 0.987 | 0.820 |
| SVM (polynomial kernel) | 0.734 | 0.971 | 0.977 | 0.885 | 0.734 | 0.823 | 0.974 | 0.877 |
| MLP | 0.983 | 0.044 | 0.993 | 0.953 | 0.982 | 0.030 | 0.990 | 0.950 |

|  | Standard deviations of above training values | | | | Standard deviations of above test values | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Classifier | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy |
| Random Forest | 0.000 | 0.003 | 0.000 | 0.000 | 0.010 | 0.074 | 0.004 | 0.005 |
| SVM (linear kernel) | 0.016 | 0.030 | 0.006 | 0.028 | 0.034 | 0.073 | 0.007 | 0.024 |
| SVM (polynomial kernel) | 0.020 | 0.020 | 0.007 | 0.009 | 0.034 | 0.075 | 0.009 | 0.011 |
| MLP | 0.004 | 0.039 | 0.001 | 0.003 | 0.011 | 0.037 | 0.004 | 0.007 |

All three classifiers which I used to analyse feature impact showed that air temperature was the most significant feature in predicting ice mask. SVM had much larger bars, showing that this model is more susceptible to changes in the data. This makes sense considering the mechanics of how SVM works, using particular vectors to guide the process. From this first look at classifying ice mask, we decided to move forward focusing on random forest and SVM.



### Narrowing the focus and improving results

After looking at a wide variety of models, I moved on to improving those with the most promise. I tried a variety of methods relating to feature engineering and model tuning. In feature engineering, I first created specific features that I felt could be valuable based on the context of the data. The second approach I tried was more heavy handed. For each point, I included the features of its neighbouring points as features of that point. I also tried sequential modelling, which proved to have some interesting results. For model tuning, I used a grid search with three cross validations to search a space of hyperparameters that I specified.

#### Feature Engineering

The first aspect I looked into was feature engineering. The first method I tried was creating features based on the context of the problem. I developed specific features that I felt may be able to improve the model’s ability to predict accurately. Although random forest does have some built in feature engineering capability, I wanted to look at feature engineering anyway as I felt that it could still be beneficial.

For each of the features that I created, I will list its name in the dataset, how it was created, and why I created it. Hopefully this will give you some understanding as to why I created the features I did.

| Feature Name(s): | What: | Why: |
| --- | --- | --- |
| dist | The euclidean distance between each point and the centre of the coordinate system at (0,0). Calculated using the x and y coordinates. | I felt this could be a valuable feature to include as it directly tells us something about the point which could be otherwise hard to infer from the coordinate values. Given that the Antarctic ice sheet is centred around the pole, knowing how far a point is from the pole could provide valuable information on how much ice a point may have. |
| temp\_diff | This is simply the difference between the air temperature and the ocean temperature at a point. | I felt that this may aid in capturing some of the underlying relationships between these two temperature readings. These relationships and feedbacks between the air and ocean temperature could be important for predicting changes in the ice sheet. |
| air\_roll  ocean\_roll  precip\_roll | These features correspond to the air temperature, ocean temperature, and precipitation respectively. They are the rolling average across the grid of these original features. The rolling average of a point is calculated as the average of all 8 coordinate points surrounding it. In edge and corner cases, only the available points are used. For example, the corner point would be an average of itself plus the two touching it and the one diagonally adjacent. | The idea behind these features is that by smoothing out local irregularities in the data, we may be able to capture the overall trend. This could aid in the models generalisation to unseen data. |

I wanted to see how many neighbouring points would be best to use in these rolling average values, so conducted a small test before I continued. I performed a standard random forest regression on the ice thickness and ice velocity outputs, using an increasing window size. I carried out a similar experiment for ice mask, but instead used both SVM with a polynomial kernel and random forest. The data used in these experiments was the same as used previously, with the only change being the addition of the averaged features. At this stage, I was only looking to single out the effect of changing window size, so did not include all of the engineered features in this analysis. A window size of 1 would mean looking at only the first circle of points surrounding, 2 would mean looking at 2 neighbours out and so on. The values in the ice thickness and ice velocity tables are the average r-squared across 30 for each of the different window size datasets. I have not included the standard deviation for values here, as I felt it would bloat the tables too much - particularly the ice mask tables. I felt that the standard deviations were generally not very significant in my decision making. The standard deviations have been calculated however, and are available to run in the codebase at /week 7/Feature Engineering.ipynb.

**Ice Thickness**

|  | Window Size | | | | |
| --- | --- | --- | --- | --- | --- |
| Dataset | 1 | 2 | 3 | 4 | 5 |
| Training | 0.9780 | 0.9774 | 0.9771 | 0.9770 | 0.9771 |
| Testing | 0.8447 | 0.8424 | 0.8424 | 0.8425 | 0.8418 |

**Ice Velocity**

|  | Window Size | | | | |
| --- | --- | --- | --- | --- | --- |
| Dataset | 1 | 2 | 3 | 4 | 5 |
| Training | 0.8715 | 0.8746 | 0.8761 | 0.8721 | 0.8704 |
| Testing | 0.0506 | 0.0874 | 0.0973 | 0.0729 | 0.0438 |

**Ice Mask -> Random Forest Classifier**

|  | Training Results (accuracies) | | | | Test Results (accuracies) | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Window Size | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy |
| 1 | 1.000 | 1.000 | 1.000 | 1.000 | 0.979 | 0.468 | 0.993 | 0.970 |
| 2 | 1.000 | 1.000 | 1.000 | 1.000 | 0.977 | 0.465 | 0.993 | 0.970 |
| 3 | 1.000 | 0.999 | 1.000 | 1.000 | 0.977 | 0.469 | 0.993 | 0.970 |
| 4 | 1.000 | 1.000 | 1.000 | 1.000 | 0.976 | 0.470 | 0.993 | 0.969 |
| 5 | 1.000 | 1.000 | 1.000 | 1.000 | 0.976 | 0.467 | 0.993 | 0.969 |

**Ice Mask -> Polynomial SVM Classifier**

|  | Training Results (accuracies) | | | | Test Results (accuracies) | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Window Size | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy |
| 1 | 0.715 | 0.985 | 0.971 | 0.888 | 0.713 | 0.894 | 0.970 | 0.882 |
| 2 | 0.731 | 0.987 | 0.970 | 0.892 | 0.732 | 0.903 | 0.968 | 0.887 |
| 3 | 0.748 | 0.982 | 0.969 | 0.896 | 0.749 | 0.891 | 0.967 | 0.891 |
| 4 | 0.765 | 0.978 | 0.970 | 0.902 | 0.765 | 0.885 | 0.969 | 0.897 |
| 5 | 0.778 | 0.976 | 0.972 | 0.908 | 0.774 | 0.894 | 0.971 | 0.902 |

Looking at the above tables, I decided to use a window size of 1 for ice thickness, and of 3 for ice velocity. In the random forest classifier for ice mask, I noticed that the floating ice category was still scoring quite low comparatively. Since this is such an important category, I decided to continue work looking at only the polynomial SVM for this classification. I decided to use a window size of 5 here.

In the tables below are the results from using the engineered features. The data for these include all of the rolling averages (with the particular window size decided from above), along with the distance and temperature difference features as well. We can gain some understanding of how these engineered features are impacting performance by comparing them with the metrics from the original dataset.

Looking at ice thickness, I notice a slight drop in performance. In ice velocity, I see a good improvement in performance. The model is still overfitting quite a lot, and I wonder if this improvement comes from the smoothing effect of the rolling average features. In saying that, the model is still not performing well enough with an r-squared of just 0.0973. Ice mask shows an improvement generally, with the exception of a small decrease in accuracy for the open ocean category.

**Ice Thickness**

|  | Results on **Training Set** | | | | Results on **Test Set** | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Average r-squared | r-squared Std.  Deviation | Average RMSE | RMSE Std. Deviation | Average r-squared | r-squared  Std.  Deviation | Average RMSE | RMSE  Std.  Deviation |
| Original RF | 0.9781 | 0.0010 | 159.09 | 3.53 | 0.8459 | 0.0158 | 422.53 | 20.21 |
| RF with new Features | 0.9780 | 0.0009 | 159.16 | 3.37 | 0.8447 | 0.0166 | 424.08 | 21.02 |

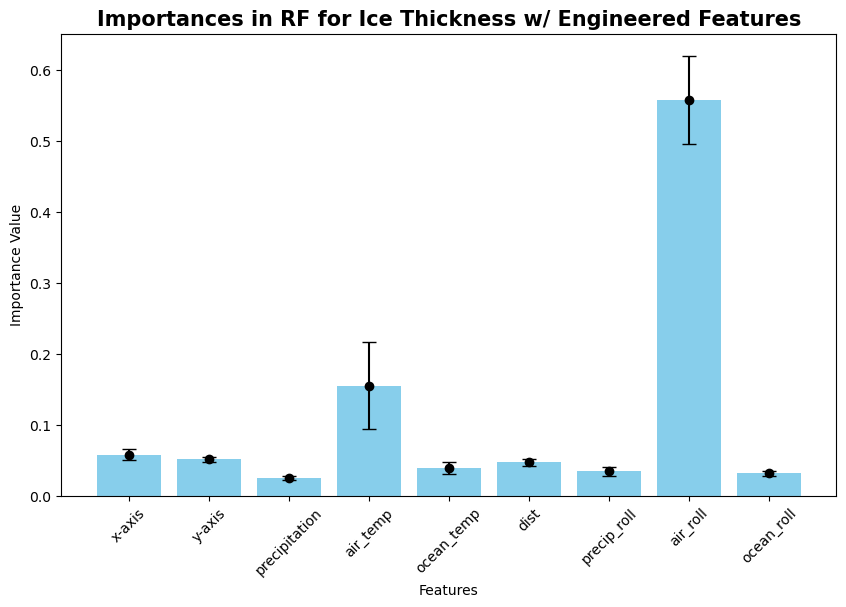
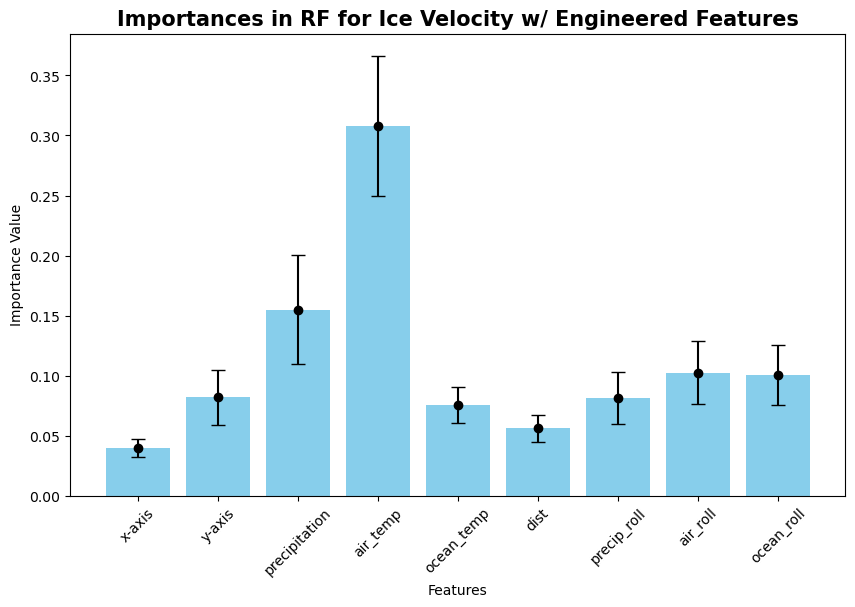
**Ice Velocity**

|  | Results on **Training Set** | | | | Results on **Test Set** | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Average r-squared | r-squared Std.  Deviation | Average RMSE | RMSE Std. Deviation | Average r-squared | r-squared  Std.  Deviation | Average RMSE | RMSE  Std.  Deviation |
| Original RF | 0.8707 | 0.0136 | 109.24 | 8.75 | 0.0014 | 0.2240 | 295.44 | 42.79 |
| RF with new Features | 0.8761 | 0.0125 | 106.87 | 7.53 | 0.0973 | 0.1930 | 281.92 | 45.74 |

**Ice Mask**

|  | Accuracies on **Training Set** | | | | Accuracies on **Test Set** | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dataset | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy |
| Original SVM | 0.734 | 0.971 | 0.977 | 0.885 | 0.734 | 0.823 | 0.974 | 0.877 |
| SVM with new features | 0.778 | 0.976 | 0.972 | 0.908 | 0.774 | 0.894 | 0.971 | 0.902 |

We can examine how these new features are impacting the model for ice thickness and ice velocity by visualising the feature importances. I notice that the air temperature rolling average is very important in the prediction for ice thickness, whereas the other engineered features are comparatively less so. In predicting ice velocity, the original air temperature is still the most important feature, however all of the engineered additions are not insignificant. This is encouraging, as it shows that the features are indeed useful.



#### 

#### 

Ultimately, I think these engineered features are having a positive impact on the predictive performance of the models. Ice thickness is the only one that does not show improvement with these new features. In saying that, the results are from models with standard hyperparameters. I wonder if comparing the results with tuned models would show something different.

#### Feature Engineering for Ice Velocity

Since the results for ice velocity were still quite poor, it was suggested that I try another method of feature engineering, focusing on ice velocity. This method involved adding in the features from neighbouring points as they were into each point.

For each point, I added the surrounding data in the format ‘feature\_type\_number’. The number denotes which of the surrounding points that feature belongs to. This numbering system started at the top left hand of a point, counting clockwise. For example, the air temperature of the point directly to the right of the current point would be ‘air\_temp\_4’. I have included a diagram below to better explain this system.

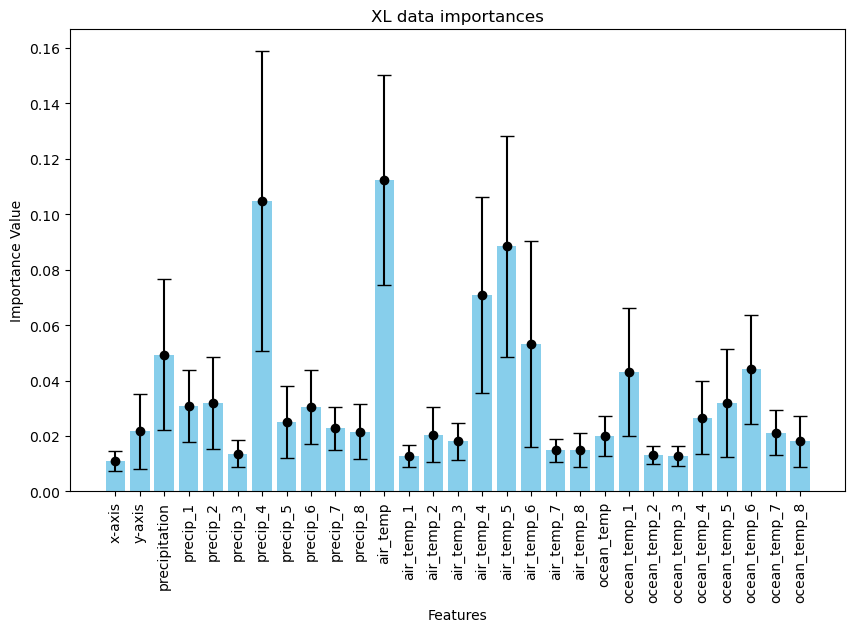


Once I had this dataframe worked out, I used it to perform a random forest regression using standard settings. Below are the results of this, with every value being the average across those 30 runs.

| **Standard** RF with surrounding features | Average r-squared | r-squared Std.  Deviation | Average RMSE | RMSE Std. Deviation |
| --- | --- | --- | --- | --- |
| Training | 0.8731 | 0.0162 | 108.16 | 9.29 |
| Test | 0.0697 | 0.2069 | 285.30 | 41.96 |

I plot the importances of the added features to get an idea of how they are impacting the final result.

I find it interesting that precip\_4, the precipitation of each cell directly to the right score so high in this. I assume this is related to natural irregularity in the data. That could perhaps be an interesting point to explore further.



I then performed a grid search to tune the hyperparameters for this new dataset. I did not do this grid search 30 times over as it took a very long time to run even once with the much larger dataset.

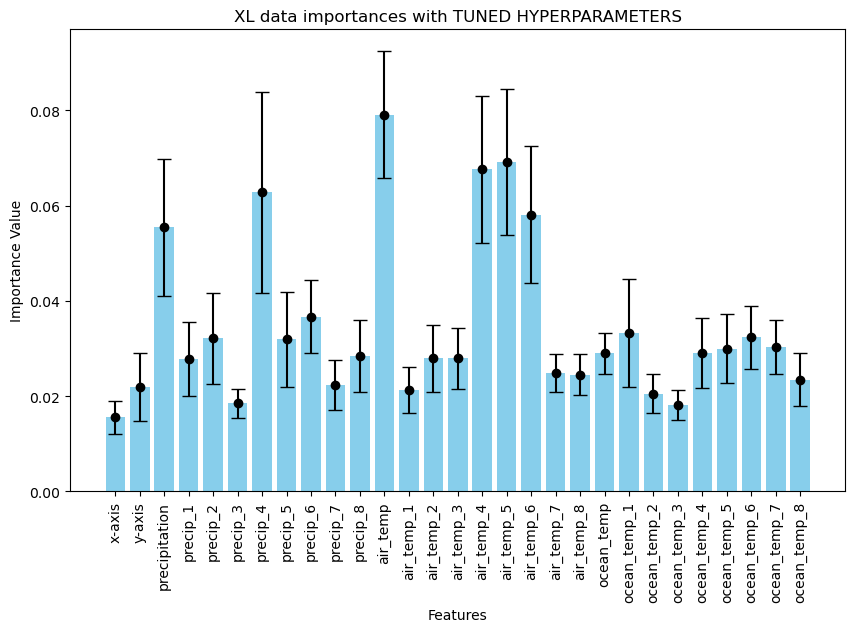
The search found the following hyperparameters for the best run. This gave the best model r-squared of 0.188 (3 d.p). This was encouraging, but does not tell the whole story. This is the r-squared for these parameters on a particular data split.

| Model | Settings |
| --- | --- |
| RF for Ice Velocity on XL dataset | n\_estimators = 200  max\_leaf\_nodes = 200  max\_depth = 30  min\_samples\_split = 2  min\_samples\_leaf = 1  max\_features = 6 |

To further investigate these hyperparameter settings, I ran the random forest 30 times again using these new settings. Below are the results from this run. We can see that this has improved the r-squared, however not to the extent that the tuning I carried out previously did. I think this is because I only did one grid search of the parameters and took the best one from that.

| **Tuned** RF with surrounding features | Average r-squared | r-squared Std.  Deviation | Average RMSE | RMSE Std. Deviation |
| --- | --- | --- | --- | --- |
| Training | 0.8845 | 0.0112 | 103.30 | 8.50 |
| Test | 0.1279 | 0.1698 | 277.30 | 42.72 |

I also plotted the importances for this tuned model to see any change there. I notice that the importance for each feature has generally increased when compared to the previous graph.



#### Model Tuning

The second aspect I looked at was model tuning. For this I looked only at the models using original data - for both tuned and untuned results. I chose to do this so I could get an idea of how significant model tuning was in terms of performance for this project. I tuned the models using a grid search method from the sklearn library - GridSearchCV. This allowed me to test a large variety of hyperparameter settings.

I first split the data into training and test sets. On the training set, I performed a grid search on all of the possible settings. Using three fold cross validation, I assessed each setting on that split of the data. After each setting had been assessed for a split, I recorded the most successful parameter settings in a dictionary. I have visualised these in bar charts - this will hopefully allow us to see what settings are successful, and any trends relating to each hyperparameter.

**Ice Thickness Tuning Results:**

|  |  |
| --- | --- |
|  |  |
|  |  |

After I did this analysis, my next step was to re-run the random forest with the best parameters above.

The settings were as follows:

n\_estimators = 300

max\_leaf\_nodes = 300

max\_depth = None

min\_samples\_split = 2

min\_samples\_leaf = 1

max\_features = 3

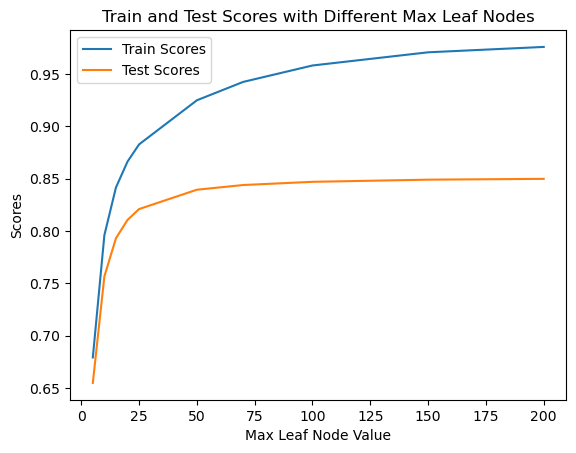
This gave me a **tuned model** with the following metrics:

| Training or Test Data | Average r-squared | r-squared Std. Deviation | Average RMSE | RMSE Std. Deviation |
| --- | --- | --- | --- | --- |
| Training | 0.9790 | 0.0009 | 155.70 | 3.21 |
| Test | 0.8503 | 0.0150 | 416.48 | 19.08 |

We can compare these metrics with the results from the **model without tuning**:

| Training or Test Data | Average r-squared | r-squared Std. Deviation | Average RMSE | RMSE Std. Deviation |
| --- | --- | --- | --- | --- |
| Training | 0.9781 | 0.0010 | 159.09 | 3.53 |
| Test | 0.8459 | 0.0158 | 422.53 | 20.21 |

There is a slight improvement in the r-squared of the test set when compared to the previous model which had the standard settings (0.8503, up from 0.8459). Although there is a small improvement, there is still some overfitting with the training set r-squared staying relatively similar with the previous one being 0.9781.



I wanted to investigate this overfitting further. I tried to do a similar gridsearch to look at every possibility, but I was unable to pass in the data I needed to assess the tradeoff between training and test results with the GridSearchCV method. I instead manually investigated this. I found the parameter that was able to reduce the gap between train and test results the most was max\_leaf\_node. I tried a variety of different settings, and have plotted them to the right.

Looking at this graph, we can see that the ‘value’ point of the number of leaf nodes is somewhere between 25-50.

**Ice Velocity Tuning Results:**

I also looked at tuning features for ice velocity. I started off by following a very similar process as above for ice thickness.

|  |  |
| --- | --- |
|  |  |
|  |  |

As with ice thickness, I then ran the ice velocity random forest model, using the best settings from the tuning.

The settings were as follows:

n\_estimators = 300

max\_leaf\_nodes = 300

min\_samples\_split = 2

min\_samples\_leaf = 1

max\_features = 1

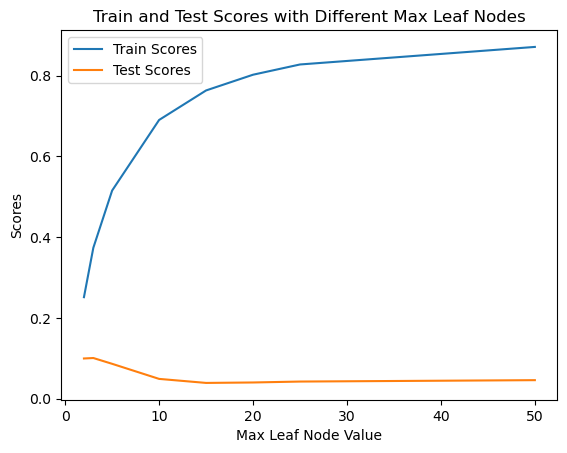
max\_depth = 20

This **tuned model** has the following metrics:

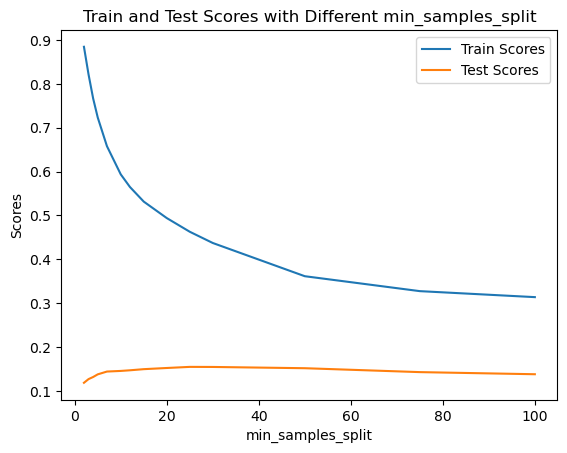
| Training or Test Data | Average r-squared | r-squared Std. Deviation | Average RMSE | RMSE Std. Deviation |
| --- | --- | --- | --- | --- |
| Training | 0.8824 | 0.0100 | 104.26 | 7.96 |
| Test | 0.1186 | 0.1315 | 280.06 | 44.83 |

We can compare these results to those of the **model without tuning**:

| Training or Test Data | Average r-squared | r-squared Std. Deviation | Average RMSE | RMSE Std. Deviation |
| --- | --- | --- | --- | --- |
| Training | 0.8707 | 0.0136 | 109.24 | 8.75 |
| Test | 0.0014 | 0.2240 | 295.44 | 42.79 |

Looking at these tables, we can see some difference between the two models. This is a much larger improvement in r-squared than I was able to achieve with the ice thickness model. In saying that, the model is still performing quite poorly with an r-squared of 0.12 (2 d.p) which would ideally be close to 1. The model is also still very overfit, with an r-squared of 0.88 (2 d.p) on the training set. 

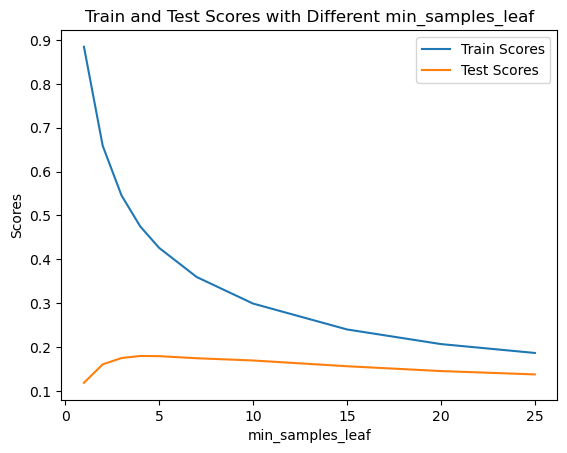
I wanted to investigate this overfitting more. The first thing I investigated was if reducing the max\_leaf\_nodes would have any impact on this overfitting. I created a similar graph to the one above for ice thickness. I used all the same parameter settings as the tuned model above, changing only max\_leaf\_node. We can see that the overfitting increases sharply beyond a very small number of nodes. Looking at this graph, I feel that the ideal max leaf node value is around **4**.



I also wanted to investigate further how changes in the other parameters would affect the model. Looking at min\_samples\_split, min\_samples\_leaf, and max\_features, we can see some evidence that increasing these values may be beneficial.

The min\_samples\_split parameter controls how many samples a node must have to be allowed to split.

Looking at the graph we can see that the ideal value for this parameter is around **25**.



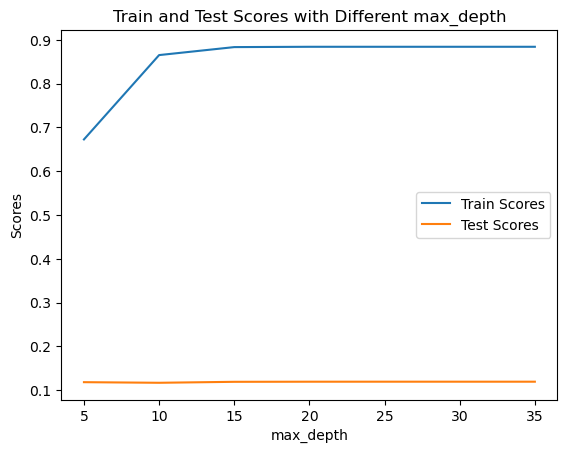
The min\_samples\_leaf parameter controls the minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min\_samples\_leaf training samples in each of the left and right branches. We can see the tradeoff between the overfitting and test set accuracy in the graph produced by modulating this parameter.

It seems to me that a beneficial range for this parameter would be around **5-10**.



The max\_features parameter controls how many features are used in each individual decision tree. The notion is that using a subset of the features in each tree will help mitigate overfitting, allowing the model to generalise to unseen data better.

Looking at the graph, we can see that changing this parameter does not seem to reduce the gap between training and test r-squared values as much as the other parameters above. In saying that, the r-squared on the test set does drop with increased max\_features. This graph shows that the best max\_features value is **1**.

Although the max\_depth parameter was already set at 20 (as opposed to None), I felt that this could still be investigated as a potential cause of overfitting. 

Looking at the graph, I see that even a maximum depth of 20 may have been causing some of the overfitting issues. I feel that a more ideal value would be **5**.

After this investigation into overfitting, I wanted to look at the metrics of a model whose parameters were seen to be the most optimal for reducing overfitting.

The settings were as follows:

n\_estimators = 300

max\_leaf\_nodes = 4

min\_samples\_split = 25

min\_samples\_leaf = 10

max\_features = 1

max\_depth = 5

This **model tuned to avoid overfitting** has the following metrics:

| Training or Test Data | Average r-squared | r-squared Std. Deviation | Average RMSE | RMSE Std. Deviation |
| --- | --- | --- | --- | --- |
| Training | 0.2366 | 0.0216 | 266.06 | 20.54 |
| Test | 0.1403 | 0.0336 | 279.15 | 51.19 |

We can compare these results to those of the **model with initial tuning**:

| Training or Test Data | Average r-squared | r-squared Std. Deviation | Average RMSE | RMSE Std. Deviation |
| --- | --- | --- | --- | --- |
| Training | 0.8824 | 0.0100 | 104.26 | 7.96 |
| Test | 0.1186 | 0.1315 | 280.06 | 44.83 |

Looking at these tables, we can see that this model has improved both in how overfit it is, and in the test r-squared. This is an encouraging sign that the model with these newly tuned parameters is more effective at capturing the underlying patterns in the data. In saying that, I feel that an improvement in the test results could be achieved by using slightly less stringent parameters for overfitting. I will explain my choices for each parameter change in the following table.

| Parameter name | Previous value | New value | Description of change |
| --- | --- | --- | --- |
| max\_leaf\_node | 4 | 10 | Looking at the graph, we see a decline of the test r-squared between 4 and 10. I am unsure why this was the case, and it interested me. I experimented with the settings and found that 10 achieved a good balance between test results and overfitting. |
| min\_samples\_split | 25 | 25 | I decided to leave this value as it was. In the output of my code, there is a (very) small peak in the test r-squared at 25. I experimented with 20 and found that this did not improve test results and increased the training r-squared. |
| min\_samples\_leaf | 10 | 5 | Looking at the graph, we can see a slight increase in the test r-squared from 5-10. |
| max\_features | 1 | 1 | I decided to leave this as it is, since there seems to be no benefit to increasing it looking at the graph. |
| max\_depth | 5 | 10 | I felt that increasing the maximum depth may allow the detail of the data to be captured more by the model. I felt that 10 was a good middle ground between 5 from the model tuned for overfitting and 20 from the previous analysis. |

With the above settings, the metrics for the **final tuned model** are as follows:

| Training or Test Data | Average r-squared | r-squared Std. Deviation | Average RMSE | RMSE Std. Deviation |
| --- | --- | --- | --- | --- |
| Training | 0.3466 | 0.0326 | 246.06 | 19.36 |
| Test | 0.1744 | 0.0486 | 273.24 | 49.04 |

This seems to be the model that performs the best, with a higher r-squared value than the underfit, overfit, and untuned model. Although there is a large improvement in performance with this model, it is still not as effective as I would like it to be.

**Ice Mask Tuning**

I followed a similar process for tuning the ice mask model as I had done previously for ice mask and ice velocity. I performed a grid search on 30 different data splits, finding the best performing hyperparameters for each split. Below are the graphs showing the distribution for each parameter from those 30 runs.

|  |  |
| --- | --- |
|  |  |

I then ran the SVC model 30 times again, with these new optimised parameters. Below are the results, compared with the original untuned SVM. In the first table I have accuracies, while in the second I have the associated standard deviations. We can see that this tuned model is generally performing much better, with a jump of overall accuracy from 0.877 to 0.944. In saying that, the r-squared for accuracy for floating ice in this tuned model is much worse. It seems that this class, being that it is the hardest to predict, has a tradeoff associated with it.

|  | Accuracies on **Training Set** | | | | Accuracies on **Test Set** | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model Type | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy |
| **Untuned** SVM | 0.734 | 0.971 | 0.977 | 0.885 | 0.734 | 0.823 | 0.974 | 0.877 |
| **Tuned** SVM | 0.961 | 0.875 | 1.000 | 0.980 | 0.924 | 0.602 | 0.978 | 0.944 |

|  | Standard deviations for above training values | | | | Standard deviations for above test values | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model Type | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy | Grounded Ice | Floating Ice | Open Ocean | Overall Accuracy |
| **Untuned** SVM | 0.020 | 0.020 | 0.007 | 0.009 | 0.034 | 0.075 | 0.009 | 0.011 |
| **Tuned** SVM | 0.008 | 0.041 | 0.001 | 0.004 | 0.016 | 0.092 | 0.007 | 0.005 |

In SVM models, ‘C’ represents the regularisation parameter. This is a way to control model complexity and overfitting. Higher C values tell the model to prioritise fitting the training data. It will try to correctly classify all training examples, even if it has to draw a very complex decision boundary. A lower C value will encourage the model to find a simpler decision boundary, meaning that some training points may be misclassified. A more simple model may generalise to unseen instances better, leading to better performance on test data.

I see some evidence of overfitting in this tuned model, particularly in the floating ice category. I wanted to try running the SVM with most of the same tuned parameters, only changing C. Below are the results from this model. I used the default setting, C = 1.0.

| SVM - tuned but with lower C value | Category Accuracies | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Grounded Ice | Grounded  Std. | Floating Ice | Floating  Std. | Open Ocean | Ocean  Std. | Overall Accuracy | Overall  Std. |
| Train | 0.911 | 0.012 | 0.999 | 0.004 | 0.993 | 0.003 | 0.962 | 0.005 |
| Test | 0.889 | 0.020 | 0.748 | 0.081 | 0.982 | 0.007 | 0.939 | 0.009 |

Although the overall accuracy is a little lower at 0.939 in comparison to the previous tuning, I feel that this is an acceptable tradeoff for the higher floating ice accuracy of 0.748 compared to 0.602.

#### Sequential Modelling

An idea I had to improve the performance of ice velocity was to first predict ice mask, then use that predicted value as a feature for ice velocity. This seems logical to me, as ice movement is more significant around the border of the ice sheet where it interfaces with the ocean. I suspected that the floating ice category would be a good predictor of ice velocity. I ensured no data leak happened during this process by being sure to not include ice velocity as a feature when predicting ice mask in the first instance. I wanted to ensure that the ice mask predictions would be as accurate as possible, to bolster the performance of this feature. I first decided to re-tune the model using the data set which contained the additional engineered features. Below are the resulting graphs from that tuning.

|  |  |
| --- | --- |
|  |  |

I then ran the model again, using the most popular hyperparameters from the tuning. For the parameter coef0 1, 10, and 15 were all tied at the best on 8. I chose 10 for this as it seemed like a reasonable middle ground. This gave me the accuracies in the table below.

| SVM - tuned with engineered features | Category Accuracies | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Grounded Ice | Grounded  Std. | Floating Ice | Floating  Std. | Open Ocean | Ocean  Std. | Overall Accuracy | Overall  Std. |
| Train | 0.976 | 0.009 | 0.898 | 0.035 | 1.000 | 0.000 | 0.987 | 0.004 |
| Test | 0.936 | 0.014 | 0.582 | 0.091 | 0.977 | 0.008 | 0.947 | 0.007 |

Again, I felt that the model was overfitting slightly with this high C value. I reran the model with the same hyperparameters, with the exception of C which I changed back to the default of 1.0. This gave the following results.

| SVM - tuned with with  C = 1.0, with engineered features | Category Accuracies | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Grounded Ice | Grounded  Std. | Floating Ice | Floating  Std. | Open Ocean | Ocean  Std. | Overall Accuracy | Overall  Std. |
| Train | 0.957 | 0.008 | 1.000 | 0.000 | 0.999 | 0.001 | 0.983 | 0.003 |
| Test | 0.920 | 0.015 | 0.669 | 0.091 | 0.980 | 0.008 | 0.946 | 0.007 |

I felt that the slight drop in overall accuracy was worth the increase in floating ice accuracy. In the following sequential modelling, this is the model I used to predict ice mask.

I have prepared results from the same random forest model with standard settings, using four slightly different data sets. In my analysis for predicting ice velocity so far, I have only looked at the results for predicting where there aren't fill values for that target - so only looking at areas where there is ice. In my very first regression I did for this project, I predicted ice velocity and thickness for across the whole domain. I used velocity = -1 and thickness = 0.0 in the areas of open ocean. The results were so poor, that we decided to only focus on predicting these values within the ice sheet. With this ice mask prediction covering the entire domain, I wanted to go back to predicting velocity across the whole domain to further my understanding of how effective or ineffective this feature was. Below is a table explaining the four different data sets - the titles are quite cryptic to avoid the long descriptions in the results table.

| **Trimmed OFf**  This is the Original Features, looking only at points where there is ice only. I titled these ones as ‘trimmed’ since we are trimming away any open ocean points. The original features are x-axis, y-axis, precipitation, ocean temperature and air temperature. | **Trimmed OF+PIM**  This is the Original Features, plus the Predicted Ice Mask value. Similarly to the data set to the left, this one is only using the instances where there is ice. |
| --- | --- |
| **Whole Domain OF**  This is the Original Features, looking at the whole domain. The whole domain in this case is not the full square, but the trimmed ‘squircle’ created by removing instances with a fill ocean temperature. | **Whole Domain OF+PIM**  This is the Original Features, plus the Predicted Ice Mask value. Again, this data set contains all of the points including those which are open ocean. Similarly to the dataset to the left, this one has excluded those points with fill values for ocean temperature, giving a rounded square kind of shape. |

|  | Training Results | | | | Test Results | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Data type | Average r^2 | r^2 Std.  Deviation | Average RMSE | RMSE Std. Deviation | Average r^2 | r^2  Std.  Deviation | Average RMSE | RMSE  Std.  Deviation |
| **Trimmed OF** | 0.8707 | 0.0136 | 109.24 | 8.75 | 0.0014 | 0.2240 | 295.44 | 42.79 |
| **Whole Domain OF** | 0.8466 | 0.0128 | 79.94 | 7.30 | -0.1711 | 0.2552 | 198.79 | 32.56 |
| **Trimmed OF+PIM** | 0.8756 | 0.0146 | 105.32 | 12.21 | 0.1394 | 0.1741 | 281.51 | 55.68 |
| **Whole Domain OF+PIM** | 0.8683 | 0.0135 | 73.05 | 7.73 | 0.0129 | 0.2193 | 191.41 | 33.04 |

These results are quite interesting. If we were using only RMSE as a metric, the clear winner would be to use the whole domain plus the predicted ice mask. If we were using only r-squared, the winner would be the trimmed domain plus predicted ice mask. Another aspect that cannot be ignored with the r-squared metrics is their standard deviations. Across the board, they are very high. This again indicates that the ice velocity prediction shifts greatly depending on the particular train test split. Either way, it seems that including ice mask is having a positive effect on the results. I wonder if the low r-squared in the two models that use the whole domain is due to the -1s used in areas of open ocean. I am unsure, this could be something interesting to investigate further. It would be good to know if these r-squared values are really reflective of the model's performance or not, as they conflict with the RMSE results.

### Genetic Programming

Another method I explored for tackling this problem was genetic programming. I used the dataset of points only where there was ice for this GP section of the report. I did not normalise the data in this case. I used an ephemeral random constant generated randomly between -1 and 1. I implemented a genetic program using the DEAP library, with the following settings:

Population size = 512

Generations = 50

Crossover = 0.8

Mutation = 0.2

Number of elites = 10 (around 2% of total population)

For the primitives, I used the following operators:

* Add
* Multiply
* Subtract
* Protected division
* Protected log
* Negation (eg. 12 -> -12)
* Absolute (eg. 12 -> 12, -12 -> 12)
* Tan
* Sin
* Cos

I decided to include the primitives of tan, sin, and cos, as I felt the underlying pattern was likely to not be linear. I used the basic operators of add, multiply, subtract and protected division - these are the standard operator set. Protected division is used to avoid the possibility of dividing by zero within a tree/individual. I included negation and the absolute functions largely because of the x–axis and y-axis coordinate features. I wanted there to be options for calculating the distance of a point from the pole. I included a protected log to enable some direct dynamic range compression. The variables can cover a large range of values - eg. precipitation has a maximum value of 2333.78 (2 d.p) and a minimum of -26.34 (2 d.p), while air temp fluctuates between 214.17 (2 d.p) and 275.54 (2 d.p).

I defined the fitness function of an individual as the RMSE on the training set. I set my program so that the goal would be to minimise this fitness. I wanted to get a fair comparison to the other methods I had used so far, so went back to using the trimmed domain. Below are the results for predicting ice thickness and ice velocity with GP. I have included a graph of the best found individual for each, as well as the associated RMSE metrics.

| Ice Thickness Tree | Ice Velocity Tree |
| --- | --- |
|  |  |

The average RMSE is the average fitness of the best individual from each of the 30 runs of the GP. The standard deviation is how much change in this number we see between runs. The best RMSE is the best recorded fitness ever. The final value for each of the targets is the RMSE on the test set. This is how well the individual above, the best individual found, performs on the test data that was separated out at the beginning.

| Ice Thickness | | | | Ice Velocity | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Average RMSE | RMSE Std. Deviation | Best RMSE | RMSE on test set | Average RMSE | RMSE Std. Deviation | Best Fitness | RMSE on test set |
| 874.09 | 69.70 | 672.61 | 695.90 | 309.02 | 10.76 | 293.30 | 224.45 |

Looking at the table above, I notice that the RMSE for ice velocity is generally quite good. The average best individual across 30 runs is 309.02, with a standard deviation of 10.76. I find it interesting that the RMSE for the test set is much lower than the best fitness found overall. This indicates to me that the solution found could be quite sensitive to the particular set of points used in the test set. I am surprised to see that the GP is performing quite poorly on ice thickness. Up until this point, ice thickness predictions have always been better than those for ice velocity.

### Conclusions and future work

The focus of my work for this project has been quite broad. Since this is a new project, and not something being carried on from previous years, I felt it was important to work in quite an exploratory manner. This is the start of a much larger possible body of work. I did not want to narrow my focus too much in these early stages, so there are many avenues that could and should be explored further.

This was my first time doing a project of this scale. I had a lot to learn along the way in a lot of areas, so I feel that I didn't cover as much ground as I would have liked. As a result, there are many ideas I have for how to continue this work.

One area that I think could use a lot of work is sequential modelling. The results for this are quite promising so far. With three targets to predict on the same dataset of features, there are several options for the order in which each one is predicted. In the correlation heatmaps, we can see that ice thickness is the most highly correlated with ice velocity. One option I would try would be predicting ice velocity using a predicted ice thickness feature. I would also try sequentially modelling all three targets in the order or ice mask, ice thickness, then ice velocity. This may provide a better result than predicting just thickness followed by velocity, however it would also be more computationally expensive.

Currently, the sequential modelling uses engineered features and model tuning in the first stage for predicting ice mask, but does not continue that trend for ice velocity. Using engineered features and carrying out model tuning for each following stage could further improve results. I also wonder if this sequential predicting style could be used with GP. Instead of using SVM or random forest to produce the predicted features, I wonder if the best individual from a GP could be used instead.

So far, I have only had some initial results using GP. I would like to expand the code further in several ways. The first and most obvious would be to expand to predicting ice mask as well. Adding an r-squared metric as a fitness function would be a valuable step. I’d also like to implement a more in depth analysis of the GP program. Currently, the GP runs 30 times on one train test split. It's hard to know if changes in the parameters like crossover or mutation are actually having a significant impact on the final individual and its associated RMSE, or if any change seen is just associated with randomness in the data and GP process. I'd like to extend the program to automatically run 30 times on 30 different data splits, recording the performance/metrics of the best individual for each run. Taking the average of these values and looking at the standard deviation would give a much clearer picture of how changes being made to the GP are impacting its behaviour.

Another area which could have potential improvements is in feature engineering. There could be additional features created which I am not considering. The ones I have implemented are only the ones that I could think of, and I wonder if there are other better ones that I haven’t considered. It may be valuable to discuss this with some domain experts.

Additionally, it could be worth expanding on the second feature engineering technique I tried. This was the one where I added in features from the surrounding points into each point. I only implemented it for ice velocity, as this was the hardest target to predict. It could be a good technique for improving performance in the other two targets also. This feature engineering method could also be layered with the sequential modelling, adding in the predicted target value(s) of surrounding points as well. I am also curious about if adding in features from the surrounding two rows of points would be beneficial. This would create a huge number of features for each point (77 I believe, if using just the original data), so perhaps some feature selection could be done here.

Further down the track still, I would like to expand to predict across time intervals. This would be a much closer step to the goal of the work. Climate models predict how the climate will change over time periods, so it follows that we would want to do the same here. Before doing this, I think it would be wise to improve the prediction for just one time slice first. Even small errors in prediction would be magnified when used as input across many iterations.

Thinking about additional time slices, it would be worth first extending the EDA to reflect this. I am not quite sure of the details of how it would be best done. I did have one idea for EDA over time on the correlations. You could start by calculating the correlations between variables for each time slice. Then, plot these correlations as a line graph with correlation for the y axis and time for the x axis, with each correlation being a line of a different colour. This could reveal changes in how tightly correlated the variables are over time, and influence decisions about feature engineering.

Another idea I had was regarding the metrics used to understand the efficacy of models. It could be interesting to look into plotting a heat map of each point, using the error. These images may tell us more about areas that are not being predicted as well or areas that are easier to predict. I wonder if these images could be presented to domain experts for their input. Another area that would be good to consult experts on is regarding the ocean temperature variable. I decided early on in the work that it would be worth removing all points where there was a fill value for ocean temperature. Bach has suggested that we may lose some information doing this. I agree with him that it would be worth consulting Nicholas or another domain expert regarding that decision.

### Code details

The repository for this project has very much been a workspace, it is not a finalised or polished codebase. Each week I created a new folder to contain what I was working on. There is a lot of repeated code, as I put more effort into ensuring my work was correct and readable over the code being compact or reusable. In saying that, you should be able to replicate all of the values and graphs from this report using the code. For each table or graph, I will name the file that you can use to reproduce those numbers or images. In some parts of the report, I reuse results previously seen, so that we can compare them to the new findings. In those cases, I will not list where to find the previously seen results. Another important general note, in my code when I print out ‘score’, I am referring to the sci-kit learn model.score function, which gives the r-squared of a model.

EDA

The first images in the preprocessing subheading of this section can be found in AIS\_data/IO-fields.png.

All work for the rest of this section can be found at /week 8/EDA on new data.ipynb. Part way through this project, I received some very similar but new data from Nicholas to use. I did not re-do the EDA until week 8, as I wanted the report to have completely correct figures.

Casting a wide net

For ice thickness and ice velocity, the tables of results, along with the associated bar charts can be found at /week 5/Extended Regression.ipynb. At this stage, I had the idea to use the absolute values for the x and y axis features instead, and decided to test this theory against all the models. In hindsight, I should have waited until feature engineering to do something like this - you can just ignore those results. For ice mask, the table of results and associated bar graphs are available at /week 5/Extended Classification.ipynb. Again, there are results for using the absolute x and y values which you can ignore.

Narrowing the focus - feature engineering

I generated additional features at /week 7/Feature Engineering.ipynb. This is also the file where I look at the different window sizes for the rolling average features and produce the results in those tables. You will also find here the final results for all of the engineered features, and their associated importances graphs. There is also code in that file testing that the rolling average method works on a smaller toy dataset which I have left in, just for sanity checking. The file /week 7/RF results w engineered features.ipynb can be ignored, this is where I started tuning the models with the engineered features.

Narrowing the focus - feature engineering for ice velocity

The code for this section can be found at /week 8/predicting IV with surrounding raw features.ipynb. The image in this part explaining the numbering system is just a diagram I made using the drawing tool in google docs, there is no code for this.

Model Tuning

The tuning carried out for ice thickness and velocity can be found in /week 6/Ice Thickness RF Regression.ipynb and /week 6/Ice Velocity RF Regression.ipynb respectively. For ice mask, it is in /week 10/Tuning SVC.ipynb.

Sequential Modelling

My work for this can be found in /week 10/Sequential Modelling v2.ipynb.

Genetic Programming

This can be found at /week 9/ice\_thickness\_GP\_v2.py and /week 9/ice\_velocity\_GP\_v2.py. For reference, the data set files these GP programs use are generated in /week 4/New Data Preparation.ipynb. The graphs of solutions were generated using pygraphviz, and are in the repository as ‘ice\_thickness\_tree.pdf’ and ‘ice\_velocity\_tree.pdf’. These will be overwritten each time the programs are run, so keep that in mind.

Version Listings:

I also wanted to make a note here of the versions used for various things. Below is a (hopefully) complete list, for your reference. I have also included the output from the terminal of the current versions of all packages on my machine at the time of writing this in the file ‘packages\_info.txt’. Most of this will be irrelevant but I wanted to include it in case there is anything I forgot. The version of python I am using is 3.9.6 64-bit.

| Library | Version |
| --- | --- |
| Pandas | 2.1.1 |
| Numpy | 1.24.3 |
| Matplotlib | 3.7.2 |
| pygraphviz | 1.12 |
| Sklearn | 1.3.0 |
| seaborn | 0.12.2 |
| Deap | 1.4.1 |

### References and Acknowledgments

#### References

1. Chen, X., Zhang, X., Church, J. A., Watson, C. S., King, M. A., Monselesan, D., … Harig, C. (2017). The increasing rate of global mean sea level rise during 1993–2014. Nature Climate Change, 7, 492–495.
2. Edwards, T.L., Nowicki, S., Marzeion, B. et al. Projected land ice contributions to twenty-first-century sea level rise. Nature 593, 74–82 (2021). <https://doi.org/10.1038/s41586-021-03302-y>
3. Levermann, A.; Clark, P.U.; Marzeion, B.; Milne, G.A.; Pollard, D.; Radic, V.; Robinson, A. The multimillennial sea-level commitment of global warming. Proc. Natl. Acad. Sci. USA 2013, 110, 13745–13750.
4. Le Brocq AM, Payne AJ, Vieli A (2010) An improved Antarctic dataset for high resolution numerical ice sheet models (ALBMAP v1). Earth Syst Sci Data 2(2):247–260.
5. Hinkel, J., Lincke, D., Vafeidis, A. T., Perrette, M., Nicholls, R. J., Tol, R. S., … Levermann, A. (2014). Coastal flood damage and adaptation costs under 21st century sea-level rise. Proceedings of the National Academy of Sciences, 111(9), 3292–3297.
6. Huybrechts, P., & Oerlemans, J. (1988). Evolution of the East Antarctic ice sheet: A numerical study of thermo-mechanical response patterns with changing climate. Annals of Glaciology, 11, 52–59.
7. Golledge N. R. (2020). Long-term projections of sea-level rise from ice sheets. Wiley Interdiscip. Reviews: Climate Change 11 (2), e634. doi: 10.1002/wcc.634

#### Acknowledgements

I'd like to acknowledge Nicholas Golledge from the Antarctic research centre for providing the data for this project. It would not be possible to mimic a climate simulation without data from a climate simulation to use. I'd also like to thank and acknowledge my supervisors Bing Xue and Bach Nguyen for their support and guidance. The idea for this project was theirs and not mine, so all credit goes to them in that respect.