Linear Regression, Polynomial Regression and MLP for predicting Success rates for climbing a Mountain

Pranav Bajoria
CIDSE
Arizona State University
Tempe, Arizona, USA
p.bajoria@asu.edu

Naga Sai Teja Vinnakota
CIDSE
Arizona State University
Tempe, Arizona, USA
nvinnako@asu.edu

I. Introduction

The task at hand was to perform Machine Learning Regression using Linear Regression, Polynomial Linear Regression, Multi-Layer Perceptron (single hidden Layer) on linear features and Multi-Layer Perceptron (single hidden Layer) on polynomial features models. These models were to be trained to predict the success ratio of an attempt made towards climbing a mountain given a series of features that describe the weather on the day the attempt is made. The dataset was taken from Kaggle and the performance of all the aforementioned models are discussed in the paper.

II. Method / Approach

A. Data Pre-processing

The available dataset has 2 sheets of data for the 2 years 2014 and 2015. The first one is “climbing_statistics”, which explains the parameters related to mountain climbing like attempts made, number of successful attempts and the success ratio, and the second one is “Rainier_Weather” which explains the different features of weather observed on each day.

We first change the date column from a string value to a date-time usable format which will later help us for further processing. By visualizing the climbing stats data, we found that there were duplicate entries for the same date, even when the routes were same. Hence, we first separated the data based on the required routes, which is Disappointment Cleaver (from now called Route 1) and Emmons-Winthrop (from now called Route 2). Our next objective was to merge the duplicate entries. We achieved this by summing the attempts made and succeeded columns for the same dates and recalculating the success percentage column by dividing the two.

We also found that the climbing statistics data were recorded for only the days when climbing happened and Rainier Weather data had entries which were missing a few dates that were present in the climbing stats data. Therefore, to combine the data, we used a Pandas function “pd.merge”. By keeping the parameter on = “Date”, we make sure that only the weather data for which there is a corresponding date in the climbing statistics is merged. The parameter of pd.merge that helped us achieve this was setting “how = inner”. This takes the intersection of both the data-frames according to date and makes sure there are no NAN values in the merged data. Since the entire model solely depends on features derived from parameters, it would be a bad idea to fill these NAN values with the respective means of their columns.

Next, we dropped unnecessary columns like 'Date', 'Attempted', 'Succeeded' which do not help in training the regression models. We then split the data into training and testing so that it can be fed to the model for training.

B. Selecting the optimum features for training the models.

To get a better perspective on how the success rate depended on the various features, and how they all were correlated with each other, we used different plotting methods available in matplotlib. The figures are shown in figures 1, 2, 3 & 4.

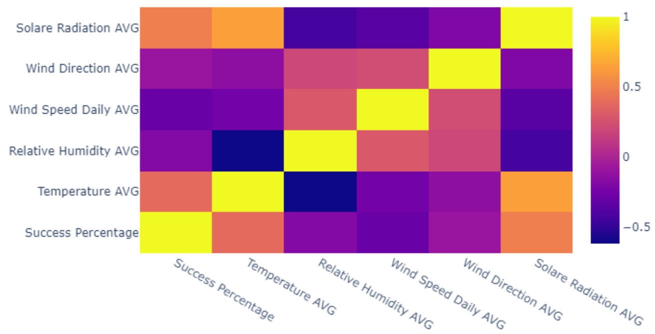


Figure 1 Covariance matrix for Disappointment Cleaver (Route 1)

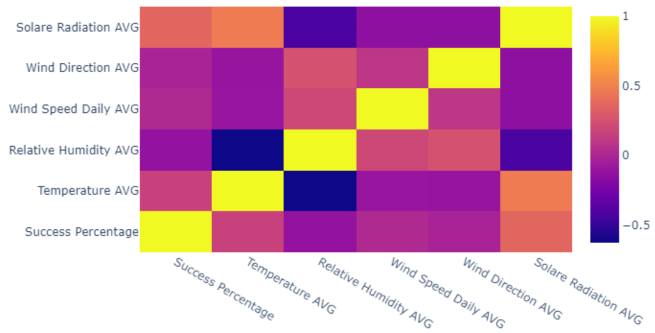


Figure 2: Covariance matrix for Emmons-Winthrop (Route 2)

We dropped another column “Battery Voltage Average” since its covariance with the required output label was very low. That means, most of the values for this feature were the same on most dates.

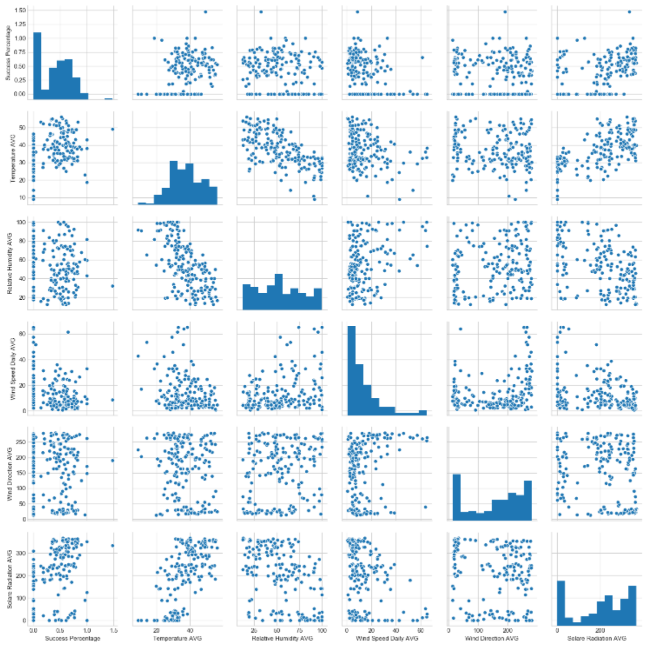


Figure 3: Route 1's feature selection plots

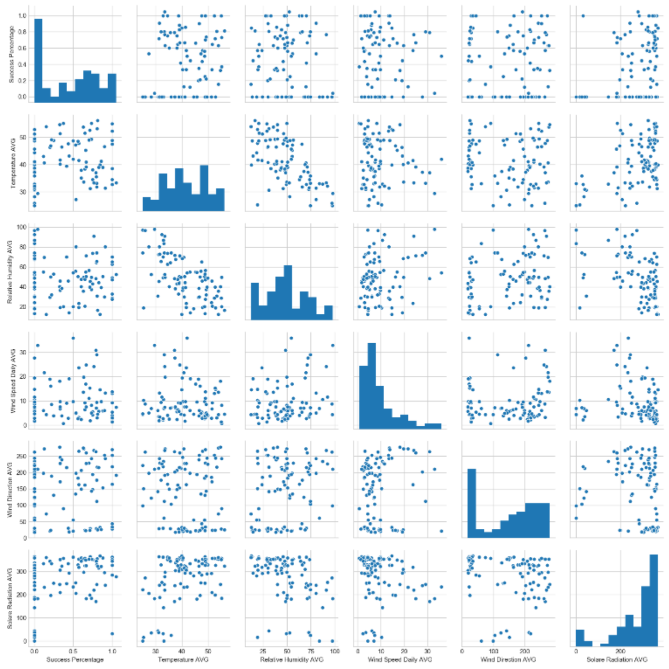


Figure 4: Route 2's feature selection plots

For Polynomial Feature Selection, we were allowed to use only 12 columns. The approach we used to achieve the best combinations of polynomial parameters is that we first observed the correlations from the precious plots and appended squared and cubed columns of the features that were highly correlated with the success rate and with each other and then by again plotting their covariance, we eliminated the features for which the correlation with respect to success rate dropped. The final features selected were again plotted to show how only the highest correlated features were selected. The final selected features for polynomial regression can be found in figures 5 and 6.

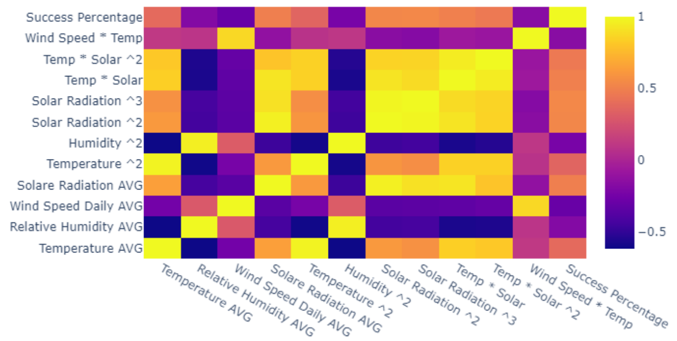


Figure 5: Covariance matrix - selected polynomial features (Route 1)

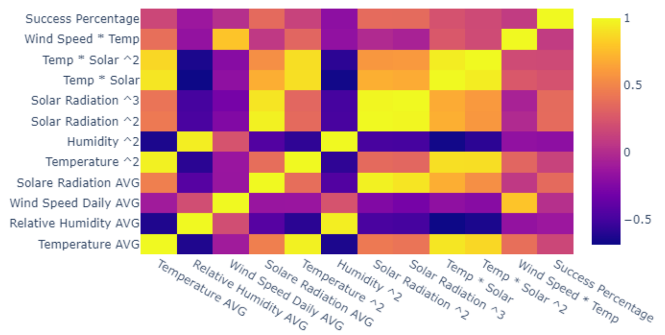
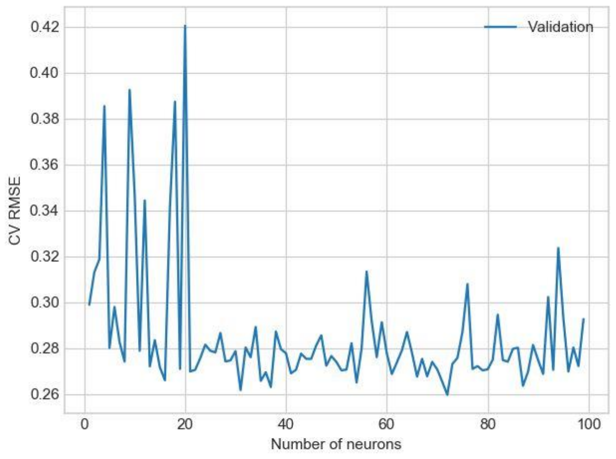


Figure 6: Covariance matrix - selected polynomial features (Route 2)

C. *Machine Learning and Model Selection.*

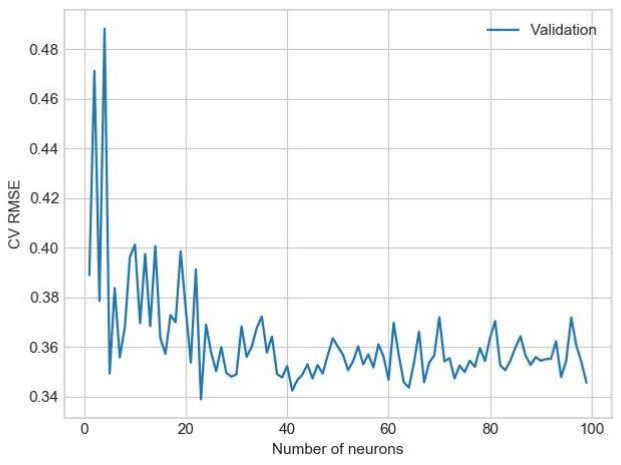
Splitting the data: The first measure we took was to make sure the metric for comparing the performance of each of the models should be measured on the same set of test data. Hence, we first kept 20% of test data from the preprocessed data. Now, for training, the same 80% data will be used across all the models. Linear Regression and Polynomial Regression was done on the train data and the performance was reported on the test data.



```
opt num_neurons : 71
Min RMSE : 0.25964394181944916
```

Figure 7: Route 1 linear features LOOCV RMSE vs no. of neurons

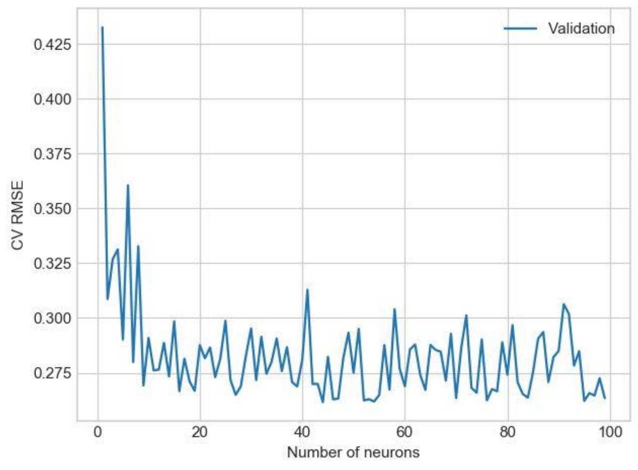
Leave One Out Cross Validation: LOOCV is a very useful process for deciding hyperparameters in models like the Multi-Layer Perceptron. To achieve this, we first used the K-Fold function to split the train data into 5 equal portions randomly. We then run the model by leaving 1 portion and storing the RMSEs of the model's performance on the left-out portion. This is done 5 times for each fold and then the mean of the final list of the 5 cross validation error is used to tune the hyperparameters.



```
opt num_neurons : 22
Min RMSE : 0.3389247094972335
```

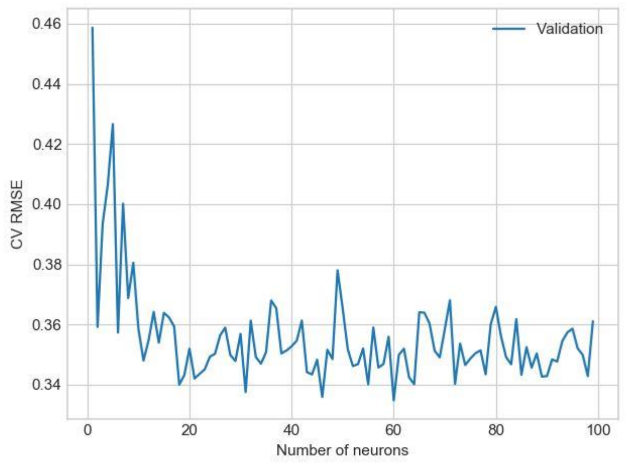
Figure 8: Route 2 linear feautures LOOCV RMSE vs no. of neurons

We used this method to find the number of optimum neurons in the hidden layer of our MLP, for which the mean cross validation error was the least. The plots and the selected number of neurons can be seen in figures 7, 8, 9 and 10 for each route and linear and polynomial features.



```
opt num_neurons : 43
Min RMSE : 0.2616522523275326
```

Figure 9: Route 1 polynomial features LOOCV RMSE vs no. of neurons



```
opt num_neurons : 59
Min RMSE : 0.33476402871646443
```

Figure 10: Route 2 ploynomial feautures LOOCV RMSE vs no. of neurons

III. RESULTS

Each model's performance was measured by the training error, cross validation error and test error.

Following are the tables of results for each route's regression models for predicting the scores.

Route: Disappointment Cleaver			
Model	RMSE		
	Train	Test	Cross validation(mean)
Linear Regression	0.2642616	0.27212845	-
Polynomial Regression	0.249164469	0.24916484	-
Multi-Layer Perceptron - Linear Features	0.259237473	0.2711288	0.263466852
Multi-Layer Perceptron - Polynomial Features	0.25166658	0.25311049	0.261652252

Route: Emmons-Winthrop			
Model	RMSE		
	Train	Test	Cross validation(mean)
Linear Regression	0.332937707	0.39642939	-
Polynomial Regression	0.319637701	0.37146081	-
Multi-Layer Perceptron - Linear Features	0.333076891	0.39660781	0.338924709
Multi-Layer Perceptron - Polynomial Features	0.322556452	0.38496061	0.334764029

Each model's performance can be seen from a 3d plot of how well the model fits the data. These can be seen in the figures below.

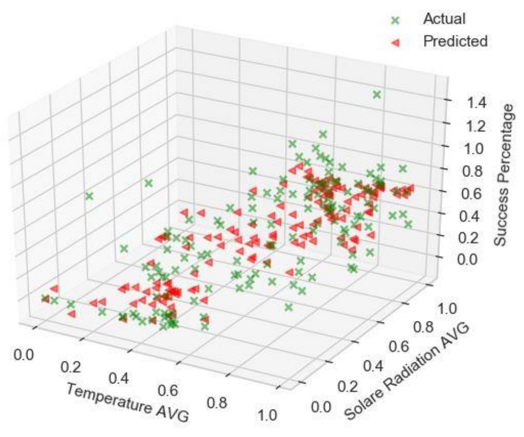


Figure 11: Route 1 Linear Regression with linear features

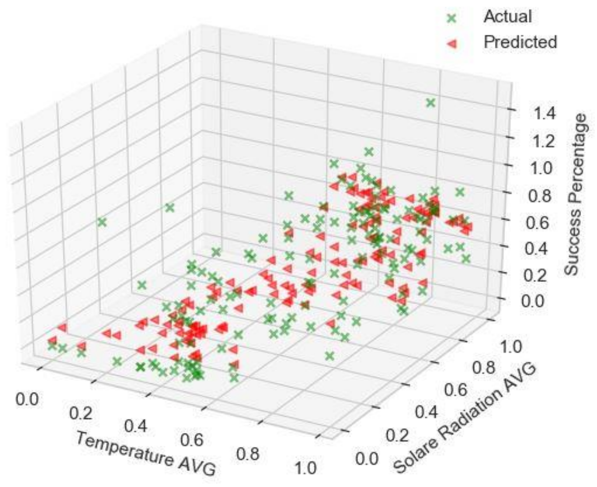


Figure 12: Route 1 Linear Regression with polynomial features

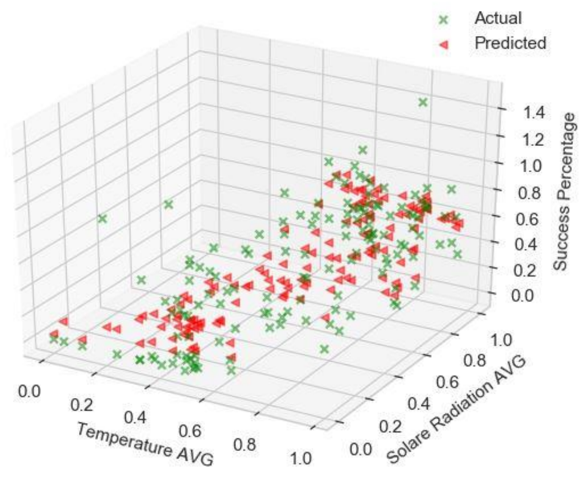


Figure 13: Route 1 Multi-Layer Perceptron with linear features

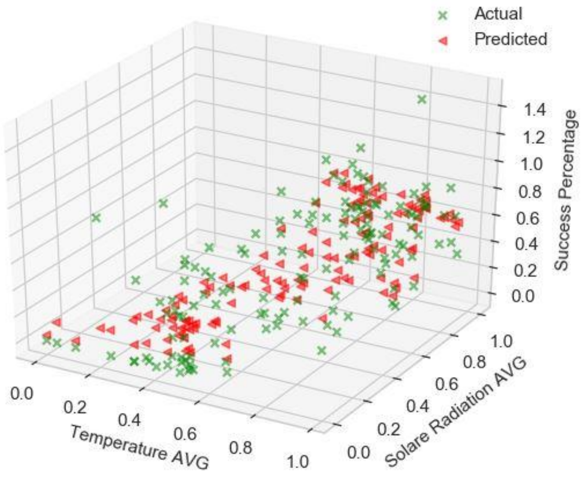


Figure 14: Route 1 MLP with polynomial features

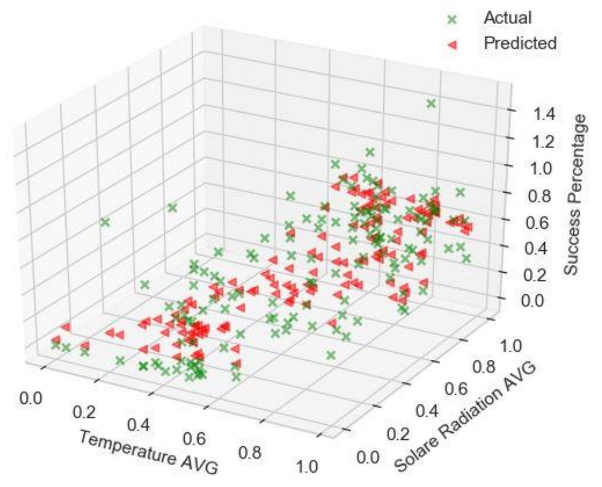


Figure 15: Route 2 Linear Regression with linear features

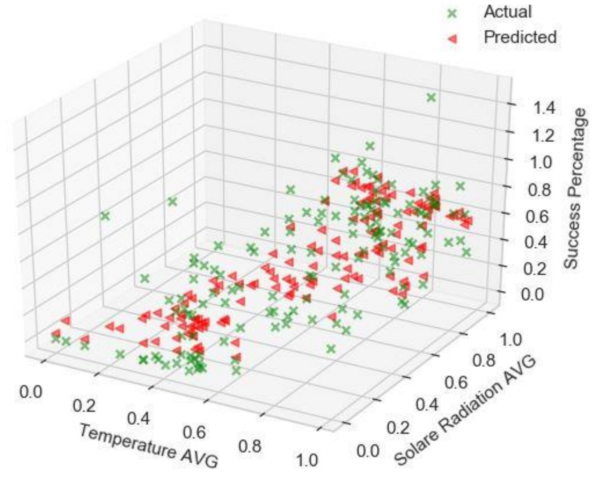


Figure 16: Route 2 Linear Regression with polynomial features

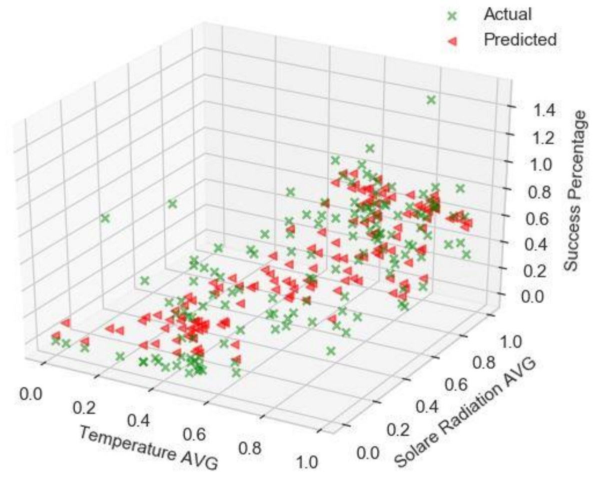


Figure 17: Route 2 Multi-Layer Perceptron with linear features

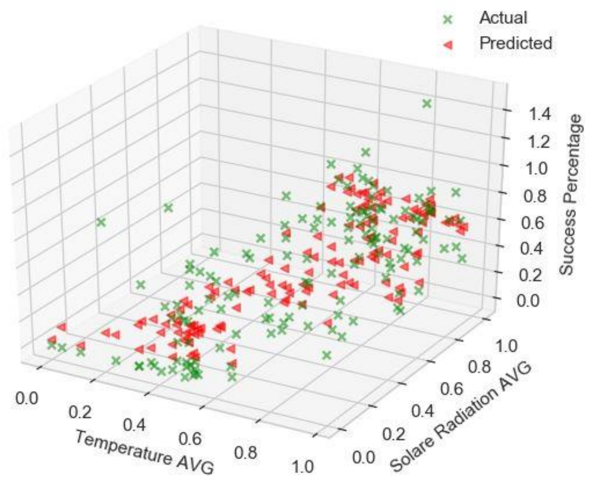


Figure 18: Route 2 MLP with polynomial features

IV. Conclusion

By performing the above procedures, it can be inferred that, a real-life data set might not be in the most usable form for naively training machine learning models. They need to be well analyzed and processed before being fed to the models for learning.

Machine Learning models perform best, only if the hyperparameters involved in the training process, are well tuned. The importance of Leave One out Cross Validation achieved by implementing K-Fold cross validation can be observed from the results.

Lastly, by comparing the models' performances, it can be inferred that, the data that is not perfectly linearly separable can be fit by using models learnt on the polynomial functions of the features. Hence, the performance of polynomial Regression models for both Linear Regression and MLP performed better than their counter parts.