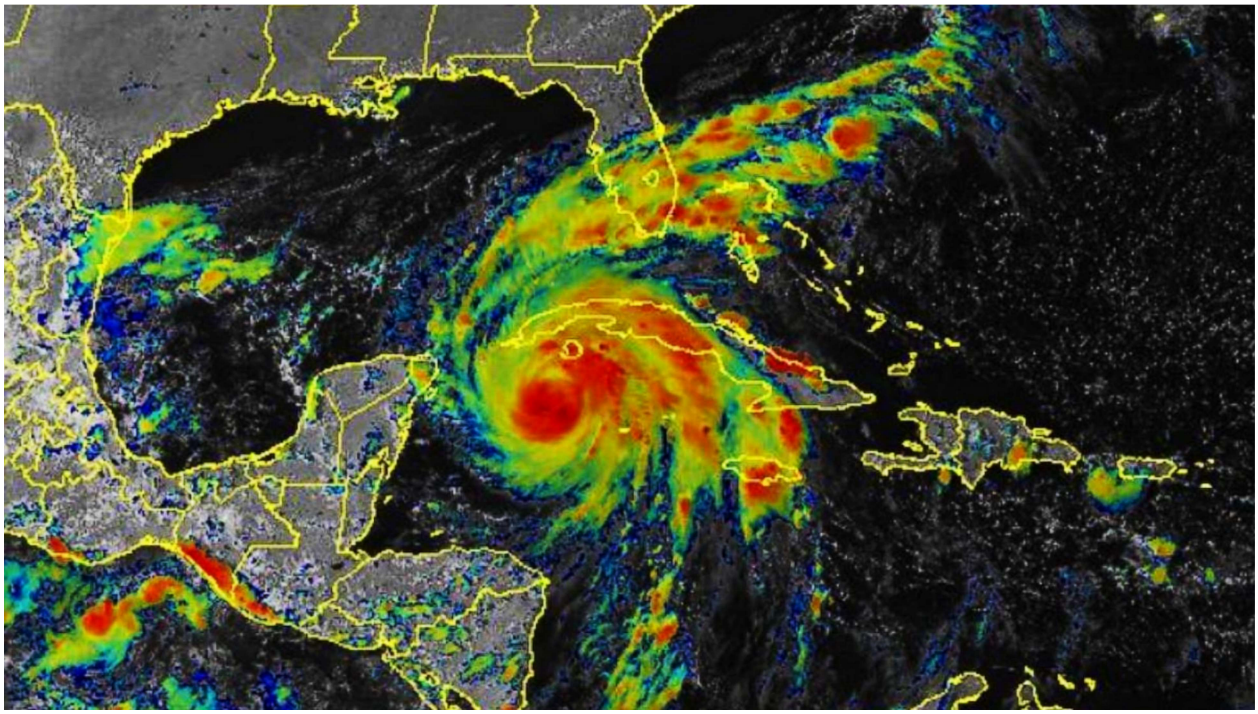

REPORT

Storm Intensity Change Forecast

Ishan Singh Bhullar
27 September 2024

Background on Subject Matter

Over the last 50 years, Tropical cyclones, or hurricanes or typhoons, have led to the deaths of over **780,000** people and caused an estimated **US$ 1.4 Trillion** in economic damages. This averages out to 43 deaths/day and about US$78 million in damages per day. This makes the need for better early warning systems obvious.

Problem Statement

Accurately predict whether there will be an increase in intensity (increase in surface winds) of a given Tropical storm over a 6 hour period. The eventual goal is to develop a model that allows us to more accurately predict a storm's intensity 24 hours or more in advance, so that people can evacuate in a timely manner.

Data Collection

I gathered two types of data from two separate sources.

1. Storm track data for the North Atlantic from IBTrACS - International Best Track Archive for Climate Stewardship maintained by National Centers for Environmental Information (USA).
    a. This contains time indexed track information about storms in the North Atlantic from the year 1851 onwards.
    b. Shape: 125,452 rows x 163 columns.
    c. Time field has a granularity of 3 hours.
    d. Also contains location and surface weather data for each reading.
2. ERA5 reanalysis dataset from ECMWF - European Centre for Medium-Range Weather Forecasts (EU).
    a. Hourly information about world weather at different pressure levels since 1959.
    b. I wrote an API script to extract the relevant fields for the relevant months and times from 1959 to 1999.
    c. Came in the form of hard to process .grib files for each year. Each file was about 5.5 GB.

Data Process

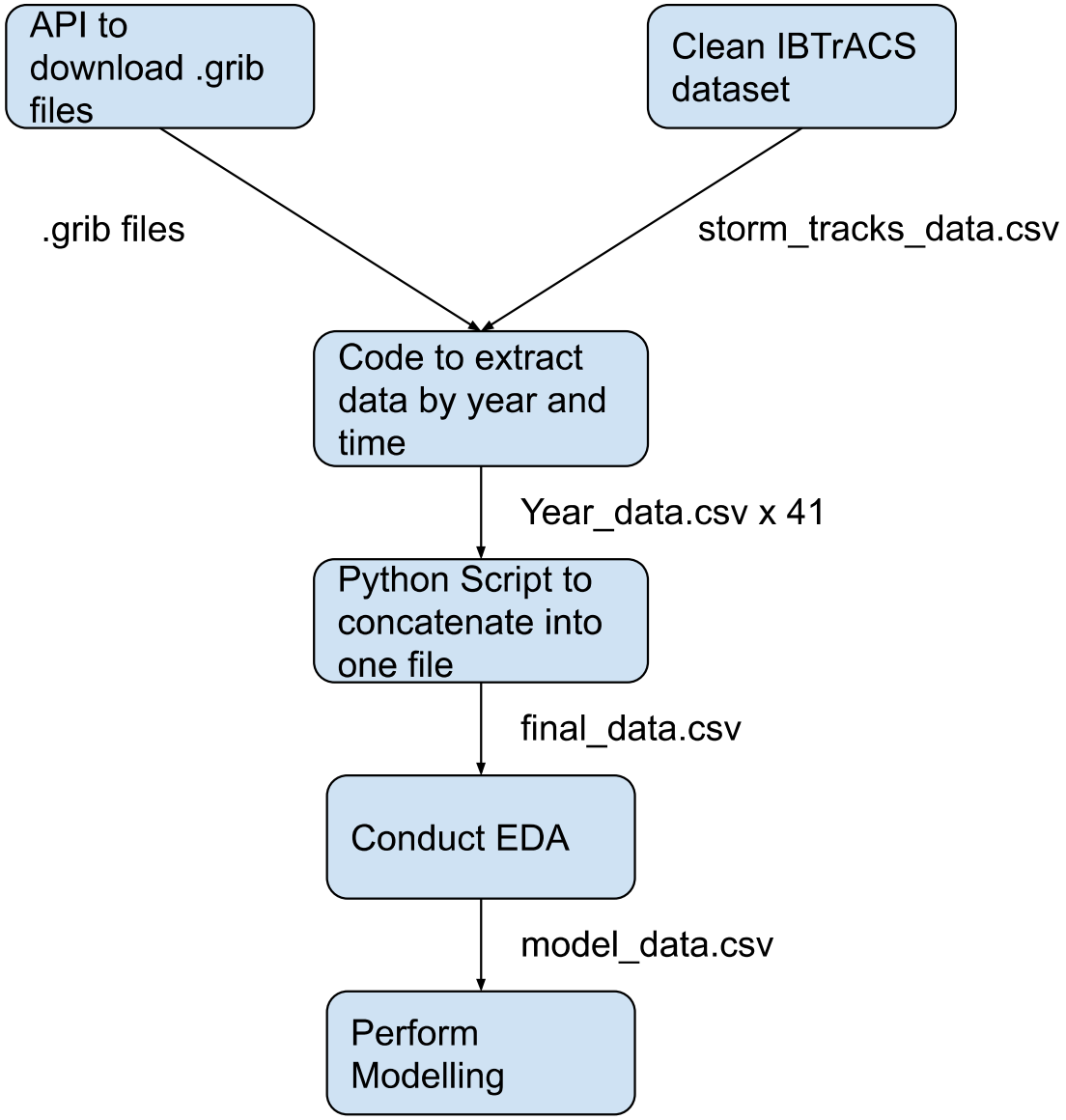

Fig 1: Data process flow chart.

Cleaning and Preprocessing

Most of the cleaning I had to do was for the IBTrACS storm track dataset. After taking care of null values, duplicates and dropping redundant columns I was left with a dataset with 50,593 rows and 14 columns. This was exported to the 'storm_tracks_data.csv' file.

Once I extracted data from the .grib files and concatenated all the 'year_data.csv' files together, I was left with 28,800 rows and 29 columns. At this stage I had to conduct some feature engineering to add the calculated target variable. For this I first calculated a wind lag field with surface wind values of 6 hours later. Using this, I created a binary column denoting whether surface winds increased or not over the 6 hour window. I also had to do some feature engineering and encoding to make categorical and vector variables usable.

Exploratory Data Analysis (EDA)

The main motivation behind performing EDA was to identify any trends in the data and to find correlations between features. I started with checking the distributions of each variable and found some interesting insights.

As shown in Fig 2 and Fig 3, the geopotential is a good indicator of storm severity. If it is low, there are higher chances of a storm being more severe.

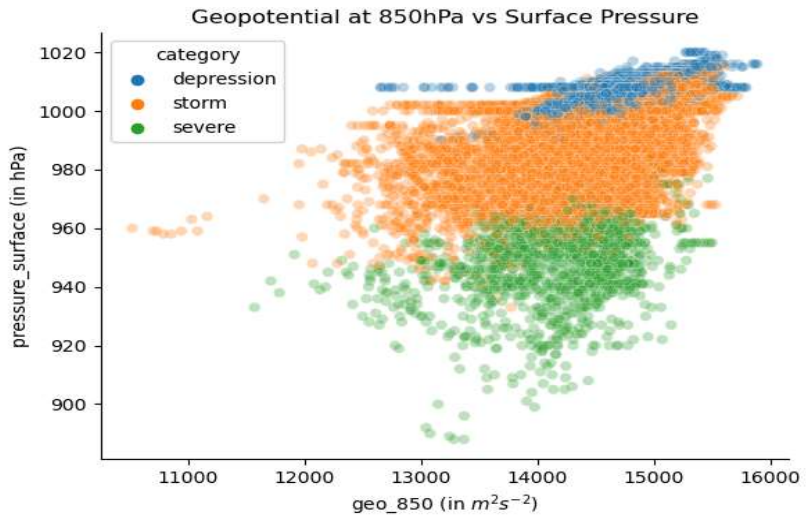

Fig 2: Geopotential at 850hPa vs Surface pressure.

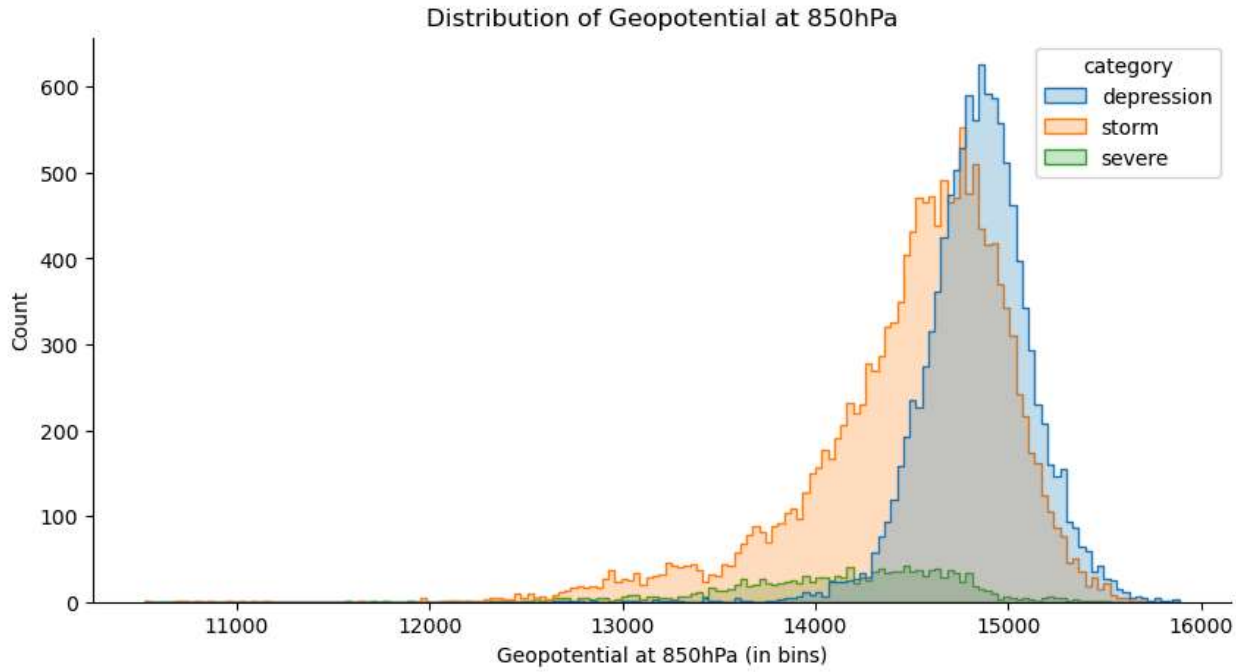

Fig 3: Distribution of Geopotential at 850hPa

Another observation was the near perfect correlation between surface winds and pressure as shown in Fig 4. This led me to exclude surface pressure from modeling efforts to avoid the mulit-collinearity problem.

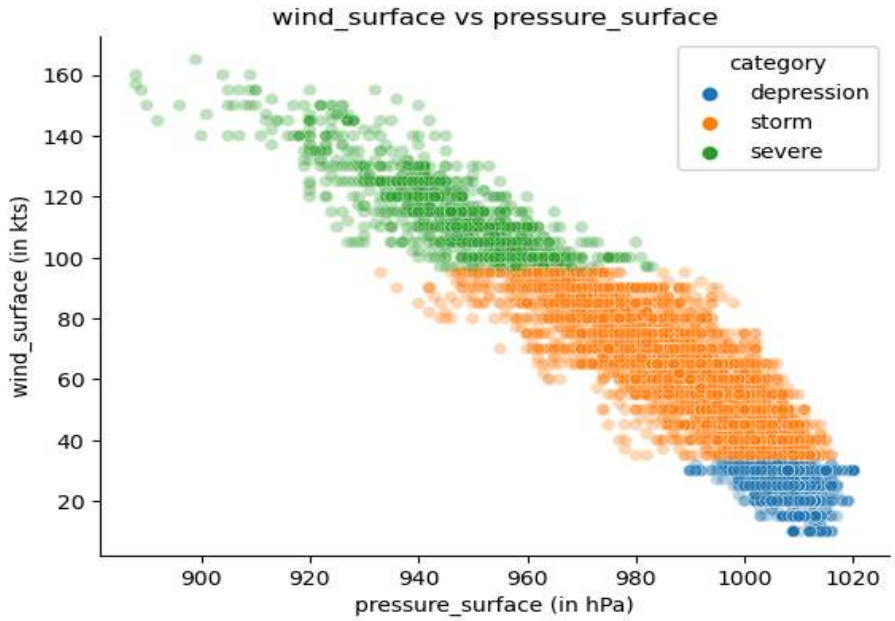

Fig 4: Surface Wind vs Surface Pressure

Modeling and Evaluation

I trained the following models and optimized the hyper-parameters using GridSearchCV and/or for loops where applicable:

- **Logistic Regression** - The non-linear nature of the data limited the predictive efficacy of this model. However, it served as a base-line to compare other models against. It helped pick out important features for further modeling via backwards selection.
- **Decision Trees** - These models did slightly better due to their ability to map non-linear boundaries. However, the results still weren't great.
- **Random Forests and Gradient Boosted Decision Trees (using XGBoost) -** The XGBoost models delivered much more satisfactory results, while the Sci-kit Learn Random Forest Classifier did not fare much better than individual decision tree models. Hyperparameter tuning for XGBoost was very time consuming and computationally expensive.
- **Support Vector Machine (SVM) -** As expected, given the binary nature of the target variable, SVM models performed quite well. Overfitting was an area of concern, requiring meticulous hyperparameter tuning. I expect SVM modeling to be less useful in future iterations when more data is added.
- **Neural Networks -** These models gave the best results by far and I can expect their efficacy to increase with future iterations as more data and features are added to the dataset.

When predicting calamitous events, it is important to not miss one in order to minimize casualties.. At the same time, false alarms can be very expensive economically. For evaluation I want to balance between minimizing False Positives and False Negatives. I will be using the following metrics to evaluate models:

- Accuracy Scores - How well the model predicts the target variable.
- Recall (True Positive Rate) - The ratio of correct positive predictions to total positive values.

$$R=TPTP+FN$$

- Precision - The ratio of correct positive predictions to total positive predictions.

$$R=\frac{TP}{TP+FP}$$

- F1 Score - The harmonic mean of Recall and Precision. Useful to try and balance Precision and Recall.

Summary

Best Models:

Model	Remainder Accuracy	Test Accuracy	Recall	Precision	F1 Score	Best Parameters
XGBoost 4	0.838	0.795	0.54	0.73	0.62	booster='gbtree', gamma=5.3, learning_rate=0.04, max_depth=19, min_child_weight=54, n_estimators=140, reg_alpha=1.6
SVM 2	0.823	0.777	0.51	0.69	0.59	C=1, gamma=0.1, kernel='rbf', scale='StandardScaler()
Neural 9	0.823	0.784	0.59	0.67	0.63	Layers=10x32swish(SiLU), w/BatchNormalization, w/L1regularizer=0.001, Output=sigmoid, epochs=500, Opt=Adam, loss=binarycrossentropy

Neural 9 model gave us the best performance. The ability to change the soft prediction threshold also gives us a lot of flexibility when it comes to tuning for Recall vs Precision. XGBoost is also very promising, however it requires precise hyperparameter tuning due to its propensity for producing overfitting models.

Next Steps

The next steps from here are:

1. Improve the hyper-parameter optimization for the XGBoost model.
2. Incorporate data from more years.
3. Add other features to the dataset.
4. Experiment further with Recurrent Neural Networks
5. Perform additional feature engineering.
6. Train models to predict storm tracks.