Alma Mater Studiorum - Università di Bologna

BDA – PROJECT

Australia weather classification

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Artificial Intelligence

Riccardo Grassi

Academic year 2022-2023
JULY

PROJECT OVERVIEW

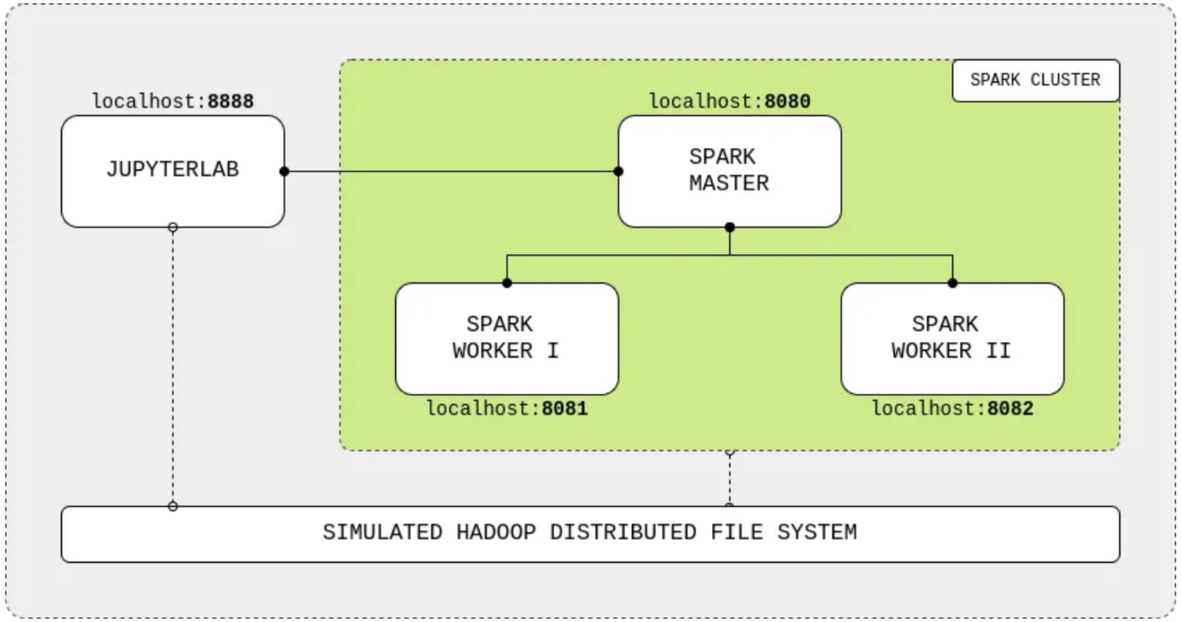
Objective

- Predict **next-day rain** by training classification models on the target variable **RainTomorrow.**
- PySpark library is used to analyze the dataset and build a machine learning model to identify if will it rain tomorrow or not.

Models

- Decision Tree
- Random Forest
- Logistic Regression
- GBT Classifier

Cluster set up



The cluster is composed of four main components: the JupyterLab IDE, the Spark master node and two Spark workers nodes (5GB of memory). The user connects to the master node and submits Spark commands through the nice GUI provided by Jupyter notebooks.

Data overview

The Dataset is available on Kaggle at the folllowing link: https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package.

Some properties:

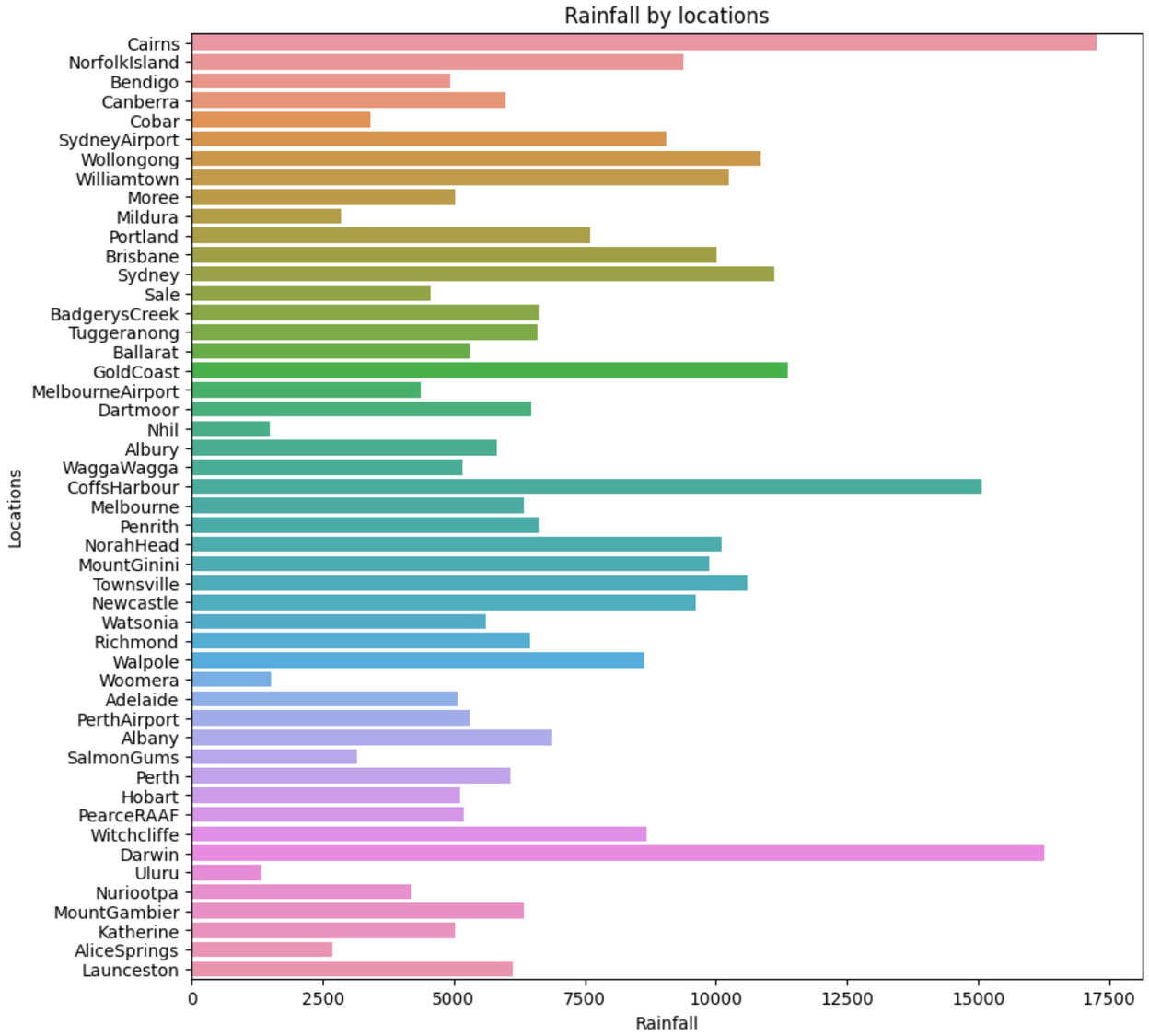
- It consists of 145460 rows and 23 features.
- 16 categorical and 7 numerical features.
- This dataset contains about 10 years of daily weather observations from many locations across Australia.
- Each row contains weather observations of a single day.

Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm
2008-12-01	Albury	13.4	22.9	0.6	NA	NA	W	44	W	WNW	20	24
2008-12-02	Albury	7.4	25.1	0	NA	NA	WNW	44	NNW	WSW	4	22
2008-12-03	Albury	12.9	25.7	0	NA	NA	WSW	46	W	WSW	19	26
2008-12-04	Albury	9.2	28	0	NA	NA	NE	24	SE	E	11	9
2008-12-05	Albury	17.5	32.3	1	NA	NA	W	41	ENE	NW	7	20

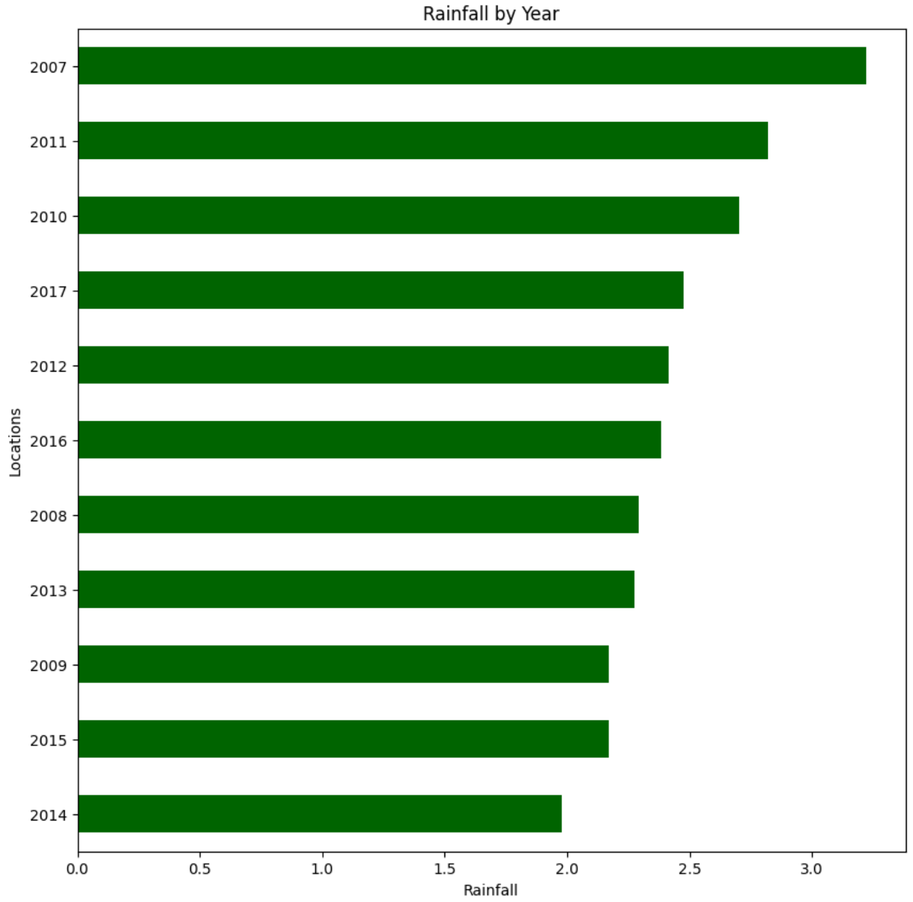
Exploratory Data Analysis

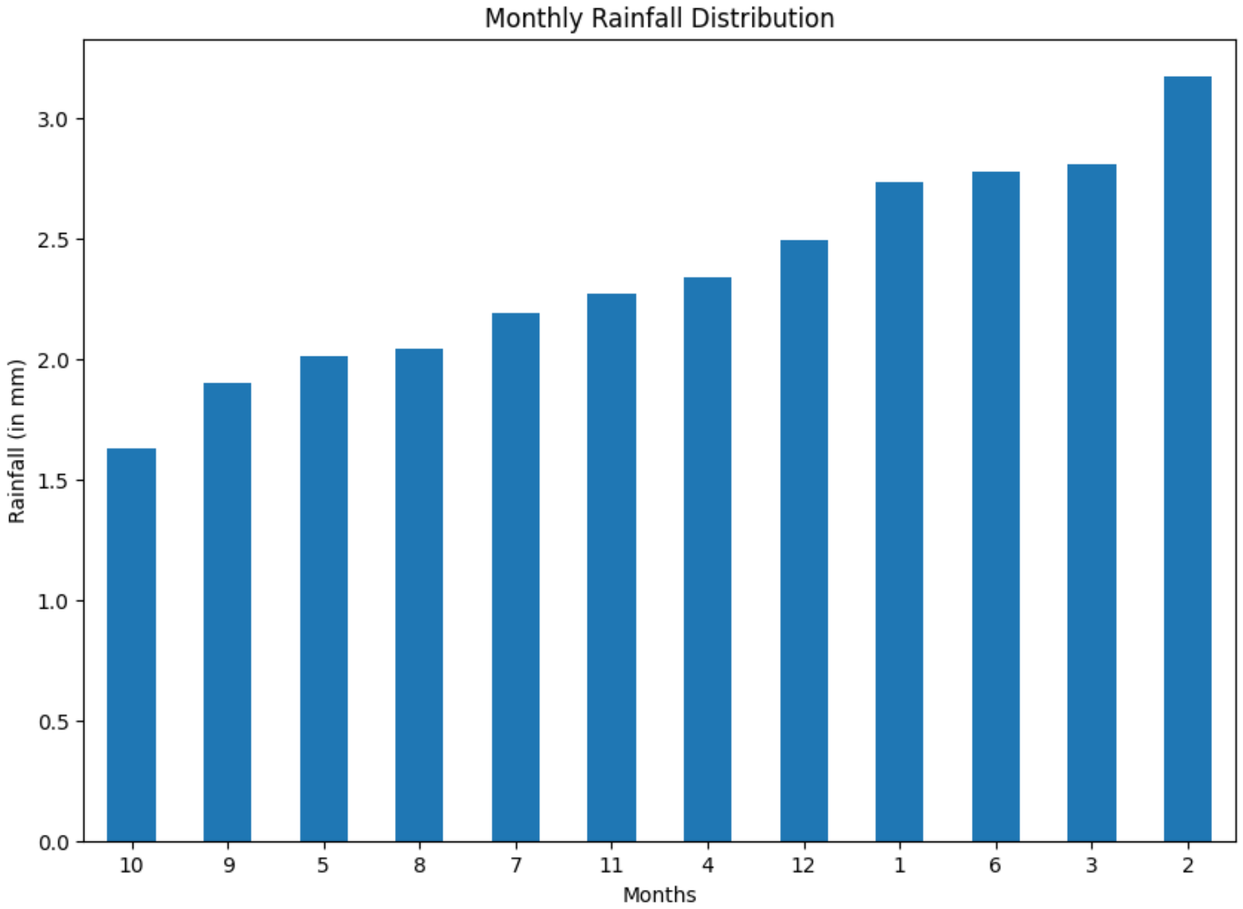
Which location received the highest Rainfall ?

Darwin and Cairs received the highsted rainfall, both situated in the North of Australia.

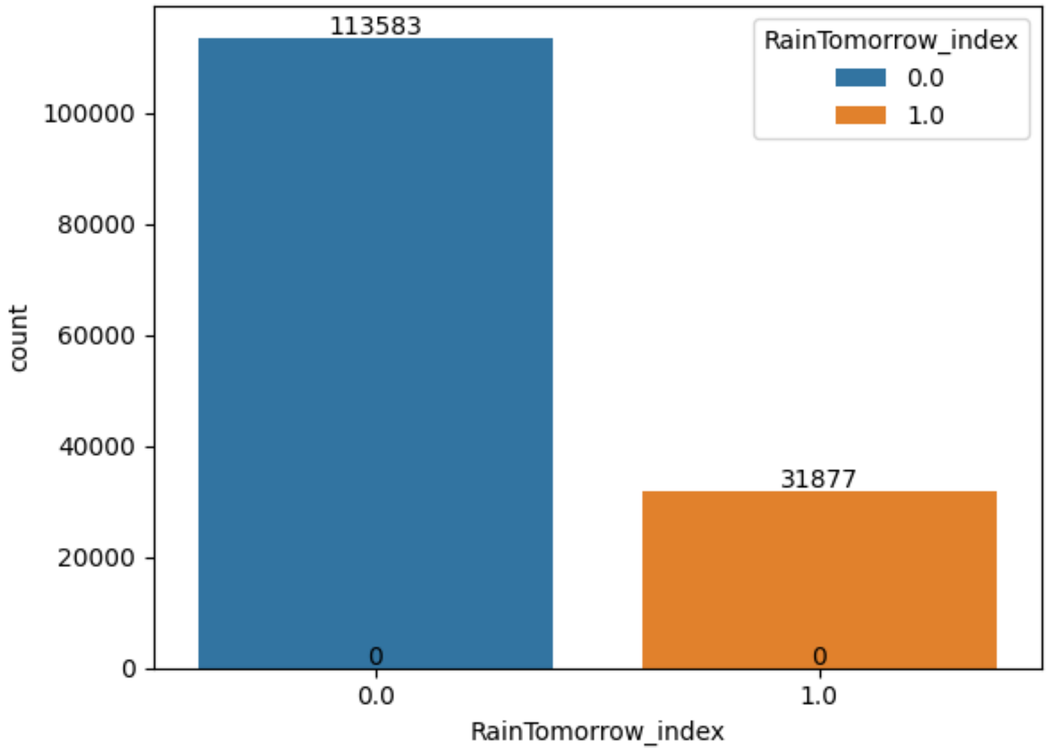


Rainfall distribution during years and months



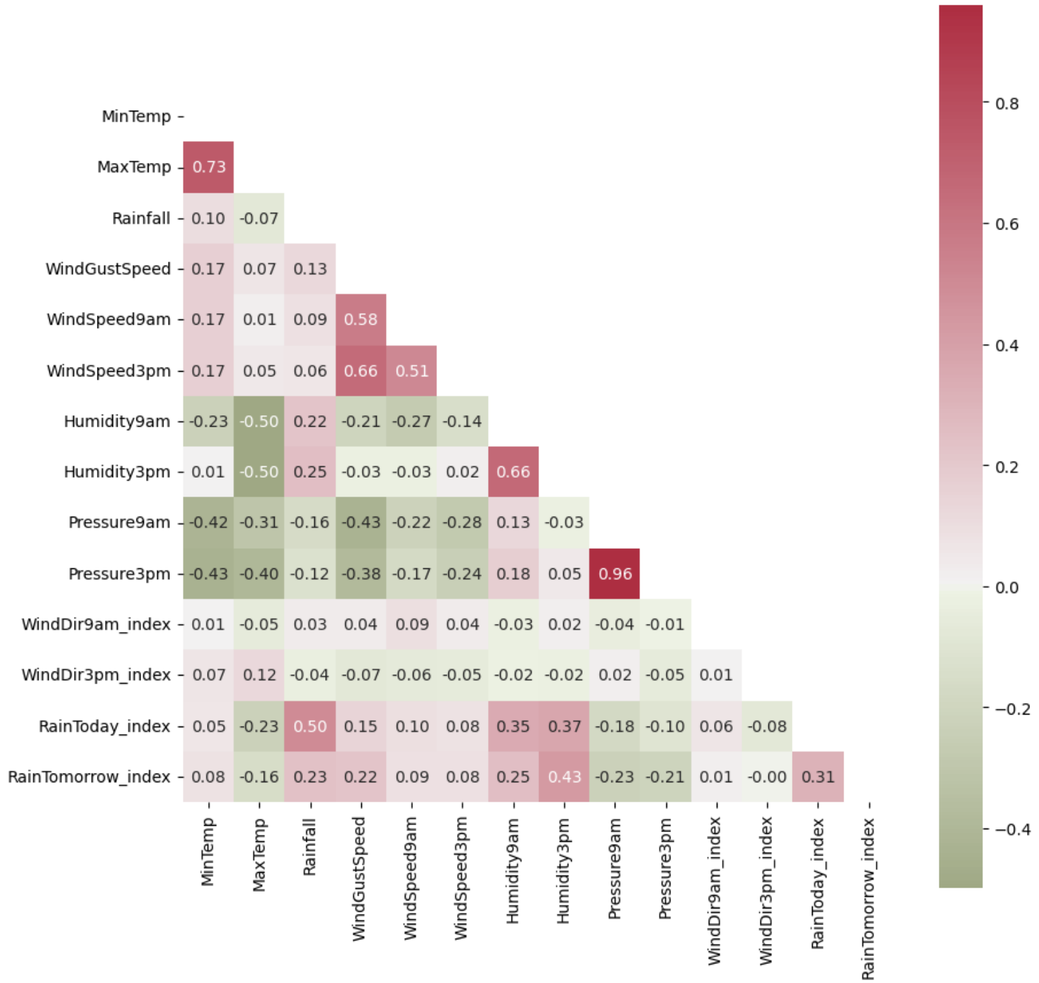


Target variable distribution



- **High imbalanced problem.**
- **F1 score is considered for measuring performances.**
- **The cost of false positives and false negatives is different.**
- **F1 score combines precision and recall to provide a balanced evaluation of the classifier's performance.**

Correlation among features



RainTomorrow_index is well correlated with Humidity3pm and **RainToday_index**

Data Preprocessing

Fill Null Values

Numerical columns

- Mean imputation

Null values are replaced with the average of the values contained in a single column.

	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN
1	9	NaN	9.0	0	7.0
2	19	17.0	NaN	9	NaN

mean()

	col1	col2	col3	col4	col5
0	2.0	5.0	3.0	6.0	7.0
1	9.0	11.0	9.0	0.0	7.0
2	19.0	17.0	6.0	9.0	7.0

Categorical columns

- Mode imputation

Null values are replaced with the most frequent item of that colum.

Make
Ford
Ford
Fiat
BMW
Ford
Kia
Fiat
Ford
Kia

Mode = Ford

Price
Ford
Ford
Fiat
BMW
Ford
Kia
Ford
Fiat
Ford
Ford
Kia

Data transformation

Numerical columns

Numerical value columns are firstly transformed by type casting them to double as they are defined as string.

```
# Convert values of numeric columns from the 'string' type into the 'double' type
for col_name in num_col:
    dataset = dataset.withColumn(col_name, col(col_name).cast('double'))
```

Categorical columns

Non numerical value columns are converted to numbers using stringindexer.

StringIndexer is a method or transformer used for converting categorical or string features into numerical indices. It assigns a unique index to each distinct string value in a column.

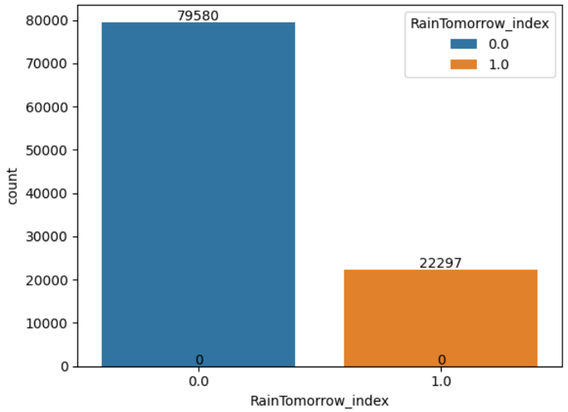
Feature Vector and Scaling

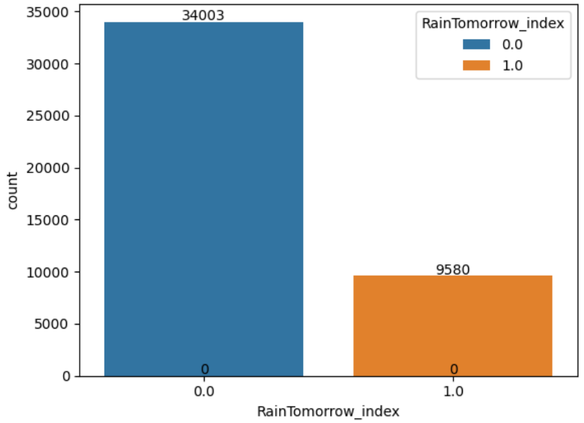
Before model fine-tuning, the feature have to be prepared in order to be processed by Machine Learning models. 2 steps have been performed:

- **Assembler**, a given list of columns is unified in one column, excluding the target feature;
- **Scaler**, a standard scaler has been applied, trasforming the data in a way that each feature has a mean of 0 and a standard deviation of 1.

Train and test split

The dataset has been splitten in 70% train set and 30% test set. Imbalanced class ratio is preserved both in train and test set.





Modeling

Models

- Decision Tree
- Random Forest
- Logistic Regression
- GBT Classifier

Decision Tree

The decision tree classifier builds a hierarchical structure of decision nodes and leaf nodes, where each decision node represents a test on a specific feature, and each leaf node represents a class label or a prediction.
Advantages:

- the resulting tree structure can be easily visualized and understood;
- They can handle both categorical and numerical features;
- they can capture non-linear relationships between features and the target variable.

Random Forest

A random forest classifier is an ensemble learning method that combines multiple decision trees to make predictions.
Advantages:

- Random forests can handle datasets with a large number of features effectively
- Random forests are less sensitive to outliers and noisy data points compared to individual decision trees
- By randomly selecting subsets of features, they can identify relevant features and reduce the impact of irrelevant ones.

Logistic Regression

Logistic regression classifier is a popular algorithm used for binary classification tasks. It is actually a probabilistic model that estimates the probability of an instance belonging to a certain class. It is widely used for its simplicity and interpretability.

Advantages:

- Logistic regression is computationally efficient and can handle large datasets with a high number of features
- Logistic regression produces probability estimates of class membership rather than just binary predictions
- It helps in identifying the most influential features in the classification process.

GBT Classifier

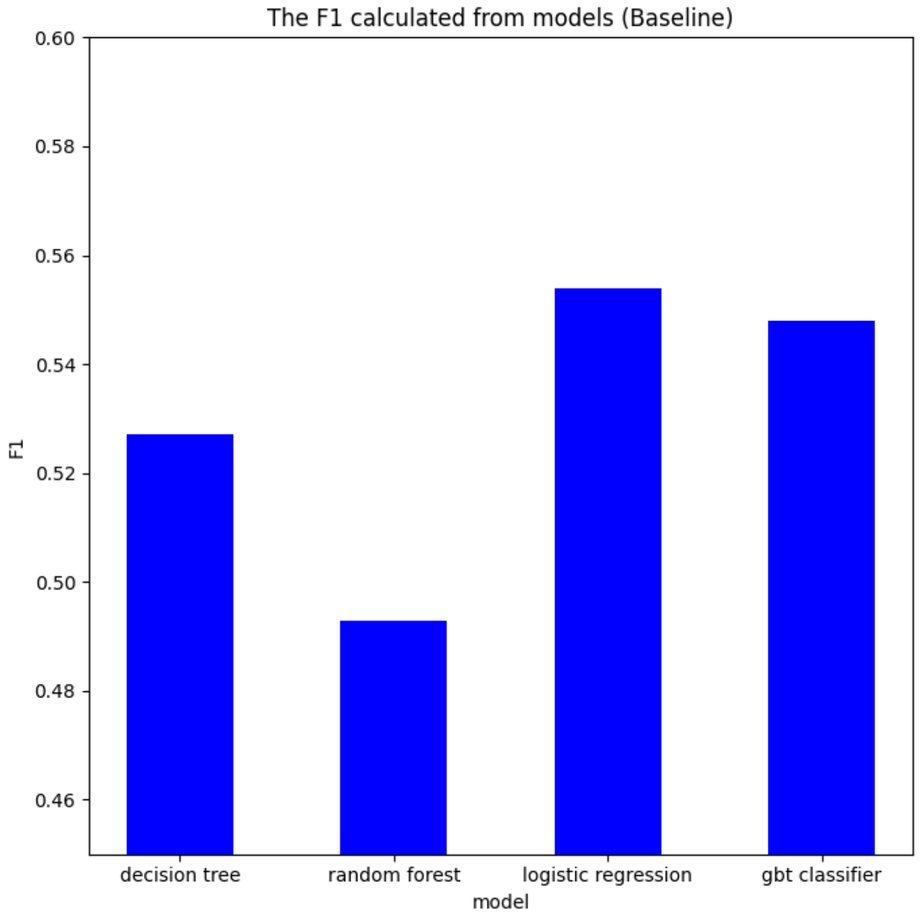
Gradient Boosting Trees is an ensemble method that combines multiple decision trees to make predictions.

Advantages:

- Gradient Boosting Trees often achieve high accuracy and predictive performance, as they can capture complex nonlinear relationships in the data
- Gradient Boosting Trees are less sensitive to outliers compared to some other algorithms.
- The algorithm provides a measure of feature importance based on how frequently the features are used in the ensemble of trees.

Results

Results



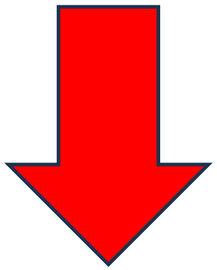
Model	F1 Train	F1 Test
Decision Tree	0.5315	0.5270
Random Forest	0.4932	0.4927
Logistic Regression	0.5600	0.5539
GBT Classifier	0.5545	0.5479

Results Improvement

Resampling

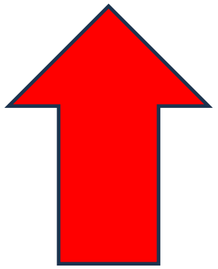
Downsampling

Refers to reducing the number of instances in a dataset, usually by randomly removing instances from the majority class. The goal of downsampling is to balance the class distribution, particularly in situations where one class is significantly overrepresented compared to the others.



Upsampling

Upsampling is the opposite of downsampling and involves increasing the number of instances in a dataset, particularly in scenarios where one class is underrepresented compared to the others. Upsampling is commonly used to address class imbalance issues, where the minority class has insufficient data for effective model training.



Reweighting

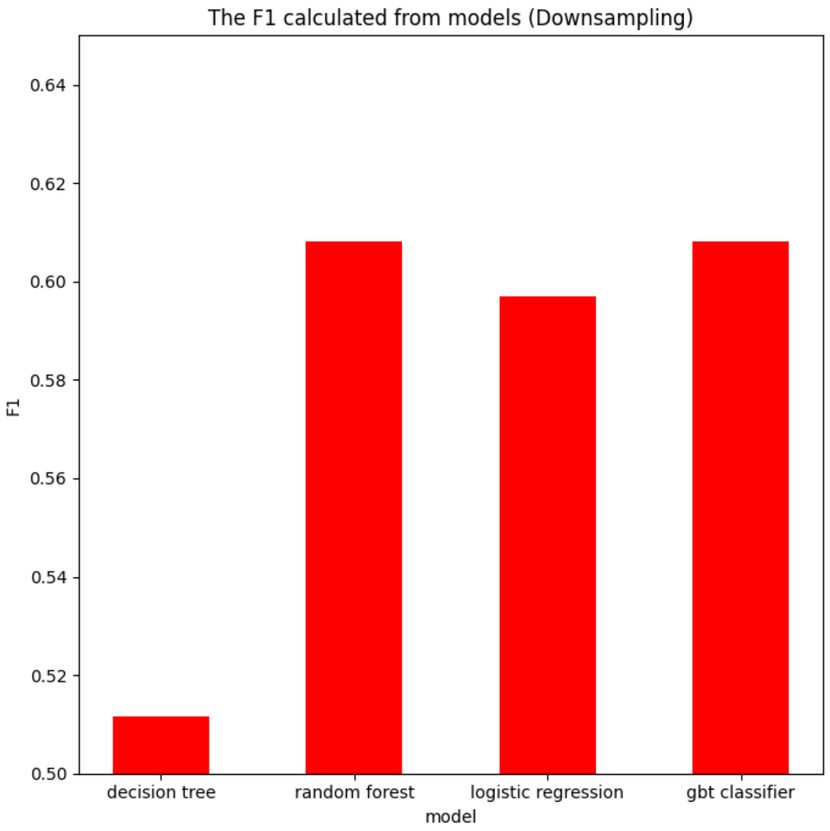
- By reweighting the dataset, you can balance the contribution of each class during model training without changing the distribution of the samples in the original dataset. This approach can be beneficial in cases where resampling techniques may not be suitable or may introduce unintended biases.

- PySpark provides the setClassWeight method, which allows you to set the class weights in each model. By specifying the appropriate class weights, you can ensure that the model pays more attention to the minority class samples, leading to a more balanced learning process.

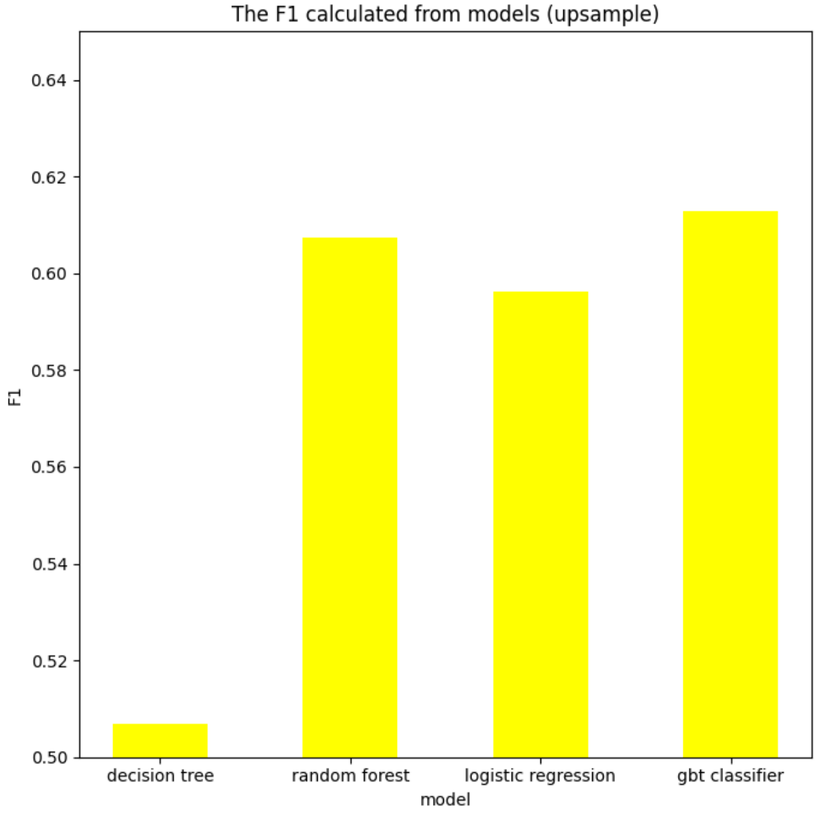
- Overall, reweighting the dataset is a technique that assigns higher weights to the minority class samples during training, effectively addressing class imbalance without altering the original distribution of the dataset. PySpark provides the necessary functionality to adjust the class weights in the machine learning models to achieve this reweighting effect.

Resampling Results

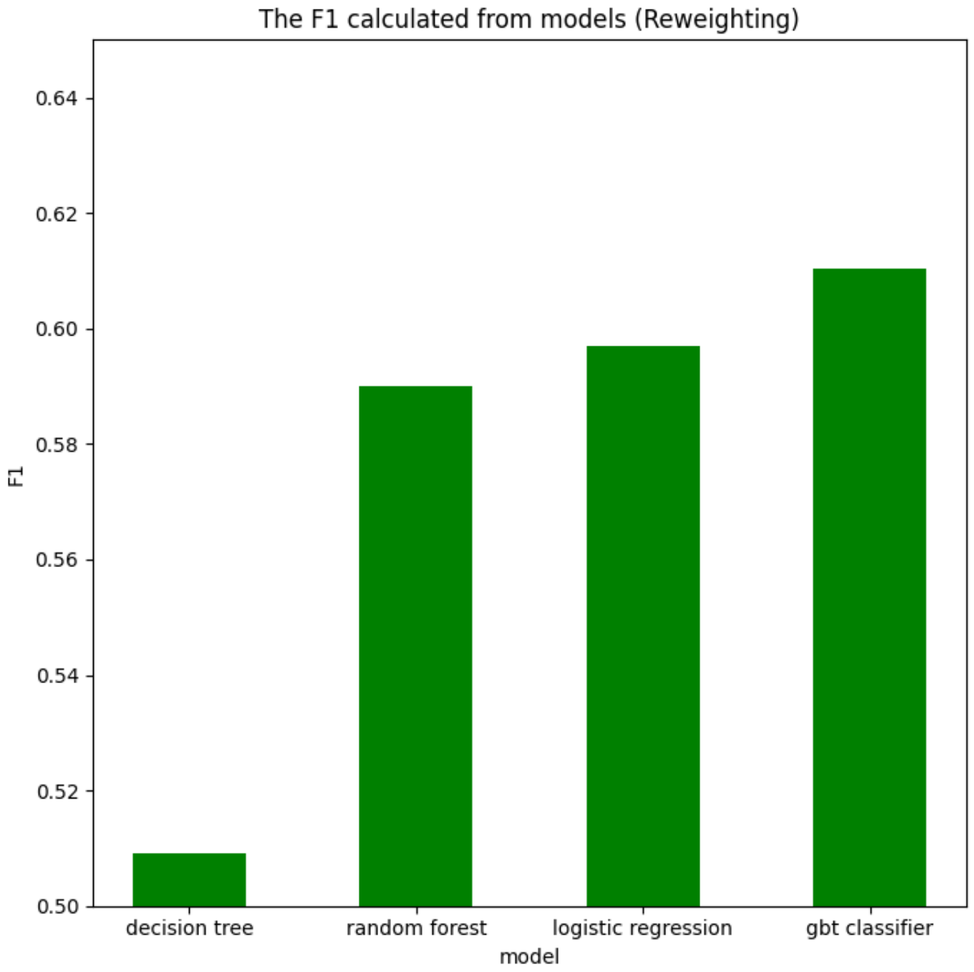
Downsampling



Upsampling



Reweighting



Overall Results

Model / F1	Baseline	Downsampling	Upsampling	Reweighting
Decision Tree	0.5270	0.5114	0.5068	0.5091
Random Forest	0.4927	0.6080	0.6074	0.5900
Logistic Regression	0.5539	0.5970	0.5963	0.5969
GBT Classifier	0.5479	0.6081	0.6128	0.6104

Overall Results

- GBT classifier reached the best result
- Resampling technique incresed F1 Score (not in Decision Tree)
- Random Forest benefit a lot with Resampling and Reweighting
- No evident diffrence between Upsampling and Downsampling
- With baseline models no overfitting

Future Improvements

- Increase parameters in the hyper-parameters tuning phase
- Increase dataset dimension
- Try other classification models
- Deriving new features from existing ones or transforming the existing features to better represent the underlying patterns in the data.
- Transform the existing features to make them more suitable for the classification task.