Kaggle Competition

Team Name: Weder Praedictio

Student #1: Avinash Nath Aita 88289773 **Student #2:** Anirudh Rachuri 47365112

Problem at our hand:

We are given data containing measurements of cloud temperature. Each data point corresponds to a particular lat-long location where the model thinks there might be rain; the extracted features include information such as IR temperature at that location, and information about the corresponding cloud (area, average temperature, etc.). We have to build a classifier trained on this data to predict the presence or absence of rainfall at a particular location.

Feature Importance and Selection:

For simplifying the purpose of selecting features, we only do a held out validation rather than cross-validation. Cross-validation is done at a later phase when the analysis is done.

Removing features with low variance:

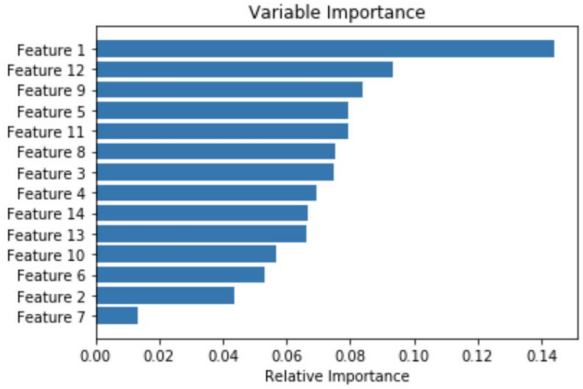
VarianceThreshold is a simple baseline approach to feature selection. It removes all features whose variance doesn't meet some threshold. By default, it removes all zero-variance features.
Here we observe that on a range of threshold values, all features seem to pass the threshold. Hence, no dimension reduction is performed based on this approach.

Feature Importance - Random Forests

Feature Importance: Importance/Significance of feature towards the end goal of classification /regression.

Random forests are among the most popular machine learning methods because of their good accuracy and easy to use. They provide two straightforward methods for feature selection:

1. Mean decrease impurity: Based on the information gain from a feature every tree is generated. Thus when training a tree, it can be computed how much each feature decreases the weighted impurity in a tree. For a forest, the impurity decrease from each feature can be averaged and the features are ranked according to this measure.
2. Mean decrease accuracy: Direct measure of impact by each feature on accuracy. Idea is to permute the values of each feature and measure how much the permutation decreases the accuracy.



Observations:

1. Feature 1- most influential
2. Feature 7- least influential
3. All other features contribute to some extent towards the classification.

To get a greater idea, we explore these combinations and compare the MSE or AUC accuracies for these models.

Testing Feature 7 removal: Random Forests - Training data set: 20,000 (10%)

- Least Influential feature- After feature removal: AUC: 0.61074784551386874

Here, we observe that AUC has increased. This might be due to the sampled validation partition of the data. But, nevertheless the AUC has not decreased dramatically which proves less significance of Feature 7.

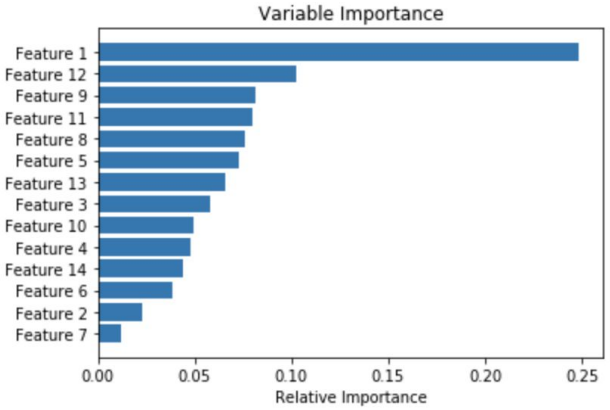
- Highest Influential feature- After feature removal: AUC: 0.60955984729155188

Here, we observe that AUC has decreased - Just as per our understanding from learning feature importance!

Exploring Regression- Random Forests:

Since, classification into 2 classes strictly, is not desired in our case: We need to only predict the probability of rainfall.

Therefore, using regression would not be a bad idea. Random Forest Regression and Linear/Polynomial regression is done to evaluate the above mentioned feature importance



Similar to Random Forest Classifier based approach for feature importance, feature importance can be calculated in case of a forest regressor also. The above graph details the results. Observation: The results are consistent with results of Random Forest Classifier.

Highest Influential feature Only: AUC: 0.628102712491 MSE: 0.215221822778

Least Influential feature Only: AUC: 0.479493541985 MSE: 0.250831635573

Here, we can clearly observe that, least influential feature when considered for AUC has less than 0.5 and hence is making no contribution to the overall regression.

We can also see that most influential feature has AUC comparable to that of all features! This fact can be used for exploration of using only 1 most influential feature for regression (used for classification though) for regressors with proper modelling other than random forests (because of their immunity towards feature importance).

Linear Regression

Since forests are immune to feature selection, we now explore it's effect on Linear regressor.

Original Feature Space: AUC: 0.669752488935 MSE: 0.205709808436
Here, the chosen degree - 2, is based on local search on range (1,5) which gives out a better AUC.

Least Influential feature only: AUC: 0.524373775208 MSE: 0.225620489899

Here, we can clearly observe that picking this feature alone will decrease the performance drastically.

Highest Influential feature only: AUC: 0.633878813268 MSE: 0.214234886932

Here, picking only 1 out 14 dimensions lead to comparable performance with respect to original feature space. Therefore this feature can be considered as most influential - just as proved in Random Forests!

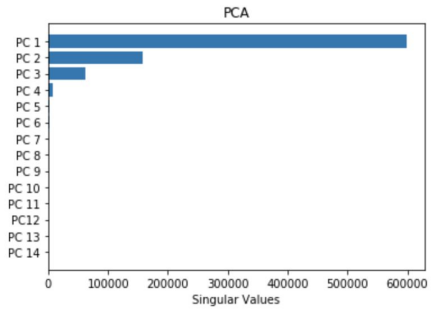
Also, there might be feature correlation for features other than feature 1(most influential) feature 7(least influential). We continue our study further by feature correlation and PCA - dimensionality reduction.

From this discussion we conclude that all features are not required for contribution towards the classification problem at hand. Therefore, the 14 dimensional space can definitely be reduced to smaller dimension.

PCA - Dimension reduction

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

PCA is done on our dataset to retrieve principal components and the singular/eigen values. These values can therefore determine the effective number of dimension/features enough for future purpose of classification/regression.

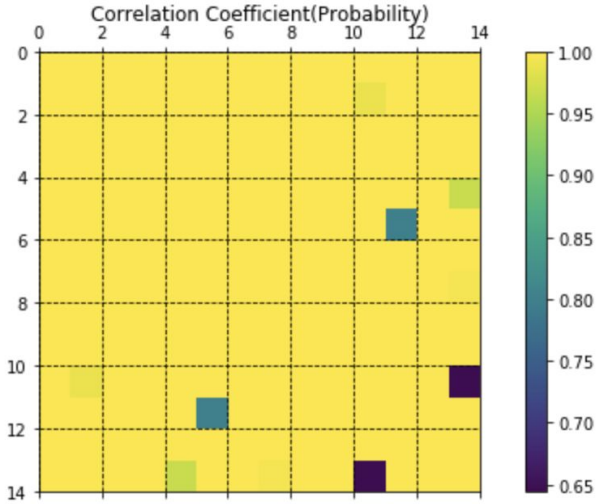


Here we observe that top 3 or 4 principal components have high significance than the others! We'll continue on feature correlation to confirm this phenomenon and later use top 3/4 principal directions for classifier tuning.

Feature Correlation: To observe the extent of inter-feature correlation, Correlation coefficients for each pair of features is computed and a matrix plot is made.

We use Pearson correlation coefficient measures - the linear relationship between two datasets. Like other correlation coefficients, this one varies between -1 and +1 with 0 implying no correlation. Correlations of -1 or +1 imply an exact linear relationship.

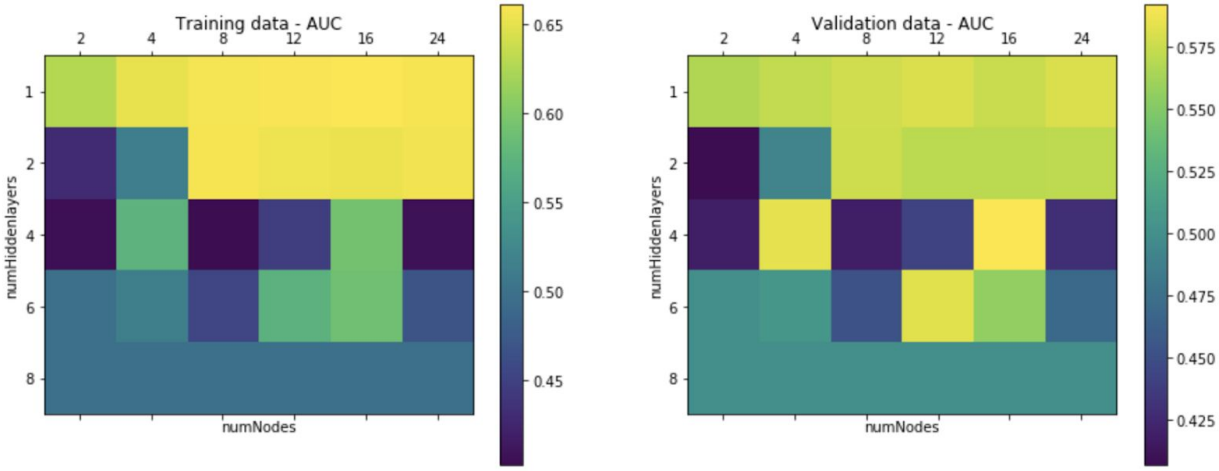
Along with these coefficients, a p-value roughly indicates the probability of an uncorrelated system producing datasets that have a Pearson correlation at least as extreme as the one computed from these datasets.



Here, we can observe the correlation is very high among features. We conclude that many of the features have high correlation and hence not required for classification as they are redundant. As per the earlier PCA analysis top 3 to 4 features are unrelated which will be picked for further detailed study. This concludes our study on dimensionality reduction and feature engineering.

Effect of PCA on Neural Network (MLFFNN):

Transform Data to top 4 PCA dimensions:



From the above plots (Validation data - AUC), we observe that highest AUC is observed at numHiddenlayers = 1 and numNodes = 16. This combination of parameters gives best continuous AUC in both train and validation data. (There are other validation AUCs with higher values but are not chosen because of the discontinous behavior.)

Also, we observe that, the next highest AUC is observed at numHiddenlayers = 2 and numNodesinHiddenLayer = 8. Therefore, a total number of 16 hidden nodes. We can conclude from these observations that, any of these parameters can be chosen.

PCA MLFFNN Conclusion:

The above mentioned parameters of 16 nodes in 1 layer and 10 nodes in 2 layers are chosen as the best parameters. Based on these parameters, the whole of training data set is trained with PCA applied.

Interesting Observations:

- The result of PCA utilized for Neural Networks tuning leads to almost the same model as concluded in Assignment 4 (without PCA and dimensionality reduction).
- This indicates that, Neural Network model does not change with PCA/dimensionality reduction which implies
    1. PCA did not effect on the overall prediction (classification/regression)
    2. PCA can be done and then equivalent results can be achieved with significantly less amount of data (here, 10/14) are reduced.

XGBoost

Why XGBoost?

XGBoost is short for “Extreme Gradient Boosting”. It uses boosted trees as model. Since it uses decision trees at the core of its algorithm, feature importance and feature selection is taken care by XGBoost ie., each feature is evaluated as a candidate for splitting the data, which makes them robust to unimportant/irrelevant features. It is robust to overfitting the data because it uses ensemble of weak classifiers to build a complex classifier. Moreover, XGBoost is computationally extremely fast.

Parameters:

We used 'binary:logistic' as objective function so that we get probabilities as output instead of 0 or 1.

max_depth: Maximum depth of a tree. Higher the max_depth, more the complex the model is.

min_child_weight: Minimum number of instances needed to be in each node.

eta: Learning rate used in shrinking the feature weights after each boosting step.

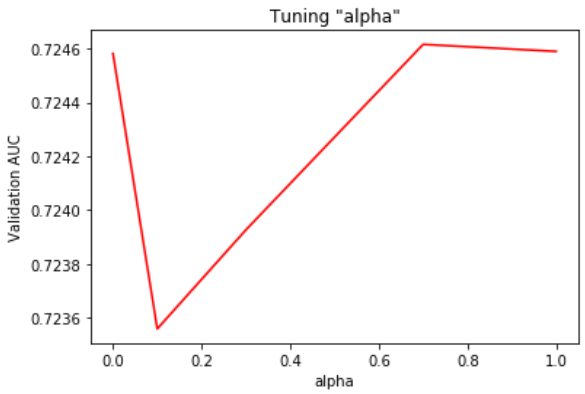
alpha: Regularization term to make it less overfitting.

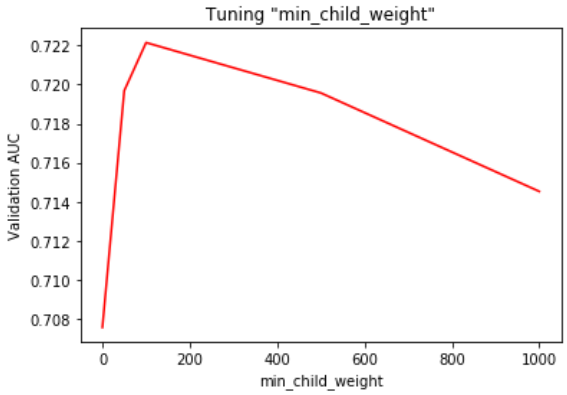
num_round: Number of rounds of boosting.

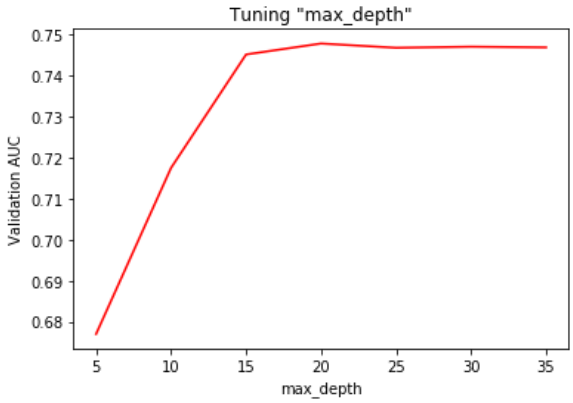
Parameter Tuning:

We used auc as evaluation metric in tuning the above parameters. We shuffled the whole input data and kept aside 20% of it for validation. We started with default values for parameters (with num_round = 100) and tried tuning them such that it maximizes the validation AUC.

We select values - max_depth = 20 from the graph 1, min_child_weight = 100 from the graph 2, alpha = 0.7 from the graph 3.







While tuning eta, we have to note that reducing eta, demands increase in num_round. So, we set num_round as 4000 and we will stop training if validation AUC doesn't improve in 50 rounds. From this, we can find the best round with highest validation AUC for each value of eta. We select eta = 0.1 from the below table.

eta	**Best iteration**	**Validation AUC**
0.1	1686	0.792076
0.3	750	0.791395
0.5	578	0.789102
1	250	0.776265

The above tuning gave us insights of what should be the parameter values. After this, we went ahead and fine tuned to maximize AUC. The final parameter values that we got are - min_child_weight = 100, max_depth = 15, eta = 0.15, alpha = 0.75, num_round = 2000. This gave us AUC's of **0.75177** and **0.75041** on public and private leaderboards respectively.

Libraries majorly used: sklearn, xgboost, matlibplot

Individual contributions

1. **Feature Engineering** - Ideation Phase: Both - Implementation Phase: Avinash
2. **Xgboost (Boosted Trees)** - Ideation Phase: Both - Implementation Phase: Anirudh