Should Sally Surf?

Elias Devadoss ETD3@WILLIAMS.EDU
CS Department

1. Introduction

In Del Mar, California, surfing conditions can be unpredictable, unstable, and ever-changing. The goal of this project is to test whether or not a neural network can successfully classify pieces of meteorological data into clusters, which will be hand-labeled in terms of their "surfability." I hypothesize that clustering the data into distinct groups, the number of which will be determined with the elbow method for k-means, will allow for surfable conditions to be pulled indirectly from the data, as opposed to direct relationships such as logistic regression. The meteorological data will be taken from the National Data Buoy Center, under the U.S. National Oceanic and Atmospheric Administration (NDBC). A supervised neural network is then trained to predict cluster labels. I expect a fine-tuned neural network to achieve accuracy greater than 80% and greater than the baseline logistic regression.

2. Preliminaries

The machine learning algorithms I will use are as follows:

MLP. The model I will be using is a Multilayer Perceptron (MLP), which is a type of neural network. It uses a directed acyclical graph (DAG) with layers of nodes. Each node has an edge leading to it from each node in the previous layer and an edge leading from it to each node in the next layer. Edges represents a learnable weight or parameter. The final layer is fed into a logistic regression to output predictions. I use a Rectified Linear Unit (ReLU) activation function for each neuron. This is a piecewise function which zeros out negative values and retains positive values.

SGD. I use mini-batch stochastic gradient descent (SGD) to train the model, which shuffles the data before dividing it into smaller batches. Then, SGD computes the gradient for each batch before updating the parameters. This iterates over epochs until the model is no longer learning from the data (early stopping) or for 50 epochs.

K-Means. I use k-means clustering to make training targets. This attempts to group the data into clusters based on centroids, recalculating the cluster centroids each iteration. Using the elbow method, I will determine the number of clusters to aim for, and k-means will be used to assign samples a cluster label. These labels are then the targets that I will use for classification.

Logistic Regression. For my baseline I will use a logistic regression, which looks at input variables to predict a binary outcome. The hyperparameters will not be fine-tuned in order to provide a simple baseline.

The activation functions, overall neural network, and gradient descent are implemented from scratch. However, I use numpy (Harris et al., 2020), pandas (pandas development team, 2020) and scikit-learn (Pedregosa et al., 2011) for the clustering and logistic regression.

3. Data

The data used is historical meteorological data collected from NOAA buoy readings, which containts the following variables:

- Date: YY (year), MM (month), DD (day), hh (hour), mm (minute)
- Wind: WDIR (direction), WSPD (speed), GST (gust)
- Waves: WVHT (height), DPD (dominant period), APD (average period), MWD (mean direction)
- Atmosphere: PRES (pressure), ATMP (air temperature), WTMP (water temperature), DEWP (dew point), VIS (visibility), TIDE (tide)

Wind direction, wind speed, gust, pressure, dew point, visibility, and tide were all missing variables, so they were omitted. In addition, air temperature was missing for around half the data entries, so it was omitted as well. Entires with missing data for any of the other features were also removed, accounting for 25 entries total.

The data was split by year, with 2019-2022 being training, 2023 being validation, and 2024 being test. After splitting the data, I removed year (YY) as a feature as well. Between the six years of data there were 87,704 entries of data in total and nine total features—month, day, hour, minute, wave height, dominant period, average period, mean wave direction, and water temperature.

Month ran from January (1) through December (12), days were in the range 1-31, hour was on a 0-23 cycle, and minute was generally 0 or 30. Wave height was measured in meters, dominant and average period in seconds, mean wave direction in degrees, and water temperature in Celsius.

All features were standardized around zero as the mean of the training data and the variance based on the standard deviation of the training data.

4. Training And Validation Of Models

While the standard tool that surfers use for wave prediction is an app called Surfline, it is rather inaccurate. It is also hard to specifically gauge the accuracy of the app without comparing it to real-life conditions, which requires resources beyond the scope of this project. Instead, the baseline I will use will be unregularized logistic regression. This will give a simple model to compare against, to see if deeper and more complex architectures actually provide insights to surfing conditions. On this data, a logistic regression was able to achieve accuracy of 94.8% for cluster labeling, setting a proficient baseline with only a simple model.

Training. I trained the model on a neural network, with a few architectures of varying depth of layers and layer sizes. The model looked between a 15-neuron hidden layer, two 8-neuron hidden layers, and four 4-neuron hidden layers. For each epoch, the data was shuffled, batched into groups of 256, and used to find the combined negative log likelihood as the loss. Then, the loss was backpropagated through the network and used to update the gradients. Early stopping was implemented, where if three consecutive epochs did not improve the accuracy, the model stopped training. This helped to prevent overfitting, as well as reduce resource load.

Clustering. Using the elbow method (on the model with the wide but shallow architecture), I determined that four clusters was the optimal number to use, as the loss in inertia flattened out and no longer had as steep of an effect.

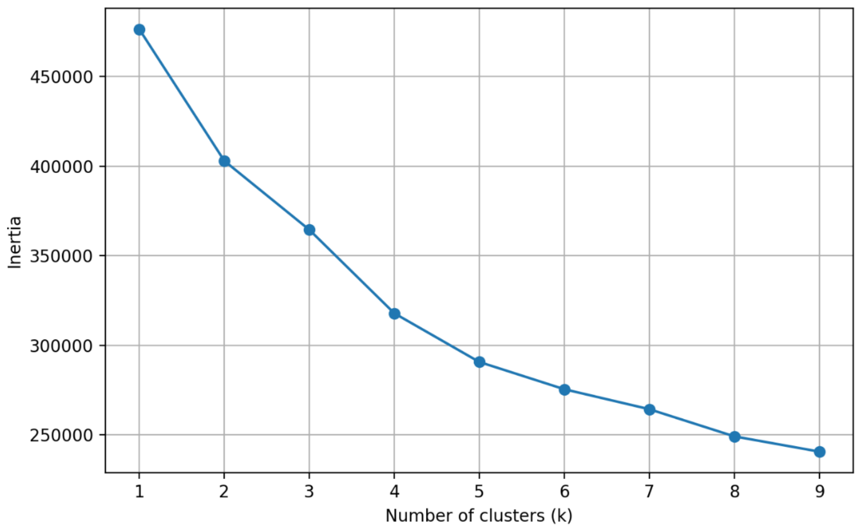


Figure 1: The Elbow Method for K-Means Clustering

I also considered k = 5, but I did not want to over-complicate the model, and the visualization of the clusters simply split an unimportant existing cluster in two. For visualizing the data, I used Principal Component Analysis (PCA) to do a dimensional reduction. This turned the nine features into only two, while still keeping most of the data. This can be seen in Figure 2.

Based on the feature values of the cluster centroids (Figure 3), I decided to label Cluster 3 (in pink) as ”good surfing conditions.” The water temperature was the highest, and it had the longest dominant period—directly correlated with how long one could ride a given wave. Though wave height was the lowest, the difference between Cluster 3 and the largest average wave height was 0.286 meters (less than a foot). Furthermore, variance in wave height is large enough that averaging 0.75 meter waves will give pleasant waves to surf for all but the most hardcore. Date, hour, and minute were not included as they were virtually useless to the clustering, being essential averages (date around 15, hour around 12, and minute around 15). Finally, Cluster 3 was the closest to peak surfing season (when waves are generally best), which lies around late summer and early fall.

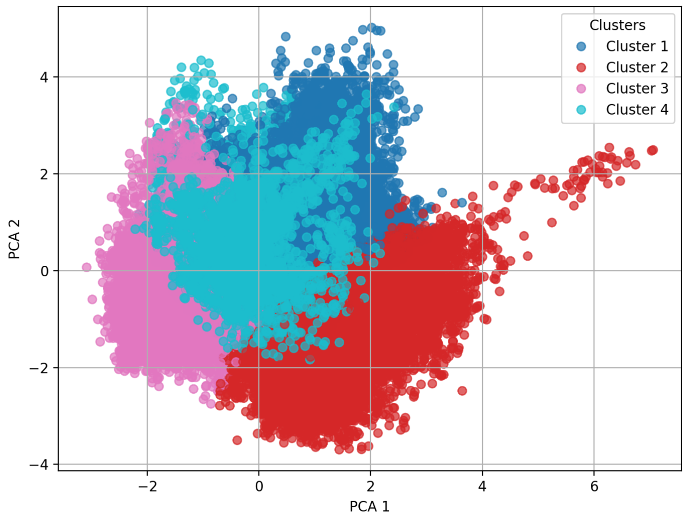
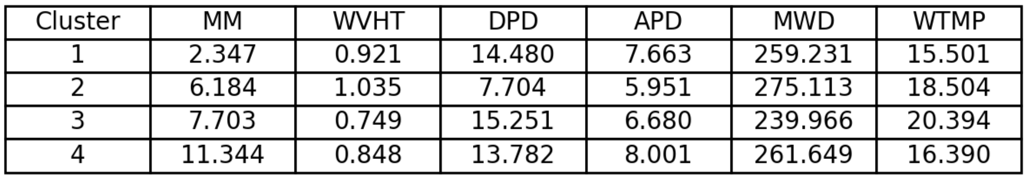


Figure 2: K-Means Clustering where k = 4

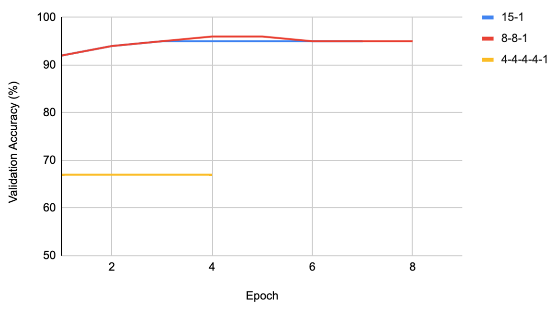


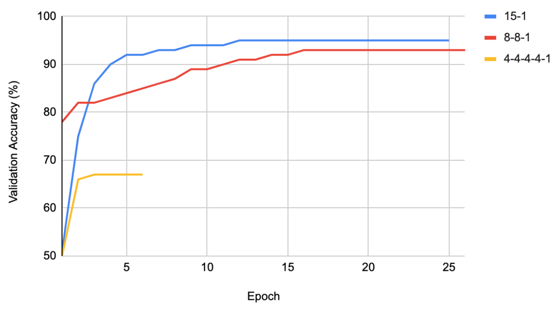
Cluster	MM	WVHT	DPD	APD	MWD	WTMP
1	2.347	0.921	14.480	7.663	259.231	15.501
2	6.184	1.035	7.704	5.951	275.113	18.504
3	7.703	0.749	15.251	6.680	239.966	20.394
4	11.344	0.848	13.782	8.001	261.649	16.390

Figure 3: Centroid Feature Values

Validation Setup. The validation set was used to determine the best of the three architectures by looking at the accuracies over the validation set. Learning rate was also tested between 0.1, 0.01. In total, there were thus six models to consider, and whichever performed best on the validation set was chosen as the final model to test on. With architectures that were not too deep, batched stochastic gradient descent, and dropout, I did not try a lower learning rate to ensure that the models did not underfit. All models used dropout = 0.5, meaning neuron activations were randomly zeroed out half the time. This value was a good balance as it helped to prevent some overfitting but did not do so too aggressively, as there were other mechanisms in place to do the same.

Validation Results. Over all models, training and validation accuracies were very closely tied, with training being generally two percent higher than validation. Therefore, only validation accuracies were graphed over epochs. For both learning rates, the model with four hidden layers of 4-neurons each performed poorly (stopping at 67% accuracy). This is likely because the low number of neurons could not accurately represent the data being presented to it. With a learning rate of 0.1, the 15-neuron model reached an accuracy of 0.95, but the two 8-neuron model achieved 0.96. Though they both ended with accuracy of 0.95 on the final epoch of training, the two 8-neuron model was performing better with 95.4 compared to 95.1. For the learning rate of 0.01, the 15-neuron model consistently outperformed the two 8-neuron model, ending with a validation accuracy of 95% compared





Learning Rate = 0.1 Learning Rate = 0.01

Figure 4: Validation Accuracies

to 93%. The best validation accuracy on the last epoch of training of any model was that of the two 8-neuron model with a learning rate of 0.1, therefore it was chosen as the final model.

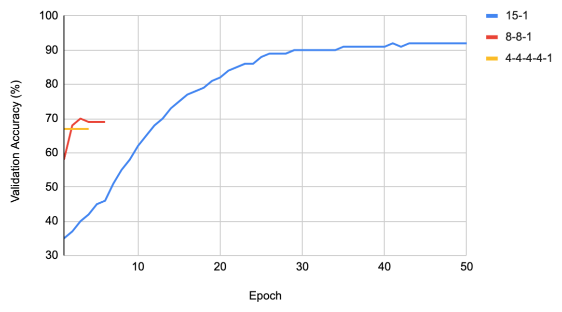
5. Results

The 8-8-1 model with a learning rate of 0.1 was able to perform well on the unseen data, achieving a test accuracy of 95%. This is the same accuracy that it had on the validation set. However, this is also the same accuracy as provided by the logistical regression baseline.

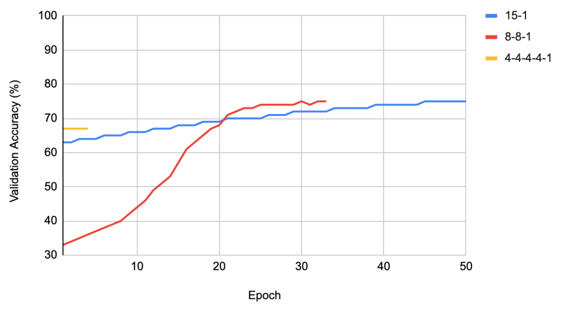
6. Ablation Study

For an ablation study, I downsampled the data to see how the models would react to less training data. All valid data entries were used to determine clustering, and then the dataset was shrunk before training began. Originally, I planned to use 10% of the original data, thus training with around 8,700 examples. However, the new model achieved near-identical performance to the original model. I instead only provided the models with 1% of the original data. This was done by randomly sampling from the data, before training on the new dataset. All six combinations of model were again tested, and the best was evaluated on the held out test set.

For the learning rate of 0.1, both models with deeper architectures quickly flattened out and had accuracies just below 0.7. However, the 15-neuron network slowly progressed over all fifty epochs and achieved 92% accuracy by the end. For the lower learning rate, the lack of data was harmful and training was much slower. The four 4-neuron network again came in and stayed at accuracy under 0.7, while the two 8-neuron network peaked at 75%. The 15-neuron network also came in at 75%, but it had not yet flattened out yet and may have been able to reach a higher accuracy. The best model—in this case the 15-neuron network with a learning rate of 0.1—achieved an accuracy rating of 0.88, 7 points lower than the unablated model. However, given the scarcity of data provided to the model, I think this study shows the model has some resistance to ablation and is still relatively well-performing.



Learning Rate = 0.1



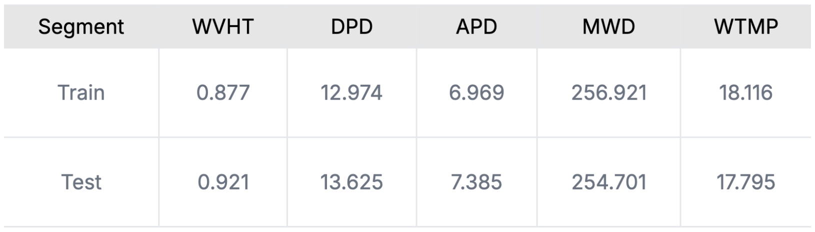
Learning Rate = 0.01

Figure 5: Ablation Accuracies

7. Discussion

Despite possible distribution shift stemming from a new year's data and the strong learning rate, the two 8-neuron network was still able to retain an accuracy of 0.95 for the testing data. Training for this model got up to 0.97 accuracy. The model was able to successfully draw inference from the data and only lost two points between the training and testing data. This high accuracy was likely aided by a large dataset with many features. Furthermore, mini-batch SGD prevented overfitting while validation ensured the optimal model was chosen.

The model did not outperform the baseline set by logistic regression, but rather tied it. This could imply that a neural network did not provide additional benefits or insights into surfing conditions. However, the model could have performed sub-optimally for multiple other reasons. Splitting by year, there are possible distribution shifts to take into account. As seen in Figure 6, many of the average values changed between the 2019-2022 training data and the 2024 testing data.



Segment	WVHT	DPD	APD	MWD	WTMP
Train	0.877	12.974	6.969	256.921	18.116
Test	0.921	13.625	7.385	254.701	17.795

Figure 6: Train vs. Test Averages

Waves grew about two inches and lasted around half a second longer. While the logistic regression also had this distribution shift, it may have affected a complex neural network more than a model with linear relationships to features. Furthermore, only three different architectures and two different learning rates were tested. There are infinitely many more combinations to try, some of which would have been more effective in this neural network.

One shortcoming of these findings is a lack of broader implications. As the clusters were labeled by looking at data from the centroids, one could argue there is no need for the model itself. If one has access to the raw values of the features, they could decide the ”surfability” of the waves themselves. Despite this critique, the model still successfully clusters data points from meteorological observations and is able to predict those clusters with 95% accuracy. This project serves as a strong baseline for further expansion, building a neural network that combines aspects of weak and strong learning to produce a balanced final product.

8. Conclusion

Through this project I realized that a better model could have come out of many things, such more hyperparameter tuning, longer stopping times, and more epochs. Having only three architectures and two learning rates stunted the possibility to find more optimal system designs for the project. In order to avoid overfitting, after three consecutive epochs without improvement models stopped training. In addition, after 50 epochs models were always stopped. Both of these were useful tools, but some models appear as though they may have improved if not for the early stopping (such as the two 8-neuron network in the ablation study) or if not for the 50-epoch limit (such as the 15-neuron network in the ablation study).

It is worth noting that both of the underfitting cases were in the ablation study, and as the larger project had much more data, these features operated better. The full-data models may have instead been prone to overfitting if not for the early-stopping. No model with the strong learning rate of 0.1 took longer than eight 8 to end training, and none longer than 26 overall. This could have been solved by testing a lower learning rate, which my hubris prevented me from trying. Additionally, a stronger dropout rate could be used.

In the future I would like to attempt label propagation to form clusters to see if the results change at all. I would also like a way to generate better labels, possibly by creating more clusters to find more precise pockets of ”good surfing.” Overall, this project was difficult as surfing conditions are hard to evaluate—especially from numbers alone—and subjective. However, the model outperformed expectations and will continue to improve as I tinker.

References

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

NDBC. Station 46266 - del mar nearshore, ca (153). URL `https://www.ndbc.noaa.gov/station_history.php?station=46266`.

The pandas development team. pandas-dev/pandas: Pandas, February 2020. URL `https://doi.org/10.5281/zenodo.3509134`.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.