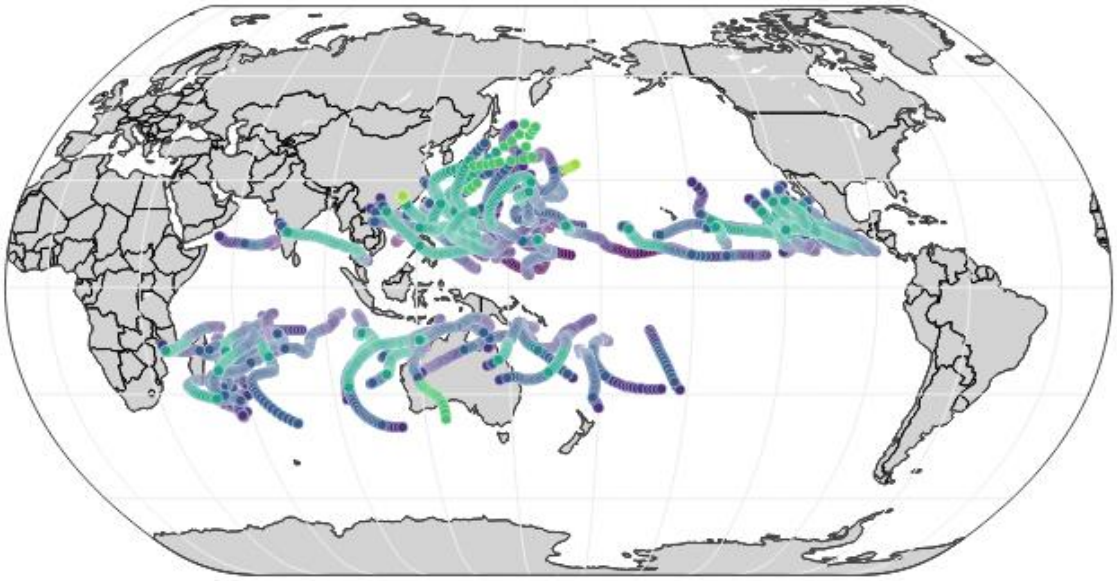
School of Engineering

Applied MSc in Data Analytics
Applied MSc in Data Science & Artificial Intelligence
Applied MSc in Data Engineering & Artificial Intelligence

A24 - Machine Learning with Python Lab

Tropical Cyclone Severity Prediction



Instructor: *Pauline Salis*

Authors: *Vincent Boettcher*
Favour Chinonyerem Okike
Audrey Costes
Lydéric Gandibleux
Arnaud Gomez
Claire Peyran

30 March 2025

Introduction

Tropical cyclones are among the most devastating weather events on Earth, causing severe damage to both natural ecosystems and human communities, with high mortality rates and significant economic losses (Wu et al., 2022). For instance, the Great Bhola Cyclone of 1970, the deadliest cyclone ever recorded, led to the deaths of at least 300,000 people in Bangladesh. In many cases, populations are caught off guard, and fatalities could be mitigated with better preparedness and forecasting. Furthermore, recent assessments agree that the frequency and intensity of tropical cyclones are expected to increase due to global warming (Sobel et al., 2016). In this context, developing reliable forecasting systems to avert future disasters is critical, which implies a deeper understanding of tropical cyclone behavior and improvement in weather prediction models.

Over the past few decades, best-track data have become instrumental in analyzing the evolution of tropical storms. Unlike real-time data, best-track data are reanalyzed post-season, incorporating additional information such as satellite observations or in-situ measurements for improved accuracy (Knapp et al., 2010). Best-track datasets are thus mostly used by researchers and forecasters to improve knowledge about weather phenomena rather than for real-time monitoring. The International Best Track Archive for Climate Stewardship (IBTrACS) is one of the most comprehensive global collections of tropical cyclone data, emerging from the collaboration between the World Meteorological Organization (WMO), Regional Specialized Meteorological Centers (RSMC) and other institutions around the world. It compiles multiple historical datasets and is continuously updated by different meteorological agencies worldwide. For example, the TD9636 dataset, on which focused this project, is a historical dataset compiled between the 1960s and late 1980s by the National Oceanic and Atmospheric Administration (NOAA) and the National Climatic Center. It represents a collection of global storm records, mostly outside the North Atlantic, gathered from multiple sources. Despite the availability of massive amounts of large-scale weather data, these datasets remain underutilized in improving the understanding and forecasting of these extreme events. However, predictive models still face significant challenges as most widely used tropical cyclone dynamical forecast models exhibit relatively low accuracy, while traditional statistical models generally struggle to deal with the complex and nonlinear relationships between tropical cyclone predictors (Chen et al., 2020).

In this context, machine learning offers a new approach to enhance the accuracy and efficiency of tropical cyclone prediction. As a core technology of artificial intelligence, machine learning consists of a series of computational models using statistics techniques to make inferences from observations. Given a model with defined features, the learning process involves optimizing these parameters using training data, enabling the model to predict future states or extract meaningful knowledge from the data. Machine Learning methods have substantial advantages in processing large-scale datasets and provide new strategies for addressing the difficulties in tropical cyclone forecasting (Wang et al., 2022).

The objective of this project is to apply a machine learning approach to classify cyclone intensity, which is typically defined by the maximum surface wind speed or the minimum sea-level pressure at the storm's center (Wu et al., 2022), using geographical and environmental input data from a best-track dataset. However, the definition of tropical cyclone intensity can vary depending on the oceanic region, and there is no universally accepted standard definition. Therefore, this project focuses on the intensity stages defined in the TD9636 framework. Specifically, the goal is to develop and refine a classification model based on the categories outlined in the TD9636_STAGE, using parameters available in the IBTrACS dataset (see notebook *ml_pipeline.ipynb*).

Methodology

Data set description

Dataset and features presentation. The dataset used in this project consisted of data extracted from the IBTrACs dataset, covering the period from 1980 onward and comprising 297,098 rows and 174 columns. The dataset gathers cyclone data from more than 10 meteorological agencies[1] and 5 data collections[2], covering storm activity

[1] The group of USA agencies, RSMC at Tokyo, Chinese Meteorological Administration (CMA) Shanghai Typhoon Institute, Hong Kong Observatory (HKO), Korea Meteorological Administration (KMA), RSMC at New Delhi, RSMC at La Reunion (Meteo France), Australian Bureau of Meteorology (BoM), New Zealand MetService at Wellington, RSMC at Nadi.

[2] DS824, TD9635, TD9636, Neumann Southern Hemisphere Dataset, Chenoweth Dataset.

across the globe. Each row is uniquely identified by a combination of a storm identification number (SID) and a time record (ISO_TIME), representing data collected from meteorological agencies worldwide.

IBTrACs provide general variables that serve as broad descriptions of the data reported by the WMO. These include summarized physical parameters and general metadata about each measurement such as:

Storm identification:
- Unique identifier for the storm (SID),
- Storm name (NAME),
- Year the storm began (SEASON),
- the number of the storm for the year (NUMBER),

Temporal and geographic information:
- Timestamp of the observation (ISO_TIME),
- Basin and sub-basin where the storm occurred (BASIN, SUBBASIN),
- Mean latitude and longitude derived from multiple agency reports (LAT, LON),
- Storm's distance to land (DIST2LAND, LANDFALL)

Physical parameters from different agency reports:
- Wind speed (WMO_WIND, STORM_SPEED, STORM_DIR)
- Atmospheric pressure (WMO_PRES)

Additional information:
- Agency responsible for this location (WMO_AGENCY)
- Storm classification, based on a combination of what is proposed by the agencies (NATURE)

Most of this data were assigned or derived by IBTrACS algorithm, and each value is associated with a corresponding interpolation flag (IFLAG) to indicate how it was processed. In addition, each agency reports numerous parameters, mostly related to the wind structure (wind speed, direction, gust period, radial extent), atmospheric conditions (pressure, eye diameter), geographic location (latitude, longitude), and storm classifications which vary depending on the agency. The common parameters across agencies include wind speed, atmospheric pressure, latitude, and longitude.

Target variable. As the project focuses on predicting the TD9636_STAGE parameter, all rows with missing values in this column were removed from further analysis because they could not have served to train or validate the model. This filtering resulted in a dataset of 48,343 rows and 99 columns. The TD9636_STAGE variables originate from the TD9639 dataset, which was constructed in the 1960s by NOAA as a global collection of storms derived from multiple sources. It has not been updated since the 1980s, thus the dataset for this project covers the period from 1980 to 1989 and does not monitor North Atlantic.

The TD9636_STAGE classification is not clearly documented, and it does not correspond to one of the official intensity scales such as the Dvorak Technique, which relies on satellite imagery, or the Saffir-Simpson Scale, which only applies to hurricanes and excludes disturbances, extratropical storms or dissipating systems. Instead, TD9636_STAGE is to be a 7-level numerically coded categorical parameter (Table 1), with an additional "unknown" category that was not present in the covered period for this project and appears to be a custom wind-based classification system developed by NOAA. One notable feature of this classification is that it uses different wind thresholds for different oceanic basins. This flexibility suggests that the TD9636_STAGE was designed to standardize historical cyclone records across multiple basins, allowing for global comparisons of storms intensity.

Table 1: Description of the intensity stages defined in the TD9636 dataset and number of observations before featuring engineering.

Level	Name	Description	Number of observations
0	Tropical disturbance	Early stage of storm formation	3,156
1	Depression	Wind < 34 knots	16,028
2	Storm	Wind between 34 – 63 knots	17,204
3	Cyclonic storm	Wind ~ 64 knots *(except in North Indian where it is wind 43 – 47 knots)*	742
4	Hurricane	Wind > 64 knots *(except in North Indian with wind > 48 knots)*	10,623
5	Extratropical	Transitioned system outside the tropics	331
6	Dissipating	Weakening storm	259
		Total:	48,343

Exploratory Data Analysis and features selection

The data set enables tracking the evolution of the storms' intensity throughout its occurrence (Fig. 1). The intensity stage typically increases along the storm's trajectory, reaching stage 4, which corresponds to a hurricane, the most violent intensity stage. Afterward, the intensity gradually decreases, often returning to stage 1 or 0. In some cases, particularly when a storm approaches land, it may transition to stage 5 or 6, indicating an extratropical or dissipating system. We therefore hypothesize that the storm's location (i.e. its geographic position and distance to land) could be key factors in predicting storm intensity.

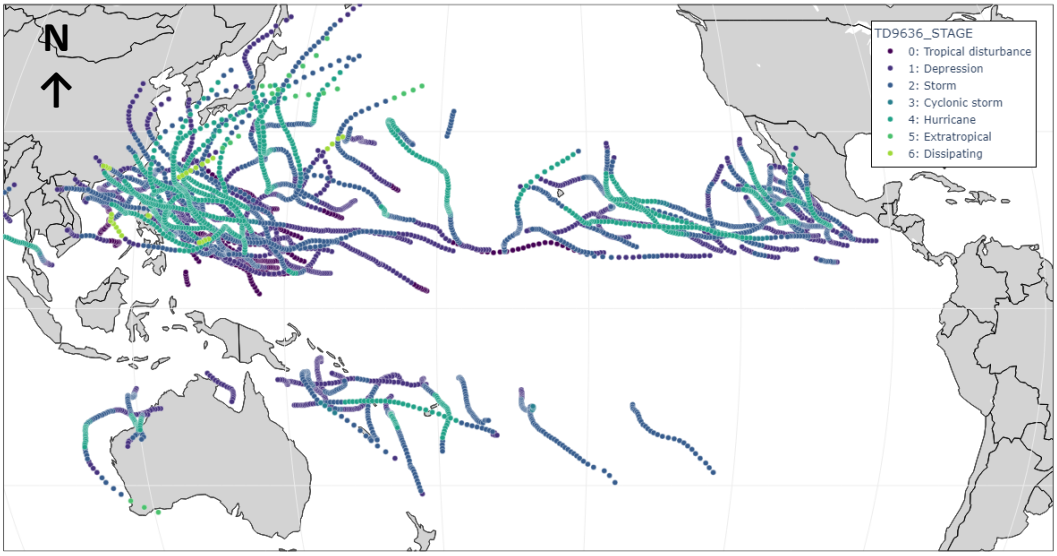
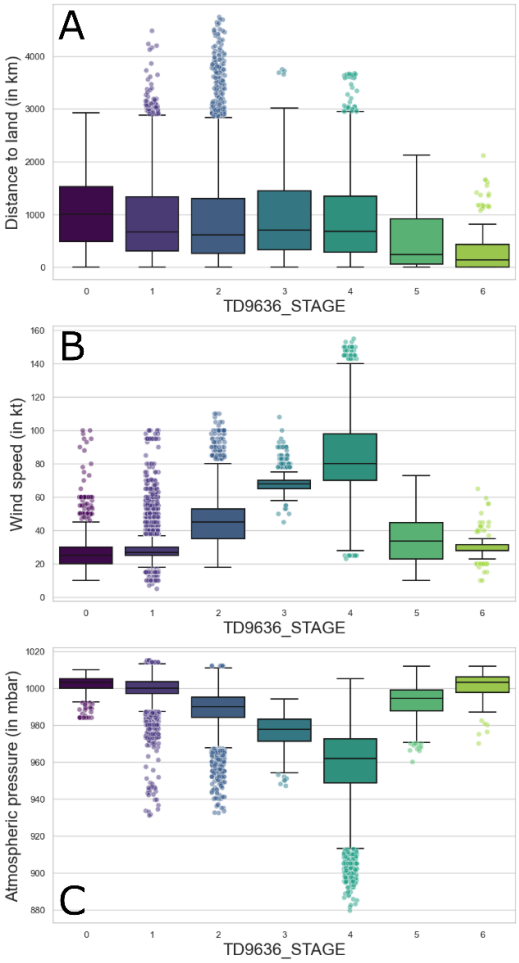


Figure 1: Evolution of the stage of intensity along the trajectory of a sample of storms extracted from the TD9636 data set.



The boxplot did not reveal a clear correlation between the distance to land and storm intensity stages, except for the highest stages (5 and 6), where the mean distance is noticeably shorter (Fig 2.A). This suggests that these stages occur as the storm approaches land, which aligns with the trend observed when mapping the TD9636_STAGE along the storm's trajectory (Fig. 1). As the storm intensity increases, wind speed tends to increase (Fig 2.B), peaking at stage 4 with a mean speed of approximately 80 knots. This is consistent with the TD9636_STAGE classification, where stage 4 (hurricane) represents the most violent storm category. On the other hand, atmospheric pressure exhibits an almost perfectly opposite relationship with wind speed (Fig 2.C), which is expected, given their strong physical connection. Indeed, wind is generated by air movement in response to variations in atmospheric pressure. These observations are consistent with the TD9636_STAGE classification description, which suggests that storm intensity increases progressively until stage 4 followed by a decrease in stages 5 and 6. However, in most cases, the storm seems to revert to stages 0 to 3 rather than continuing into stages 5 and 6 (Fig. 1 map). There is also considerable variability within each stage, across all parameters. This is not surprising, especially for wind speed, given the differences in stage definition across oceanic regions.

Figure 2: Boxplots illustrating the variations in physical parameters based on the TD9636 storm's intensity classification. (A) distance to land (in km), (B) wind speed (in kt) and (C) atmospheric pressure (in mbar). Data sourced from the IBTrACS data set.

Feature engineering and selection:

The dataset consisted of two types of features: general features provided by the WMO and additional features reported by the other meteorological agencies. Although the dataset was extensive (99 columns), only a few parameters were consistently reported across all agencies. These commonly available parameters included wind speed, atmospheric pressure, latitude, and longitude. Parameters provided by only a few agencies, such as the radial extent or eye diameter, were excluded from this analysis due to insufficient data coverage.

Storm identification. The parameters used to uniquely identify each storm (SID, NAME, NUMBER) were removed from further analysis, as they do not contribute meaningful information about storm structure for predictive modeling. Additionally, metadata-related parameters such as IFLAG and WMO_AGENCY were excluded.

Temporal and geographic information. The dataframe included two features related to temporal information: SEASON and ISO_TIME. However, despite its name, the SEASON column only contained the year in which the cyclone happened, rather than its actual seasonal classification. In contrast, ISO_TIME provided the exact date and time of the measurement occurrence. Correlation matrixes (not shown here) revealed no distinctive correlation between the cyclone's intensity stage and the specific day or month of the year. However, to retain a form of temporality in the analysis, the ISO_TIME values were transformed to accurately represent the four seasons of the year: Spring, Summer, Fall, and Winter. To fully represent the meteorological context, the seasons were defined considering the hemisphere in which the cyclone occurred, grouping months with similar weather patterns and climatic conditions according to the NOAA's Climate Prediction Center. For instance, December to February were considered winter months in the North Hemisphere, whereas in the South Hemisphere, they were regarded as summer.

Each agency reported its own latitude and longitude for observation positions. After comparing these values and confirming that variations were minimal for the same data points, only the latitude and longitude provided by the WMO were retained for consistency. To avoid multi-collinearity, the SUBBASIN column was removed, as it was redundant with the BASIN feature. Additionally, the two parameters representing the storm's proximity to land (DIST2LAND and LANDFALL) were totally correlated (Fig. 3) so only DIST2LAND, which represents the distance from the storm's current position to land was retained.

Physical parameters from agency reports. Each agency used varying units to monitor wind speed, and the WMO does not standardize these values before compiling the data. Since direct comparisons between different units are not possible, the raw WMO_WIND data were removed. Instead, wind speed values were gathered from various agency reports and converted to a common unit. Indeed, US agencies report wind speed based on a 1-min averaging period, whereas most other agencies use a 10-min averaging time. Since 1-min mean winds are roughly 12% higher than 10-min winds, all wind speed values were standardized to a 1-min averaging time. The USA_WIND column was used as the primary source since it contained the most data. Missing data were supplemented with data from DS824_WIND and NEUMANN_WIND, which also used a 1-min averaging period. For agencies that reported 10-min averaged wind speeds (e.g., TOKYO_WIND, HKO_WIND, KMA_WIND, REUNION_WIND, BOM_WIND, NADI_WIND, WELLINGTON_WIND), an adjustment factor of 1.12 was applied to estimate equivalent 1-min wind speeds. The WIND_SPEED feature represents the translation speed of the storm system (i.e. how fast the storm moves across a region) as assigned by the WMO based on latitude and longitude changes over time. Therefore, this differs from wind speed which measures the intensity of winds within the storm itself. This feature had high data coverage and showed no correlation with wind speed (Fig. 3). Therefore, it was retained for the analysis.

The wind direction (STORM_DIR), expressed in degrees, was computed by the WMO based on latitude and longitude, representing the translation direction between consecutive points. Thanks to its extensive data coverage, this feature was retained for further analysis.

Atmospheric pressure (WMO_PRES) was reported using a consistent unit across all agencies, allowing all values to be combined into a single column. When multiple agencies reported pressure values for the same data point, an average was used. However, correlation analysis (Fig. 3) revealed that wind speed and atmospheric pressure

were highly correlated. Since wind speed had a larger coverage, it was retained as a key feature, while atmospheric pressure was removed.

Additional information. The NATURE parameter, which classifies the storm type based on multiple agency reports, was retained. All rows containing at least one missing data were removed.

After feature selection and data cleaning, the dataset was constituted of 9 features (Table 2) over 45,025 observations (see Table 3). However, few features appeared to be correlated with the target variable, which could indicate that they could have low involvement in storm intensity prediction (Fig. 3).

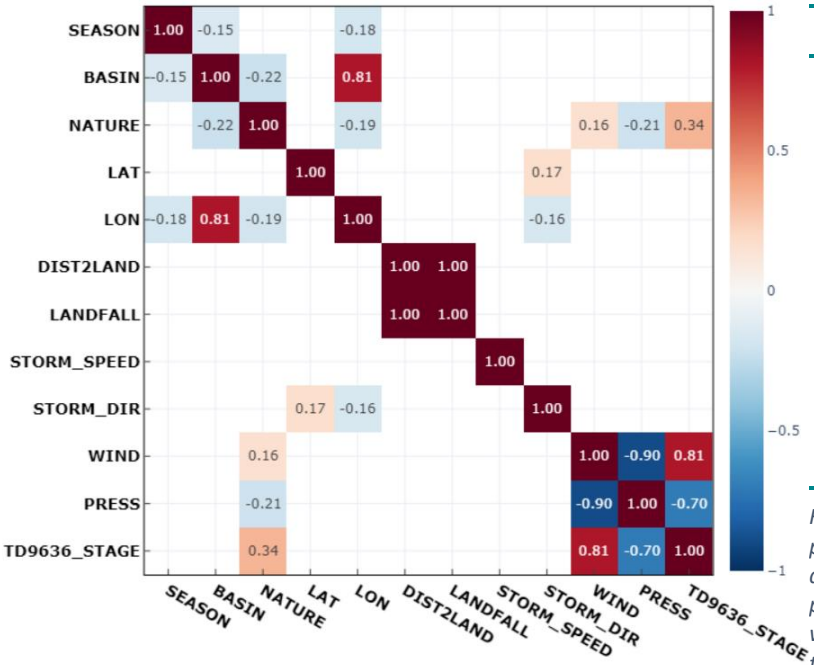


Figure 3: Heatmap of the correlation matrix displaying the pairwise correlation coefficients between features in the dataset from IBTrACS used to predict the TD9636_STAGE parameters. Only correlations with absolute value up to 0.15 were displayed. Positive correlations are shown in warmer tones, while negative correlations are shown in cooler tones.

Table 2: List of the features used in the analysis

Feature	Description
SEASON	Season where the storm occurred
BASIN	Oceanic basin
NATURE	Storm classification
LAT	Latitude
LON	Longitude
WIND	Standardized 1-min averaged wind speed in kt
STORM_SPEED	Storm translation speed in kt
STORM_DIR	Storm direction in degree
DIST2LAND	Distance to land in km

Encoding of categorical features.

Machine learning models cannot directly process non-numerical data, so categorical features must be transformed into numerical values. Various encoding methods and combinations were tested across different models, but the final model used the categorical features (SEASON, BASIN, and NATURE) encoding as follows. The NATURE and BASIN features were encoded with the `get_dummies()` method from the Pandas library. This function applies one-hot encoding, which converts categorical data into separate binary dummy columns. Each unique category becomes a new column, where a value of 1 indicates the presence of that category in a given row, and 0 indicates its absence. This approach ensures that categorical values are encoded numerically without introducing bias or ordinal relationships between them. In addition, this method follows the k-1 rules, which state that a categorical feature with k unique categories can be represented using k-1 binary variables without losing any information, allowing to reduce computation time.

Table 3: Details of the sampling effort for each category of the TD9636_STAGE parameters across the study steps.

Level	Number of observations after feature engineering	Training dataset	Training dataset after SMOTE
0	853	695	13,658
1	15,409	12,325	13,658
2	17,079	13,658	13,658
3	742	579	13,658
4	10,590	8,496	13,658
5	225	176	13,658
6	127	91	13,658
Total	45,025	36,020	95,606

On the other hand, SEASON was encoded using `OrdinalEncoder()` from Scikit-learn, which assigns a unique integer to each category in a feature, creating an ordinal relationship between the categories. Depending on the way this method is used, it can give more weight to some categories. Since storm frequency and intensity vary by season, an ordinal ranking was applied (Winter < Spring < Fall < Summer), assigning higher values to seasons associated with stronger and more frequent storms. By encoding it this way, the model can capture seasonal trends more effectively.

Data splitting

The dataset was split into two stages. First, a subset of samples was isolated to serve as new, unseen data for the web application as it is imperative that the model was not trained on these samples. Next, the remaining data were divided into an 80-20 split, with 80% used for training and 20% reserved for testing. The test set provides an unbiased evaluation of the model's performance on unseen data, offering a realistic estimate of its generalization ability. By keeping a separate test set, we can also detect overfitting (when the model performs well on training data but poorly on new data).

Handling unbalanced target variable (SMOTE). The first steps of data exploration immediately revealed how highly unbalanced the target variable was (Table 3). Although this is not surprising in the context, as certain storm intensity stages are less frequent, it can have a significant impact on the model performance. Models were tested on both the unbalanced dataset and an artificially balanced version using the SMOTE (Synthetic Minority Over-sampling Technique) procedure from the imblearn.over_sampling module. The rebalancing of the data set was applied before scaling the numerical variables and only on the training subdataset. Indeed, the test dataset as being used to validate the models, needs to stay the most empirical as possible. SMOTE procedure generates synthetic observations for the minority classes by simulating new samples through interpolation between existing minority class samples and their nearest neighbors, leading to a fully balanced dataset (Table 3).

Model selection, comparison and evaluation

This project focuses on categorizing the intensity of storms, which implies selecting a multi-class classification model. Multiple models were tested on the dataset[3], including K-Nearest Neighbors, Random Forest, Gradient Boosting, XGBoost, and Histogram-Based Gradient Boosting. To optimize model testing, the GridSearchCV method from Scikit-learn was used for hyperparameters tuning. This tool performs exhaustive research over a predefined grid of hyperparameters. It performs an exhaustive search over a predefined grid of hyperparameter values for a given machine learning model. By using cross-validation, it evaluates all possible combinations of hyperparameters to identify, for a given model, the optimal values that lead to the best performance metrics (accuracy or precision). For instance, to find the optimal model, GridSearchCV was applied over the Histogram-based Gradient Boosting model to test each hyperparameter in a specific range. Additionally, GridSearchCV integrates cross-validation techniques that allow to evaluate the model performance by re-partitioning the dataset into multiple subsets. It involves training the model on some of these subsets and testing it on the remaining data and rotating the subsets, ensuring the data point is used for both training and testing during the hyperparameter tuning process.

Results

After testing several models with different encoding techniques and hyperparameters, the Histogram-based Gradient Boosting (HGB) appeared to be the best solution to classify storms intensity stages from the TD9636 dataset, and show an accuracy of 93.2 % (Table 5). HGB is based on the Gradient Boosting (GB) method which is an ensemble learning method that sequentially builds series of decision trees, each integrating and improving the performance of the previous one. The HBG methods is an optimized version of the GC that accelerate training by bucketing features into discrete bins (like in a histogram), instead of considering each individual observation. Therefore, this approach is significantly more efficient than the traditional GB in reducing the computation time and memory consumption. It also decreases the sensitivity to noise and potential outliers.

[3] All the model test trials are available in the *model* folder on the GitHub repository.

Table 4: Some of the best models tested on both unbalanced and balanced (using SMOTE datasets) using the best hyperparameters determined by GridSearchCV. The model with the best parameters and selected for the final implementation is displayed in bold.

Model	Hyperparameters	Balanced?	Scores
Random Forest	*n_estimators=100* *max_depth=None* *min_samples_split=5* *random_state=42*	No	Precision: 92.4 % Recall: 92.5 % F1-score: 92.3 % Accuracy: 92.5 %
Random Forest	*n_estimators=100* *max_depth=None* *min_samples_split=5* *random_state=42*	Yes	Precision: 92.6 % Recall: 92.6 % F1-score: 92.6 % Accuracy: 92.6 %
Gradient Boosting	*n_estimators=100* *learning_rate=0.1* *max_depth=3* *random_state=42*	Yes	F1-score: 92 % Accuracy: 92%
XGBoost	*n_estimators=100* *learning_rate=0.1* *max_depth=6* *eval_metric='mlogloss'*	Yes	F1-score: 92 % Accuracy: 92%
Histogram Gradient Boosting	*L2_regularization=1* *max_bins=128* *max_iter=500* *random_state=42*	No	Precision: 92.6 % Recall: 92.7 % F1-score: 92.6 % Accuracy: 92.7 %
Histogram Gradient Boosting	***L2_regularization=0.1*** ***max_bins=128*** ***learning_rate=0.2*** ***max_iter=500*** ***random_state=42***	**Yes**	**Precision: 93.2 %** **Recall: 93.3 %** **F1-score: 93.2 %** **Accuracy: 93.2 %**
K-Nearest Neighbors	*N_neighbors=2* *Weights='uniform'* *P=2* *knn_algorithm='auto'*	No	Precision: 90 % Recall: 90 % F1-score: 90. % Accuracy: 90.1 %
K-Nearest Neighbors	*N_neighbors=2* *Weights='uniform'* *P=2* *knn_algorithm='auto'*	Yes	Precision: 90.3 % Recall: 90.1 % F1-score: 90.0 % Accuracy: 90.1 %

The retained HGB model was parametrized using GridSearchCV which enables to find the best hyperparameters for each case. The optimal hyperparameters were as described above and the other were let as default.

- *l2_regularization=0.1*, which means that the model applied a very small correction to the leaves of the trees, allowing the model to fit closer to the data;
- *max_bins=128*, which is the number of bins created from the dataset. It is reduced from the default value (255), therefore reducing computation time.
- *learning_rate=0.2*, which is a parameter to scale down the contribution of each new tree during the iterative computation. This controls the number of trees that will be needed until finding the best models. 0.2 is an intermediate learning rate, which is a compromise to have a good generalizable model with reasonable computation time.
- *max_iter=500*, which refers to the maximum number of iterations that will be made. It is higher than the default value (100 iterations) but the combination of a low learning rate with a high maximum of iteration enables a generalizable model.
- *random_state=42*, which introduces randomness in certain parts of the computation to improve the reproducibility of the model each time it is run. If not random_state (as default) the model will produce slight variations in results.

The learning curve of the HGB model (Fig. 4) was computed to assess its generalization ability to unseen data by analyzing how accuracy evolves when increasing the training sample size. The training accuracy increased

quickly before reaching a plateau, while the cross-validation curve rapidly increased to reach 0.7 of accuracy with a training set size of less than 10,000 and then gradually increased until reaching its highest value of 0.93 at around 75,000 observations. Both curves eventually stabilized, indicating that the model has reached its learning capacity given the available data. This suggests that further increasing the sampling size or extending computation time would not significantly improve its performance. In addition, the slow increase in validation accuracy suggests high variance, which may be a sign of overfitting. This means the model is capturing not only the underlying pattern in the data but also noise and minor variations, reducing its ability to generalize to new data. Overfitting often results from excessive model complexity, preventing it from recognizing the simpler real patterns existing in the data. However, while some slight signs of overfitting may be present, the training and the validating curves converged, with the validation accuracy remaining slightly lower than the training accuracy. This suggests the model is still fitting the data quite well.

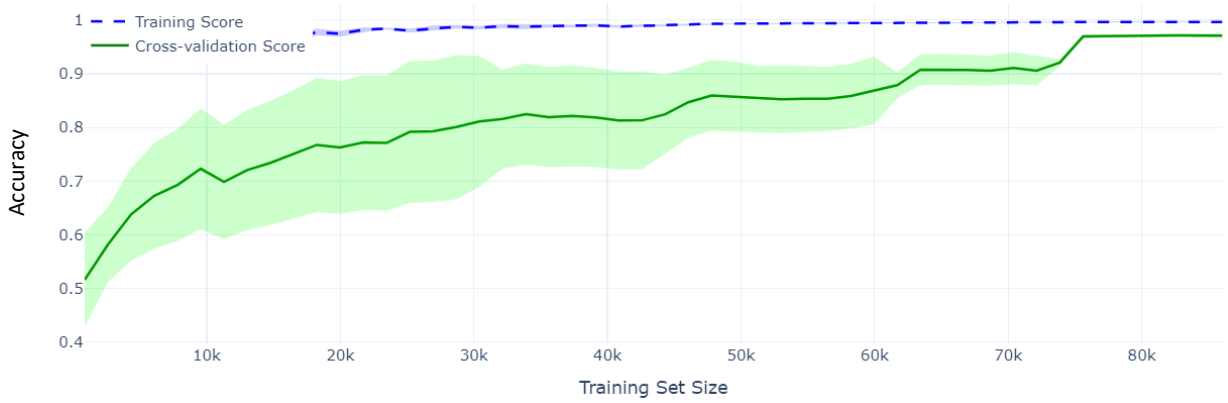


Figure 4: Learning curve of the Histogram-Based Gradient Boosting model for storm intensity classification, illustrating model performance as the training size increases. Blue: training curve and green: validation curve.

A confusion matrix was also used to evaluate the model (Fig. 5A and B), as it provides insights into where the model misclassified one stage as another and allows a comparison between predicted and actual values. Although the model performed globally well, most misclassifications occurred between stages 1 and 6 (with a confusion rate of 0.33 and 0.25 using SMOTE) and between stages 3 and 4 (with a confusion rate of 0.31 reducing to 0.26 using SMOTE). The confusion between these stages could be explained by the inner definition of the stages in the TD9639 dataset, which is mostly based on wind speed as stage 1 and stage 6 observations have similar wind speed (Fig. 2B). Then, the wind speed threshold depends on the oceanic basis, especially for the stage 3 and 4 (see Table 1) that could potentially be responsible for the large variability observed in Fig. 2B and which could explain the difficulty for the model to distinguish between these two categories.

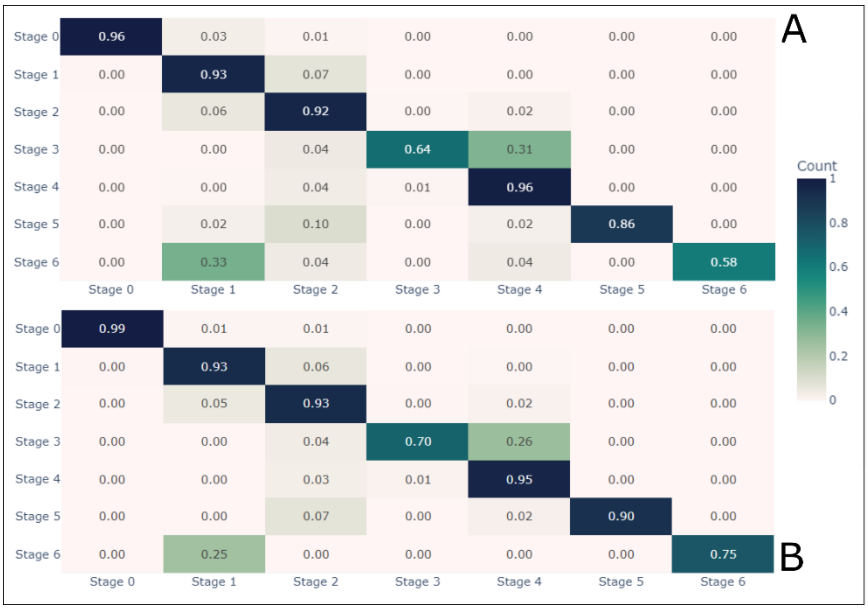


Figure 5: Confusion matrixes of the Histogram-Based Gradient Boosting model used to classify storm's intensity, comparing results with unbalanced sampling (A) and with balanced sampling using SMOTE procedure (B). The diagonal represents true positives (correct classifications), while values above the diagonal indicate false negatives (misclassified storms), and values below diagonal represent false positives (i.e. where the model assign an observation to a stage when it is not from this stage).

Finally, the confusion matrix revealed the low performance improvement with the addition of the SMOTE procedure (Fig. 5B), while it

significantly increases the computation time. Even if it is improved by HGB, one of the major problems of GB is that it is very slow to train, especially with large datasets. This is due to the creation of trees that is made sequentially, unlike other ensemble models such as random forest where it can be trained in parallel. Regarding the slow. Therefore, regarding the low performance improvement, rebalancing the data using SMOTE may not be very necessary in this particular case.

Deployment

Pipeline

As the algorithm will be used to classify new data in a web application, it was integrated into a multi-step pipeline[4] including both the preprocessing of the features and the machine learning model. This ensures that the same preprocessing steps are consistently applied to new data before it is passed to the model. The trained model, along with the preprocessing steps and their corresponding weights are stored in a .pkl file, which can be directly loaded into the application for seamless deployment.

Dataset splitting and preprocessing the features. The pipeline incorporated the same steps as described earlier, including dataset splitting (with an 80/20 ratio between training and testing sets) and all necessary preprocessing steps, especially encoding scaling (Fig. 6). Categorical features were encoded using `OrdinalEncoder()`, for SEASON, and `OneHotEncoder()`(which has the same behavior as `pd.get.dummies()`), for NATURE and BASIN, while numerical data (WIND, DIST2LAND) were standardized using `StandardScaler()`. Since STORM_DIR was expressed in degree, the custom class `StormDirTransformer()` needed to be created to pass in the pipeline. After preprocessing, SMOTE was applied to balance the dataset before training the model.

```
preprocessor = ColumnTransformer(
    transformers=[
        ("ord", OrdinalEncoder(categories=[['Winter', 'Spring', 'Fall', 'Summer']]), ["SEASON"]),
        ("storm_dir_transform", StormDirTransformer(), ["STORM_DIR"]),
        ("num", StandardScaler(), ["WIND", "DIST2LAND", "STORM_SPEED"]),
        ("cat", OneHotEncoder(drop="first", handle_unknown="ignore"), ["BASIN", "NATURE"])
    ],
    remainder="passthrough"
)
```

Figure 6: Encoding procedure of the non-numeric features, in the machine learning pipeline.

Incorporating the machine Learning model. The Scikit-learn `ImbPipeline` was employed to link the different steps of the workflow. The first step handled data preprocessing, and the second implemented the model (Fig. 7).

A variable was created (`classifier`) for the model implementation, enabling the use of any classification model working with numerical features.

```
pipe = ImbPipeline(steps=[
    ("preprocessor", preprocessor),
    ("smote", SMOTE(sampling_strategy="auto", random_state=42)),
    ("classifier", classifier)
])
```

Figure 7: Pipeline to implement to model in the web application.

The pipeline was configured with the retained model with optimal hyperparameters (i.e. `HistGradientBoostingClassifier(L2_regularization=0.1, max_bins=128, learning_rate=0.2, max_iter=500, random_state=42)`. No GridSearchCV was implemented since the optimal hyperparameters were previously established. The model's weights (and the preprocessing step) were then stored in a .pkl file.

Web application

The machine learning model was deployed in a single-page web application using the Django framework to manage the backend and an HTML file for the frontend (directly inside *index.html*). This application displayed both historical and newly recorded cyclones on a map, following their progression over time (see Annexes I & II). A single-table database was integrated to store new cyclone entries, which can be submitted through a form managed by a dedicated API POST request. Upon receiving new data, the model generates predictions, which are then returned to the frontend for real-time display. During execution, a specific warning could appear in the logs

[4] Available in the notebook *ml_app_model.ipynb* in the GitHub *web_app/ml_model* folder

(Fig. 8). This warning arose due to the use of the k-1 rules during one-hot encoding for NATURE and BASIN features. In indicates that during the transformation (prediction phase), the encoder encountered a category in the input data that was not seen during training. One-hot encoding converts categorical variables into a binary matrix, where each unique category is represented as a separate column, with 1 indicating the presence of a category and 0 for all others. During preprocessing the first column was dropped, following the k-1 rules. Consequently, the unseen category in fact refers to the dropped column, and all encoded columns are set to zero, thus correctly corresponding to the value it represents. Furthermore, predefined category options were enforced for categorical features to ensure prevent users cannot input values outside the predefined categories.

```
/home/audrey/.pyenv/versions/3.13.0/envs/tropical_cyclones/lib/python3.13/site-packages/
sklearn/preprocessing/_encoders.py:246: UserWarning: Found unknown categories in columns
 [0] during transform. These unknown categories will be encoded as all zeros
  warnings.warn(
```

Figure 8: Warning that could be displayed during the execution

Conclusion

The objective of this project was to use a machine learning approach to classify cyclone intensity, focusing on the intensity stages defined in the TD9636 framework. After testing several models with various encoding methods and hyperparameters combinations, the Histogram-Based Gradient Boosting classifier emerged as the most suitable model for our multi-class classification problem. We addressed data challenges, such as imbalanced class distributions, by applying SMOTE which generates simulated samples for underrepresented storm stages. However, rebalancing finally, did not significantly improved performance while considerably increasing the computing time and therefore did not seem to be the most suitable solution in the present case. Although some signs of overfitting were observed, the overall performance metrics suggested that the final model generalizes reasonably well to new data. Further improvement could be achieved by reassessing the significance of certain features, particularly by removing those with low weight in the correlation matrix. This would reduce model complexity and computation time, leading to a more efficient model.

Machine learning has proven effective in globally predicting the tropical cyclone intensity, as in most cases the change in intensity typically occurs slowly and steadily (Chen et al., 2020). However, rapid and anomalous intensity changes remain challenging to predict and classify due to the inherent complexity of the underlying physical processes. Factors such as air-ocean interactions or large-scale atmospheric dynamics contribute to this complexity, making some meteorological phenomena difficult for humans to fully understand. Nevertheless, pure data-driven approaches have already shown significant contributions to improving tropical cyclone prediction (Chen et al., 2020). In this context, machine learning holds great promise for enhancing our understanding and forecasting of tropical cyclones, with future challenges focusing on predicting rapid-intensity changes.

References

GitHub repository

https://github.com/dsti-labs/tropical-cyclones

Bibliography

Chen, R., Zhang, W., Wang, X., 2020. **Machine Learning in Tropical Cyclone Forecast Modeling: A Review**. Atmosphere (Basel). 11, 676. https://doi.org/10.3390/atmos11070676

Knapp, K.R., Kruk, M.C., Levinson, D.H., Diamond, H.J., Neumann, C.J., 2010. **The International Best Track Archive for Climate Stewardship (IBTrACS).** Bull. Am. Meteorol. Soc. 91, 363–376. https://doi.org/10.1175/2009BAMS2755.1

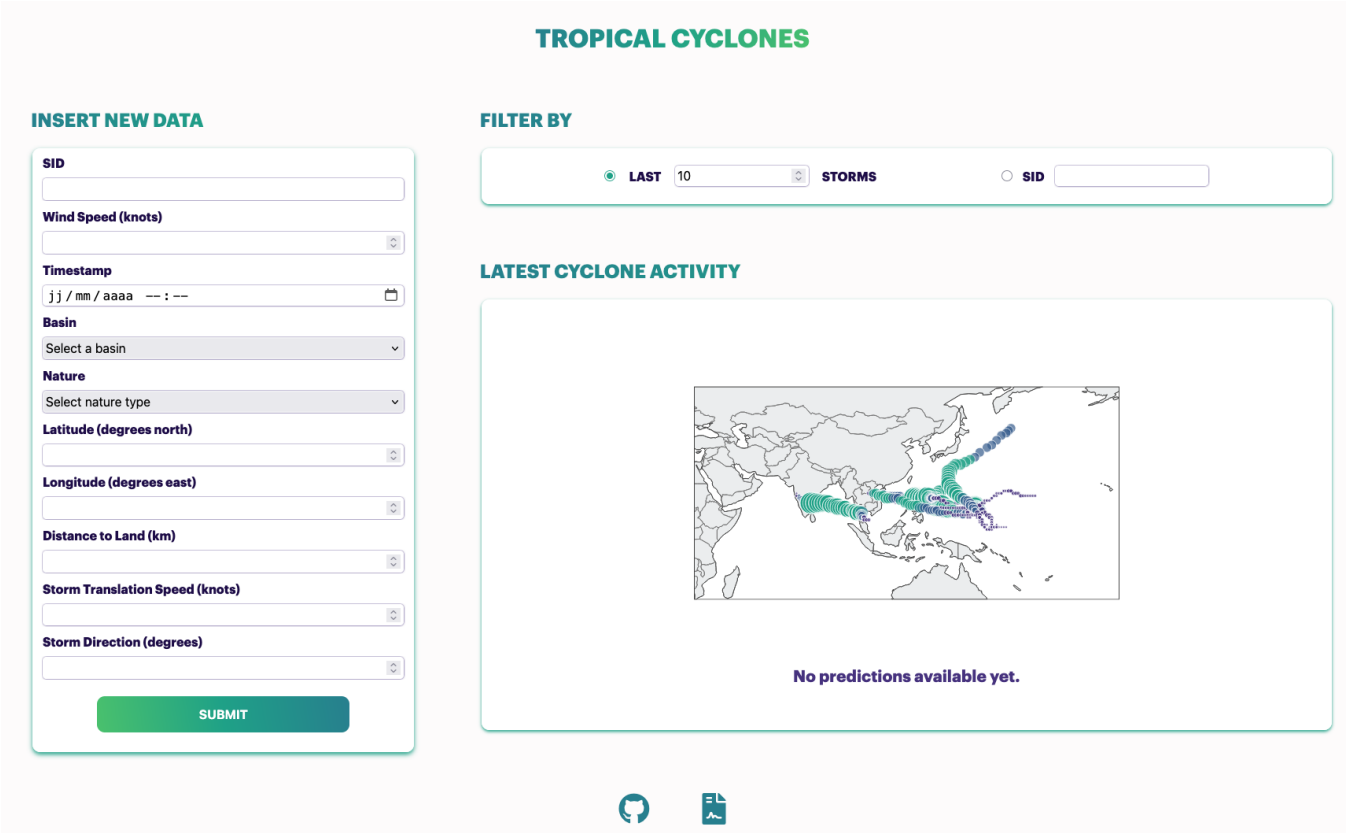
Sobel, A.H., Camargo, S.J., Hall, T.M., Lee, C.-Y., Tippett, M.K., Wing, A.A., 2016**. Human influence on tropical cyclone intensity**. Science (80-.). 353, 242–246. https://doi.org/10.1126/science.aaf6574

Wang, Z., Zhao, J., Huang, H., Wang, X., 2022**. A Review on the Application of Machine Learning Methods in Tropical Cyclone Forecasting.** Front. Earth Sci. 10, 1–17. https://doi.org/10.3389/feart.2022.902596

Wu, L., Zhao, H., Wang, C., Cao, J., Liang, J., 2022. **Understanding of the Effect of Climate Change on Tropical Cyclone Intensity: A Review**. Adv. Atmos. Sci. 39, 205–221. https://doi.org/10.1007/s00376-021-1026-x

Annexes

Annex I: Screenshot of the web application before predicting a storm intensity stage.



Annex II: Screenshot of the web application after predicting a storm intensity stage.