Statement of the Problem:

Given the dataset **Higgs_**1, my goal was to predict whether the signal was noise or a particle.

Description of the dataset:

The dataset consists of 550,000 data points with each datapoint comprising of 28 features. I also had a label attached that tells us whether it's Noise or a Particle. There are 291854 instances of Particles and 258146 instances of Noise.

The first step, I did some exploratory data analysis, and I was checking the data for any Outliers or High Leverage points, and I removed them. Then, I visualized the data to look for any trends.

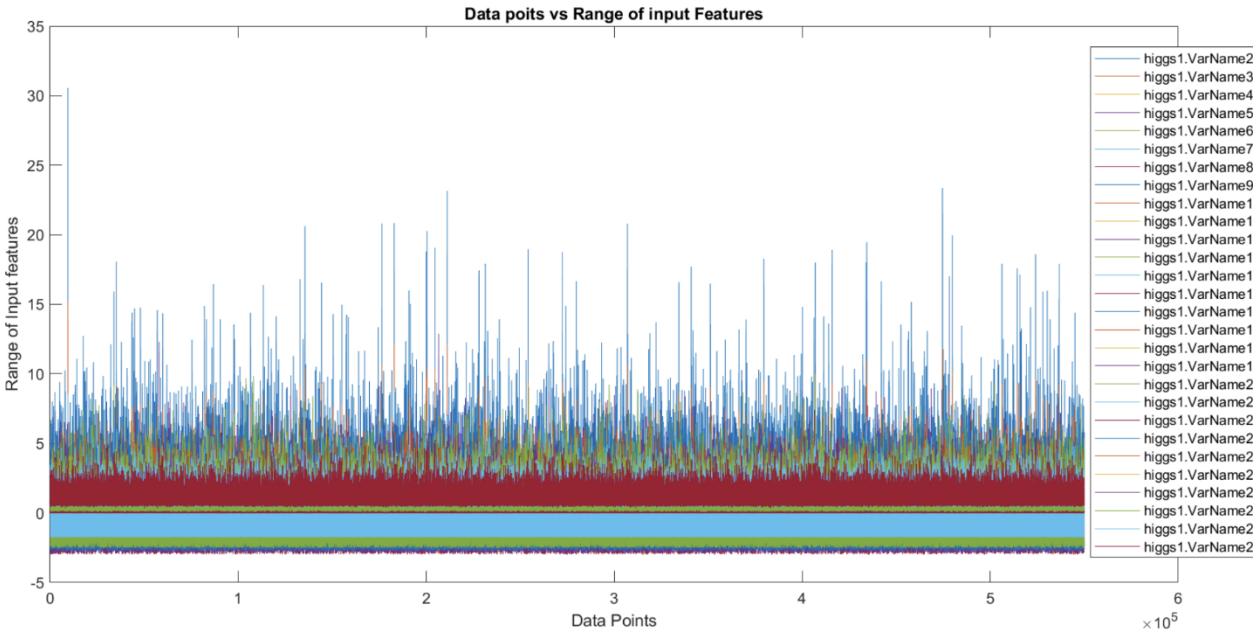

Figure 1 – Data points Vs Input Features

The above plot shows the variations in input features. This plot forced us to scale the data since different input features covered ranges. So, the scaling justifies that even though input features cover different ranges, I can bring them all back to a similar range.

Methodology:

My goal throughout the project was to predict the output of the "Higgs_1" problem as accurately as possible. So, I needed a good model that could learn from the historical data and could predict the results with accuracy.

I was aware that I was working with a very big size of data, with a dimension of 550000*28 and it will be hard to try the various combinations of Hyperparameters manually.

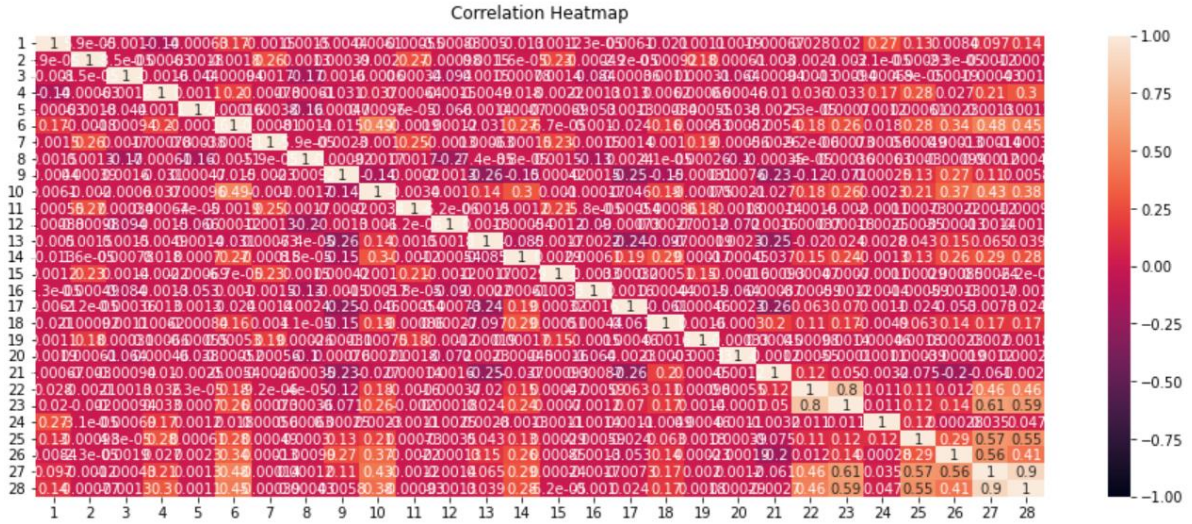

Figure 2 – Correlation Heatmap

Here in figure 2, I checked the correlation between the input features. This helped us to remove the input features which are correlated between them.

I first tried doing the simple model with 1 hidden layer using the “Relu” activation function and “GlorotNormal” initializer and 20 nodes in the hidden layer. The output layer used the sigmoid activation function. The accuracy I was looking at was 74.57%. So, I tried working with a rather complex model with 4 hidden layers and 5 dropout layers. The accuracy maxed out at 69%. So, a deep network did not seem like a great way to proceed. I then experimented with drastically increasing the number of nodes. I built a model with 200 nodes in the hidden layer, using the “relu” activation and GlorotNormal as the initializer. The output layer used the same sigmoid activation function. I also added a dropout layer to randomly reduce overfitting. This model provided us with an accuracy of 70.9%, which again was not an improvement on my simpler model. So, I added another hidden layer to my model with the same configuration. This provided us with an accuracy of around 71%. As the larger networks were more computationally intensive, I decided the tradeoffs and similar accuracy made them not worth investigating further.

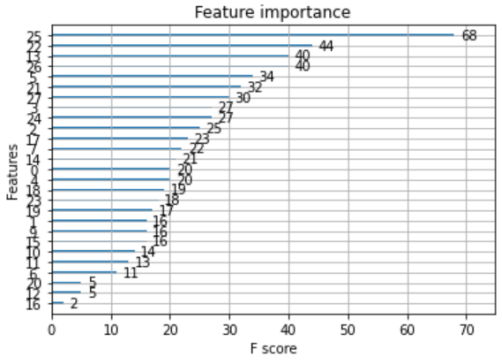

Figure 3 – Feature importance plot I obtained from XGBoost classifier

The above feature importance plot helped us to understand which input features are playing key role in predicting the output. This also helped us in performing dimensionality reduction.

It became clear that there were vast number of different combinations of parameters to be looked at. So, I tried the GridSearchCV method to automate the process of trying out the different parameters. This way I was trying to eradicate the issue of feeding different combinations manually. For example, if I was trying to work with 10 different combinations, which itself will be 10! Times of models to be run manually.

I used the GridSearchCV function with a small subset of my datapoints to get a general idea of the type of network model I should be building. After running GridSearchCV to produce around 300 models, the best performing model was: This gave us an accuracy of 58.9%. Therefore, I decided that there was still some room for improvement.

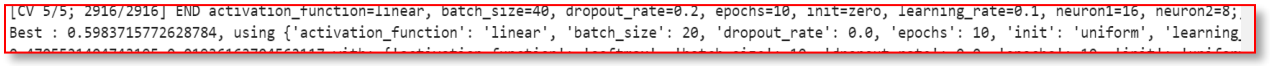

Figure 4 – Results from GridSearch CV

Since I had a very large dataset with lots of features, I decided to try to find the most impactful features using PCA and XgBoost. PCA can help reduce the dimensionality of my data while projecting the dataset to the maximum variance in the features. I used the PCA function from sklearn to transform my dataset and I wanted to reduce the dimensionality by half to 14 features and so I set the n_components parameter in the PCA function to 14. I used the transformed data in my best performing model so far, with the resulting accuracy of 63.07%, a large drop in performance. Therefore, I then used PCA to transform the data, while keeping the original dimensionality but transforming the data to the highest variance possible as reflected in the dataset. However, this also failed to improve results appreciably, plateauing around 71% accuracy. Therefore, this approach also did not produce any appreciable improvements.

Conclusion:

My experiments showed that simpler models can often outperform complex ones even with very large datasets. The best model I decided to choose was the first model I tried, that being the one with one hidden layer using 20 nodes, the “relu” activation function and the “GlorotNormal” initializer. The output layer used the sigmoid activation function and had 1 node. This model was simple, took little time to run, and produced accuracy scores that were not beaten by other models.

If I could have more time and a high-performance computer, I could have run the Grid Search CV for the whole dataset and could have tried different hyperparameter tunings. Due to this limitation, I tried Grid Search CV for a very small amount of data.