Predicting El Niño and La Niña Occurrences Using Historical Climate Data - Final Report

Gaurav Chawla, Bo Batten, Achyutan Narayanan, Saanvi Molugu, Jack Xu

Click here to go to project proposal

Click here to go to midterm checkpoint

Introduction/Background

Introduction

El Niño and La Niña, described as El Niño-Southern Oscillation (ENSO), are global climate phenomena caused by variations in the Pacific Ocean Sea Surface Temperature (SST) and the strength of easterly trade winds over the tropical Pacific [1]. El Niño is characterized by the warming of SSTs in the central/eastern Pacific Ocean, as opposed to La Niña events. ENSO events are characterized by various effects, like variable temperatures, rainfall, hurricanes, and droughts around the world.

Dataset Features

ENSO is reliant on equatorial Pacific temperatures, rainfall, and trade winds, so we plan to use CMIP6 (Climate Model Intercomparison 6) data with those values as features. CMIP6 contains simulated and recorded sea surface temperature both historically and for future climate change scenarios. We will use historical data for three variables: sea surface temperature, precipitation flux, and eastward wind. The data contains values from around the world, although we plan to narrow down to the Nino-3.4 region (5N to 5S, 120W to 170W). Our data is from https://aims2.llnl.gov/search with variables tos (sea surface temperature), pr (precipitation flux), and ua (eastward wind).

Since the proposal, we have decided to broaden our approach to include the other Niño regions - Niño 1+2 (0 to 10S, 80W to 90W), Niño 3 (5N to 5S, 150W to 190W), and Niño 4 (5N to 5S, 150W to 160E) as well. An illustration of the different Niño regions from the National Oceanic and Atmospheric Administration (NOAA) can be seen below.

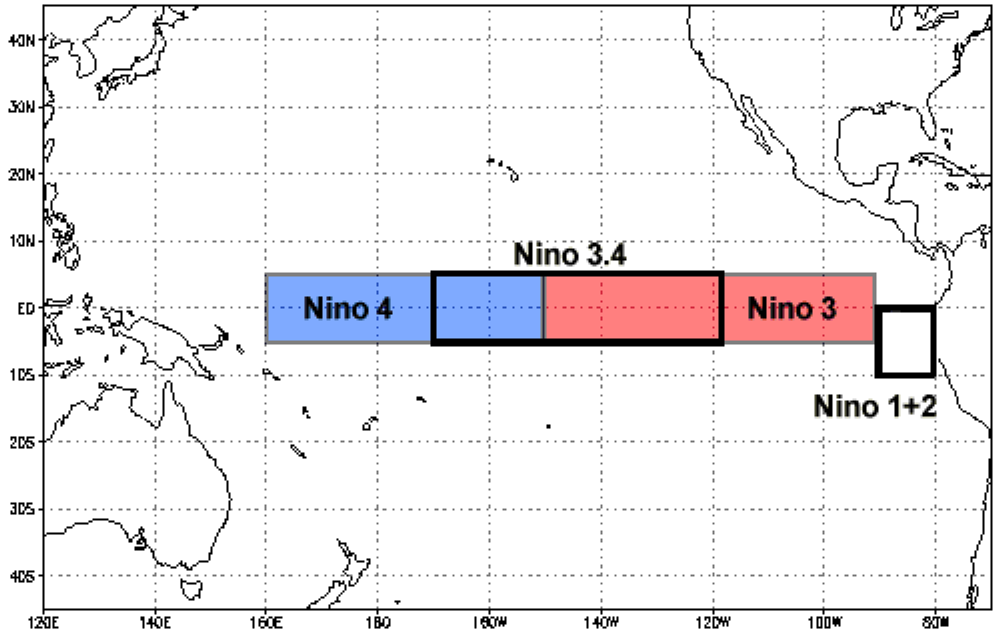


Figure 1: The different Niño regions as defined by NOAA. Source: [4]

Additionally, instead of treating every unique combination of latitude and longitude as a unique datapoint, we decided to look at the average value of a specific variable for a given month as a feature. For example, in our dataset, we identified 12 features - the average sea surface temperature, average eastern wind speed, and average precipitation amount for each of the 4 Niño regions. Our first datapoint would be the average value for each of these features for January 1950, the second datapoint the average value for these features for February 1950, and so on. This helped us reduce the complexity of our dataset, reduce the size of the dataset we are trying to work with, and reduce the amount of interpolation we had to do to fill in our datasets, as the resolution of the datasets did not match properly.

Problem/Motivation

ENSO events are sensitive to various factors. Recently, climate change has increased severity of droughts/floods globally, signifying the need for accurate prediction measures [2]. Considering that these events can generally only be accurately predicted one year in advance, [3] training ML models on past data is very useful, and has been used in the past to produce novel findings.

We are also interested in finding out how accurately a model can predict the existence and intensity of an ENSO event 6 months in advance, given monthly average data for key variables (sea surface temperature, eastward wind, precipitation) in each of the 4 key Niño regions. To test this, we will assign each of our datapoints of average monthly data to a label of the intensity of the ENSO conditions 6 months in the future. For example, our data from January 1950 would be labeled with the ENSO intensity score for July 1950. Using this labelling approach, we hope to train a model that will predict the intensity and existence of an ENSO event 6 months in advance, which would give emergency services and other public officials time to prepare.

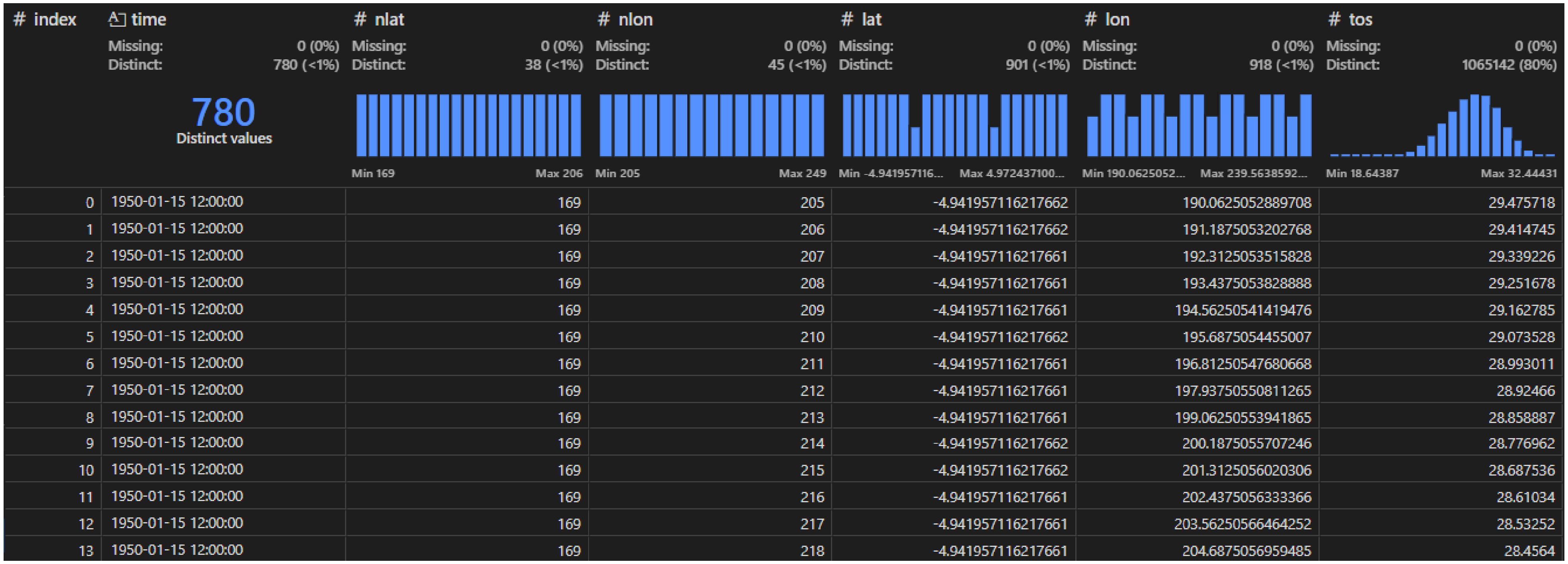
A commonly accepted standard for measuring ENSO event intensity is using the sea surface temperature anomaly value in the Niño 3.4 region. A SST anomaly of 0.5°C or greater indicates El Niño conditions, while a SST anomaly of -0.5°C or lower indicates La Niña conditions. Any anomaly values in between this is considered neutral conditions, which is neither El Niño nor La Niña.

Methods

Data Preprocessing

Data Cleaning:

When we first downloaded the datasets from our source, each of the variables that we were focused on came in its own separate file. Each file consisted of many datapoints, where each datapoint consisted of a timestamp, latitude/longitude value, then a value for the target variable (sea surface temperature, eastern wind, precipitation) at that specific time/location.



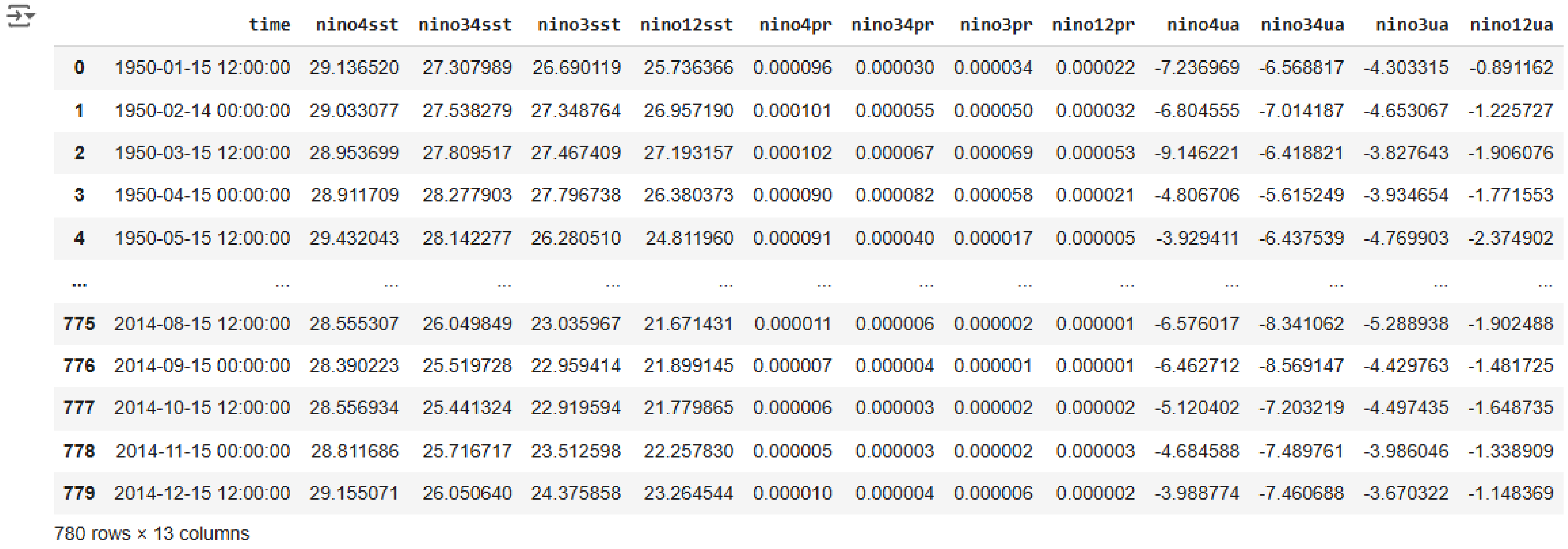
# index	time	# nlat	# nlon	# lat	# lon	# tos
	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)
	Distinct: 780 (<1%)	Distinct: 38 (<1%)	Distinct: 45 (<1%)	Distinct: 901 (<1%)	Distinct: 918 (<1%)	Distinct: 1065142 (80%)
	780 Distinct values	Min 169 Max 206	Min 205 Max 249	Min -4.941957116... Max 4.972437100...	Min 190.0625052... Max 239.5638592...	Min 18.64387 Max 32.44431
0	1950-01-15 12:00:00	169	205	-4.941957116217662	190.0625052889708	29.475718
1	1950-01-15 12:00:00	169	206	-4.941957116217662	191.1875053202768	29.414745
2	1950-01-15 12:00:00	169	207	-4.941957116217661	192.3125053515828	29.339226
3	1950-01-15 12:00:00	169	208	-4.941957116217661	193.4375053828888	29.251678
4	1950-01-15 12:00:00	169	209	-4.941957116217661	194.56250541419476	29.162785
5	1950-01-15 12:00:00	169	210	-4.941957116217662	195.6875054455007	29.073528
6	1950-01-15 12:00:00	169	211	-4.941957116217661	196.81250547680668	28.993011
7	1950-01-15 12:00:00	169	212	-4.941957116217661	197.93750550811265	28.92466
8	1950-01-15 12:00:00	169	213	-4.941957116217661	199.06250553941865	28.858887
9	1950-01-15 12:00:00	169	214	-4.941957116217662	200.1875055707246	28.776962
10	1950-01-15 12:00:00	169	215	-4.941957116217662	201.3125056020306	28.687536
11	1950-01-15 12:00:00	169	216	-4.941957116217661	202.4375056333366	28.61034
12	1950-01-15 12:00:00	169	217	-4.941957116217661	203.56250566464252	28.53252
13	1950-01-15 12:00:00	169	218	-4.941957116217661	204.6875056959485	28.4564

Here is an example of one of the files when plugged into a data visualizer tool. It is similar for the other two files.

Additionally, all of these files were Network Common Data Form (netCDF) files, meaning we had to use the `xarray` python library to parse them. Using the built in functionality of this library, for each we first trimmed the start date of the dataset to start January 1, 1950, and isolated the data corresponding to the 4 Niño regions.

We noticed that some of the data had extreme outliers, such as some datapoints having a wind speed of 10^36 m/s, which is clearly not possible, and may have occurred due to a sensor failure or other technical issue. To avoid having outliers affect our data and potentially our model, we only included wind speed values less than or equal to 100m/s which is ~224 mph.

We then used numpy's mean functionality to calculate the mean value for each month's values for each feature, and then concatenated the features together into a pandas dataframe.



	time	nino4sst	nino34sst	nino3sst	nino12sst	nino4pr	nino34pr	nino3pr	nino12pr	nino4ua	nino34ua	nino3ua	nino12ua
0	1950-01-15 12:00:00	29.136520	27.307989	26.690119	25.736366	0.000096	0.000030	0.000034	0.000022	-7.236969	-6.568817	-4.303315	-0.891162
1	1950-02-14 00:00:00	29.033077	27.538279	27.348764	26.957190	0.000101	0.000055	0.000050	0.000032	-6.804555	-7.014187	-4.653067	-1.225727
2	1950-03-15 12:00:00	28.953699	27.809517	27.467409	27.193157	0.000102	0.000067	0.000069	0.000053	-9.146221	-6.418821	-3.827643	-1.906076
3	1950-04-15 00:00:00	28.911709	28.277903	27.796738	26.380373	0.000090	0.000082	0.000058	0.000021	-4.806706	-5.615249	-3.934654	-1.771553
4	1950-05-15 12:00:00	29.432043	28.142277	26.280510	24.811960	0.000091	0.000040	0.000017	0.000005	-3.929411	-6.437539	-4.769903	-2.374902
...	...	...	...	...	...	...	...	...	...	...	...	...	...
775	2014-08-15 12:00:00	28.555307	26.049849	23.035967	21.671431	0.000011	0.000006	0.000002	0.000001	-6.576017	-8.341062	-5.288938	-1.902488
776	2014-09-15 00:00:00	28.390223	25.519728	22.959414	21.899145	0.000007	0.000004	0.000001	0.000001	-6.462712	-8.569147	-4.429763	-1.481725
777	2014-10-15 12:00:00	28.556934	25.441324	22.919594	21.779865	0.000006	0.000003	0.000002	0.000002	-5.120402	-7.203219	-4.497435	-1.648735
778	2014-11-15 00:00:00	28.811686	25.716717	23.512598	22.257830	0.000005	0.000003	0.000002	0.000003	-4.684588	-7.489761	-3.986046	-1.338909
779	2014-12-15 12:00:00	29.155071	26.050640	24.375858	23.264544	0.000010	0.000004	0.000006	0.000002	-3.988774	-7.460688	-3.670322	-1.148369

780 rows × 13 columns

For example, "nino4sst" column refers to the average monthly sea surface temperature for the Niño 4 region. Each of our 4 regions has one feature for each of the 3 target variables we are interested in (12 total features).

By doing this data cleaning, we were able to convert our raw data, which was very large (200-300 MB per file, so ~1 GB in total), in a foreign format (.nc filetype), and had different data resolutions (the sea surface temperature file had observations in more/different latitude/longitude values than the other two variables) into a concise, friendly data format which we can train our models on.

Data Transformation:

In order to train our models, we needed to make sure that there were no NaN values or similar outliers that would not allow us to run the model properly. Upon, inspection, we found that the Niño 4 eastern wind feature had 37 NaN values out of a total of 780 values (~4.74% NaN) and the Niño 3.4 eastern wind feature had one NaN value out of the total 780 values (~0.13% NaN).

```
[ ] print(nino_df.isna().sum())
```

```
time          0
nino4sst      0
nino34sst     0
nino3sst      0
nino12sst     0
nino4pr       0
nino34pr      0
nino3pr       0
nino12pr      0
nino4ua      37
nino34ua      1
nino3ua       0
nino12ua      0
dtype: int64
```

Since the percentage of NaN values were low compared to the overall dataset, we decided to use linear interpolation to fill in these NaN values, as the NaN values were fairly well spread out through the column.

Dimensionality Reduction Considerations:

We decided not to use PCA for dimensionality reduction due to the way it creates new features as linear combination of the original features. One of the core motivations behind our project is to identify the specific features from our dataset that contribute most to ENSO events. If we ran PCA on our dataset, we would lose the interpretability of our original features, meaning that we would be unable to identify specfic features and their contributions in various models.

ML Methods

Through analysis of the problem and defining our expected outcomes, we selected several ML methods to better understand the data and improve our predictive accuracy. Detailed quantitative results and analysis for each of these ML methods are provided in the “Results and Discussion” section below.

Gaussian Mixture Model (GMM):

Our original thoughts were that running GMM could improve our project by allowing us to see major clusters related to data behavior on each ENSO region we are looking at. By doing this, we could get an idea of how the features for each region behave, giving us insights into potential relationships within the data that were not as obvious from just looking at the tabular data that we constructed.

Linear Regression:

Our original thoughts were that since linear regression is best used to predict continuous dependent variables, we can use it to predict the intensity of ENSO events in the future based on past data. Our dependent variable in this case is the intensity of the ENSO event measured 6 months after the measurements of each datapoint were measured, so we can try to predict ENSO events and intensity 6 months in the future using these labels. Furthermore, by using linear regression, we could also get a measure of the relative importance of each feature, allowing us to gain insight into which regions and environmental variables have the greatest effect on future ENSO intensities. Based on the results of our linear regression, we planned to explore other forms of regression, such as Lasso Regression, in order to perform some dimensionality reduction and see if our accuracy improves with fewer features. In a way, our initial run of Linear Regression is a baseline, allowing us to expand upon our predictions and experiment with our model to improve accuracy.

Regularized Regression and Feature Engineering:

Given that we saw unacceptable training and testing RMSE in our initial linear regression implementation, we realized that our initial linear regression model may have been underfitting to our data. To combat this, we tried a form of feature engineering called polynomial features, which helps our models capture nonlinearity in the data. Additionally, since this adds a risk of overfitting, we decided to test out both ridge and lasso regression. Both of these forms of regression have regularization “penalty” terms that can help reduce the effects of overfitting, while lasso can also perform a type of dimensionality reduction by zeroing out unimportant features.

Neural Networks:

After noticing a persistently poor RMSE even after using regularized regression techniques and feature engineering, we decided to use neural networks to see if we could improve our prediction accuracy and lower our testing error. Specifically, we used LSTM (Long Short-Term Memory) neural networks, because LSTMs are optimized for analysis and making predictions on time-series data.

Results and Discussion

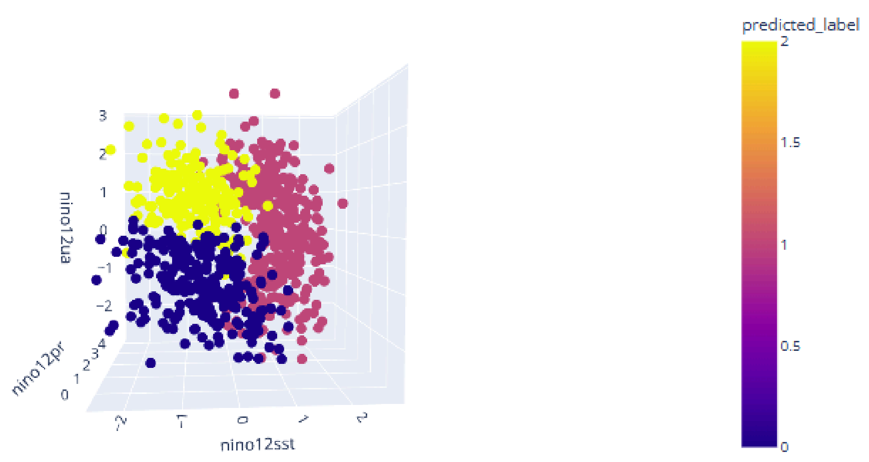
Visualizations, Quantitative Metrics, and Analysis

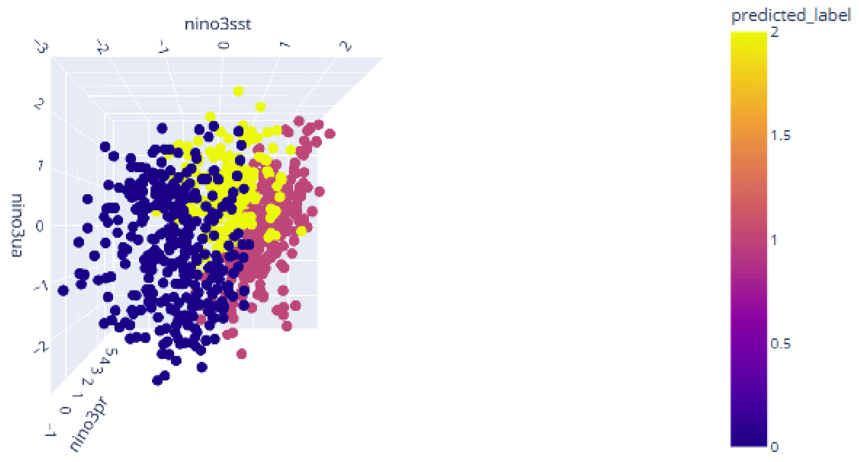
Gaussian Mixture Model (GMM):

Region	Silhouette Score
Niño Region 4	0.2147
Niño Region 3.4	0.1999
Niño Region 3	0.1912
Niño Region 1+2	0.2697

From our GMM implementation, we found that the clusters generated had low silhouette scores, with all four regions having silhouette scores close to 0.2-0.25. One reason we believe this might be the case is that there is not a lot of separation between the clusters, as our data models natural phenomena which does not have any clear separation between any parts of the data.

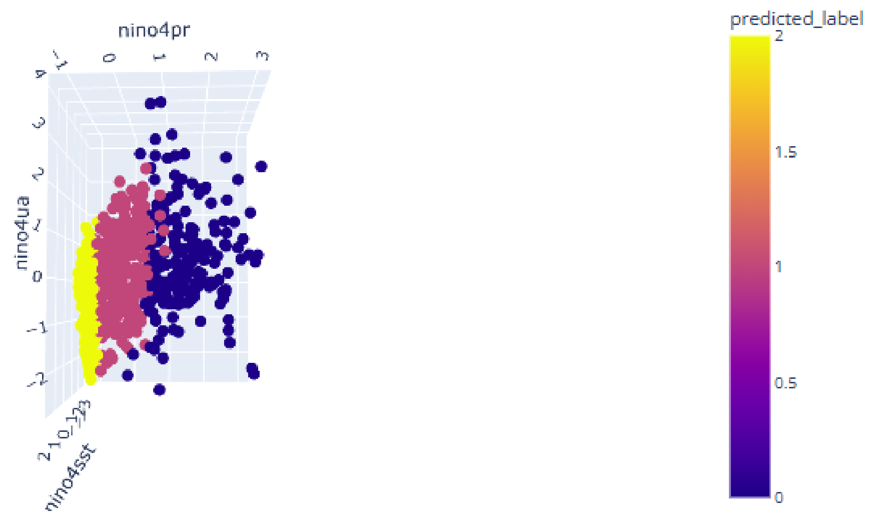
However, we found some indications of ENSO phases similar to what happens in real life in all four Niño regions.

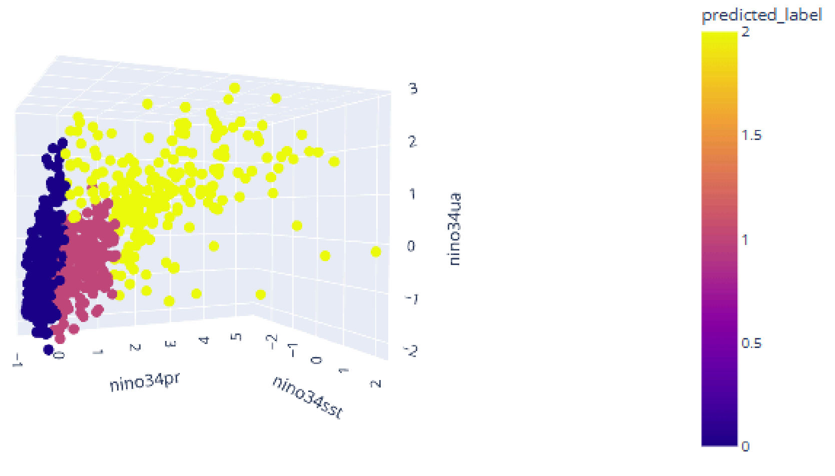




Left: Niño 1+2 region. Right: Niño 3 region.

We were able to notice in each region's clustering that there was almost always a component with a small variance in precipitation AND precipitation values consistently below normal - this component also generally had lower than average sea surface temperatures and eastward wind values that tended towards negative values, indicating lower than normal wind values. These combination of factors indicate a circulation similar to La Niña, as it is defined as a period of time where the eastward wind in the pacific is lower than normal as the western trade winds intensify, as well as lower sea surface temperatures than average.





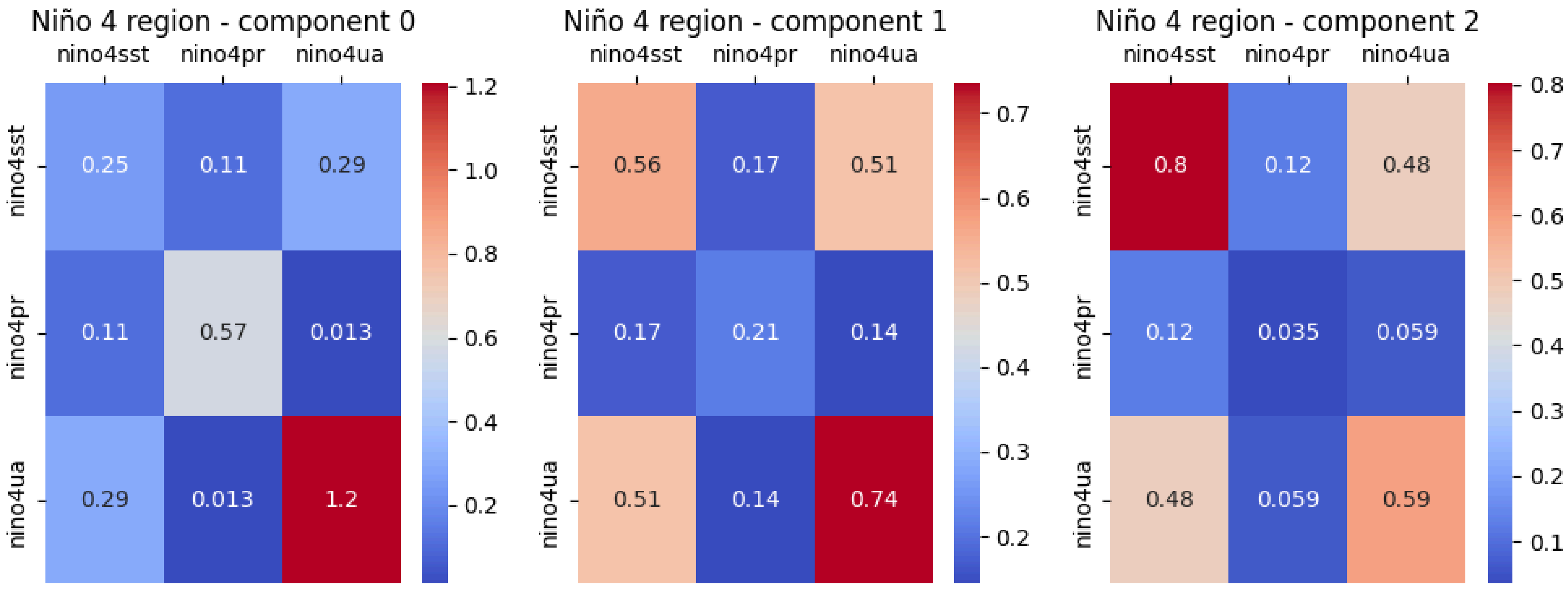
Left: Niño 4 region. Right: Niño 3.4 region.

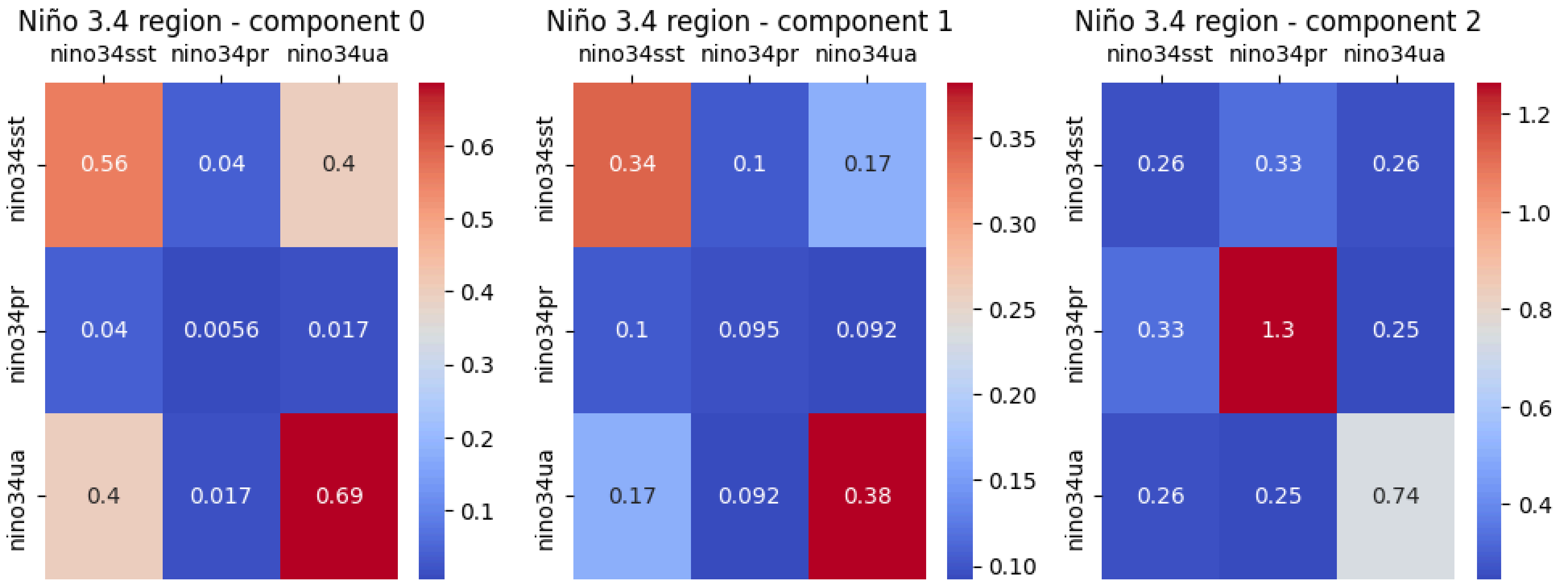
There were also components that generally had higher variance in precipitation, but always towards the positive side, indicating higher than average precipitation. Additionally, the sea surface temperature values were consistently higher than normal, and eastward wind values higher than normal as well. These combinations of factors are similar to El Niño conditions, which are characterized by weakened or even reversed trade winds (east to west winds), leading to higher eastward winds and warmer sea surface temperatures, as well as higher precipitation values.

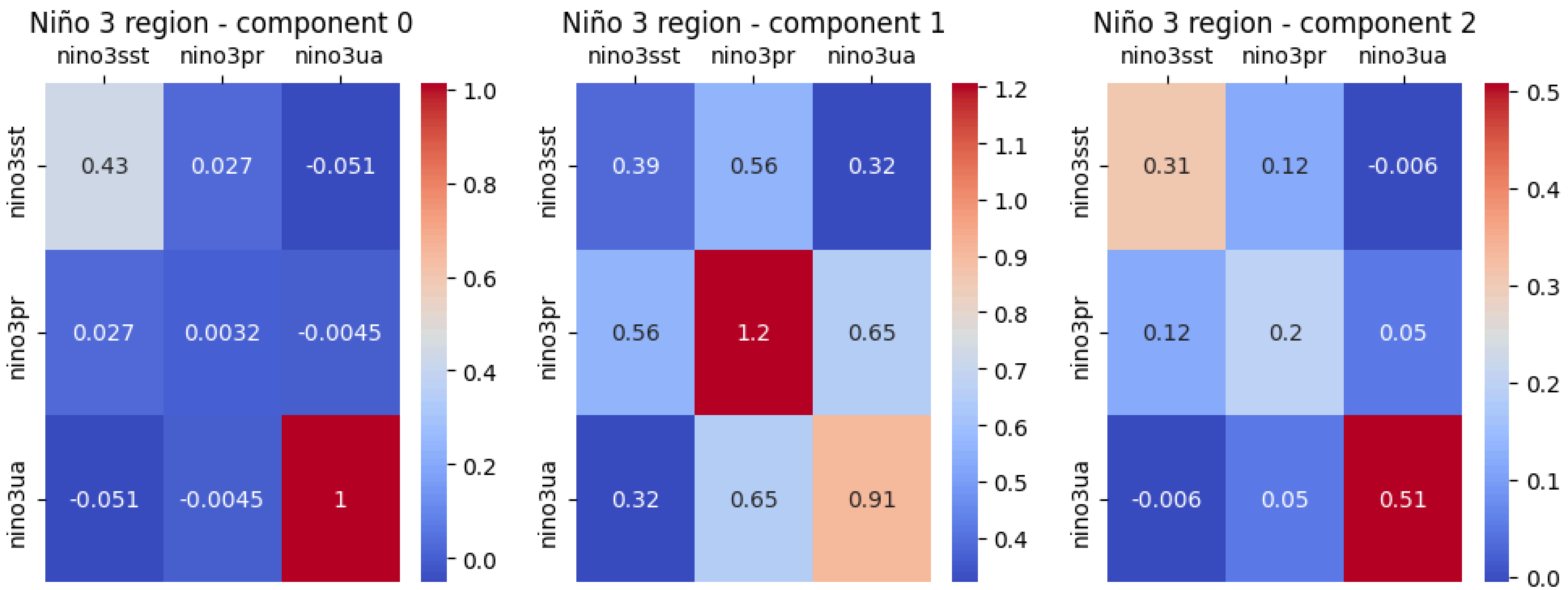
Finally, there seems to always be a component sandwiched in between the components we identified as similar to El Niño and La Niña clusters. This cluster likely identifies neutral conditions (neither El Niño nor La Niña), given that the values, which are centered, tend to be closer to 0. This means that there are no extreme shifts or variation in our three target variables, and likely represents neutral conditions.

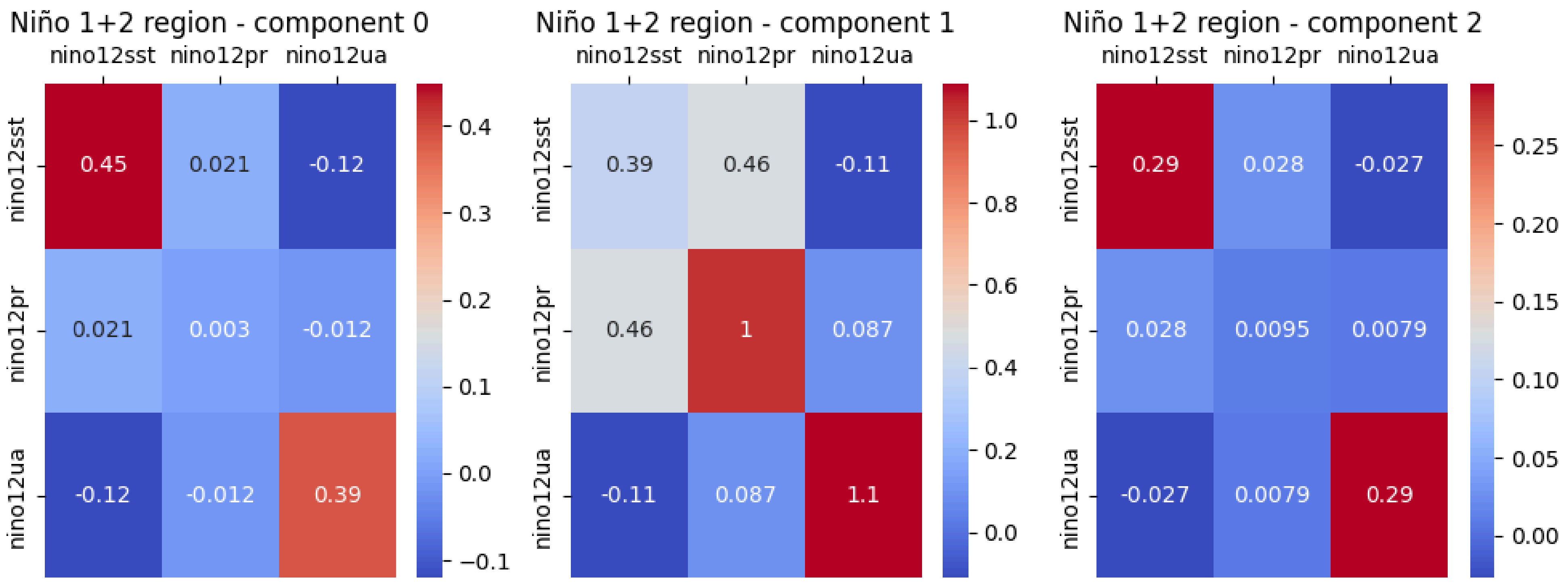
We identified that our clustering for the Niño 1+2 region had a noticeably higher silhouette coefficient (0.2697, compared to next highest 0.2147) and noticeably better cluster compared to the other three regions. This is likely due to the fact that this region sees very large variations in precipitation during El Niño/La Niña, due to its proximity to the west South American Coastline. During neutral ENSO conditions, the Niño 1+2 region is very dry, but during El Niño the precipitation values can be much higher than normal. This large difference in conditions based on ENSO status leads to natural “separation” in the data, which probably leads to a higher silhouette score.

We also constructed heatmaps of the covariance matrices of each of the components that GMM created. This allowed us to quickly visualize and quantify some of the observations about variance and also how each of the features correlated to each other (heatmaps for each region/component shown below).


Niño 4 region - component 0
nino4sst nino4pr nino4ua
nino4sst: 0.25 0.11 0.29
nino4pr: 0.11 0.57 0.013
nino4ua: 0.29 0.013 1.2
0.2 0.4 0.6 0.8 1.0 1.2
Niño 4 region - component 1
nino4sst nino4pr nino4ua
nino4sst: 0.56 0.17 0.51
nino4pr: 0.17 0.21 0.14
nino4ua: 0.51 0.14 0.74
0.2 0.3 0.4 0.5 0.6 0.7
Niño 4 region - component 2
nino4sst nino4pr nino4ua
nino4sst: 0.8 0.12 0.48
nino4pr: 0.12 0.035 0.059
nino4ua: 0.48 0.059 0.59
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8


Niño 3.4 region - component 0
nino34sst
nino34pr
nino34ua
0.56
0.04
0.4
0.04
0.0056
0.017
0.4
0.017
0.69
Niño 3.4 region - component 1
0.34
0.1
0.17
0.1
0.095
0.092
0.17
0.092
0.38
Niño 3.4 region - component 2
0.26
0.33
0.26
0.33
1.3
0.25
0.26
0.25
0.74


Niño 3 region - component 0
nino3sst nino3pr nino3ua
nino3sst 0.43 0.027 -0.051
nino3pr 0.027 0.0032 -0.0045
nino3ua -0.051 -0.0045 1
1.0 0.8 0.6 0.4 0.2 0.0
Niño 3 region - component 1
nino3sst nino3pr nino3ua
nino3sst 0.39 0.56 0.32
nino3pr 0.56 1.2 0.65
nino3ua 0.32 0.65 0.91
1.2 1.1 1.0 0.9 0.8 0.7 0.6 0.5 0.4
Niño 3 region - component 2
nino3sst nino3pr nino3ua
nino3sst 0.31 0.12 -0.006
nino3pr 0.12 0.2 0.05
nino3ua -0.006 0.05 0.51
0.5 0.4 0.3 0.2 0.1 0.0



Linear Regression:

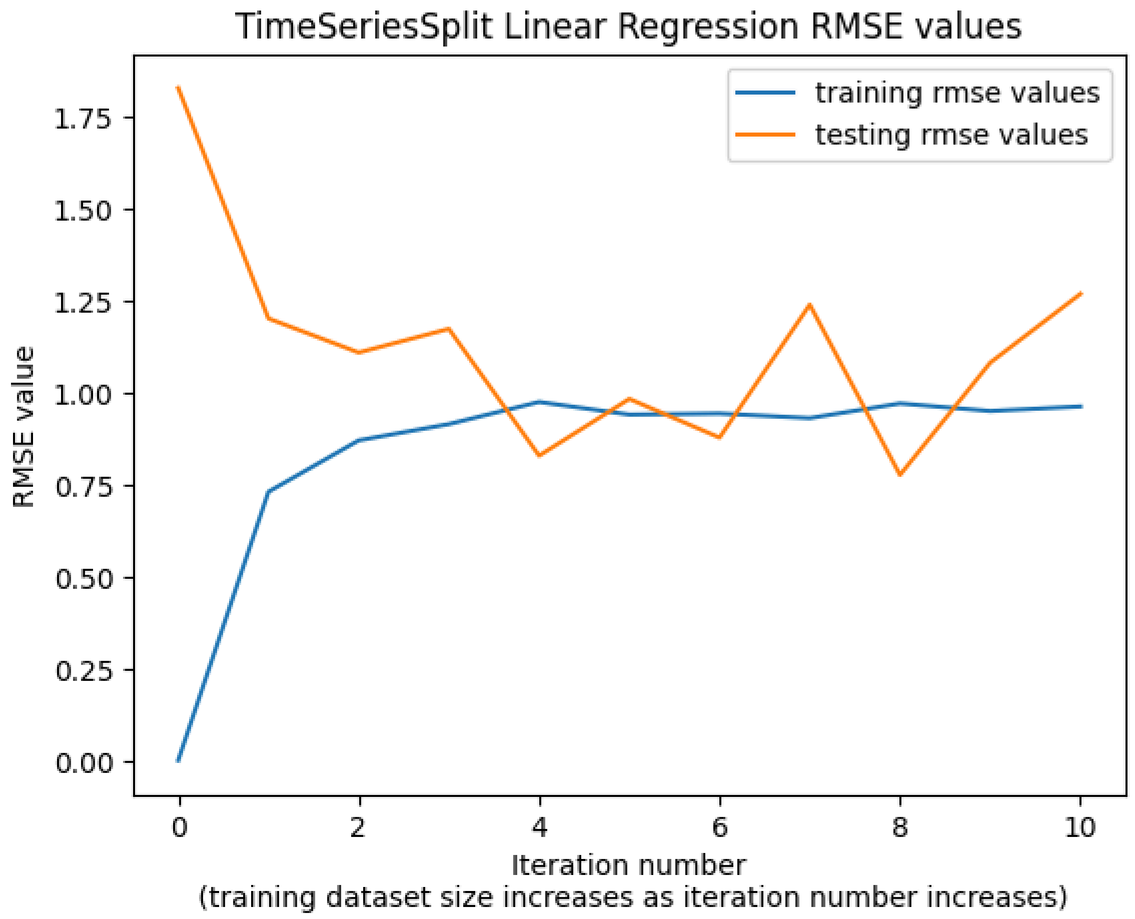
We tried two methods of splitting our data into testing and training sets for training/validating our linear regression model. The first method was randomly splitting our dataset into 80% training data, and 20% testing data. Running our basic linear regression model on this training dataset, and then validating with the test dataset, we saw that the regression model had a training RMSE of 1.0112 and a testing RMSE of 0.9214. Since our dependent variable is measured in degrees celsius, and values less than -0.5°C indicate La Nina events and values greater than 0.5°C indicate El Niño events, this means that this error is quite high and our predictions could be misrepresenting the actual true ENSO state. Additionally, the weights of the model were:

nino4sst: 0.3026	**nino4pr:** 0.2765	**nino4ua:** 0.28724883
nino34sst: -0.71249241	**nino34pr:** 0.09486572	**nino34ua:** 0.30722729
nino3sst: 2.10335957	**nino3pr:** 0.50133125	**nino3ua:** -0.47491717

nino12sst: 0.85297788	**nino12pr:** 0.24804669	**nino12ua:** 0.42084209

These weights help us rank the importance of each feature for our model. From the low weights, we can see that nino34pr and nino12pr, the precipitation values for the Niño 3.4 and 1+2 regions, may be less important. Additionally, from the high weights, we can see that nino12sst and nino34sst, the sea surface temperature values for the Niño 1+2 and 3.4 regions, may be more important.

Additionally, we also trained a linear regression model using scikit-learn's `TimeSeriesSplit` method, which ensures that the testing data was always chosen from a time period in the future from any testing data. This is a common method used when training ML models on time series data, since in real life future data shouldn't be used to predict past events, which is what could happen if we randomly split the data. This method successively trains the model with a larger and larger dataset on each iteration, so we also get to see how the model's RMSE was affected by different amounts of data. We graphed the testing/training RMSE values over each iteration to see where they converged.



The average training RMSE was 0.8358, and the average testing RMSE was 1.1251, though it was possibly skewed by the initial few iterations, where the model overfit on a smaller dataset and had a value of 0 for training RMSE and a high test RMSE. Both the initial run of linear regression and the TimeSeriesSplit run made us realize that our model might be underfitting, because both of our training and testing error are unacceptable in context of the dependent variable units. One way we plan to improve our model is to use more features by using the polynomial features feature engineering method to increase the number of features in our dataset and help our dataset model non-linear behavior.

Regularized Regression and Feature Engineering:

	Without Polynomial Feature Engineering		**With Polynomial Feature Engineering**			
	Linear Regression	**Ridge Regression**	**Lasso Regression**	**Linear Regression**	**Ridge Regression**	**Lasso Regression**
Best Alpha Value	--	1.90996	0.00049	--	0.13416	0.00064
R^2 Score	0.3933	0.4178	0.4205	0.6441	0.6431	0.6377
Training RMSE	1.0112	0.9574	0.9631	0.7745	0.7755	0.7813
Testing RMSE	0.9214	1.1461	1.2681	1.0511	1.0302	1.0081

At first, we ran ridge and lasso regression without any feature engineering to measure a baseline of how much these regularized regression techniques would improve our prediction accuracy by themselves. Scikit-learn has classes for both ridge regression and lasso regression with cross validation (`RidgeCV` and `LassoCV`) that allowed us to verify that we were using the correct value for the hyperparameter alpha by performing cross validation. Compared to the baseline linear regression, we noticed increases in both the R^2 Score and better training RMSEs for both types of regularized regression. However, the testing RMSE increased dramatically, meaning that these regularized regression techniques alone were not sufficient to prevent overfitting in our data.

After implementing polynomial feature engineering (using the `PolynomialFeatures` class in scikit-learn), our dataset was transformed from 774 datapoints x 12 features to 774 datapoints x 91 features, with the extra features being all the possible combinations of the original features, squares of the original features, and all the different combinations of the cross-product terms between features. Doing this allows our data to capture any nonlinearity in the original phenomenon better and help our model make more accurate predictions. With these extra features, the training RMSE was dramatically lowered from the baseline linear regression, but the testing RMSE remained unacceptably high. Given that El Niño events are represented by label values over 0.5 (measured in degrees Celsius), while La Niña events are represented by label values less than -0.5, an RMSE around 1 would have very poor performance predicting the type and intensity of a given ENSO event. At this point, it was clear that different iterations of linear regression with different data preprocessing techniques would not significantly lower the testing RMSE to a more reasonable level that indicates good model prediction accuracy, so we decided to pursue a different type of machine learning algorithm altogether.

Neural Networks:

Configuration	RMSE	R^2	Epochs
Basic Linear Regression	1.011	0.393	–
ReLU activation	0.888	0.501	60
SGD w/ 0.8 momentum	0.978	0.418	25
SGD w/ 0.9 momentum	0.953	0.418	25
SeLU activation	0.882	0.508	60
SeLU with 0.1 dropout	0.953	0.418	20
SeLU with L2 regularization	0.880	0.496	30

Table 1: Our final error, R^2, and epoch convergence values after various tunings of hyperparameters or slight changes of architecture. Epoch convergence values are the number of epochs the model required for testing loss to stabilize and stop improving. The RMSE and R^2 are averaged over 30 runs of each NN.

Hidden Layers	RMSE	R^2	Epochs
1	0.882	0.506	60
2	0.882	0.495	30
3	0.884	0.496	18

Table 2: Our final error, R^2, and epoch convergence values after changing the number of hidden layers. Epoch convergence values are the number of epochs the model required for testing loss to stabilize and stop improving. The RMSE and R^2 are averaged over 30 runs of each NN.

Our neural network architecture was relatively simple - we used one hidden layer with 50 neurons, with a selu activation function and the default ADAM optimizer used frequently in Keras. This architecture was chosen based on extensive hyperparameter tuning, as it minimized both the training and testing RMSE.

We originally started our neural network architecture with one hidden layer and a ReLU (rectified linear unit) activation function - which led to a significant improvement from the baseline linear regression. The testing RMSE went down from over 1 (measured in degrees Celsius) to approximately 0.886, with the R^2 value significantly increasing too (from 0.393 to 0.501). After testing changing the activation function to SeLU (Scaled Exponential Linear Unit) to potentially improve accuracy, we concluded that SeLU would be the best one by a very small, but still consistent margin in testing RMSE. Additionally, we found that a sigmoid activation function would take over 400 epochs to converge, while SiLU (Sigmoid Linear Unit) did not show any improvement over ReLU.

Unfortunately, from this point on, we were only able to make minor improvements to the testing RMSE by changing the architecture of our neural network. Changing the number of hidden layers of our neural network did not result in any improvement, only leading to faster model convergence times, as indicated by table 2. Unsurprisingly, with more layers, a fit could be reached more quickly, but we likely reached the elbow point of the bias-variance tradeoff, even with just one layer.

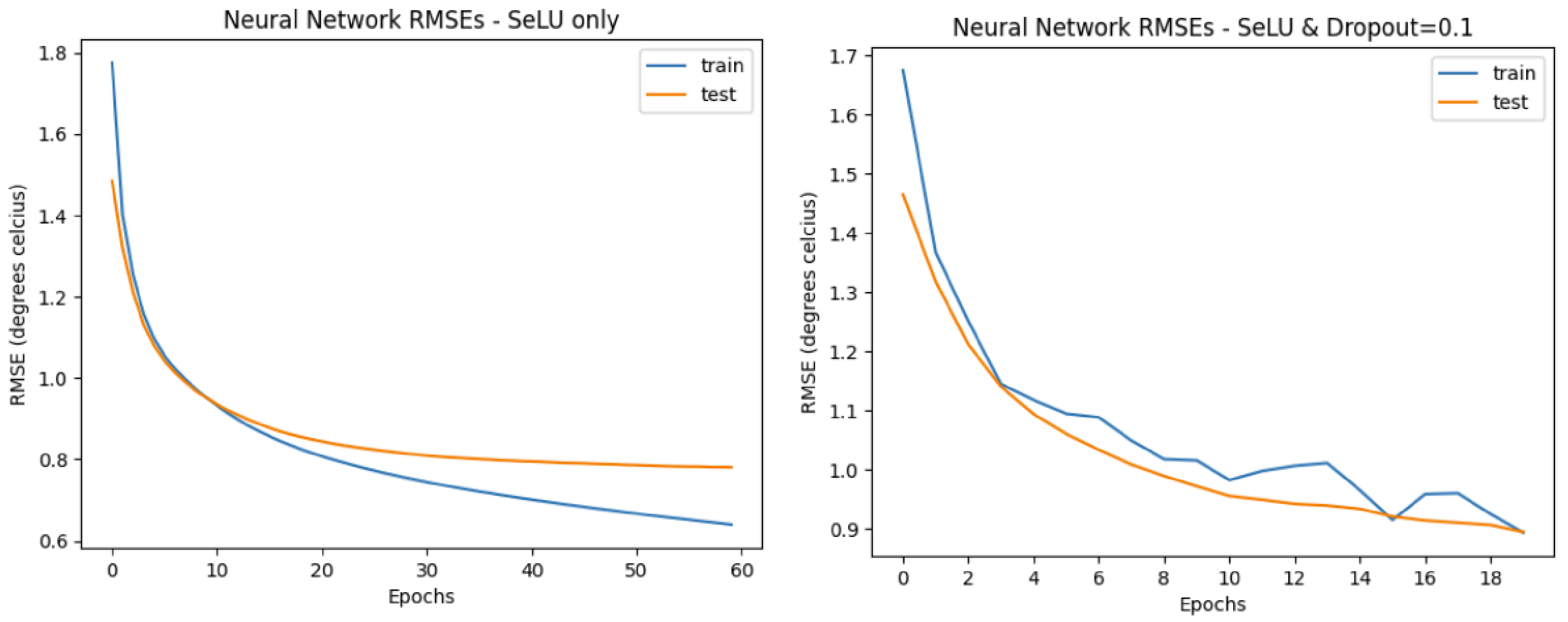


Figure 1: Plot on the left is the train/test RMSE over each epoch for a neural network with just a SeLU activation function. Plot on the right is train/test RMSE over each epoch with both SeLU activation function and a dropout probability of 0.1.

The next thing we tried to add to our neural network architecture was dropout - as we saw that the training RMSE ended up being significantly lower than the testing RMSE in all of our initial trainings, so there was a potential overfitting problem that adding dropout could potentially fix. However, by adding a fairly low dropout probability of 0.1, the model actually seemed to be underfitting, even after the test loss stabilized, as the training loss was greater than the testing loss, and the R^2 value and RMSE were similar to the linear regression. We also attempted to add L2 regularization to the weights to reduce any overfitting, but did not see any significant effects on the output of the model.

The other major hyperparameter we tried was using stochastic gradient descent (SGD) with momentum as an optimizer instead of ADAM. While ADAM does have an adaptive learning rate, which greatly helps with converging into a minimum for loss, we hoped that using the momentum parameter for SGD could help with prediction by allowing the model to partly take into account how the features were changing in the previous months. This, however, also did not work, with loss and correlation being comparable to the original linear regression.

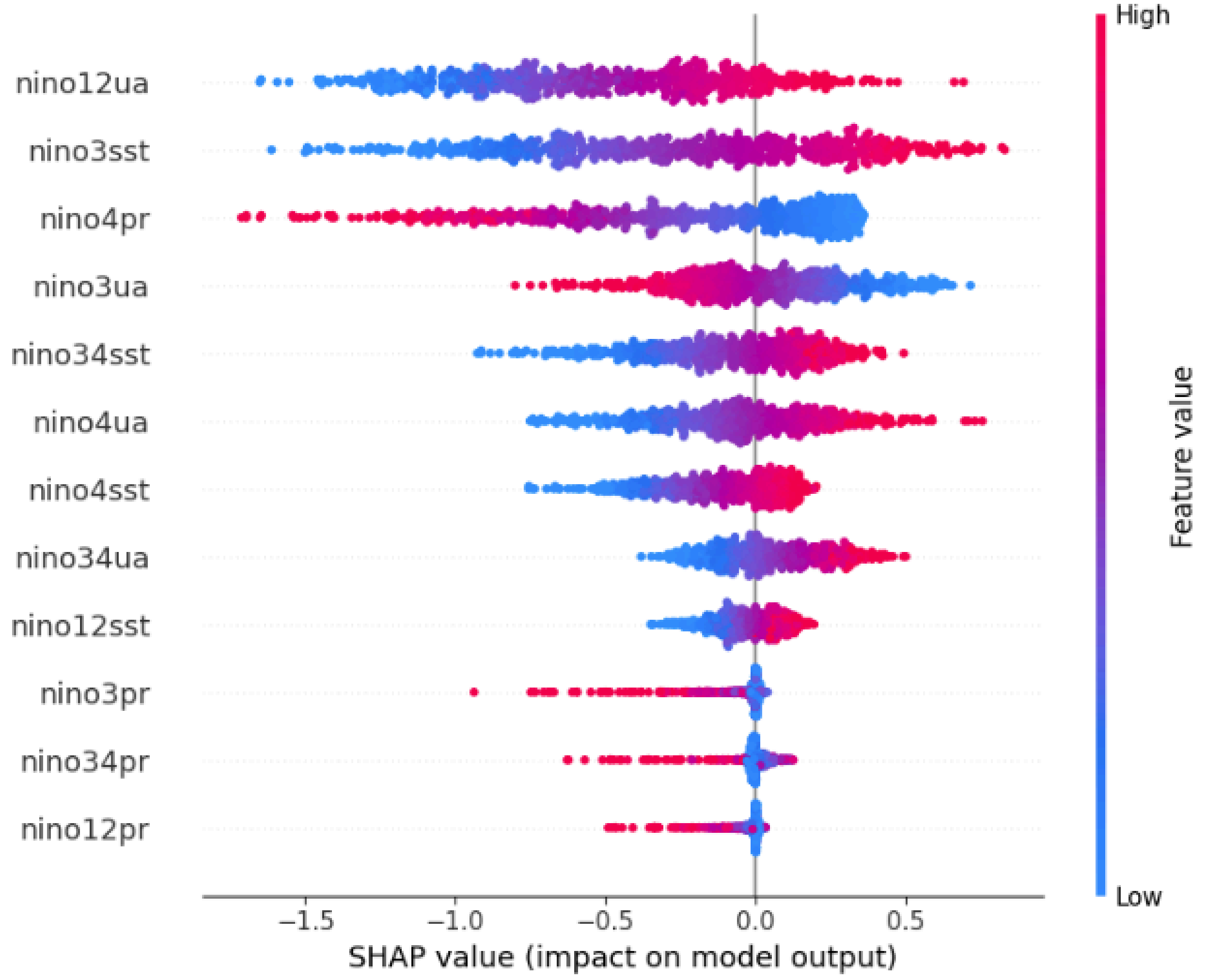


Figure 2: The results of the SHAP analysis on the neural network model.

In order to interpret our results from the neural network, and clearly identify which features contribute more towards the predictions in the neural network, we did a SHAP (Shapley Additive Explanations) analysis. While SHAP analyses are not specifically for neural networks (it is model-agnostic), it can be applied to find feature importance in neural networks.

The most visible result was that the average sea surface temperature (SST) in the Niño 3 region (represented by the nino3sst feature) has a large effect on the ENSO condition 6 months in advance. If the Niño 3 region's SST is higher than normal at the start, then 6 months in the future, El Niño conditions will likely stay (given that this indicates that the tropical pacific is warmer than normal). It also indicates that in the case that the Niño 3.4 region's SSTs aren't warm enough to be considered an El Niño event, then an El Niño event will likely arrive in the following 6 months. Interestingly, the SSTs in the Niño 4 and Niño 3.4 regions were not found to be as crucial for predicting the output. This likely suggests that ENSO conditions generally remain stable within the same phase (El Niño/La Niña/Neutral) over a 6-month period. As a result, these SSTs have limited predictive power, except in rare cases where temperatures in these regions are significantly warmer than usual.

Unsurprisingly, in 3 of the 4 Niño regions, the trade winds blowing westward slower than average or completely reversing direction implied El Niño conditions. Our model results also indicated that in this scenario, El Niño conditions would either strengthen or remain steady within the following months. However in one of the 4 Niño regions, Niño 3, the opposite correlation was true. The explanation is unclear - it likely involves negative feedback loops in that region causing any unusual in one direction to tend towards normal. This phenomenon likely needs more research, including using complex analysis techniques to see if specific groups of conditions lead to specific changes in ENSO.

In all Niño regions, high precipitation values indicated that SSTs will decrease later on in the Nino 3.4 region, leading to lower SST anomaly (our label) values. The explanation for this is likely that high precipitation implies a higher cloud cover, which would lead to more light being reflected out to space, causing cooling air temperatures (and likely sea temperatures too). Furthermore, rainfall would cause both precipitative and evaporative cooling, further leading to overall cooling of the region very slowly over time - a negative feedback loop. This phenomenon is also visible in La Niña conditions, but to a far lesser extent - the reason for this is also unknown and likely requires more research into atmospheric dynamics.

Comparison

When selecting models, we wanted each to provide us with a helpful and unique lens for either understanding the data or predicting the effects of El Niño/La Niña. For instance, when comparing our GMM usage with our linear regression usage, GMM provided us with more information about the dataset, while linear regression helped us understand feature importance. Specifically, GMM showed us whether clusters existed and whether or not they represented the certain phenomena we wanted to measure, and linear regression gave us a baseline understanding of which features are most helpful to use when predicting the weather effects, and also helped us understand how regression would work on our dataset.

Due to the established differences in goals of these models, they did not contradict each other. As previously mentioned, GMM gave us insights into the clustering behavior of the data, and confirmed that ENSO behavior has recognizable patterns, and linear regression showed us which features are linearly related to ENSO effects. Despite having different goals, you can still see that some features, such as nino34pr, had a very large variance in GMM visualization but a very small weight in linear regression. Findings like these illustrate the importance of approaching the data from different angles using different models, because it gives us more perspective to what is actually going on, rather than just trusting the results of a specific model.

When comparing how successful each model was for this specific dataset, both also had their own unique struggles. Due to the nature of our dataset, there was not much natural separation of the data, so clustering it was challenging, which made GMM less successful than it could have been. The existence of nonlinear relationships in our dataset made it hard for linear regression to understand fully the relationship between our features and ENSO effects, due to the fact that linear regression can only find linear relationships in the data, and to find non-linear relationships we would have to use a more complex model, such as neural networks.

In summary, GMM and linear regression both had their own strengths, weaknesses, perspectives, and struggles that led to a better understanding of our dataset and how its features interacted with each other and ENSO effects.

Due to the nonlinear relationships within the data, we tried to use polynomial regression. Despite polynomial regression having more features, our accuracy did not improve significantly. We also tried to regularize our weights to prevent overfitting with both polynomial and linear regression. The result, yet again, wasn’t significant. The reason for this is likely because polynomial regression, while taking into account interaction between features, is limited in how many interactions it can take into account before overfitting - if we were to pick a higher degree polynomial, it would likely overfit despite there being a large amount of terms that are interactions between features.

Furthermore, our Lasso and /Ridge regularization did not have a significant effect on our regression because, as indicated before, there were lots of features with high importance. Thus, Lasso or Ridge couldn’t reduce the coefficients without it reducing the quality of the regression significantly.

Given that polynomial regression wasn’t working, the next thing we tried were neural networks. Specifically, LSTM neural networks were used so that any time series trends can be captured more easily, thus allowing for modeling of complex relationships within the data, but without there being too many features. This time, there was a significant improvement in the regression, with the neural network capturing a significantly larger part of the relationship, as indicated by the SHAP analysis. However, we also noticed similar results to the linear regression, with a lot of features having relatively high weights, and weight regularization not having a significant effect on the regression. This is likely because there could be more complex factors playing into ENSO prediction that we weren't able to model due to time constraints. However, overall, LSTM neural networks still proved to be significantly more effective due to their ability to capture momentum in data.

Next Steps

Knowing the outcomes of our ML models and analyzing the areas for improvement, we have several ideas of how we would extend this ENSO prediction project to further improve prediction accuracy.

- To reduce the size of the dataset and simplify the problem/analysis, we used monthly average values for each of the features we were interested in. To further improve our model's accuracy, we could use the raw data in the original data resolution provided from our data sources. This removes a layer of data preprocessing (finding mean values per month across the feature values) and could potentially enhance our model's accuracy if taking the mean across a feature loses some of the patterns in the data that helps the models predict ENSO events.
- While literature is clear that sea surface temperature, eastward wind speed, and precipitation values are the three most important variables to determine an ENSO event, there are certainly many other variables in play when determining such complicated climate events. If we introduced more features representing additional climate variables (such as surface air temperature, or sea level pressure) or included data from more regions, we could potentially improve our model's accuracy as we are including information not covered in the three variables we studied in this project.
- We could also explore additional ML models that we learned in class for dimensionality reduction such as random forest feature selection, or explore ML models for prediction such as support vector machines (SVM). Doing this would give us more tools to simplify our dataset or improve our model's accuracy.

References

[1] G. a. R. Tello, K. Takahashi, and C. Karamperidou, "Explained predictions of strong eastern Pacific El Niño events using deep learning," Scientific Reports, vol. 13, no. 1, Nov. 2023, doi: 10.1038/s41598-023-45739-3.

[2] N. Lenssen et al., “Strong El Niño events lead to robust Multi-Year ENSO predictability,” Geophysical Research Letters, vol. 51, no. 12, Jun. 2024, doi: 10.1029/2023gl106988.

[3] S. Wang, L. Mu, and D. Liu, “A hybrid approach for El Niño prediction based on Empirical Mode Decomposition and convolutional LSTM Encoder-Decoder,” Computers & Geosciences, vol. 149, p. 104695, Apr. 2021, doi: 10.1016/j.cageo.2021.104695.

[4] NCEI.Monitoring.Info@noaa.gov, “El Niño/Southern Oscillation (ENSO),” El Niño/Southern Oscillation (ENSO) | National Centers for Environmental Information (NCEI), https://www.ncei.noaa.gov/access/monitoring/enso/sst (accessed Nov. 3, 2024).

Gantt Chart

Download Excel file of our Gantt Chart

View Our Gantt Chart on Microsoft Office Online

Contribution Table

Name	Midterm Contributions
Gaurav Chawla	Maintained/edited content on GitHub pages site. Maintained GitHub repo and wrote descriptions for all files. Coded/wrote the analysis for regularized regression/feature engineering section. Helped edit neural networks analysis section. Helped modify midterm proposal to fit final report format.
Bo Batten	Helped write comparison section between GMM and linear regression. Helped modify midterm proposal to fit final report format. Helped brainstorm next steps.
Achyutan Narayanan	Helped code Neural Network implementation, including tuning hyperparameters. Wrote neural network analysis section. Incorporated neural networks/regularized regression into comparison section. Helped modify midterm proposal to fit final report format. Helped create final slide deck. Helped record final video.

Name	Midterm Contributions
Saanvi Molugu	Helped modify midterm proposal to fit final report format. Helped create final slide deck. Helped record final video. Helped brainstorm next steps.
Jack Xu	Helped code Neural Network implementation, including tuning hyperparameters. Coded/Generated SHAP analysis visualizations. Helped brainstorm details for next steps section. Created slide deck for final video. Helped modify midterm proposal to fit final report format. Helped create final slide deck. Helped record final video.