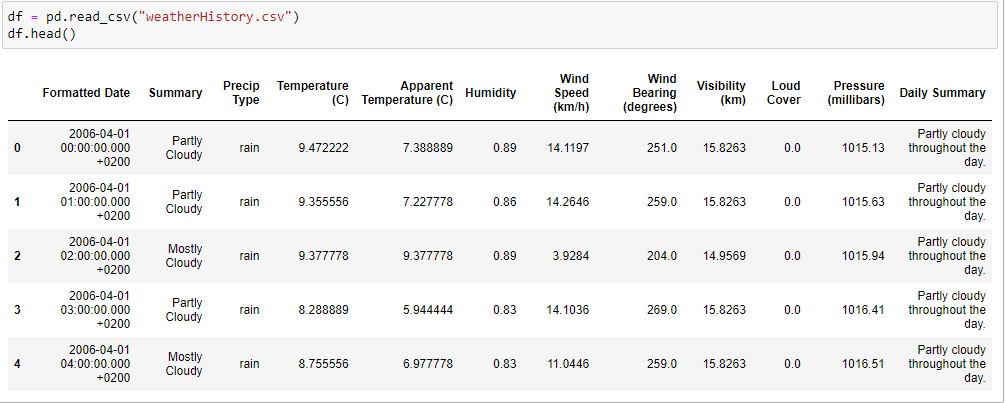
**Machine Learning CEP Report**

**Dataset:** Weather Dataset

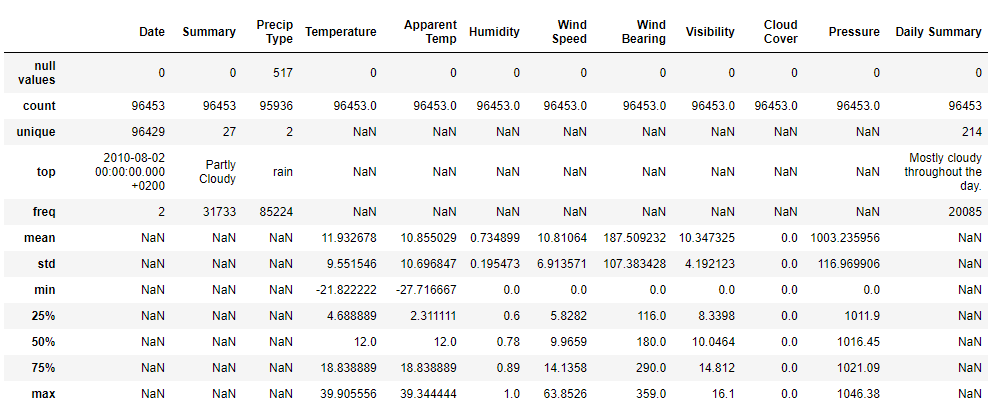
**Type of data:** Structured tables

**Data preprocessing:**

First we import raw data



As names of columns are large so we rename it and then we display basic description of dataset and number of null values in each column.

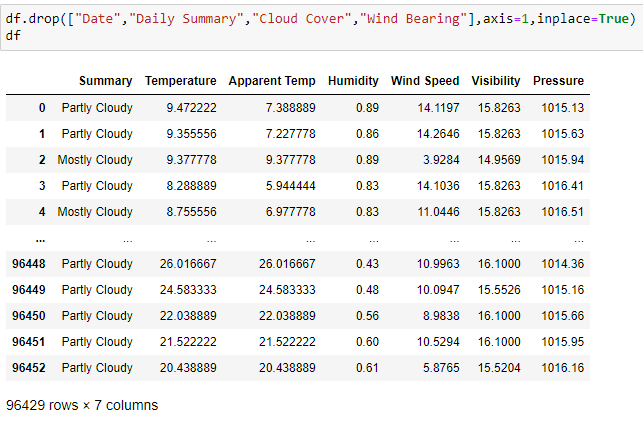


From above table we can see that total data count is greater than unique values for dates column which shows that the data contains duplicates. also we can see that all the null values present in the dataset are in column "Precip Type", Since the column does not appear to be as important feature therefore the Whole column can be dropped.

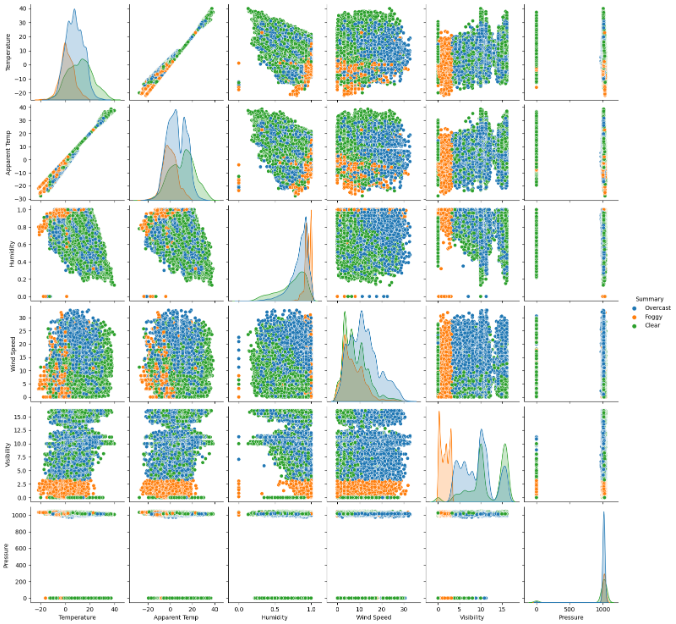


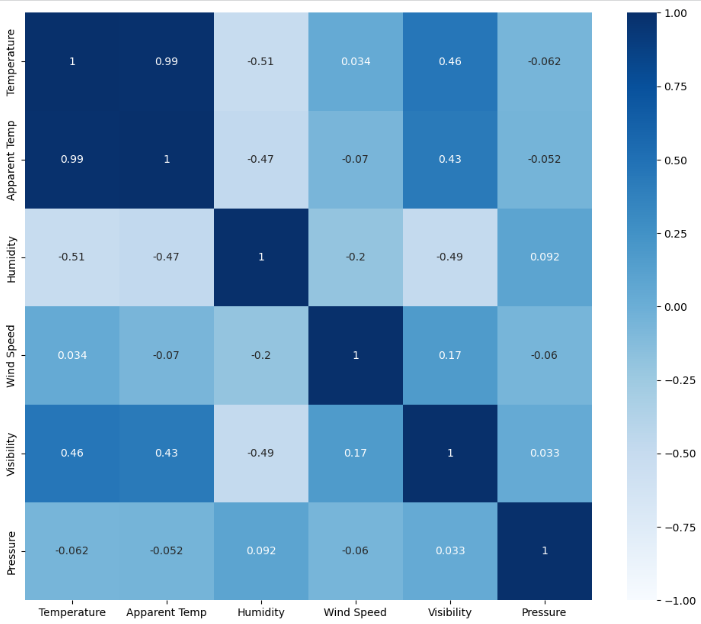
We took some steps to remove irrelevant data as it does not help to conclude any outcome.

* Removing all irrelevant columns such as **"Date"** and **"Daily Summary".**
* removing **"Cloud Cover"** since it has only 1 unique value for all instances and does not help in identifying target variable.
* by using domain knowledge, we know that direction of winds does not contribute much to identify the weather therefore also removing **"Wind Bearing"** column from the dataset.



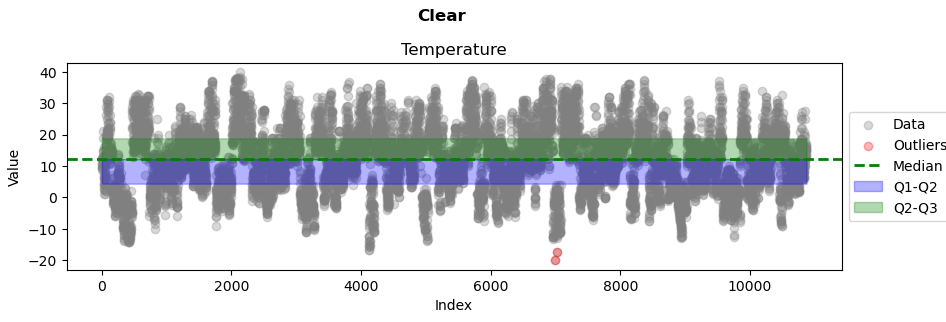
Now analyzing the remaining features through graphs and plots.





From above plots we can conclude following results

* The variables **"Temperature"** and **"Apparent Temp"** are highly correlated and may lead to multi-collinearity therefore we can drop any one of the features in order to reduce the dimensions of the feature set.in this case based on our own domain knowledge we are dropping **"Apparent temp"**.
* The variables **"Temperature"** and **"Humidity"** have moderate negative correlation. but from the domain knowledge we know that humidity can be important feature in determining weather therefore considering this correlation to be insignificant
* We can see that we have outliers in our dataset mostly in **"Pressure"** and **"Humidity"** which should be removed
* We can also from the KDEplot of **"Pressure"** that the pressure values for all **"Summary"** classes are same and there for we can conclude that pressure would not be a much helpful feature in predicting the Summary.

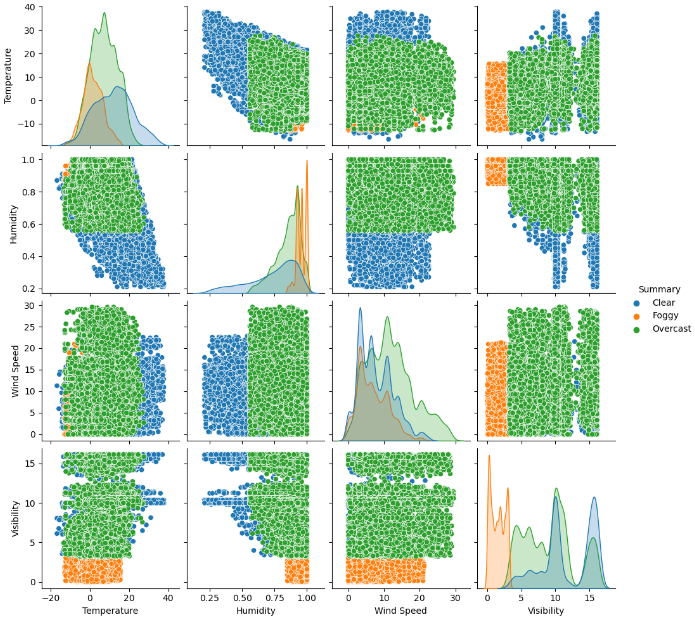


From the above Plots we can see that there are a few outliers that we need to remove in order to remove outliers we are using **"Interquartile Ranges"** with a value of **k = 1.5** (k is used to find the upper and lower boundaries out of which any data will be considered Outlier)

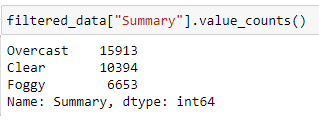
* lower\_bound = Q1 - k \* IQR
* upper\_bound = Q3 + k \* IQR

see resource link for further details on [Outlier Detection using IQR](https://machinelearningmastery.com/how-to-use-statistics-to-identify-outliers-in-data/)

<https://machinelearningmastery.com/how-to-use-statistics-to-identify-outliers-in-data/>



The data is cleaned is cleaned but it is not balanced

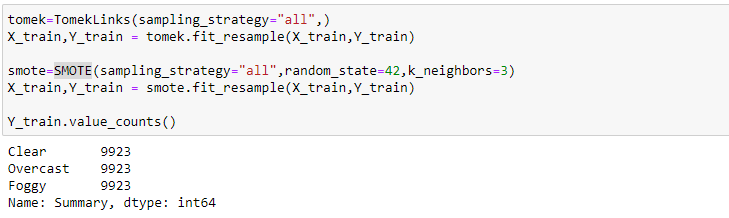


Since we know that our data contains imbalanced classes we have to resample our data to balance the classes for this we are under sampling the data using "Tomek Links" and then we will perform oversampling using "SMOTE"

for further description about Tomek Links and SMOTE you can visit the reference links below

<https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/>

its important to note that we perform these up sampling and down sampling techniques only on the training data, while keeping the original imbalanced dataset for evaluation purposes.



**Model Training and Evaluation**

we will be training our models using three algorithms

* non-parametric algorithm **(K Nearest Neighbor)**
* parametric shallow algorithm **(Logistic Regression)**
* neural network architecture **(Multi-Layer Perceptron)**

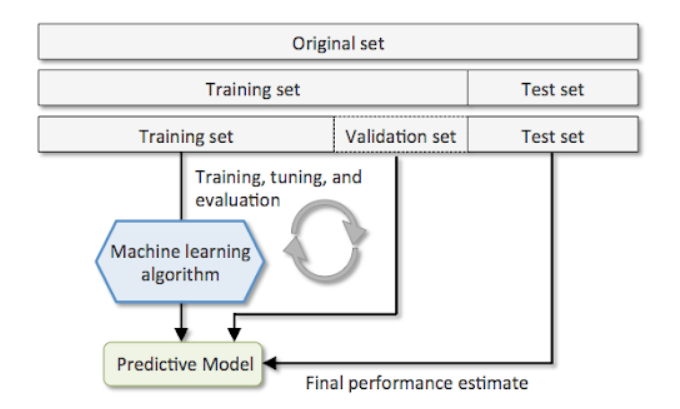
Note that we will be using Hold out method for Hyper Parameters selection and Model Evaluation. given below is reference link for further details

<https://vitalflux.com/hold-out-method-for-training-machine-learning-model/>

**Hold out method – Training – Validation – Test Dataset**

The following process represents the hold-out method for model selection:

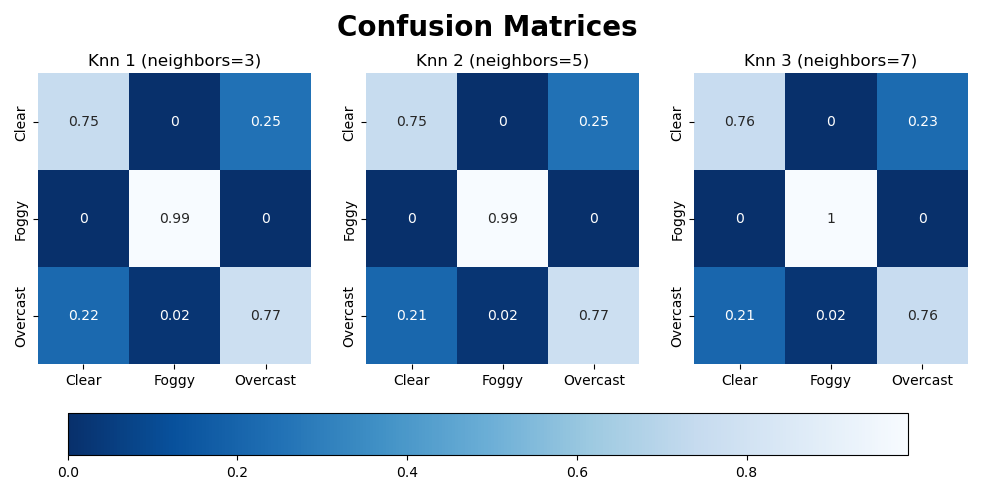
1. Split the dataset in three parts – Training dataset, validation dataset and test dataset.
2. Train different models using different machine learning algorithms. For example, train the classification model using logistic regression, knn, neural network.
3. For Classifiers: the models trained with different algorithms, tune the hyper-parameters and come up with different models. For each of the algorithms mentioned in step 2, change hyper parameters settings and come with multiple models.
4. Test the performance of each of these models (belonging to each of the algorithms) on the validation dataset.
5. Select the most optimal model out of the models tested on the validation dataset. The most optimal model will have the most optimal hyper parameters settings for a specific algorithm.
6. Test the performance of the most optimal model on the test dataset.



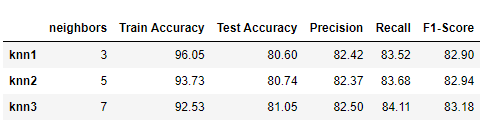
**Machine algorithm:**

**KNN**

We make 3 KNN model with different values of n\_neighbors 3, 5 and 7.



**Result:**

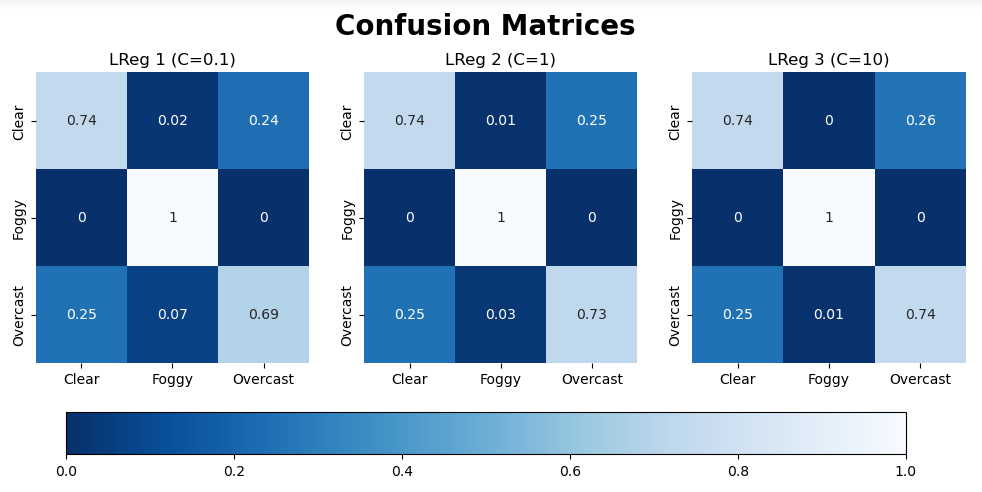
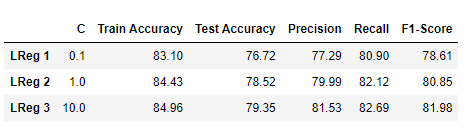


As we can see that KNN model with 7 neighbors provides highest Test Accuracy, Precision, Recall and F1-Score.

**Logistic Regression:**

We make 3 Logistic Regression model with different values of C 0.1, 1 and 10.

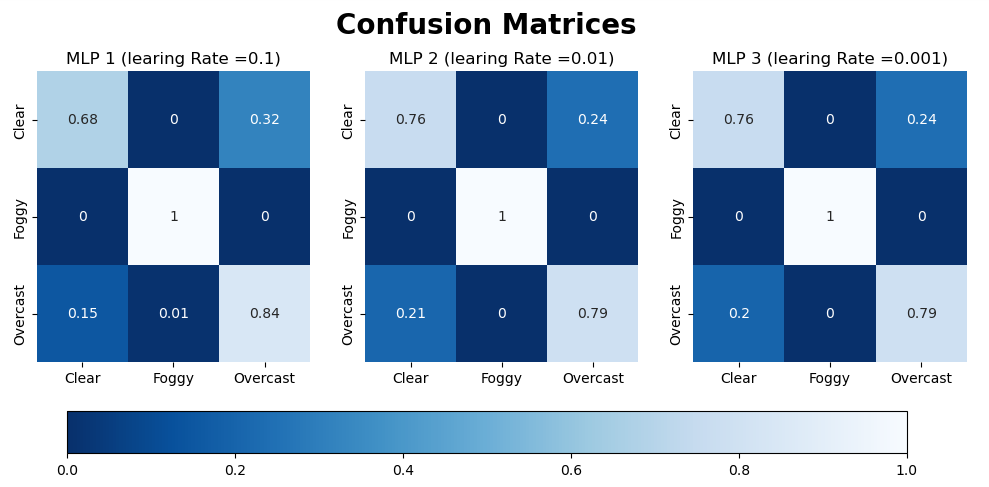
The C value in Logistic Regression is a user adjustable parameter that controls regularization. In simple terms, higher values of C will instruct our model to fit the training set as best as possible, while lower C values will favor a simple model with coefficients closer to zero.

**Result:** 

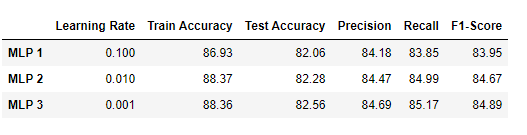
As we can see that Logistic Regression model with C value 10 provides highest Test Accuracy, Precision, Recall and F1-Score.

**Neural network:**

We make 3 Neural network model with different values of Learning Rate 0.1, 0.01 and 0.001



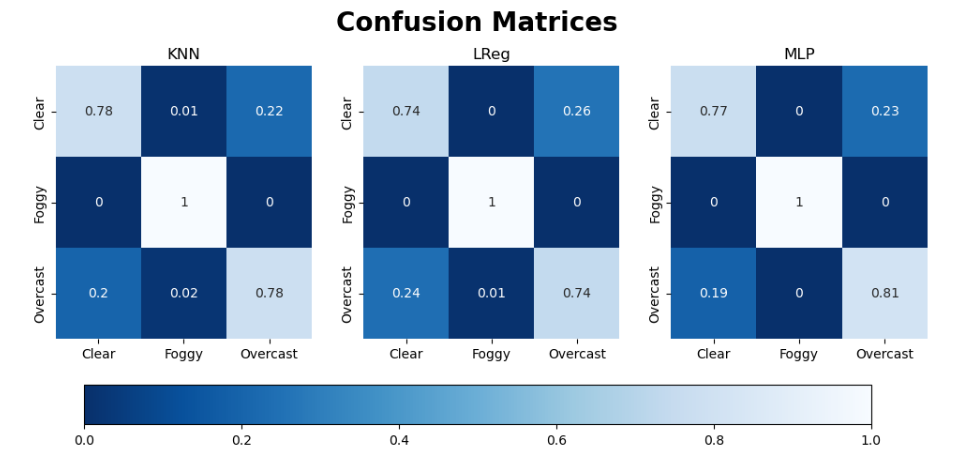
**Result*:***

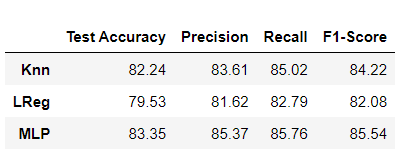


As we can see that Neural network model with learning rate of 0.001 provides highest Test Accuracy, Precision, Recall and F1-Score.

## Best Model Selection:

for finding the model with best performance we will test the performance of the each of the algorithms with their best hyper parameters of the test set. note that we have previously separated and test set and there for none of the selected models have been previously trained or evaluated of test set





### best model:

By analyzing above confusion matrices and performance results we can conclude that neural network **(Multi-Layer Perceptron)** with hyper parameter **learning rate = 0.001** performs best among all the other models

**Comparing results of all models**

