**MACHINE LEARNING PROJECT**

**Topic**: Rain Prediction

SUBMITTED BY: Bhumika(102103114)

Jiya(102283024)

SUBMITTED TO: Ms. Archana Singh

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Particulars** | **Page No.** |
| 1 | Introduction |  |
| 2 | Libraries Used |  |
| 3 | Algorithms Used |  |
| 4 | Code and Screenshots |  |

1. **Introduction**
   1. **Name of the dataset: weatherAUS**

**Dataset Link: weatherAUS.csv**

* 1. **Description:**

The rain prediction model serves various practical purposes across different sectors. In agriculture, it aids farmers in planning irrigation schedules and crop management based on anticipated rainfall. For city planning and infrastructure management, the model assists in preparing for potential flooding or drainage issues during rainy periods. Transportation and logistics benefit from improved scheduling and routing, minimizing disruptions caused by adverse weather conditions. Additionally, the model can enhance disaster preparedness and response by providing timely information for emergency services. In the context of water resource management, predicting rainfall helps in monitoring water levels in reservoirs and planning for water conservation or release. Overall, the rain prediction model contributes to informed decision-making across diverse domains, improving efficiency and mitigating the impact of weather-related challenges.

To build a rain prediction model, historical weather data is collected, encompassing key features like temperature, humidity, and wind speed, along with a binary target variable indicating the occurrence of rain on a given day. After data preprocessing, which involves exploration, cleaning, and encoding, a machine learning algorithm is chosen, such as logistic regression or decision trees. The model is trained on a designated training set and evaluated using metrics like accuracy and recall on a separate testing set. If the model performs well, it can be deployed for real-time predictions. Continuous monitoring in a production environment allows for necessary updates and maintenance, ensuring the model's relevance and accuracy over time.

1. **Libraries Used:**
2. **Scikit-learn (sklearn):**
   * **Preprocessing**: This module provides functions for preprocessing data before training a model. It includes scaling, normalization, encoding categorical variables, and more.
   * **Metrics**: This module contains various metrics for evaluating the performance of machine learning models, such as accuracy, precision, recall, F1-score, and more.
3. **Scipy.stats:**
   * This module from SciPy, a scientific computing library, includes statistical functions. In your code, it might be used for statistical operations and tests.
4. **Collections:**
   * This is a built-in Python module providing specialized container datatypes. In your code, **Counter** is used to count the occurrences of elements in a list.
5. **Imbalanced-learn (imblearn):**
   * **Over\_sampling**: This module includes algorithms for oversampling the minority class in imbalanced datasets. **SMOTE** stands for Synthetic Minority Over-sampling Technique, **ADASYN** is Adaptive Synthetic Sampling, and **RandomOverSampler** is a simple random oversampling technique.
6. **Joblib:**
   * This library provides tools for working with Python functions as lightweight tasks. In your code, it's likely used for parallel processing or efficient storage of objects.
7. **Pandas:**
   * A powerful data manipulation and analysis library. It provides data structures like DataFrames, which are helpful for handling and analyzing structured data.
8. **Numpy:**
   * A fundamental package for scientific computing with Python. It provides support for large, multi-dimensional arrays and matrices, along with mathematical functions.
9. **Matplotlib.pyplot:**
   * This is a plotting library used for creating static, animated, and interactive visualizations in Python.
10. **Seaborn:**
    * Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
11. **LabelEncoder:**
    * A utility class in scikit-learn for encoding labels with values between 0 and n\_classes-1.
12. **StandardScaler:**
    * A class in scikit-learn for standardizing features by removing the mean and scaling to unit variance.
13. **train\_test\_split:**
    * A function in scikit-learn for splitting datasets into training and testing sets.
14. **StratifiedKFold:**
    * A cross-validation method that ensures each fold has the same distribution of target classes as the entire dataset.
15. **GridSearchCV and RandomizedSearchCV:**
    * Functions in scikit-learn for performing hyperparameter tuning using grid search and randomized search, respectively.
16. **LogisticRegression, DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier, ExtraTreesRegressor, CatBoostClassifier, XGBClassifier, SVC, GaussianNB, KNeighborsClassifier:**
    * These are various machine learning models available in scikit-learn, CatBoost, and XGBoost for classification and regression tasks.
17. **pickle:**
    * A module for serializing and deserializing Python objects. In your code, it's used for saving and loading model objects.

These libraries collectively provide a comprehensive set of tools for data preprocessing, model development, and evaluation in machine learning and data analysis tasks.

1. **Algorithms Used:**

**3.1 Random Forest Model:** A Random Forest model is an ensemble learning technique used for both classification and regression tasks. It operates by constructing a multitude of decision trees during training and outputs the mode (classification) or average prediction (regression) of the individual trees.

Here's a brief explanation of key concepts related to Random Forest:

1. **Decision Trees:**
   * Random Forest is built on the foundation of decision trees, which are hierarchical structures that recursively split the data based on feature conditions to make predictions.
2. **Ensemble Learning:**
   * Random Forest is an ensemble of multiple decision trees. The "forest" aspect comes from combining the predictions of these trees to make a more robust and accurate overall prediction.
3. **Random Feature Selection:**
   * Each decision tree in the forest is trained on a random subset of features, introducing diversity among the trees. This helps prevent overfitting and improves the model's generalization to new, unseen data.
4. **Bootstrapping:**
   * Random Forest builds each tree on a random subset of the training data, chosen with replacement. This process is known as bootstrapping, and it creates diverse training sets for each tree.
5. **Voting (Classification) or Averaging (Regression):**
   * For classification tasks, the Random Forest combines the predictions of individual trees through a majority voting mechanism. For regression tasks, it averages the predictions.
6. **Out-of-Bag (OOB) Score:**
   * Since each tree is trained on a subset of the data, there is a portion of the data not used for training each tree. This allows for an out-of-bag score, which serves as an additional evaluation metric.
7. **Robustness and Generalization:**
   * Random Forest is known for its robustness and resistance to overfitting. By aggregating the predictions of multiple trees, it tends to generalize well to different datasets.
8. **Feature Importance:**
   * Random Forest provides a measure of feature importance, indicating which features contribute more to the model's predictions. This can be useful for understanding the factors driving the model's decisions.

Random Forest models are widely used in practice due to their flexibility, ease of use, and strong performance across various types of datasets. They are effective in handling high-dimensional data, capturing complex relationships, and providing reliable predictions in both classification and regression scenarios.

**3.2 Logistic Regression:** Logistic Regression is a statistical method used for binary classification, which means it helps us predict whether something belongs to one of two categories. For instance, it can be used to predict whether an email is spam (1) or not spam (0), or whether a student will pass (1) or fail (0) an exam.

Here's a step-by-step breakdown:

1. **Mathematical Basis:**
   * Logistic Regression is based on the logistic function (or sigmoid function). The logistic function looks like an elongated S-shape and is used to map any real-valued number into a value between 0 and 1.
2. **Linear Combination:**
   * Logistic Regression starts with a linear combination of input features. Each feature is multiplied by a weight, and all these products are added together along with a bias term.
3. **Log-Odds Transformation:**
   * The linear combination is then transformed using the logistic function, which converts the result into a value between 0 and 1. The output can be interpreted as the probability of the instance belonging to the positive class.
4. **Decision Boundary:**
   * A threshold is chosen (often 0.5). If the predicted probability is above this threshold, the instance is classified as belonging to the positive class; otherwise, it's classified as belonging to the negative class.
5. **Training the Model:**
   * During the training phase, the algorithm adjusts the weights and bias to minimize the difference between the predicted probabilities and the actual class labels in the training data. This is typically done using optimization techniques like gradient descent.
6. **Interpretability:**
   * Logistic Regression provides interpretable coefficients for each feature, indicating the strength and direction of their influence on the prediction.
7. **Use Cases:**
   * Logistic Regression is widely used in various fields for binary classification tasks due to its simplicity and effectiveness. It's a go-to choose when you want to understand the impact of individual features on the likelihood of an event happening**.**
8. **Assumptions:**
   * Logistic Regression assumes that the relationship between the features and the log-odds of the target variable is linear. It also assumes that the observations are independent of each other.

In summary, Logistic Regression is a valuable tool for predicting binary outcomes, and its simplicity and interpretability make it a popular choice for a wide range of applications in machine learning and statistics.

**3.3 XGBoost**: XGBoost, short for eXtreme Gradient Boosting, is a powerful and popular machine learning algorithm known for its efficiency and performance in various types of data science tasks. It belongs to the family of gradient boosting algorithms and is particularly well-suited for regression and classification problems.

Here's a breakdown of XGBoost:

1. **Gradient Boosting:**
   * XGBoost is an ensemble learning method based on the principle of gradient boosting. It builds a predictive model in a sequential manner, where each new model corrects the errors of the previous one.
2. **Decision Trees:**
   * The base learners in XGBoost are decision trees, specifically shallow trees, also known as weak learners. These trees are added sequentially to the model.
3. **Objective Function:**
   * XGBoost aims to minimize a specific objective function that consists of a loss function (measuring how well the model is performing) and a regularization term (controlling the complexity of the model to avoid overfitting).
4. **Gradient Descent Optimization:**
   * The algorithm uses gradient descent optimization to minimize the objective function. It calculates the gradient of the loss with respect to the predicted values, adjusting the model parameters to minimize the error.
5. **Tree Pruning:**
   * XGBoost includes a regularization term in the objective function that penalizes the complexity of trees. This encourages the algorithm to build simpler, more interpretable trees and helps prevent overfitting.
6. **Feature Importance:**
   * XGBoost provides a feature importance score, indicating the contribution of each feature to the model's predictions. This is valuable for understanding the factors driving the model.
7. **Handling Missing Values:**
   * XGBoost has a built-in mechanism to handle missing values in the dataset, reducing the need for extensive data preprocessing.
8. **Parallel and Distributed Computing:**
   * XGBoost is designed for efficiency and can take advantage of parallel and distributed computing, making it scalable and suitable for large datasets.
9. **Regularization:**
   * XGBoost supports both L1 (Lasso) and L2 (Ridge) regularization, allowing control over the complexity of the model.
10. **Use Cases:**
    * XGBoost is widely used in various machine learning competitions and real-world applications. It excels in tasks like classification, regression, ranking, and user preference prediction.

In summary, XGBoost is a versatile and high-performance algorithm that has become a go-to choice for many data scientists and machine learning practitioners, thanks to its speed, accuracy, and ability to handle complex relationships in data.

**3.4 CatBoost :** CatBoost is a high-performance machine learning algorithm designed for gradient boosting on decision trees. Similar to XGBoost and LightGBM, CatBoost is known for its efficiency, accuracy, and ease of use. The term "Cat" in CatBoost stands for "Category," as the algorithm is specifically optimized for categorical feature handling.

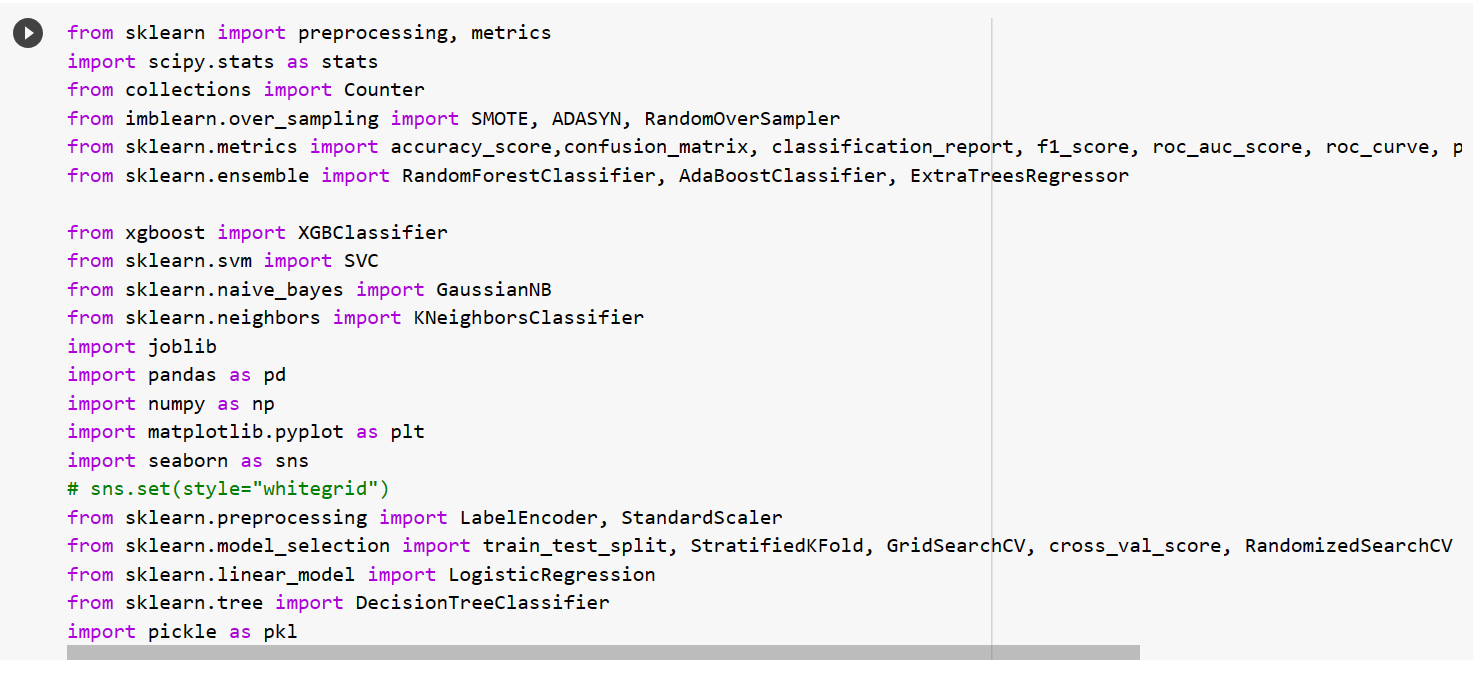
Here are key features and characteristics of CatBoost:

1. **Categorical Feature Support:**
   * CatBoost excels in handling categorical features without the need for extensive preprocessing. It internally encodes categorical variables and efficiently works with them during the training process.
2. **Built-in Feature Importance:**
   * CatBoost provides a built-in feature importance mechanism, allowing users to understand the impact of each feature on the model's predictions.
3. **Efficient Training:**
   * The algorithm is designed to be memory-efficient and scalable, making it suitable for large datasets. It implements a variant of the gradient boosting algorithm that involves a sophisticated ordering algorithm for decision tree construction.
4. **Handling Missing Values:**
   * CatBoost has a robust method for dealing with missing values, reducing the need for manual imputation or data preprocessing.
5. **Optimized Categorical Encoding:**
   * CatBoost employs an efficient method for categorical feature encoding, avoiding the need for one-hot encoding and its associated drawbacks.
6. **Regularization:**
   * The algorithm supports regularization techniques to prevent overfitting, including both L1 (Lasso) and L2 (Ridge) regularization.
7. **Cross-Validation and Hyperparameter Tuning:**
   * CatBoost simplifies the process of cross-validation and hyperparameter tuning through user-friendly interfaces, making it accessible for users with varying levels of experience.
8. **Compatibility with Other Libraries:**
   * CatBoost can be seamlessly integrated with popular data science libraries like scikit-learn, making it easy to incorporate into existing workflows.
9. **Use Cases:**
   * CatBoost is applicable to a wide range of machine learning tasks, including classification, regression, and ranking problems. It has shown competitive performance in various competitions and real-world applications.

In summary, CatBoost stands out for its efficient handling of categorical features, ease of use, and competitive performance. It has become a popular choice for machine learning practitioners, especially when dealing with datasets that contain a mix of numerical and categorical features.

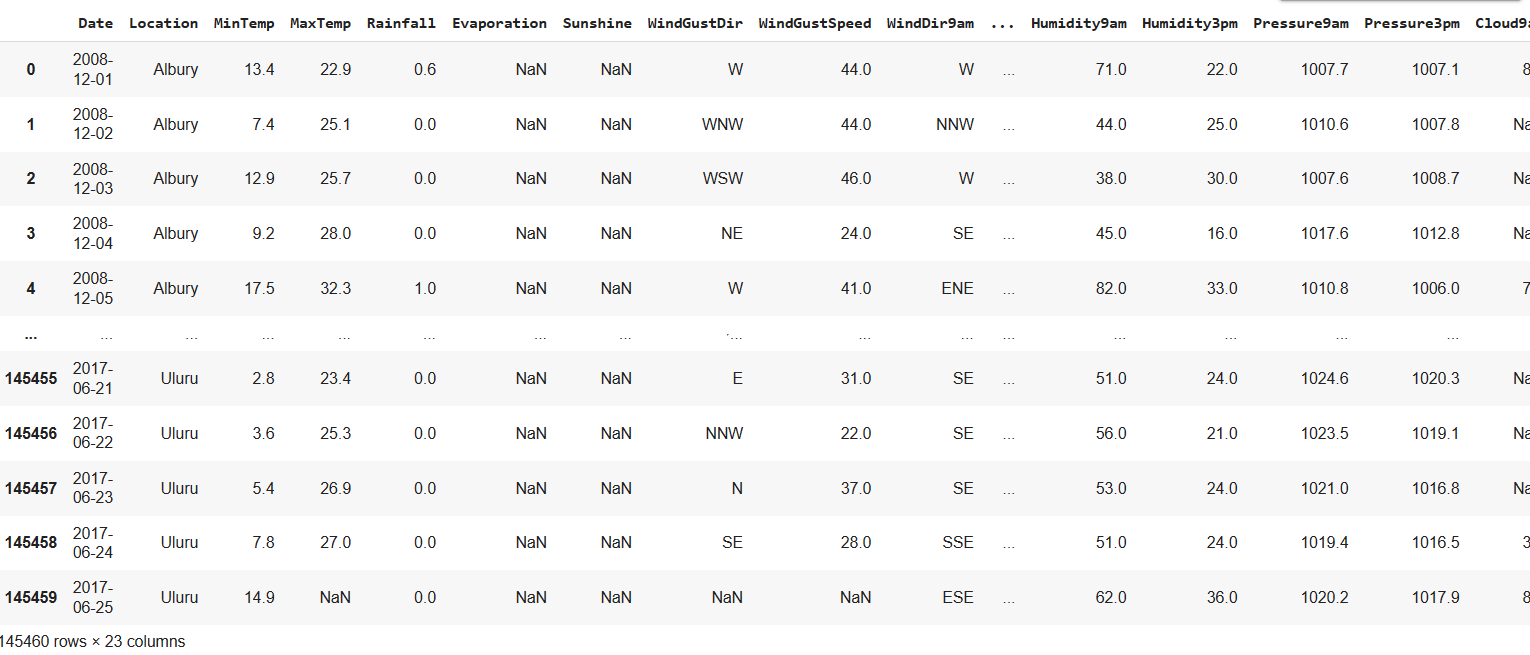
1. **Code and Screenshots**

**Importing necessary libraries:**



**Cleaning the data:**





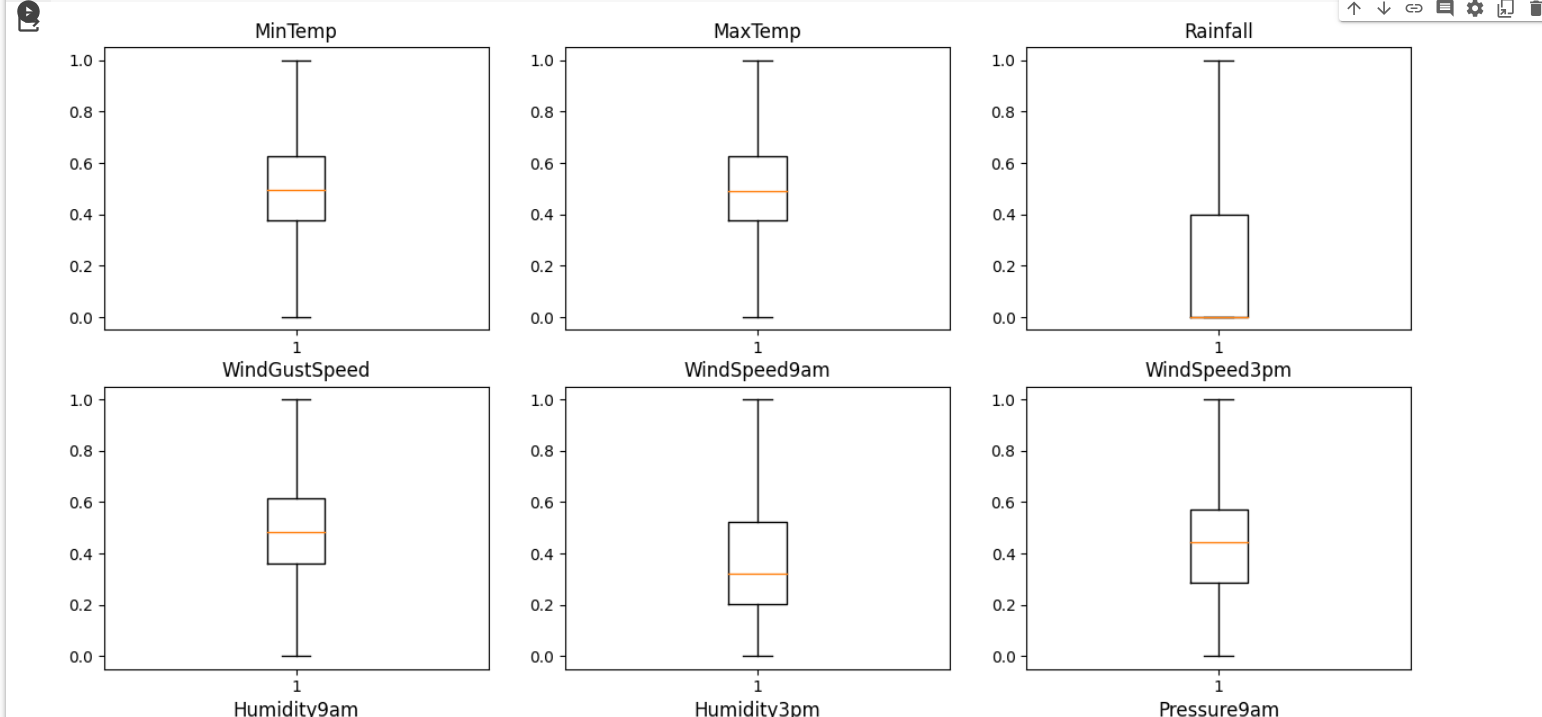
**Dealing with null values:**

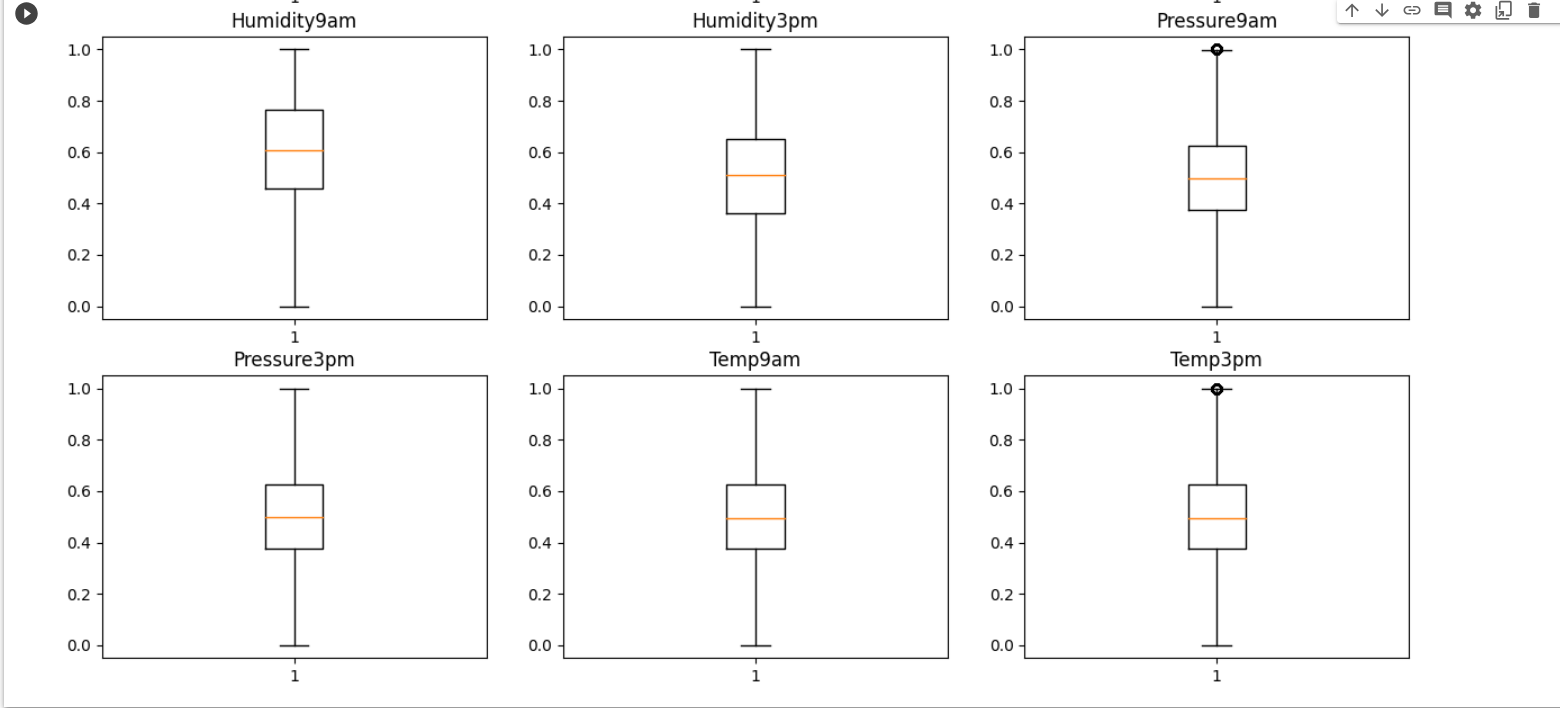


So, we lose 89,040 rows out of 145,460 rows if we decide to drop null values, i.e. ~60% of the data!

Therefore, instead, we'll drop the top 4 columns in terms of null values, then drop null values of remaining columns



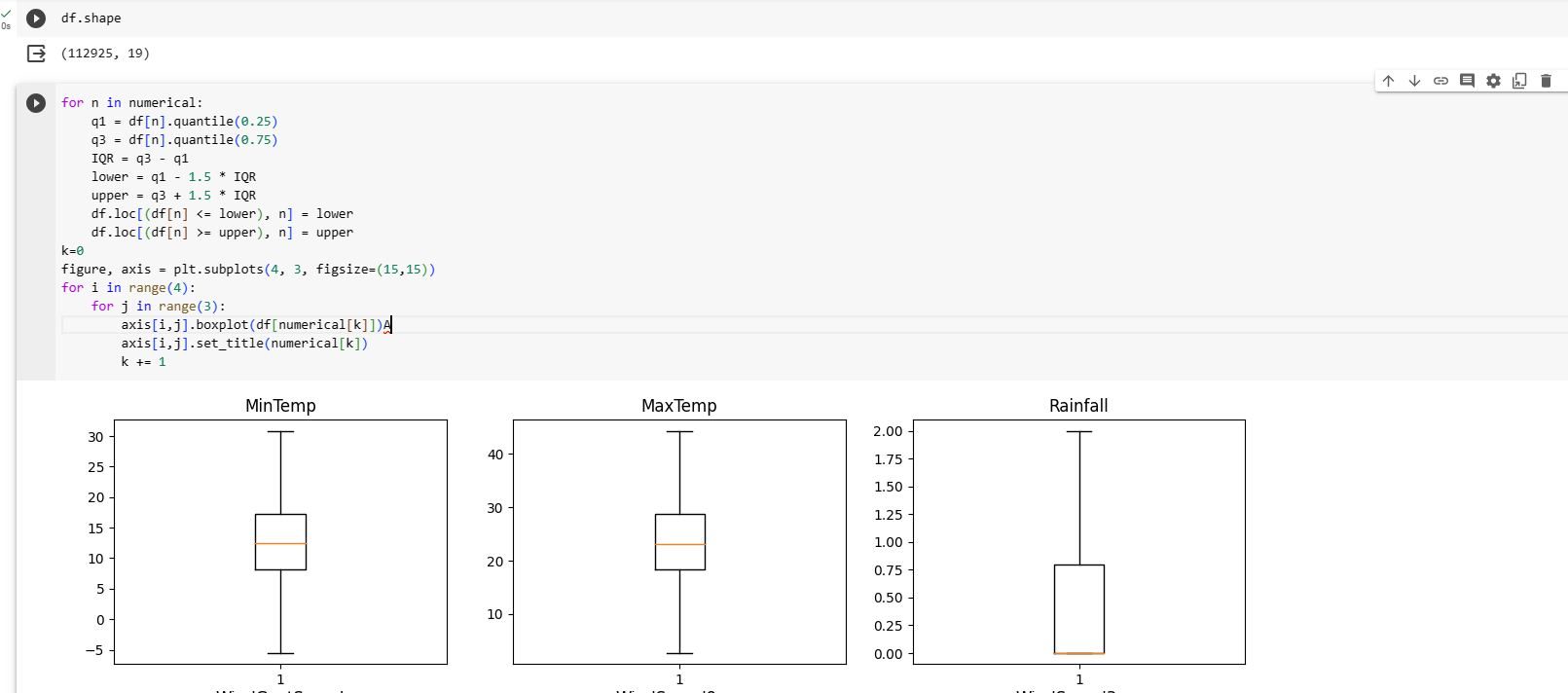


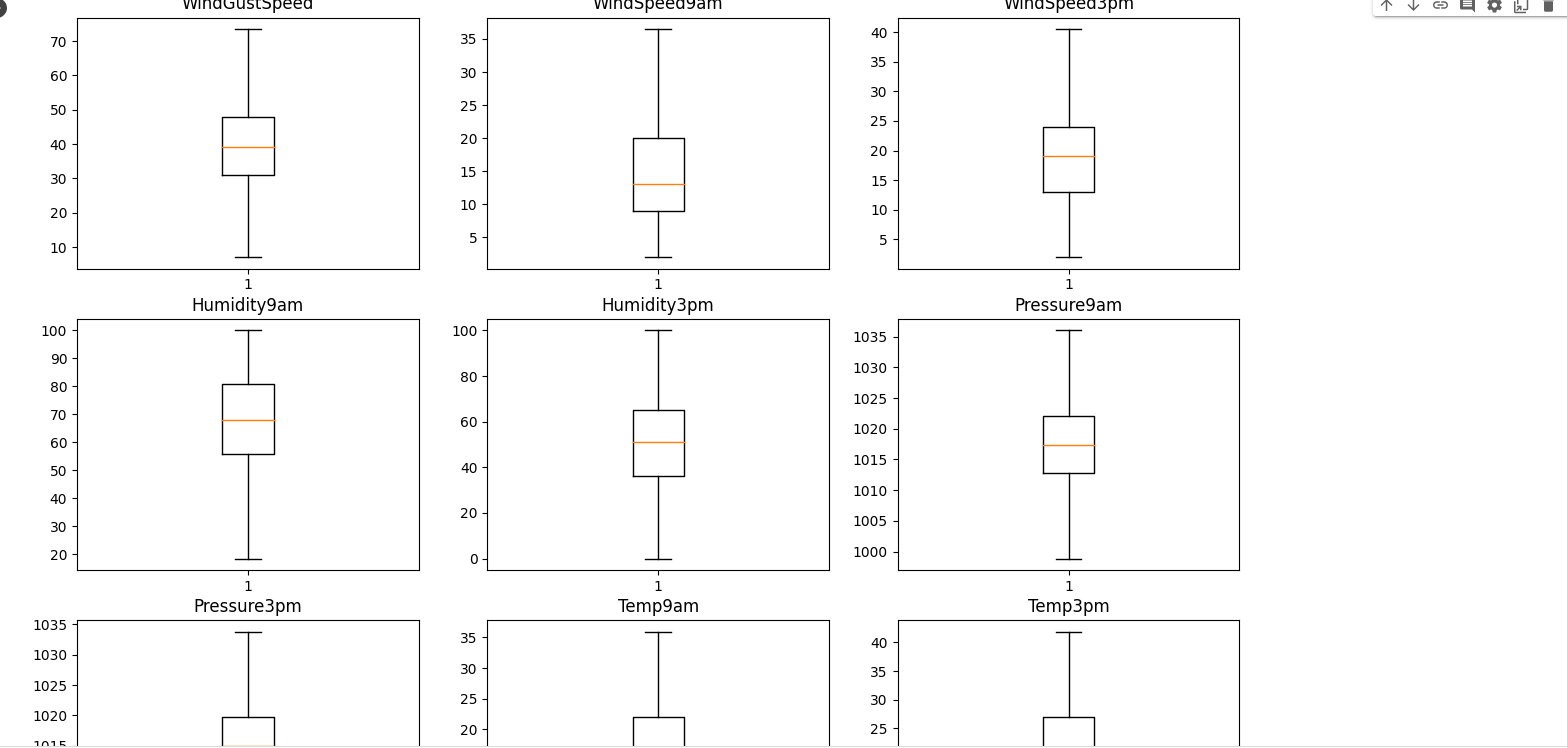


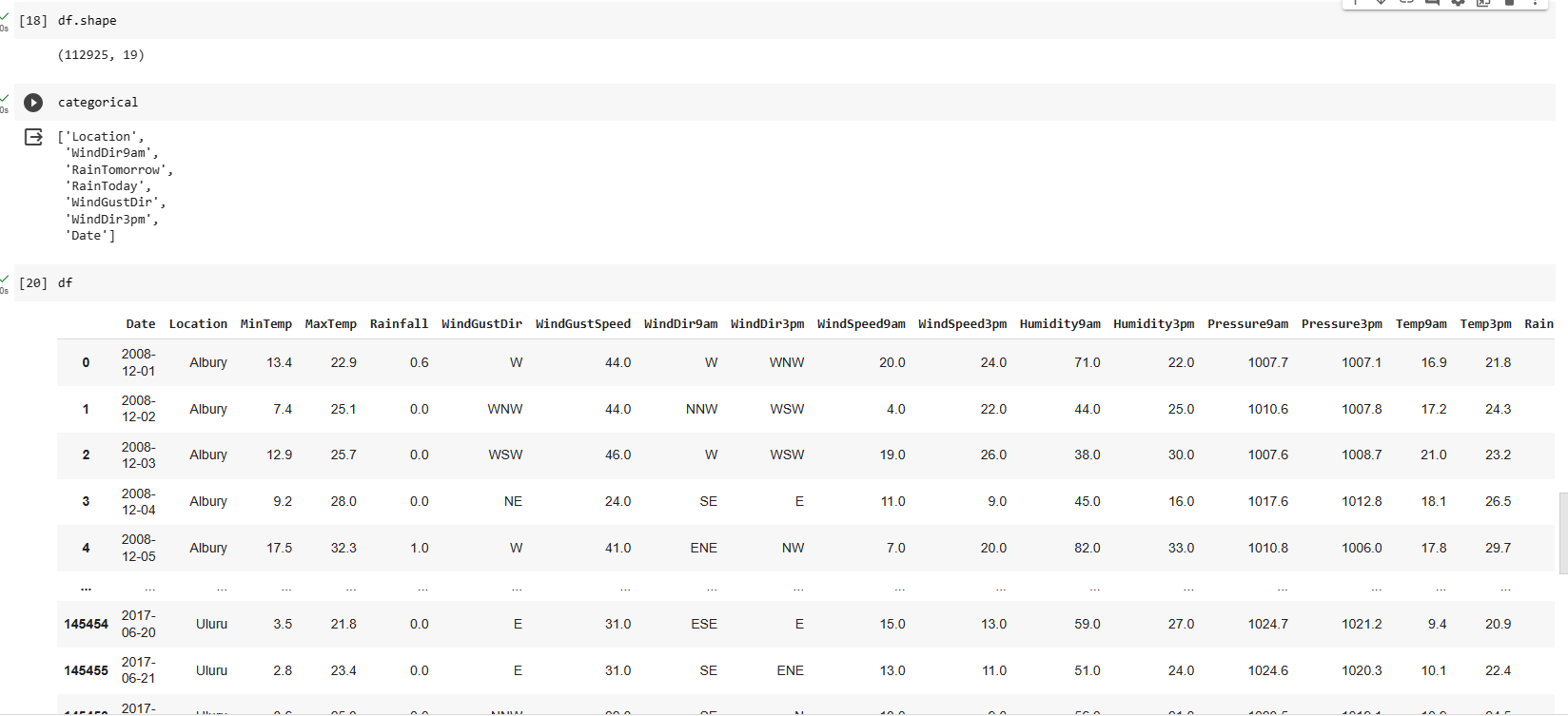
Therefore, by doing so, we're only losing ~20% of data instead of ~60%. The remaining rows are 112,925 out of 142,460.

**Preprocessing of data:**

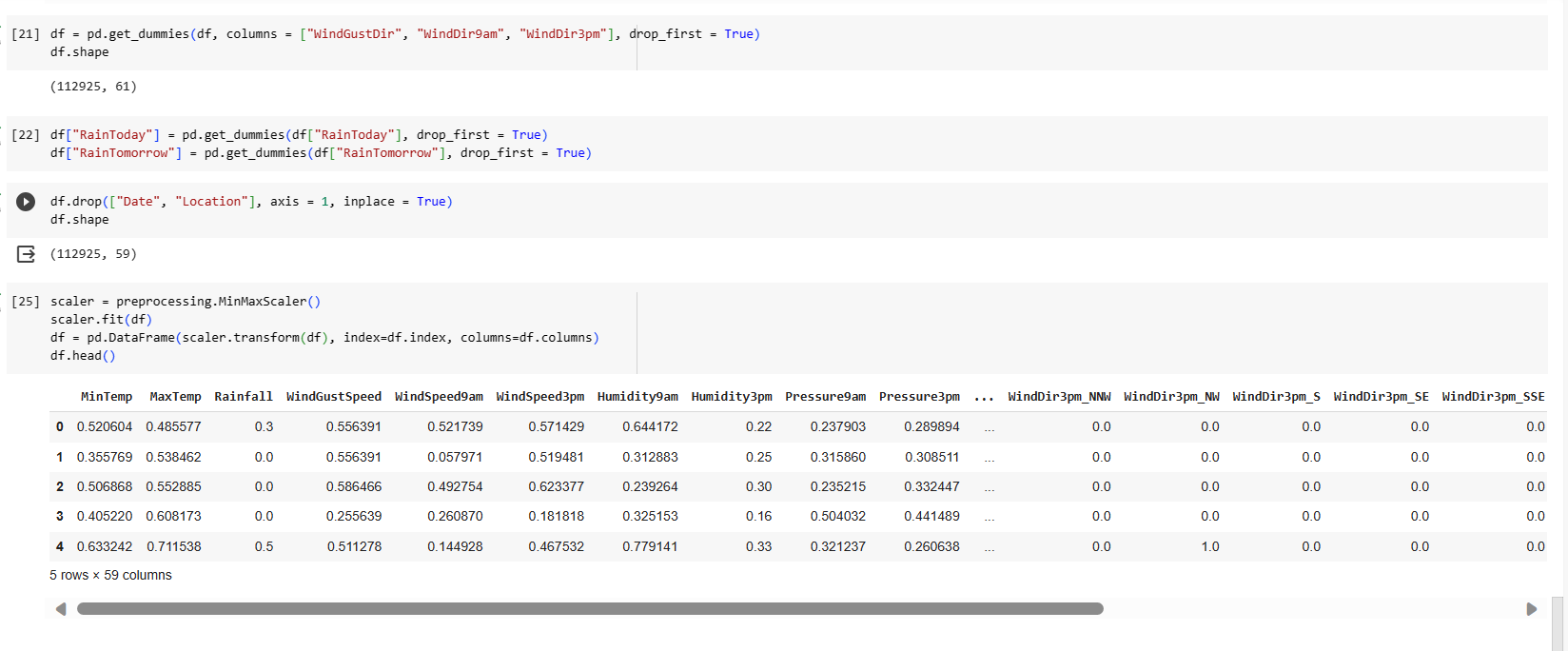
**Outliers:**



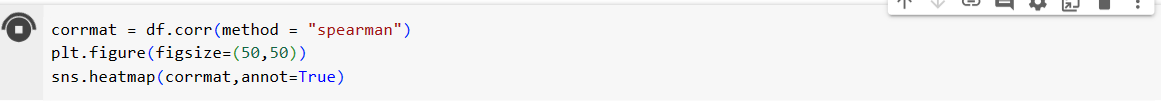


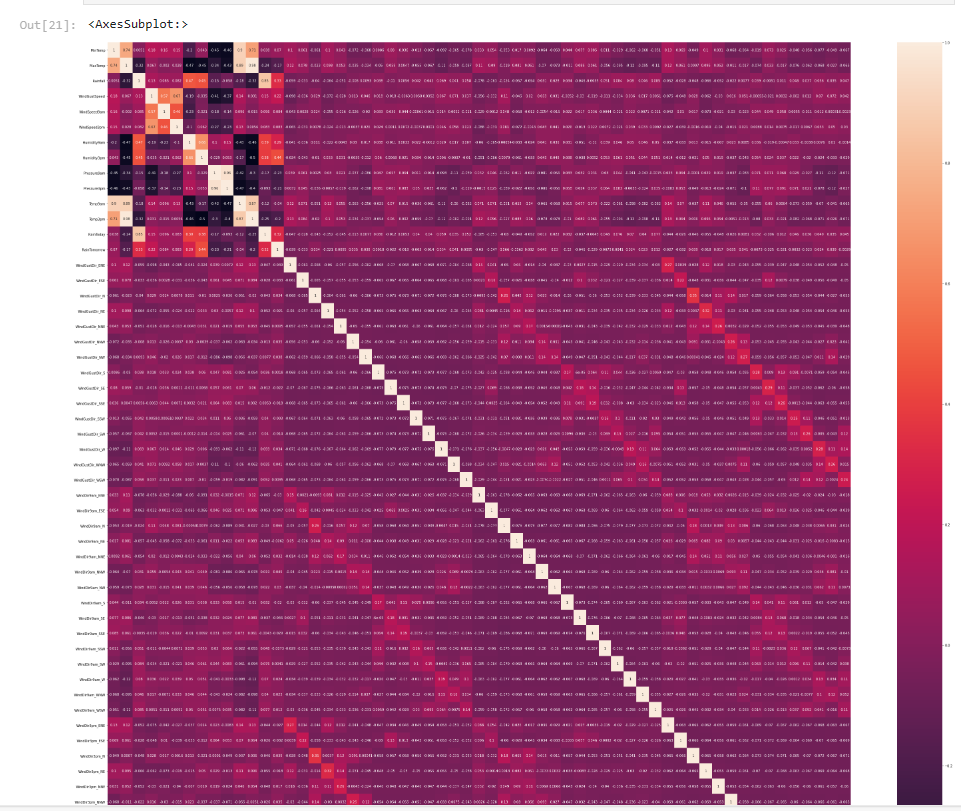


**Scaling:**



**Correlation**

****

****

### Feature Importance

### 

### Splitting of Data

### 

### Dealing with imbalance

### 

### SMOTE

### 

### ADASYN

### 

### Random oversampling

### 

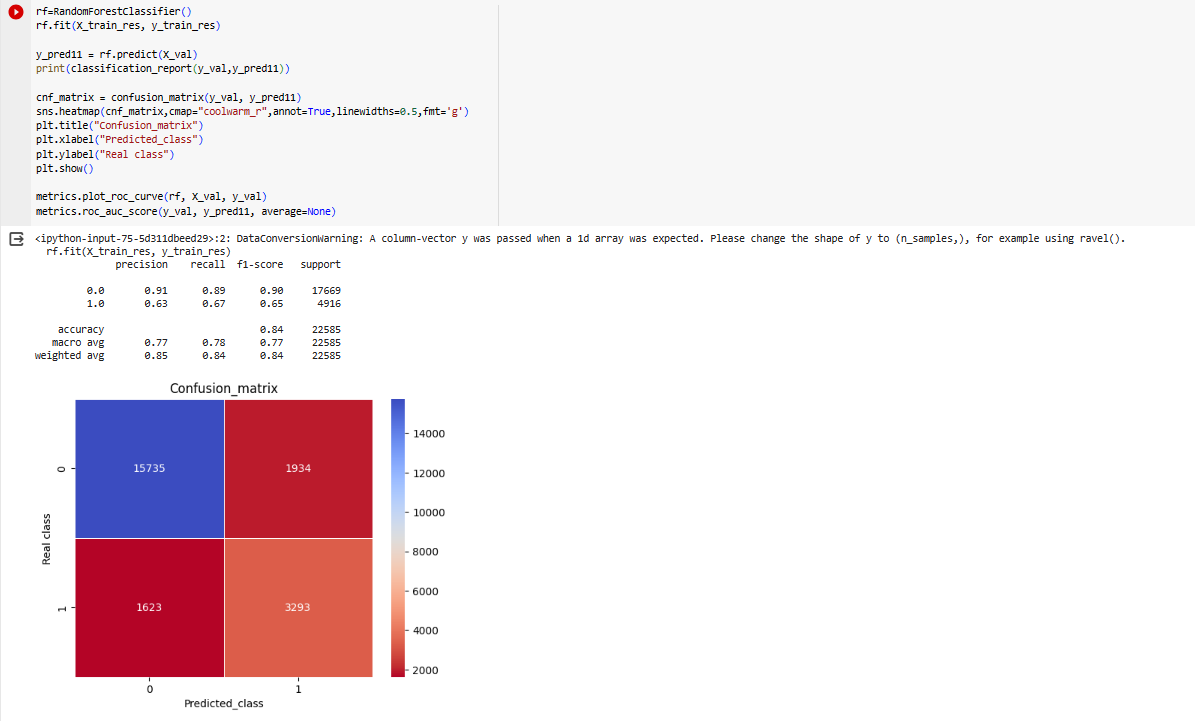
### Modelling:

### RANDOM FOREST:

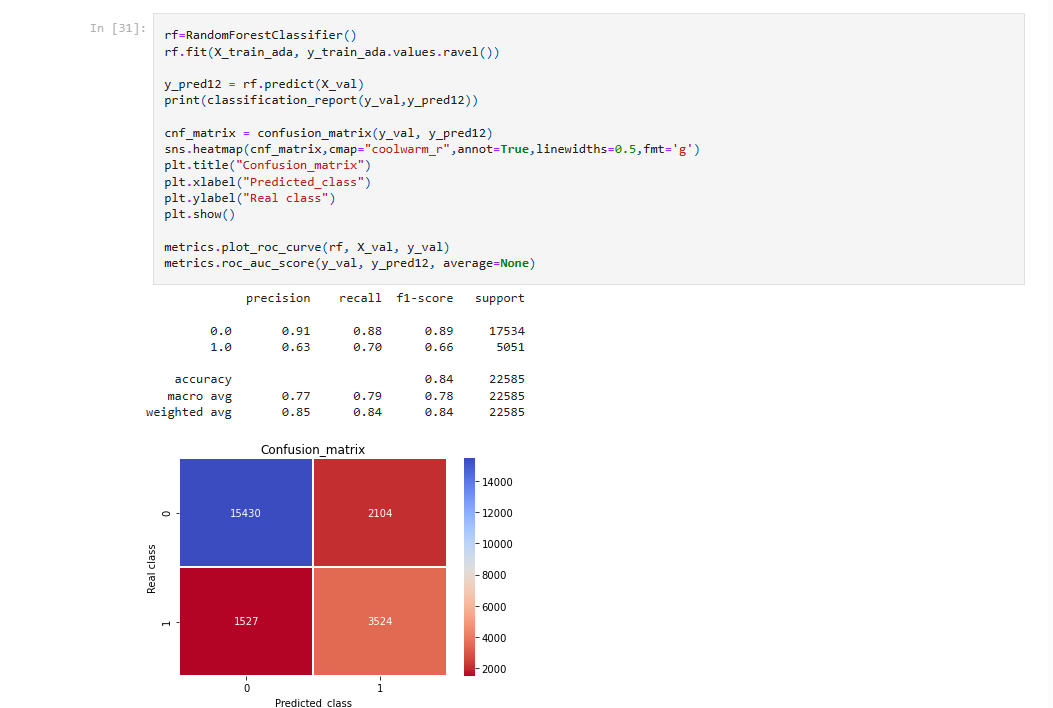
### Random Forest – Imbalanced

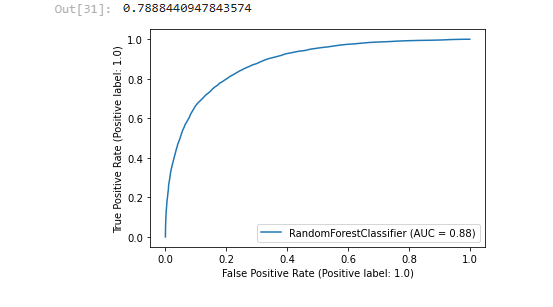
### 

### Random Forest - SMOTE

****

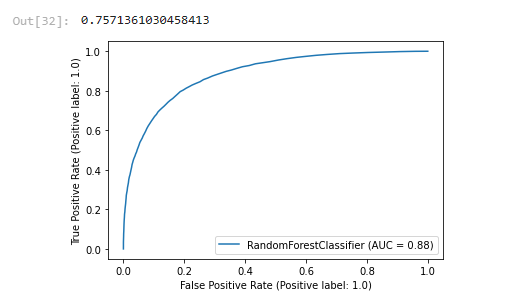
**Random Forest – ADA**

****

****

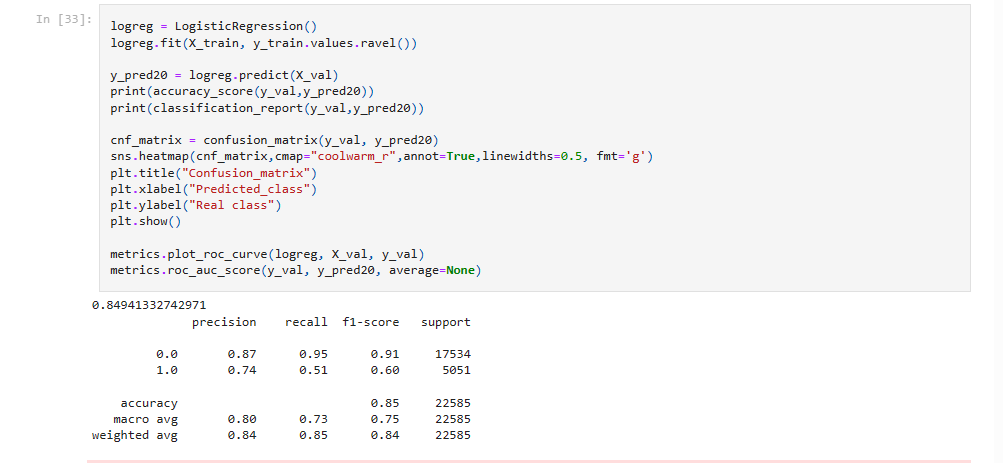
**Random Forest – Random OverSampling**

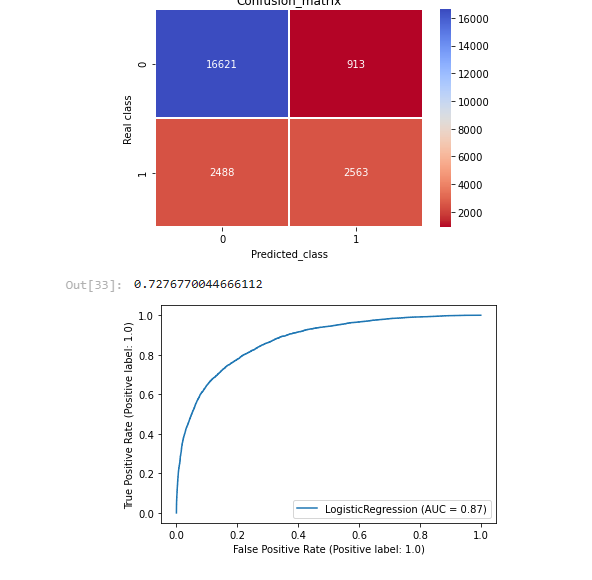
****

****

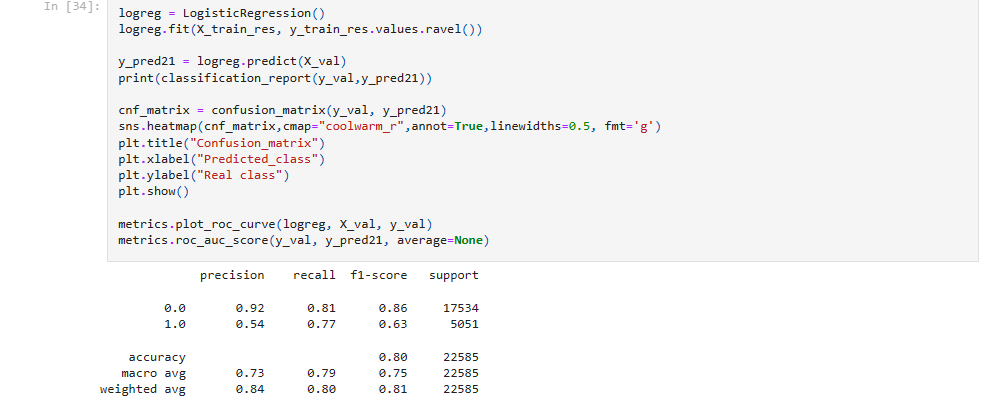
**LOGISTIC REGRESSION:**

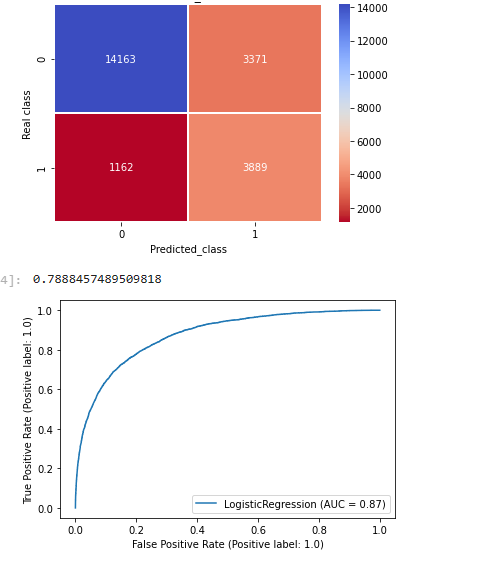
**Logistic Regression – Imbalanced**

****

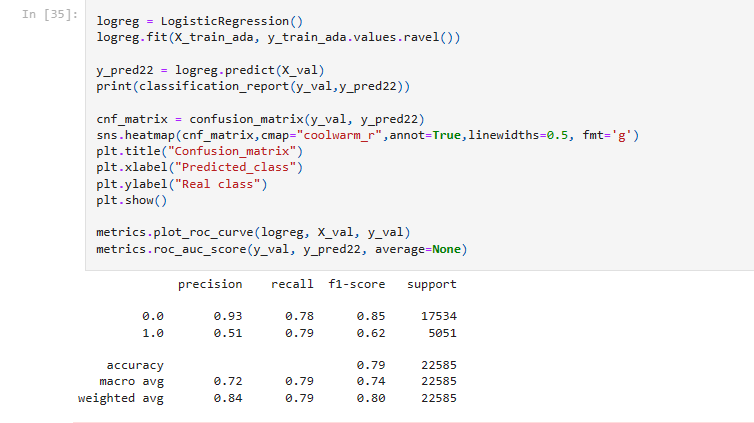
****

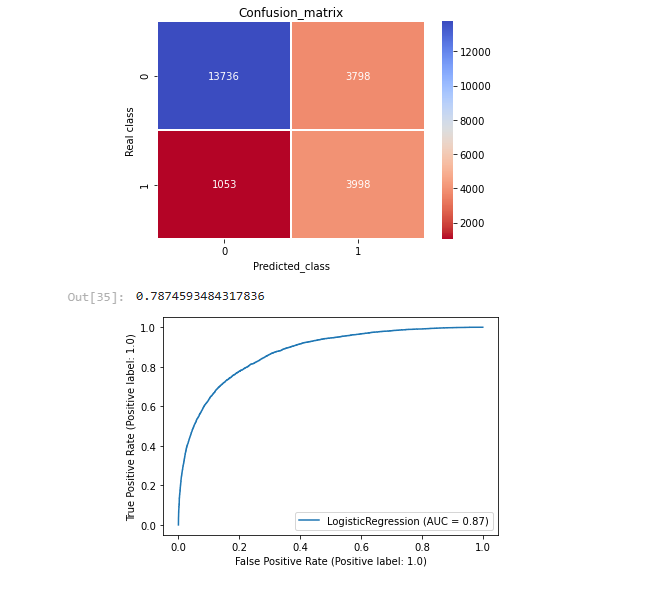
**Logistic Regression – SMOTE**

****

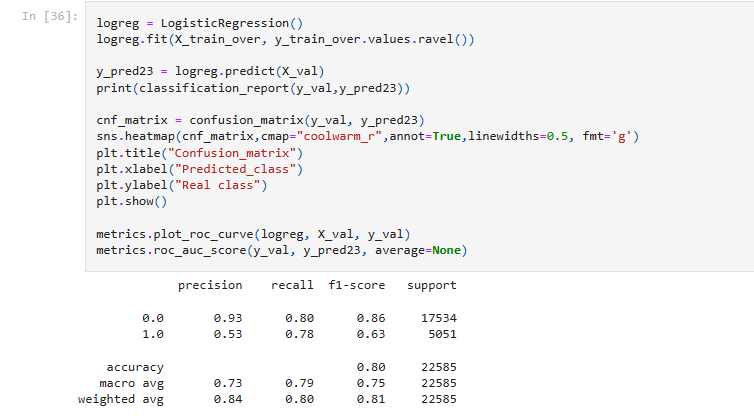
****

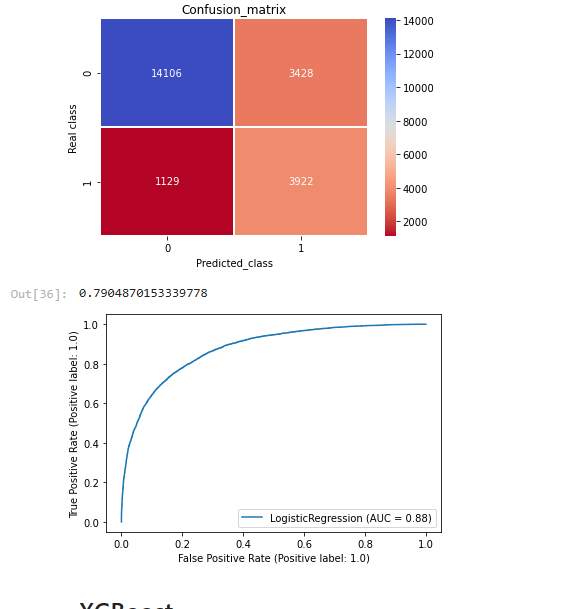
**Logistic Regression – ADA**

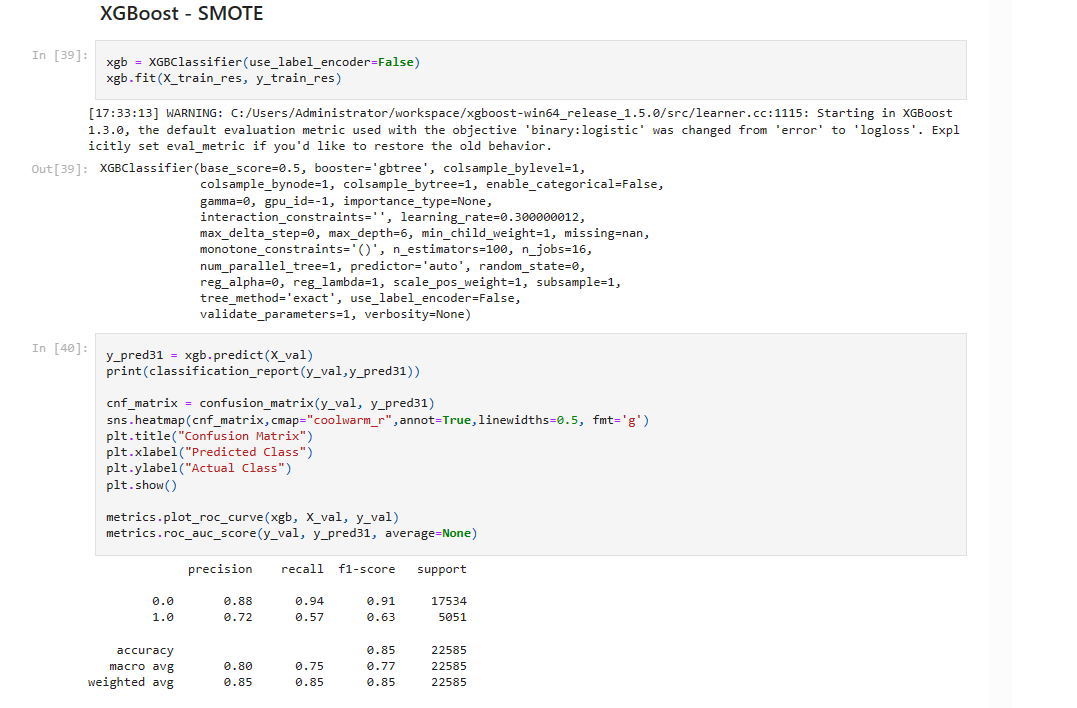
****

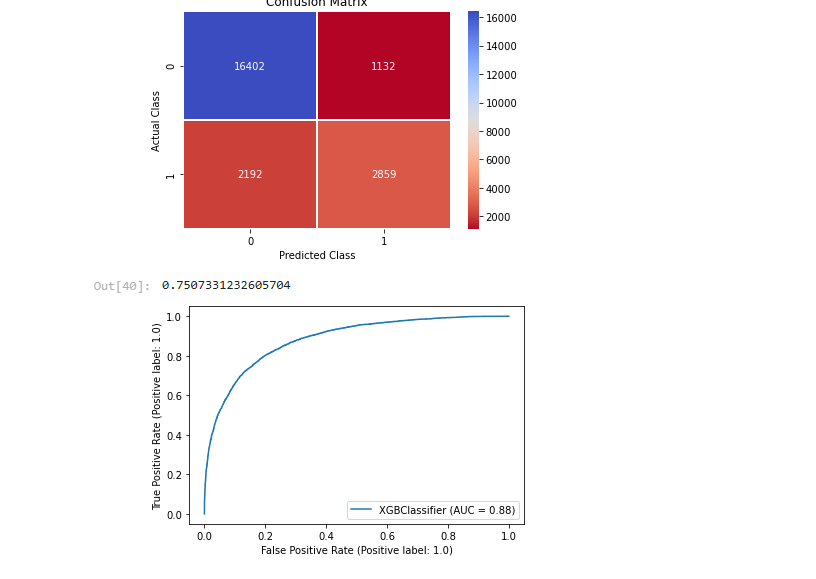
****

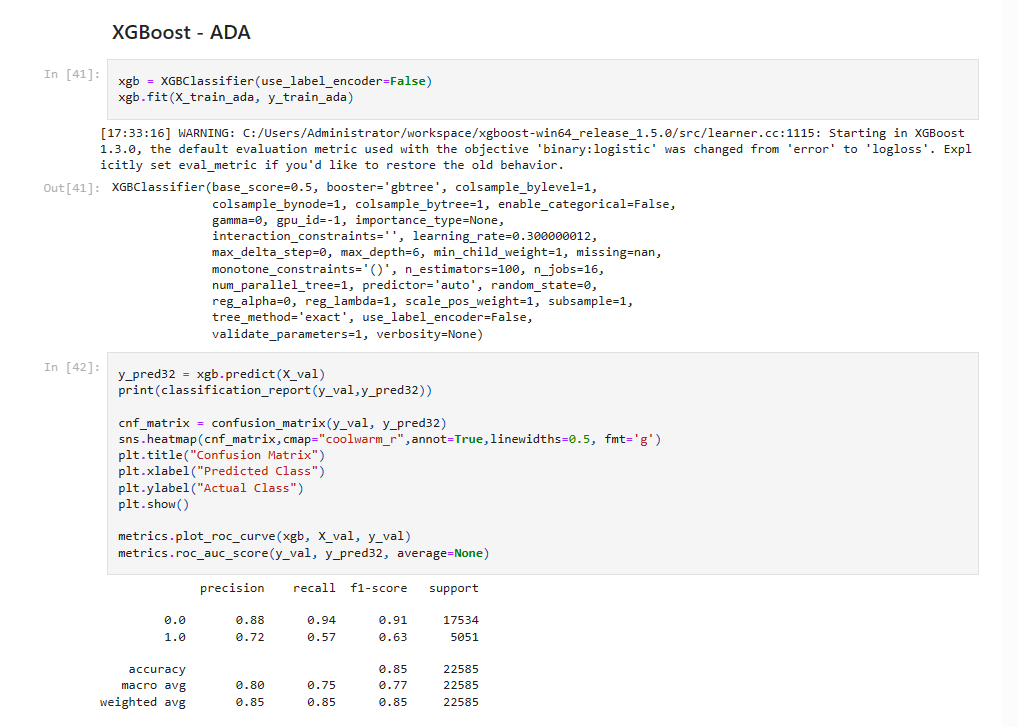
**Logistic Regression – Random OverSampling**

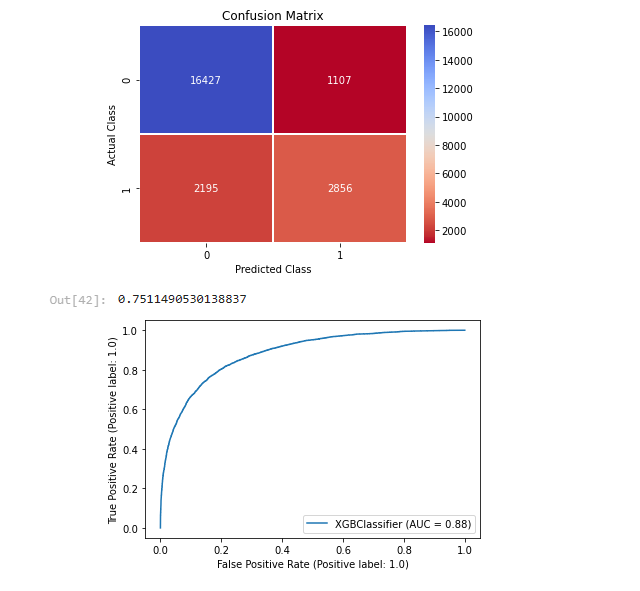
****

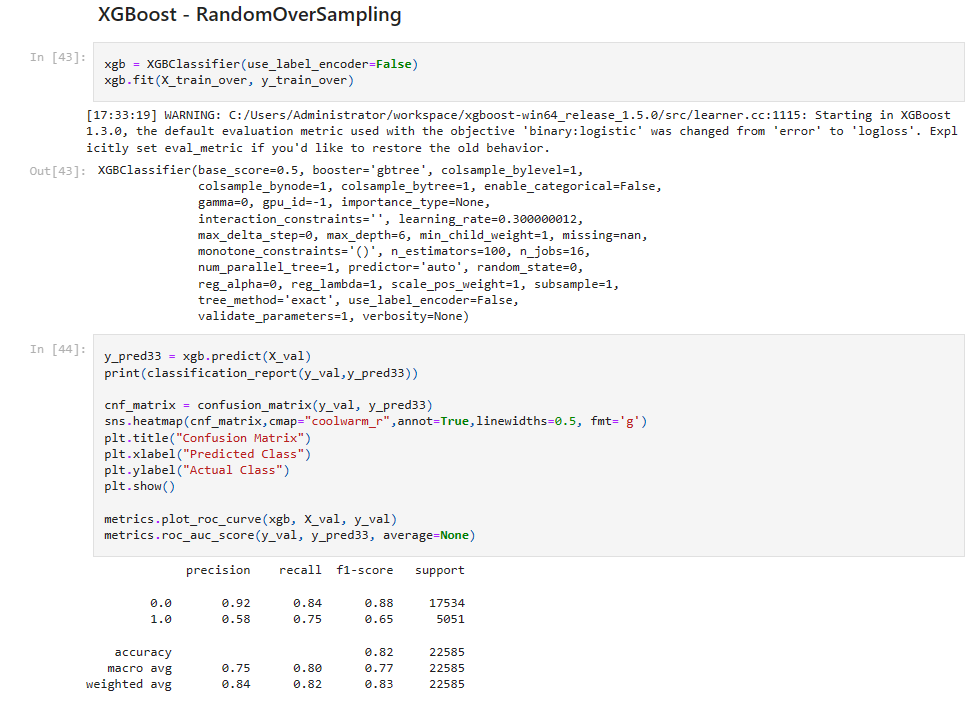
****

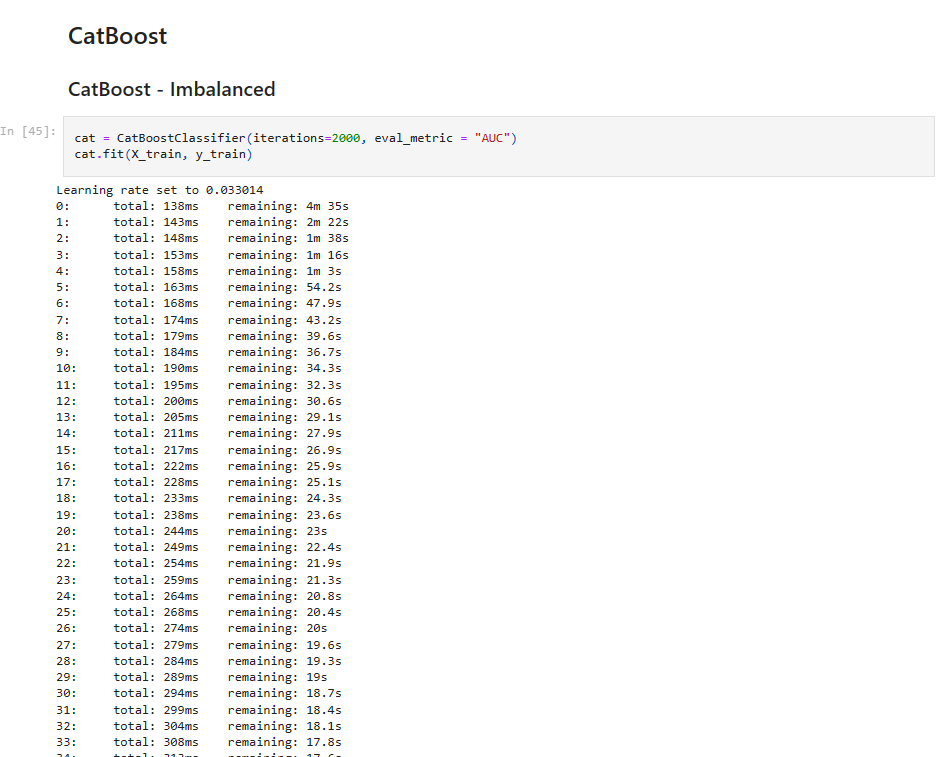
****

****

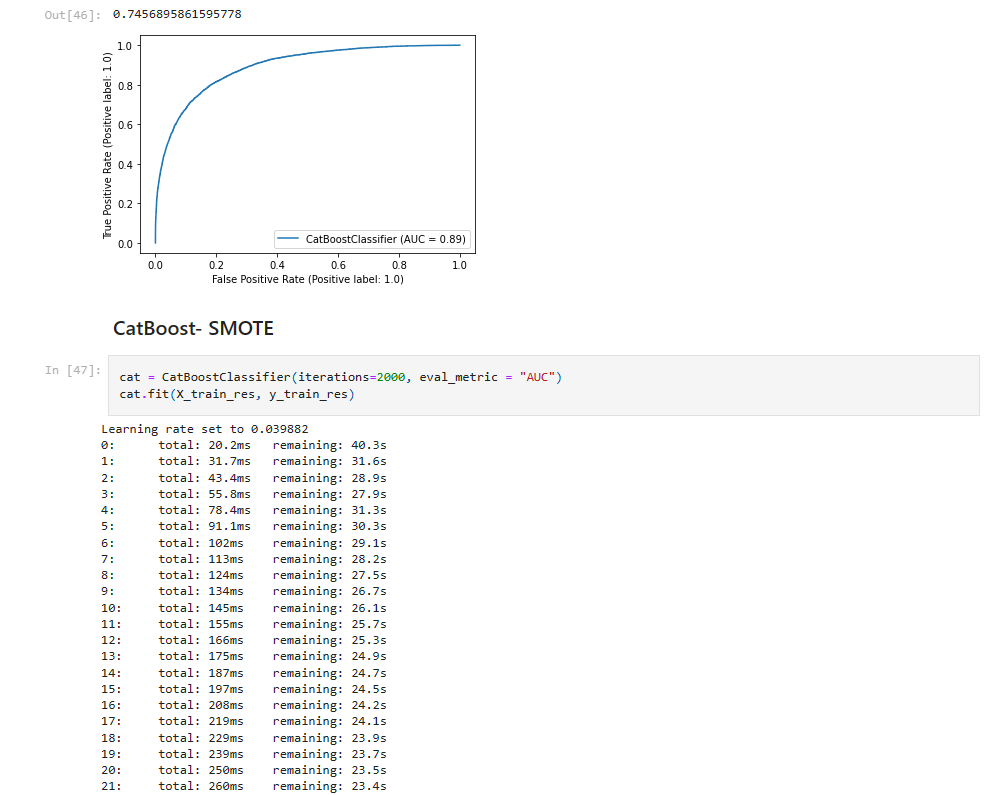
****

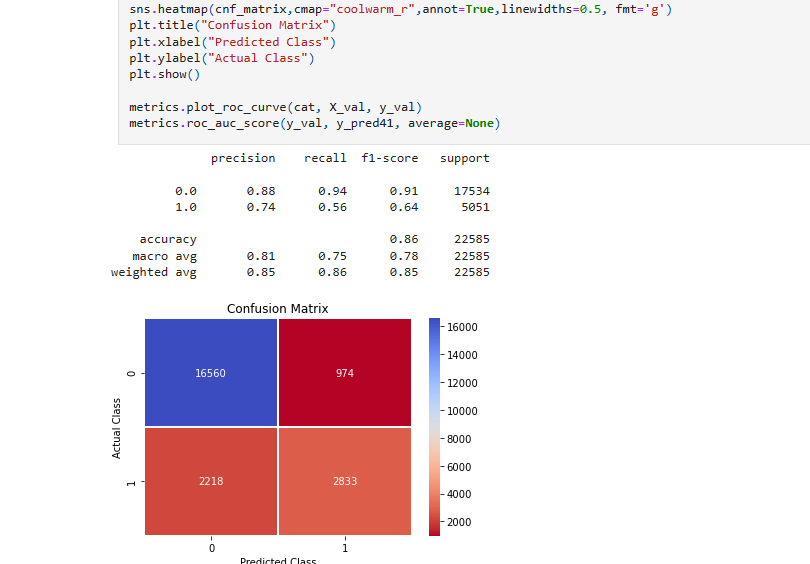
****

****

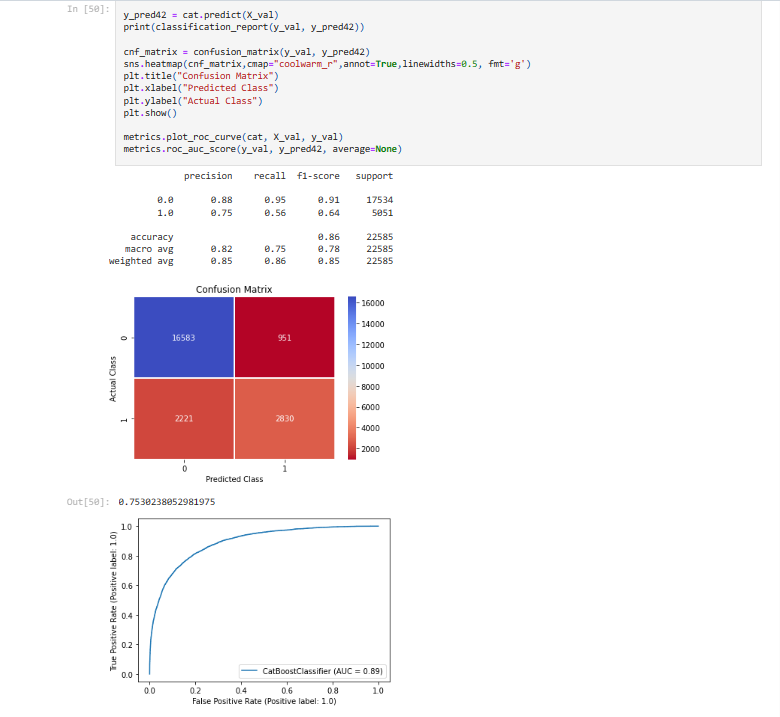
****

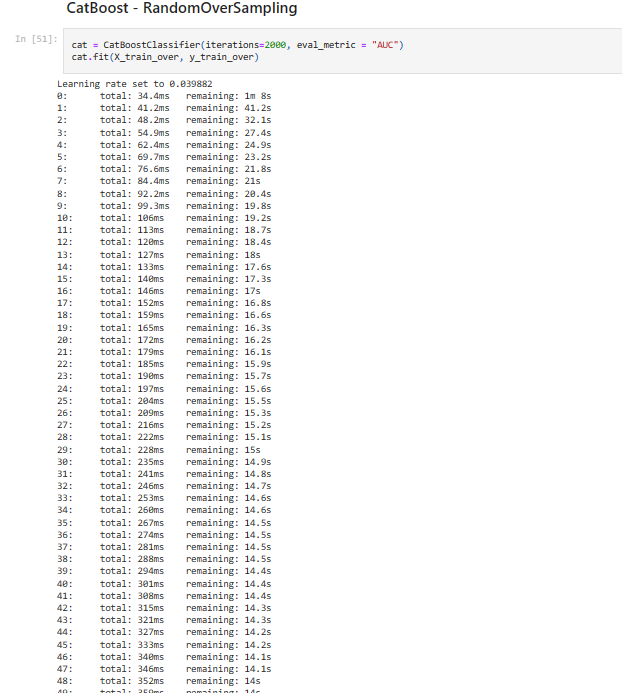
****

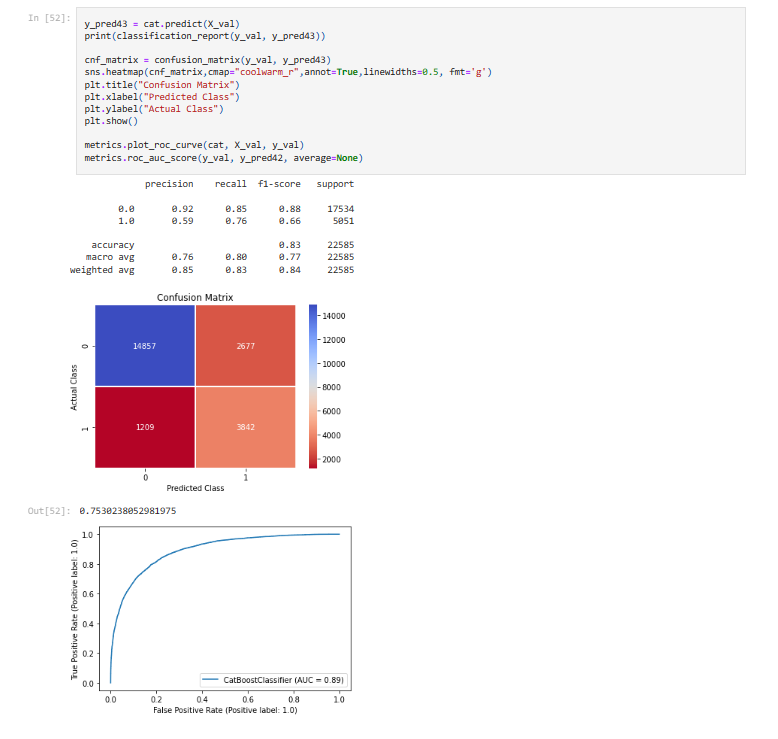
****

****

****

****

****

****

****



