**ALY 6020 PREDICTIVE ANALYTICS**

WEATHER PREDICTION IN KANSAS CITY

Final Project Report

ALY 6020, Sec 03 Fall 2018

(October 27, 2018)

Submitted By:

Venkata Kaushik Burra

Submitted To:

Andrew Long

Northeastern University, Boston, MA

Master of Professional Studies in Analytics

### **Table of Contents**

Introduction

Project Definition

Data Source

Data Cleaning

Methods

Conclusion

References

### 

**Introduction**

Weather is always changing, it never remains static. Various factors play an important role for the dynamic state of weather. Such factors are Wind, Temperature, Humidity, Dew point, Precipitation etc. To predict the weather outcome is a very important task which can help people be aware of the weather that they might embrace and take responsive actions accordingly.

Predictive analytics plays an important role in determining the weather outcome based on the factors that contribute to the dynamic state of weather.

In this project, we are using Predictive analytics in Python to determine whether the outcomes “Events” will either rain or snow based on the input variables (features) such as “Temperature, Dew Point, Humidity percentage, Sea level, Visibility, Wind speed and Precipitation Index “.

### **Project Definition**

To Predict if the outcome Event will either Rain or Snow based on the input features.

### **Data Source**

This data set is retrieved from Climate of Kansas City, <https://www.ncdc.noaa.gov/data-access>.

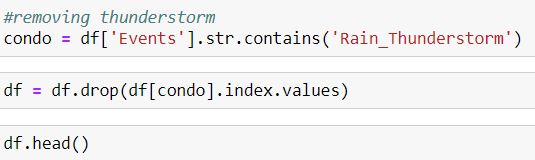
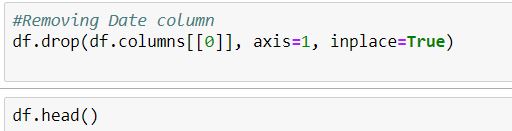
The dataset consists of seven features Temp.F, Dew\_Point.F, Humidity.percentage, Sea\_level\_press.in, Visibility.mi, Wind.mph and Precip.in which are independent and numerical, the dependent variable outcome Events which are Snow, Rain and Rain\_Thunderstorm, all together with 366 observations.

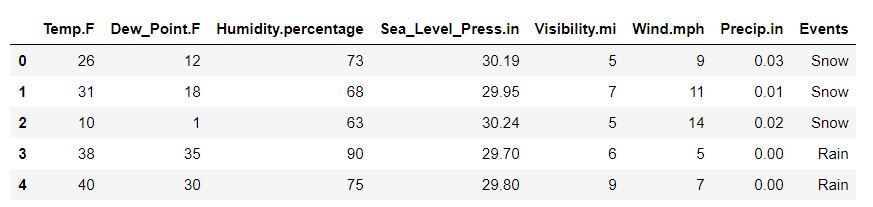
### **Data Cleaning**

This dataset required slight cleaning. I filtered the Rain\_Thunderstorm and I filtered the date column to work with my analysis.

**Hence the new data for our analysis consists of 226 observations.**

Below is the python input for the Data Cleaning.





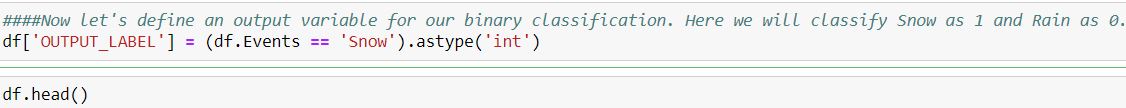
### **Methods**

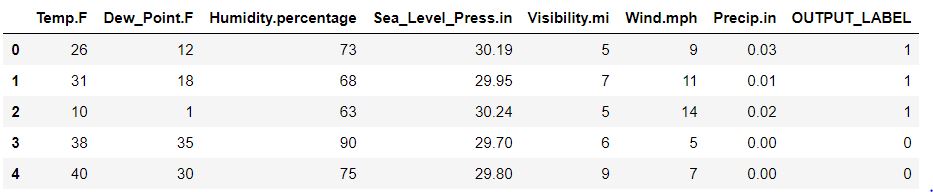
I began by exploring the dataset, then I performed Featured Engineering to select the required features for the predictive analysis, further I build the model by splitting the data into Training set, Validation set and Test set respectively. Then I performed model selection where I utilized various classification models for my analysis to predict the outcome Events and evaluate the performance metrics for the Training and Validation sets. Finally, I evaluated performance metrics for the Test set of the best model that I selected for my analysis.

### **Data Exploration**

After importing the data sets and Performed Data Cleaning, I went ahead to perform binary classification for the outcomes Events. I classified the Events Snow as 1 and Rain as 0 for our analysis.

Here is the Python code below for the binary classification.



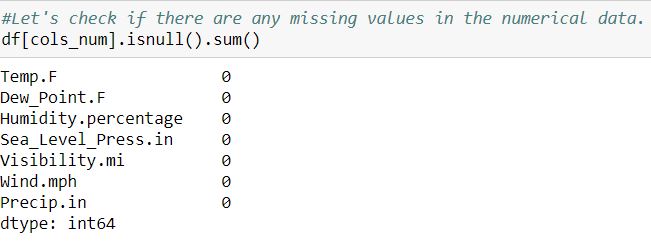


### **Feature Engineering**

There are 7 features in my dataset Temp.F, Dew\_Point.F, Humidity.percentage, Sea\_level\_press.in, Visibility.mi, Wind.mph and Precip.in, which are **independent and numerical.**



We checked weather the numerical features have any missing values.



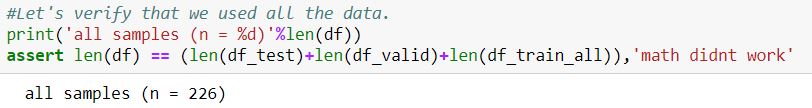
### **Building the model and fitting into Training, Validation and Test sets.**

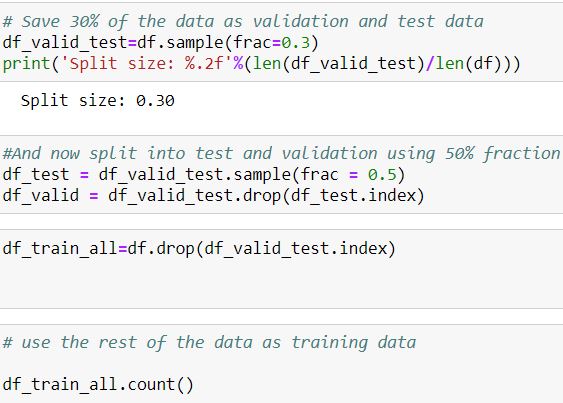
Our new data has 226 Observations after cleaning.

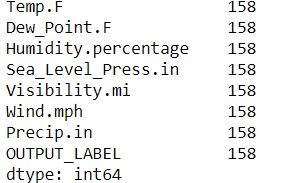
To use Regression analysis for our models, we need to first build our model.

We fit our model by splitting 30% of our data into Test and Validation set. Again, we then split 50% of that 30% data as Test and Validation sets respectively. Thereby we are going with 70-15-15 split as Training, Validation and Test sets for our analysis.

Hence the number of observations for Training set are 158 and 34 each for Validation and Test sets.

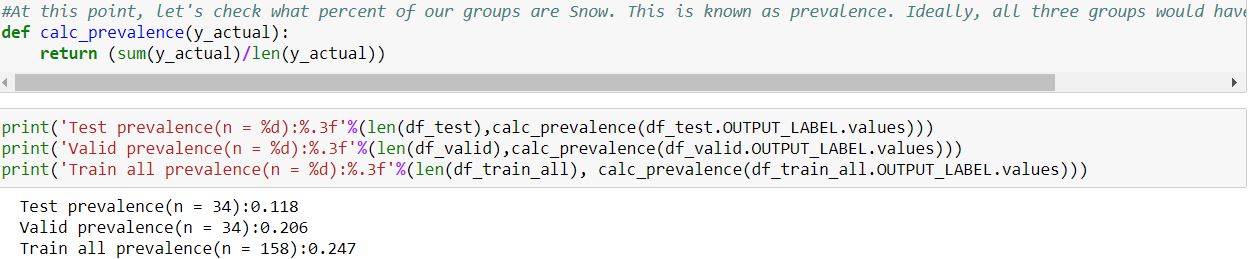






Prevalence Test:

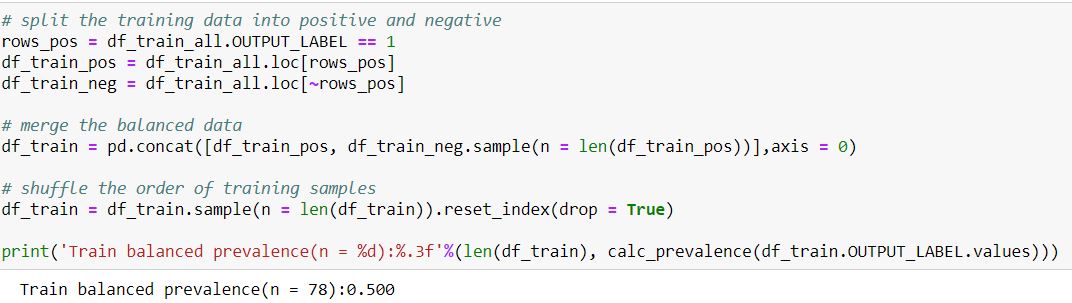
Further I tested the Prevalence to check what percent of our groups are Snow. Ideally all the three groups would have similar Prevalence.

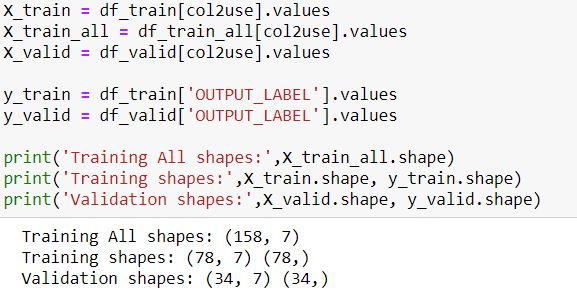


Further I saved the Split data which can be imported for further analysis.

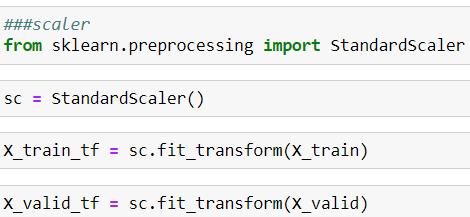


Further, I went ahead to balance and Transform (normalize) the data which is good for our analysis performance.





Additionally, I utilized scalar function to normalize the inputs so that there is no variance in sizes.



**Model Selection**

For my analysis, I trained various models such as Logistic Regression (K Nearest Neighbors),

Random Forest, Stochastic gradient descent, Naive Bayes, Decision Tree and Gradient Boosting classifier and evaluated their Performance metrics.

Before I start with training the models, let us set a threshold as 0.5 for the predicted output to be positive.

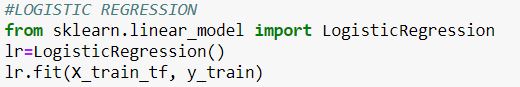


**Logistic Regression:**

Logistic Regression is primarily used to predict the outcome variable which is in categorical context. Here the input predictors are continuous and numerical. Logistic regression utilizes logarithmic functions to the output variable which makes us to analyze a non-linear model in a linear way.

Here I used the dependent outcome Snow = 1 and Rain = 0 as output and numerical, independent features as input.

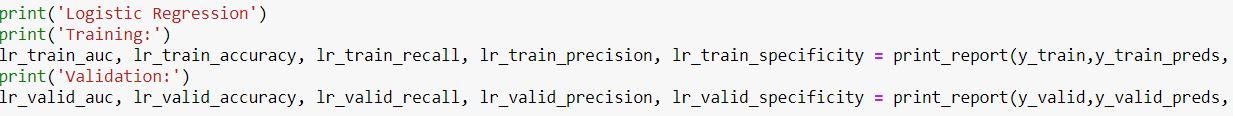
First, I import the library for the Logistic Regression and fit it to the training sets.

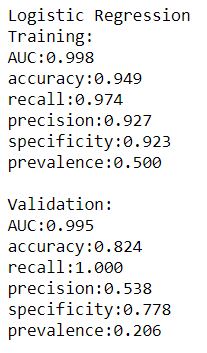


Then I tried to predict the values from the Training and Validation sets using the probability function from the input features.



Then I checked the evaluation metrics such as Accuracy, Precision, Recall, Specificity etc based on the predicted values.





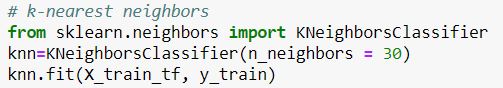
From the above metrics we analyze that the Accuracy (82.4), Recall (100%) and Specificity (77.8%) have a good set of results for the predicted model.

Then finally we plotted the ROC curve for the True Positive Rate and False Positive Rate for the Validation and Training AUC values.

**KNN Regression:**

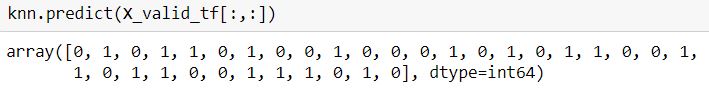
KNN (K nearest neighbors) are used primarily for the classification regression. The value of the factor k determines the number of nearest neighbors in the algorithm.

First, I tried to import the KNeighbors Classifier library and fit the values from the training set.

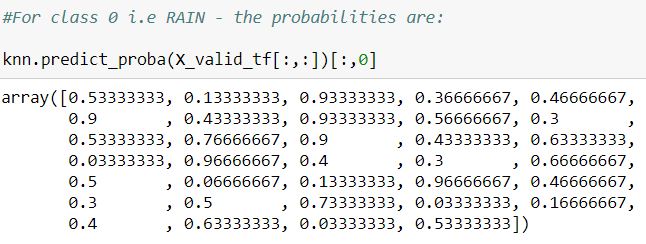


I selected the number of neighbors as 30 as shown above.

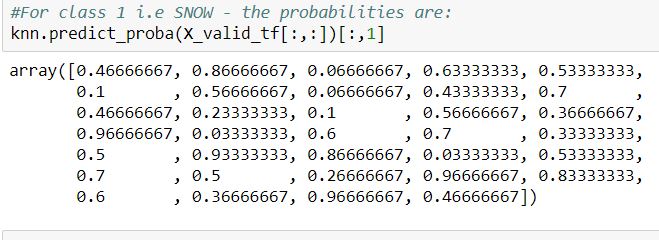
Then I used the predict function to predict the values from the Validation set. Additionally, I also used the Probability function to predict the probabilities for the Validation set.



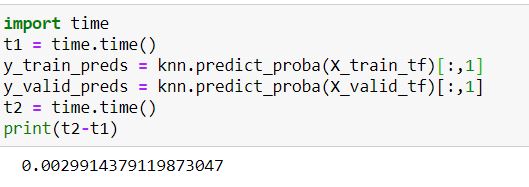
Then I used the probability function to predict the probabilities for the Rain values in the output.



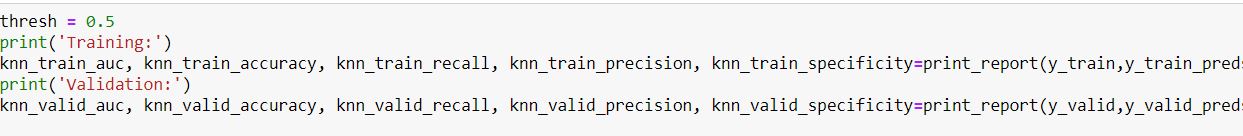
Similarly, I used the probability function to predict the probabilities foe the Snow value in the output.

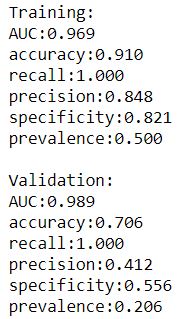


Then I used the time function to test how long the predictions take place.



Then finally, I evaluated the metrics for the Knn classifier model.





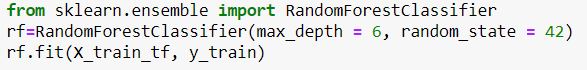
From the above analysis we can see that the Accuracy (70.6%), Recall (100%) have the high values which is good predictions.

**Random Forest:**

Random Forest is primarily used to reduce the overfitting and variance for a subsample by fitting various decision trees.

The error is reduced with the increase in number of trees.

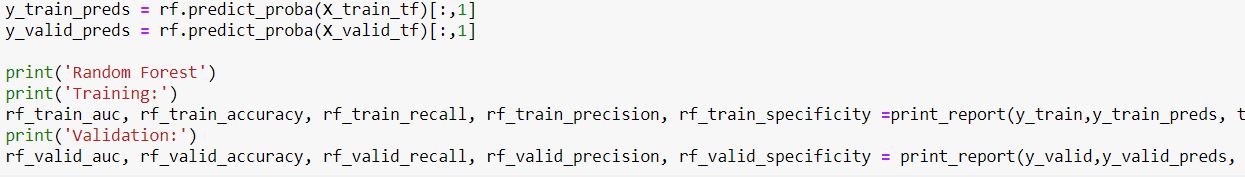
Initially we fit the Random Forest classifier for the training set values.

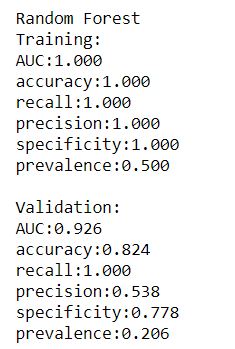


We are taking the number of maximum depths equal to 6, the greater number of depths, the better the performance.

Then we are evaluating the performance metrics for the Random Forest.

I’m calculating Probability of predicting 1 (snow).





From the above output, we can analyze that the Accuracy (82.4%) . Specificity (77.8%) and Recall (100%) have the high values and that is good for the model.

**SGD Classifier: (Stochastic Gradient Classifier)**

SGDclassifier is a simple and effective model method to use logarithmic functions in performing the predictions.

As other classifiers, SGD must be fitted with two arrays: an array X of size [n\_samples, n\_features] holding the training samples, and an array Y of size [n\_samples] holding the target values (class labels) for the training samples:

In our analysis, first we import the SGD classifier and fit it to our training set values.



Then we find the evaluation metrics for the SGD classifier.

The outcome is as follows:

Stochastic Gradient Descend

Training:

AUC:0.991

accuracy:0.956

recall:1.000

precision:0.919

specificity:0.912

prevalence:0.500

Validation:

AUC:0.995

accuracy:0.824

recall:1.000

precision:0.571

specificity:0.769

prevalence:0.235

We can see that from the above metrics, the Accuracy is 82.4% which is again indeed a good result. The recall is 100% and Precision is moderate with 57.1 %.

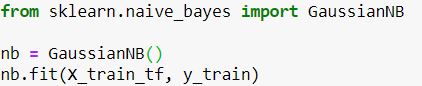
**Naive Bayes**:

The Naive Bayes Classifier is based on the Bayesian theorem and is well suited for high inputs dimension. Since our dimensions are less from our data, we might not get a good metrics results from this modelling as it supports high dimensionality inputs.

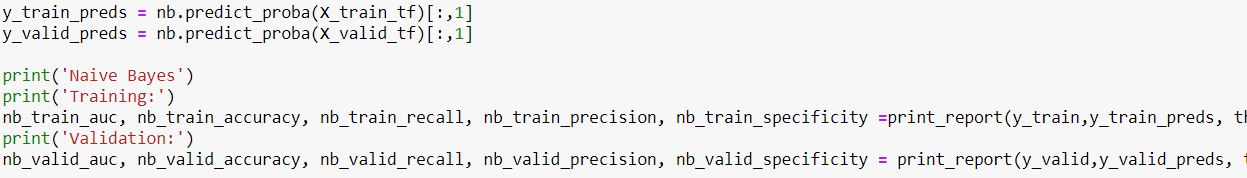
GuassianNB implements the Gaussian Naïve Bayes algorithm for classification.

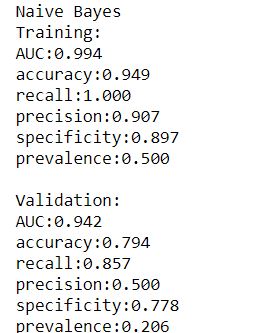
I’m calculating Probability of predicting 1 (snow).

In our analysis, we first import the Naïve Bayes and fit it to the training set values.



Then we evaluate the performance metrics for this model.





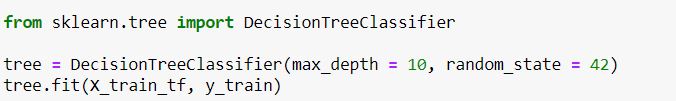
From the above metrics, we can see that the accuracy is 79.4 % which is fairly a good value, recall with 100% and Precision is moderate again with 50%.

**Decision Tree:**

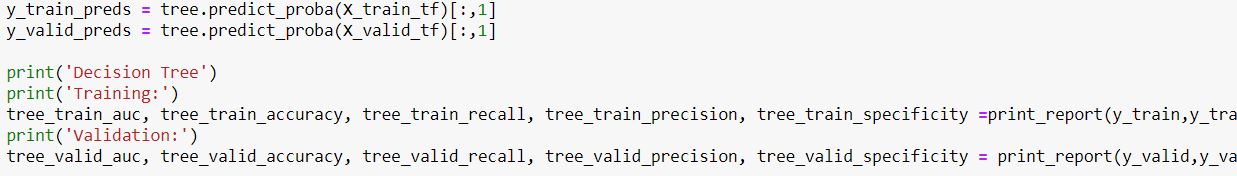
**Decision Trees are supervised learning methods used for classification and regression.**

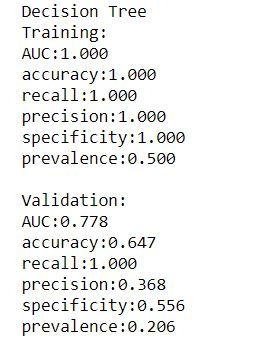
The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

For our analysis I imported the Decision Tree classifier and fit it to the Training set values.



Then I evaluated the performance metrics for this classifier.



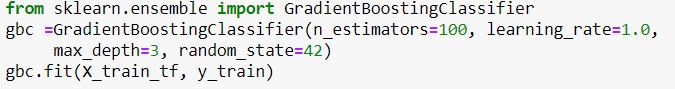


From the above metrics, we can see that Accuracy is a bit low with 64.7 % and Precision with 36.8%. This is not an ideal model for performing predictions for our analysis.

**Gradient Boost Classifier:**

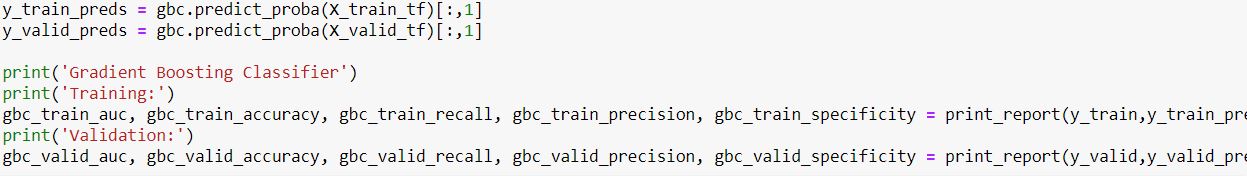
GB model builds in a much-optimized manner. In each stage n classes regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. Binary classification is a special case where only a single regression tree is induced.

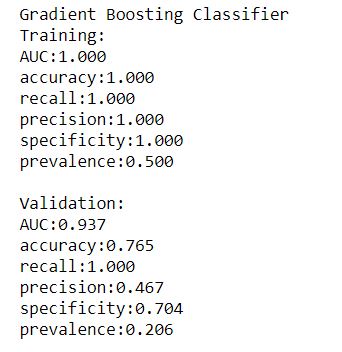
In our analysis first, I import the Gradient Boost Classifier and fit it to the training set values.



We can see that the number of estimators is 100 in our case with max depth = 3.

Then we evaluate the performance metrics.

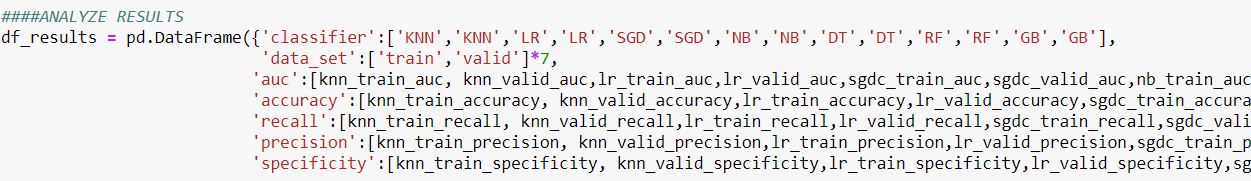




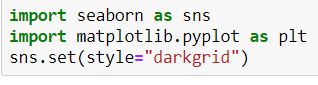
From the above metrics we can see that the Gradient Boost Classifier has the Accuracy of 76.5% which is good, but the precision is 46.7% which is low.

**Analyze the Performance Metrics:**

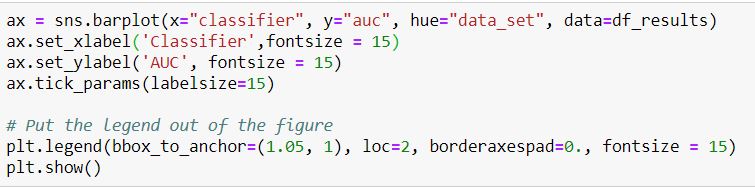
First, we import the data frames consisting of all the classifiers and the metrics.

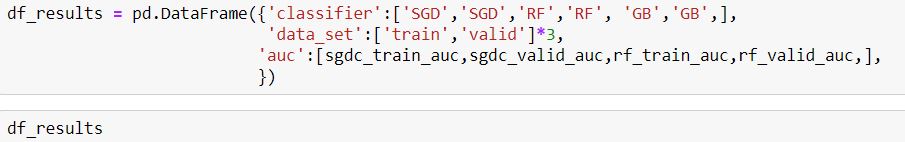


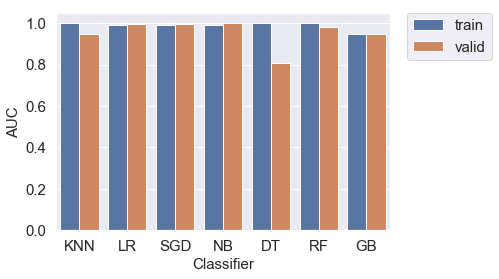
Then we plot the metrics and evaluate the results.



In this we will use the ROC curves to evaluate the best model based on AUC.







From the above graph, we can see that the SGD (Stochastic Gradient Boost), RF and GB and Naive Bayes have good AUC values based on the Validation sets values.

**Model Selection: Hyper tuning**

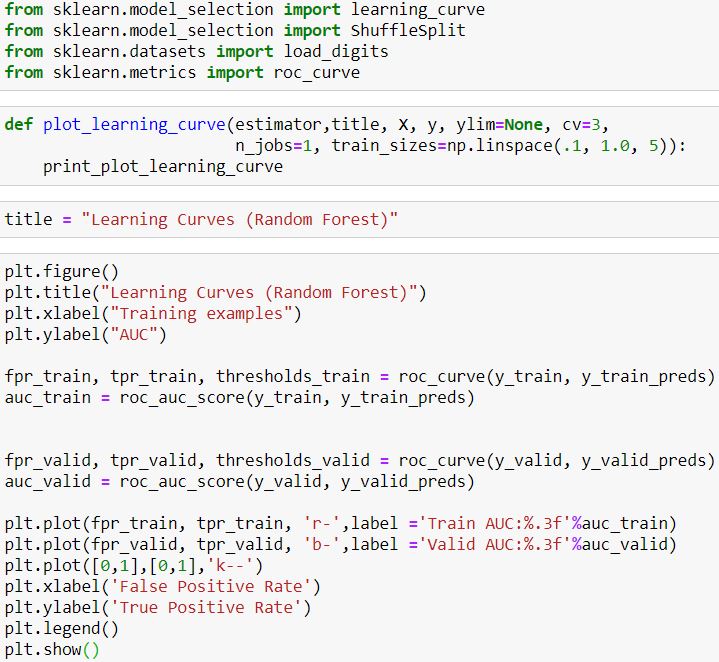
I selected the Random Forest, SGD and Gradient Boost Classifier models to check the performance in terms of AUC based on Hyper tuning.

**Random Forest:**

Methodology used:

First the AUC curve which was created from the model was biased between the Training set and Validation set.

Code:





From the above curve we can see that the Validation set, and Training set are almost equal in terms of AUC, there is no much difference in the scores.

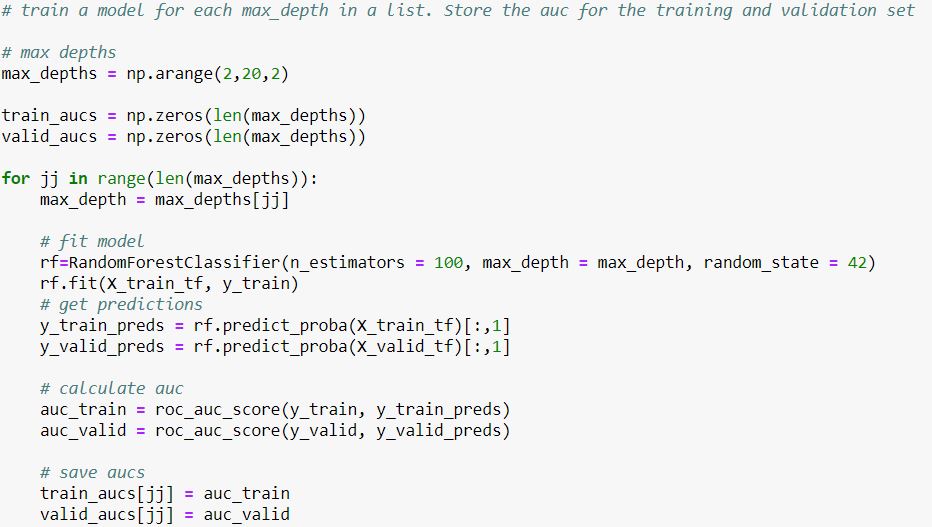
Strategy used to improve the performance:

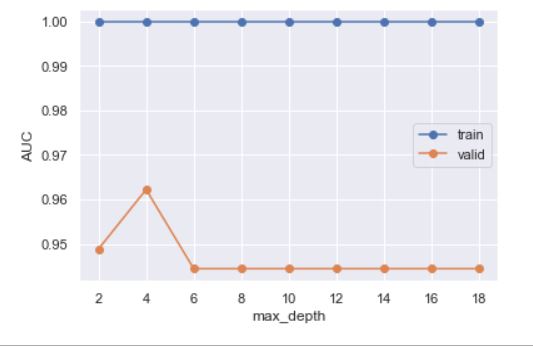
I utilized the hyperparameter tuning based on the single parameter max\_depth as max\_depth controls the tree.

I used the range of max\_depth as 2,20, 2 and after plotting the graph, I analyzed that the scores between the training set and validation varied a lot, there is a high variance.

The Performance of training set remained high, however there is not much of a performance in the validation set after using max\_depth.

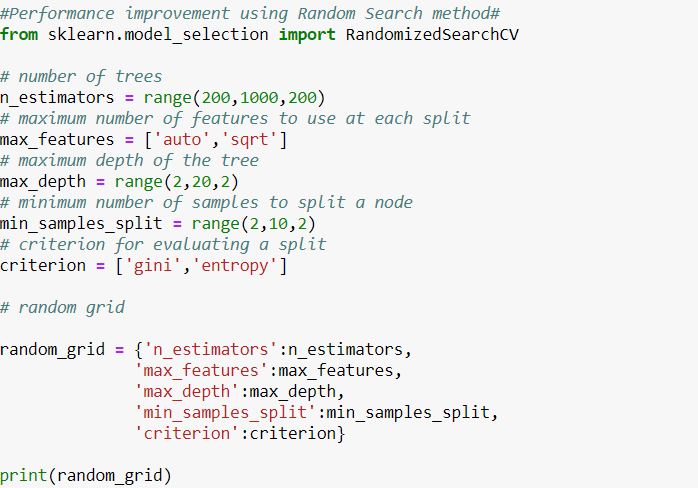
The following code:





I then utilized the Random Search method for the Random Forest Algorithm based in AUC.

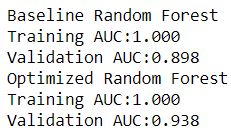
The following is the code:



The range of estimators used are 200,100,200 and the maxim depth of tree range is 2, 20.

The minimum number of samples to split are 2,10,2.

The following is the increase in the AUC performance:

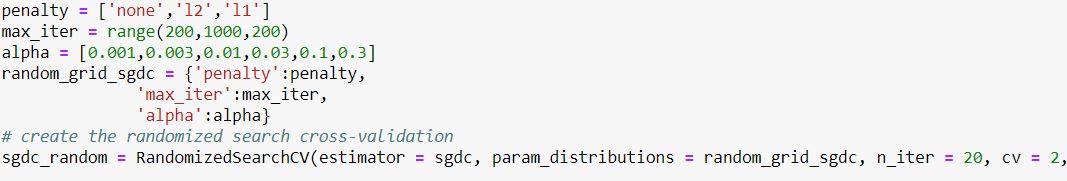


From the above results, we can see that the Validation AUC performance is increased after using Random Search hyper tuning method.

Similarly, I Utilized the Random Search Method for the SGD classifier model and Gradient Boost Classifier model to check the performances of the AUC and analyze the results.

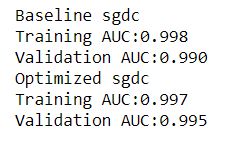
**SGD Classifier:**

The following is the code in using the Random Search Method on SGD classifier Algorithm:



Here, I used the Cross validation= 2 and Max iterations of range 200, 1000, 200.

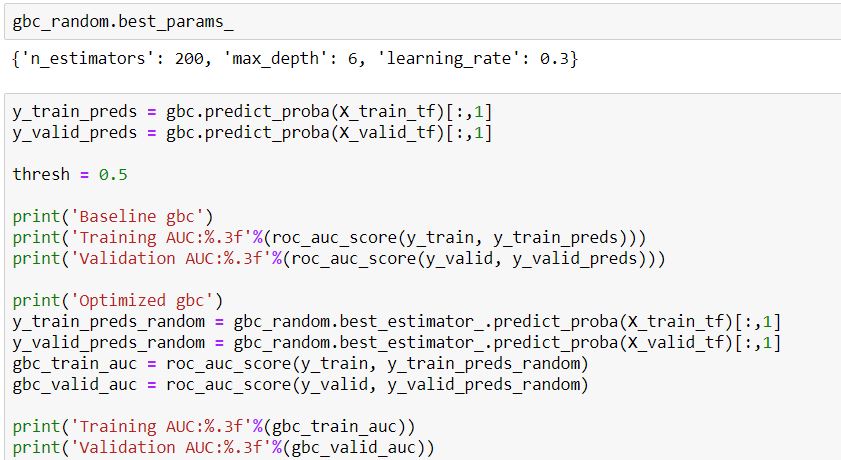
The following is the Results:



From the above metrics, we can see that the Validation AUC is increased from 0.990 to 0.995 which is good.

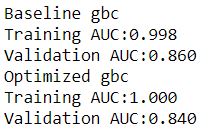
**Gradient Boost Classifier:**

The following is the code implemented:



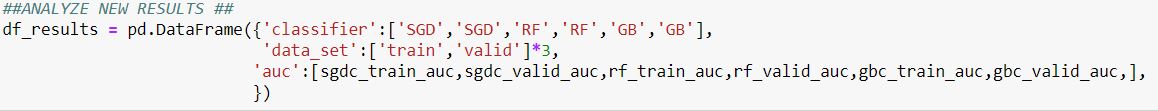
The number of estimators is 200, maximum depth is 6 and learning rate is 0.3 according to the output after training the model using Random Search Hyper tuning method.

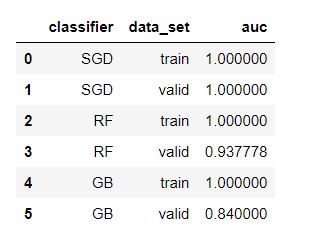
The following is the AUC metrics results:



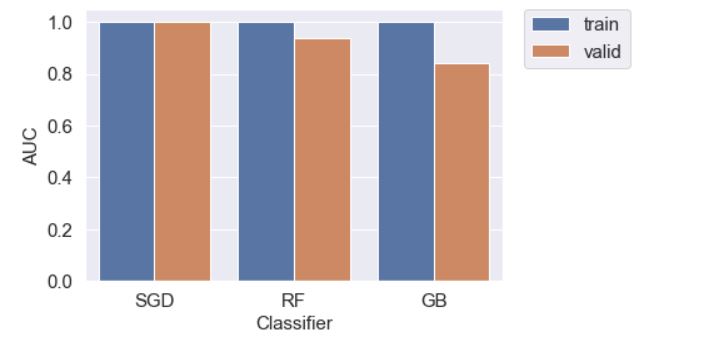
From the above metrics, we can see that the Performance of Validation Set reduced from the baseline AUC performance.

Here, we are using the following code to Analyze all these curves and plot them accordingly.





We can see that the AUC performance is maximum of 1.00 for SGD and minimum of 0.84 for Gradient boost model after Hyper tuning with respect to the Validation sets values.

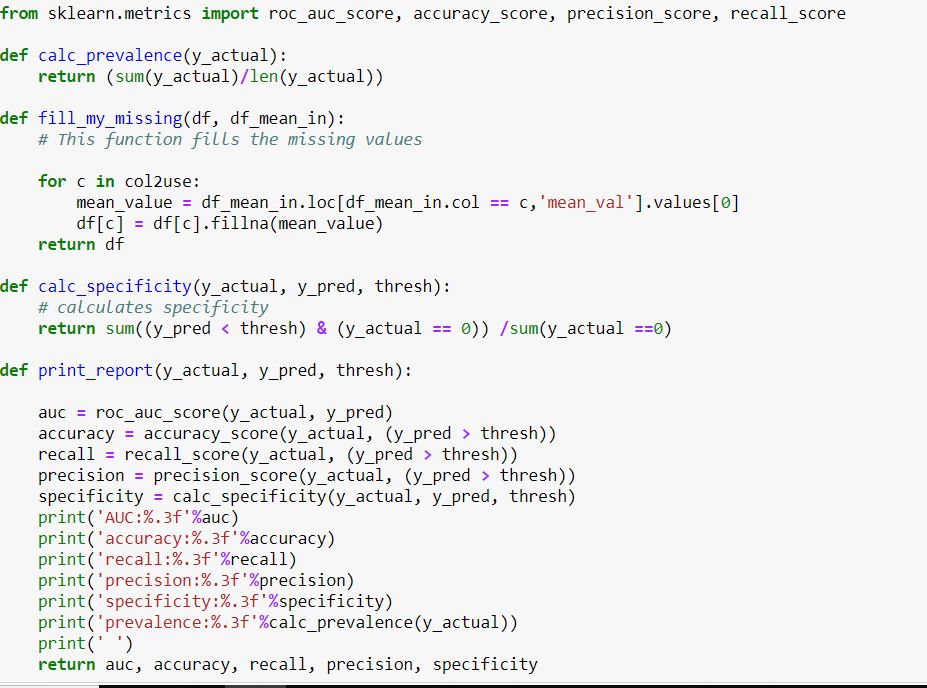


**Model Evaluation:**

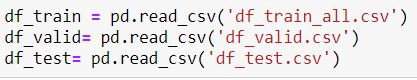
So, based on the AUC values after hyper tuning, I have selected SGD model as the best model and would like to evaluate metrics on a Test Set.

First, I save the classifier using pickle and then import them.

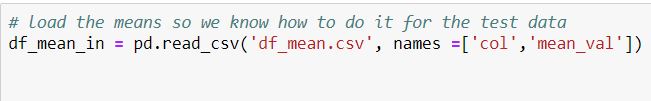


****

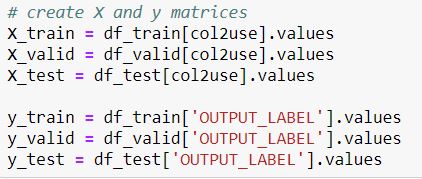
Then load the values of Training set, Validation set and Test set.

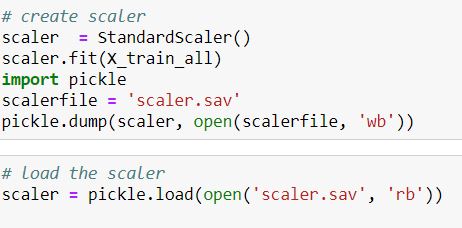
****

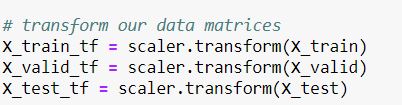
Then we load the mean values from the training set.



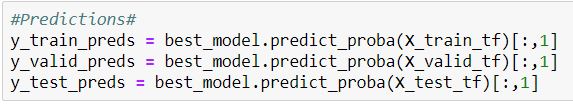
Further we try to transform the training set, validation set and the test set by using the scalar.



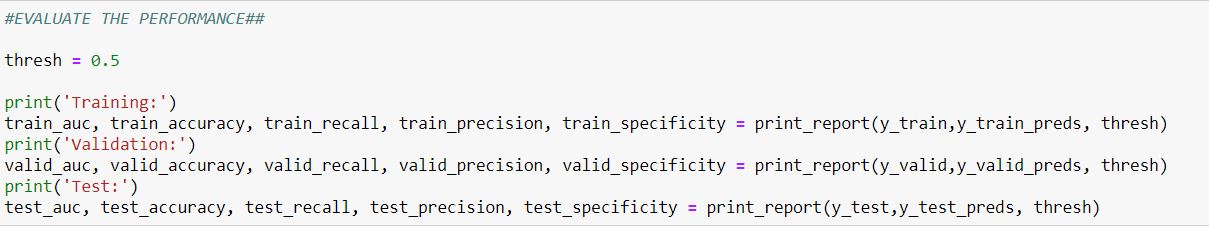


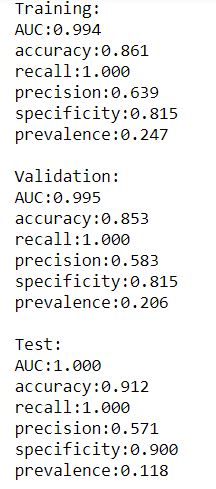


Then we train the model and predict the outcomes of training set, validation set and test set.



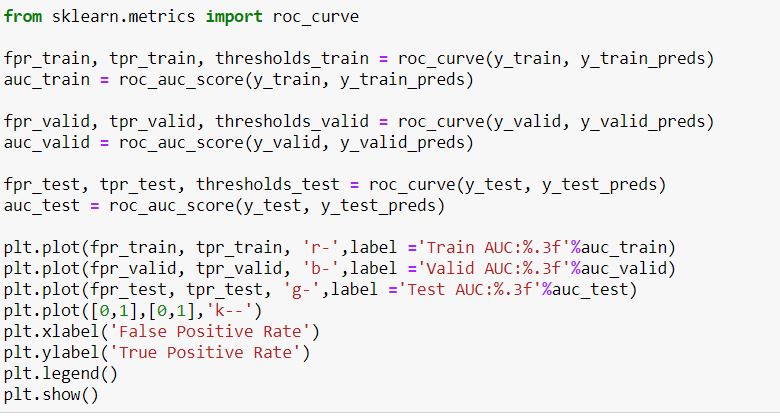
Finally, we evaluate the performance metric for the test set, training set and validation set.

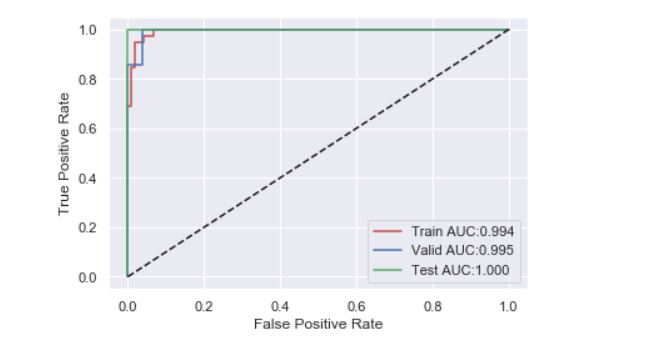




From the above metrics, we can see that the AUC is maximum for the test set with 1.00and accuracy of 91.2 %.

Let’s plot the ROC curve for the performance metrics:





From the test set, we can see that the Test set AUC value is maximum.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Training Set | Validation Set | Test Set |
| AUC | 0.994 | 0.995 | 1.000 |
| Accuracy | 0.861 | 0.853 | 0.912 |
| Recall | 1.000 | 1.000 | 1.000 |
| Precision | 0.639 | 0.583 | 0.572 |
| Specificity | 0.815 | 0.815 | 0.900 |
| Prevalence | 0.247 | 0.206 | 0.118 |

**Conclusion:**

Through this project, we implemented various Predictive modelling analysis to check whether the events will Rain or Snow.

Many machine learning models predicted with good metrics.

However, SGD (Stochastic Gradient) Classifier was the best model for me to be able to predict with high performance metrics over other models.

**References**

1. This data set is retrieved from *Climate of Kansas City*, <https://www.ncdc.noaa.gov/data-access>

2. *Hyperparameter Tuning the Random Forest in python, William Koehrsen, Jan 9 2018*, https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74