Project Milestone 3

Project Title - Predicting Rain with Machine Learning

Group 8 Members:

Dylan D'Andrea - (Team Leader)
Jainam Shah
Leon Silas
Mathew Olajide

1. Preliminary Project Statement :-

The agriculture industry is heavily reliant on weather conditions and providing forecasts and analysis of them is a surplus for farmers in order to make informed decisions on planning, cost-saving, environmental impact, and maximizing yield. However, weather prediction involves numerous factors and predicting weather is very much possible but complete accuracy is not guaranteed.The accuracy of prediction depends on various factors such as quality and quantity of historical data and complexity of weather patterns in that specific region. Moreover, localized analysis may be necessary to make more precise predictions for specific farms and fields. Overall, with proper data and tools, it is possible to forecast upcoming weather conditions which can help agricultural companies optimize their production, reduce costs, and make better decisions on farming activities such as planting and irrigation.

2. Overview of Milestone #2 :-

The milestone #2 report provides an overview of a weather forecasting project using machine learning. The dataset is split into training and testing sets, with anonymized weather data from regions A to E, including temperature, wind speed, precipitation, wind speed direction, and atmospheric pressure. The objective of the project is to predict rain for the next day based on three labels: N for no rain, L for light rain, and H for heavy rain. To accomplish this, the report utilizes Python, Jupyter-Notebook, and the Matplotlib library to perform exploratory data analysis (EDA) and create the model. The preliminary EDA includes visualizations of the imbalanced class, average temperature, maximum wind speed, maximum temperature, maximum wind direction, minimum temperature, precipitation, average wind speed, and minimum atmospheric pressure. The milestone #2 report concludes with a plan to check for missing values and build the model to predict weather conditions for rain.

3. About Dataset :-

The dataset as mentioned in the previous milestone report is scattered providing features to predict and measure weather scattered over different regions. The features include temperature,

wind speed, precipitation, wind speed direction, atmospheric pressure etc. as shown in Fig 1. The dataset has been split into parts i.e. training set and testing set. In each of the sets, there is weather data consisting of anonymized locations names from Region A to Region E.

	date	avg.temp	max.temp	min.temp	precipitation	avg.wind.speed	max.wind.speed	max.wind.speed.dir	max.inst.wind.speed
0	229b70a3	3.3	10.2	-2.4	0.0	2.9	9.3	W	14.3
1	3134f4ff	5.7	13.7	-2.9	0.0	3.6	10.7	W	15.8
2	dbfaf910	13.8	20.0	9.0	0.0	5.3	9.4	SW	15.2
3	3aea0cf0	11.4	19.3	5.8	0.0	4.2	10.1	SW	20.6
4	f0227f56	2.4	7.7	0.3	43.5	0.9	3.7	SW	5.7

Figure 1: Glimpse of the dataset

As it is seen from the dataset, the “date” column is anonymized to some random values. There are in total 10 features in the dataset. The dataset contains 5 csv files in each training and testing set along with a separate csv file named “solution_format.csv” containing target rain predictions for each of the dates, which allows us to use supervised learning when building the model.

```
labels_df.head()
```

	date	label
0	a8c6911b	N
1	eebdce12	N
2	6fb420a6	L
3	3bf8b132	N
4	e86629c2	N

Figure 2: Solution_format.csv

The goal is to predict the weather for the next day based on three labels:

- N - No rain
- L - Light Rain
- H - Heavy Rain

3.1 Exploratory Data Analysis :-

After initial exploratory data analysis in previous milestone, all the training datasets of different regions have been concatenated as they share a primary key as “date” column.It was noted from the dataset that there was an imbalanced class as label “N” in “labels_df” dataframe dominating rest of the classes. Making the model more biased towards larger samples. To check whether there exists any outliers or anomalies, data was visualized and from the plots it can be

observed that there are patterns in the data that are very similar except for regions C,D,E where minimum wind speed and average wind speed are on a lower scale.

Now that preliminary EDA is complete, there is further requirement of checking the dataset about missing values and to create the model to predict weather conditions for rain.

3.2 Missing Values :-

For finding missing values for a particular column, a custom helper function has been formulated which detects missing values in the dataframe as shown in Figure 3.

Missing Values

```
[ ]  missing_cols(train_all_lvls)
```

```
min.atmos.pressure => 2264 [80.0%]
```

Figure 3 Missing Values in train_all_lvls dataframe

A heatmap is plotted to check missing data as shown in Figure 4

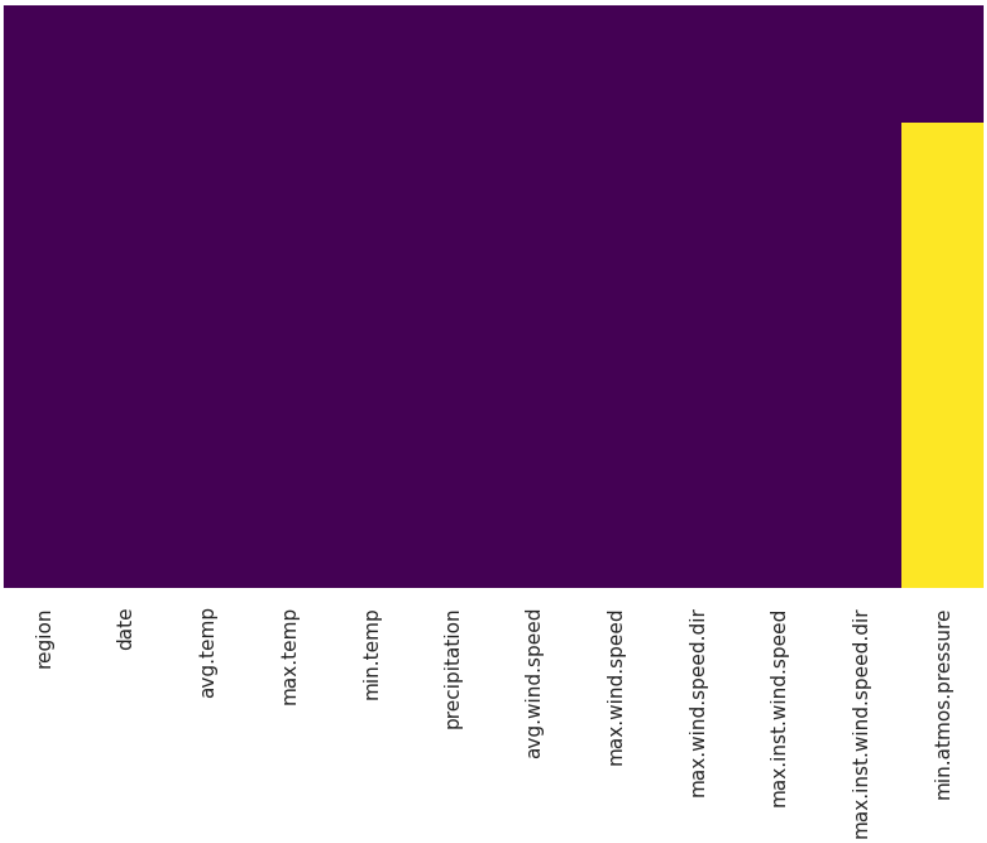


Figure 4 Heatmap of training data

Looking at the distribution of data in Figure 5 considering the “min_atmos_pressure” feature of the dataframe, missing values can be imputed using mean as distribution is normal.

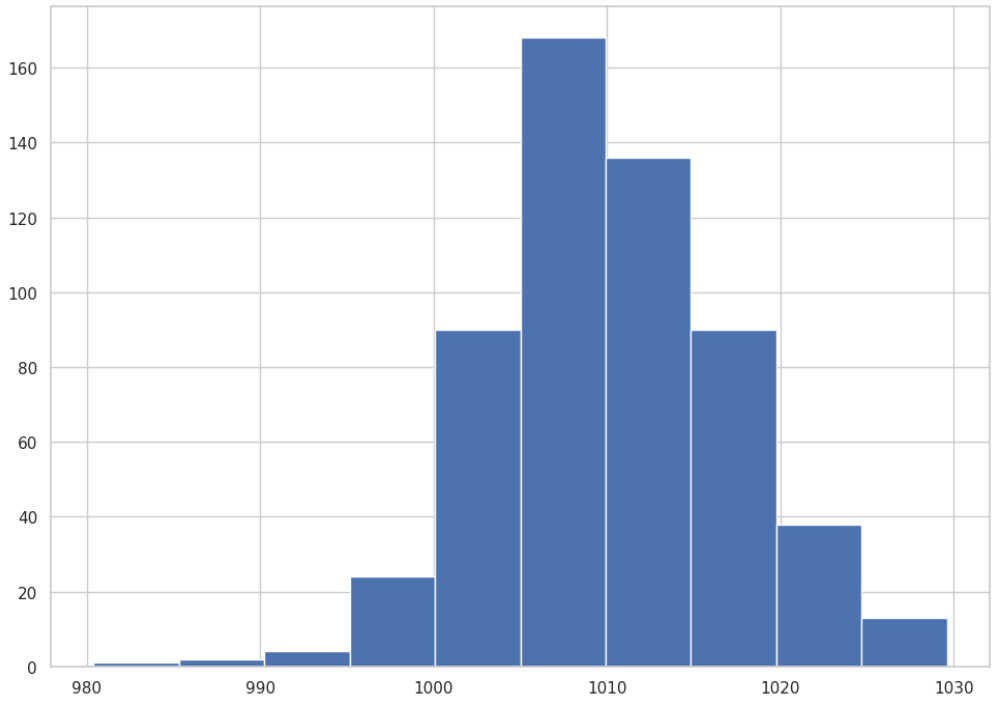


Figure 5 Distribution of min_atmos_pressure data

3.3 Feature Engineering

First, Labels were added to the data and categorical columns were converted into numeric columns using LabelEncoder().Target variable was also converted to numeric as shown in Figure 6.

```
[29] le = LabelEncoder()
     le.fit(train_all_lvls['label'])
     le_name_map = dict(zip(le.classes_, le.transform(le.classes_)))
     le_name_map

     {'H': 0, 'L': 1, 'N': 2}
```

Figure 6 Feature Engineering using LabelEncoder() function

Furthermore a Beaufort scale has been prepared which is an empirical measure that relates wind speed to observed conditions at sea or on land. The merged dataframes are further trained using the Beaufort scale.

3.4 Preparing Data

The data has been pivoted using pivot_table() function of pandas into a longer format, now each region having its own features. This way data will be in the right shape to build a model.

```
[44] train
```

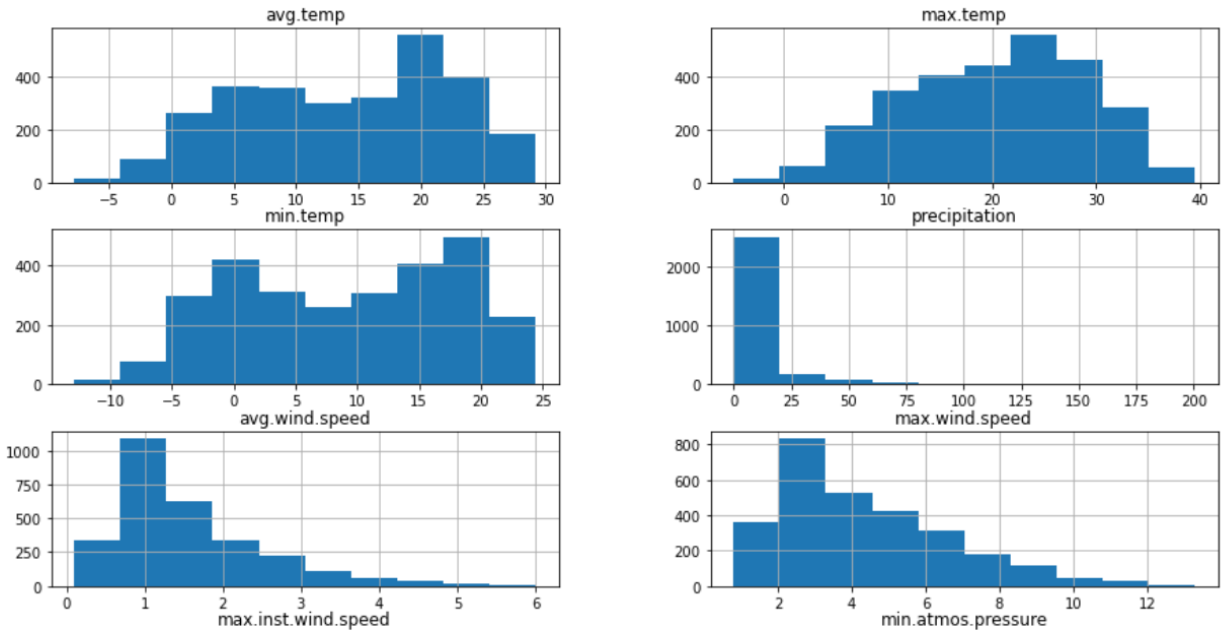
	date	label	avg.temp_A	avg.temp_B	avg.temp_C	avg.temp_D	avg.temp_E	avg.wind.speed_A	avg.wind.speed_B	avg.wind.speed_C	.
0	00173aec	2	18.7	17.6	16.9	19.5	14.3	1.6	1.8	0.9	
1	0083f291	1	13.1	12.6	12.0	13.0	10.7	1.4	1.0	0.7	
2	014cfe7b	2	19.9	19.0	17.5	19.9	16.2	3.7	3.6	0.7	
3	01947c8e	2	21.6	20.2	20.5	21.3	17.6	1.6	1.2	1.1	
4	0258884d	2	15.2	13.9	13.9	15.8	11.1	2.5	2.3	1.2	
...	...	...	...	...	...	...	...	...	...	...	
561	fe2a1385	1	2.9	1.6	1.4	3.4	0.4	1.2	0.9	0.7	
562	fe6dd99c	1	2.9	2.9	3.9	2.9	0.2	1.6	3.9	1.0	
563	ff88c3dd	1	9.8	8.9	9.0	10.2	6.3	1.9	1.5	1.4	
564	ff929090	2	10.4	8.1	7.1	11.0	5.7	4.3	4.6	0.9	
565	ffe3bd74	2	22.7	22.4	21.7	21.7	19.4	3.7	5.4	0.8	

566 rows × 57 columns

Figure 7 Training data

4. Visualizations :-

Let’s check the distribution of the continuous features given in the dataset.



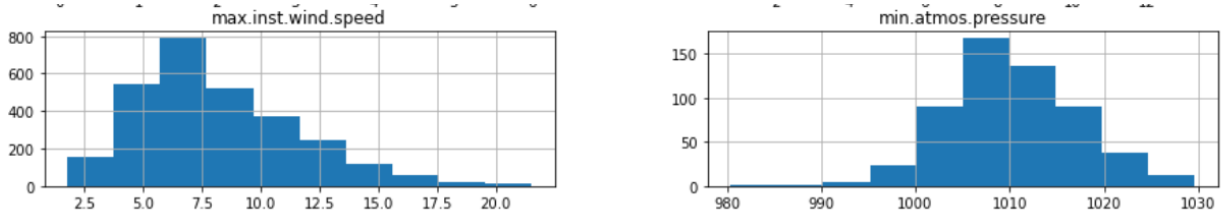


Figure 8 Distributions of continuous features

Let's draw boxplots to detect the outliers present in the data.

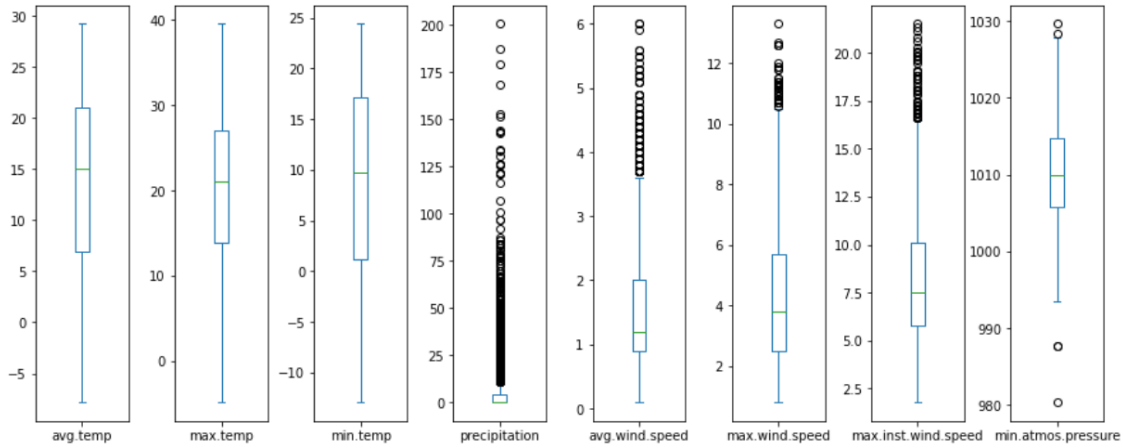


Figure 9 Boxplot of features

Takeaways:

As we can see we have outliers in the precipitation, avg windspeed, max wind speed, max inst wind speed and min atmosphere pressure columns.

More visualizations are created within the notebook based on data exploration, feature importance and finally creating the model.

5. Baseline Model

For creating baseline mode, two techniques have been utilized upto now i.e. LightGBM classification and XGB Classifier

5.1 LightGBM Classification

For LightGBM classification, categorical features are again converted into categorical from numeric as LightGBM Classifier is able to map it correctly in respect to other features and the data is splitted into 75% for training and 25% for testing as shown in Figure 8.

```
[ ] categoricals = list(train.select_dtypes(include=['int64']).columns)
    categoricals.remove("label")
    numericals = list(train.select_dtypes(include=['float64']).columns)

    feat_cols = categoricals + numericals
```

```
[ ] X = train[feat_cols]
    y = train['label']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

```
    clf = lgb.LGBMClassifier()

    clf.fit(X_train, y_train)
```

```
▾ LGBMClassifier
LGBMClassifier()
```

Figure 10 Train test split and fitting data into LGBM Classifier

First a non-normalized and normalized confusion matrix has been plotted as shown in Figure 11 and Figure 12

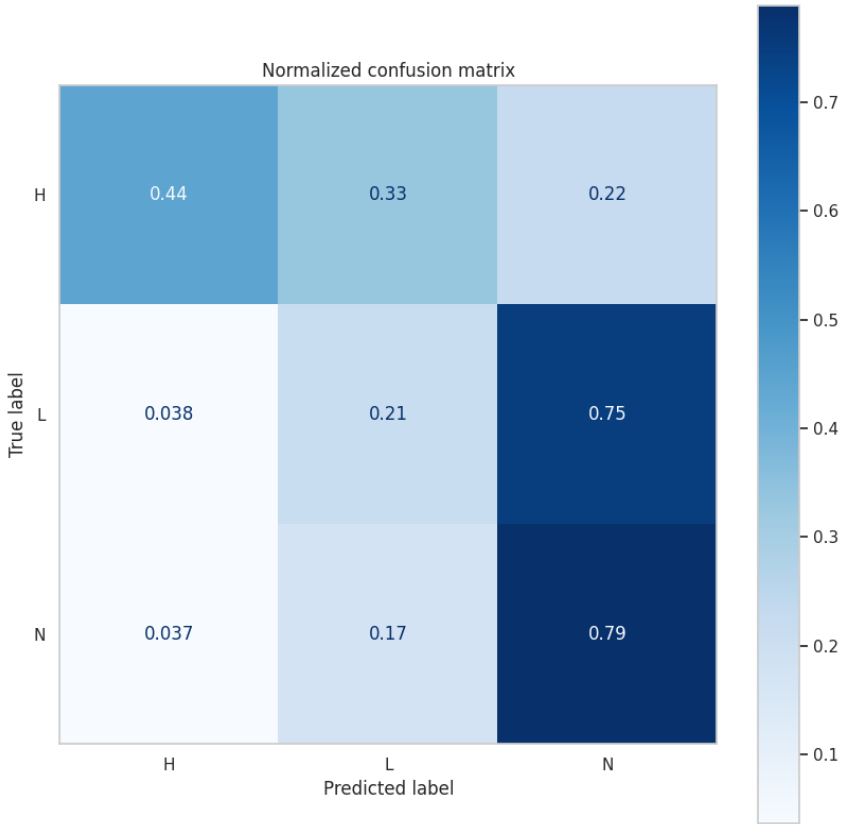


Figure 11 Normalized confusion Matrix

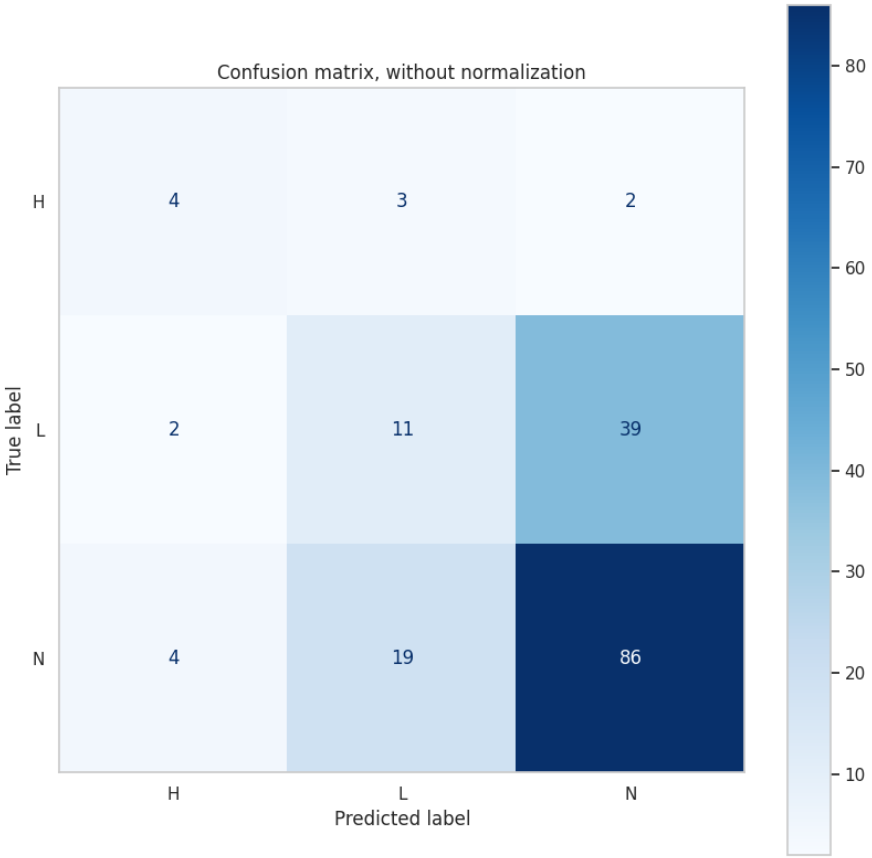


Figure 12 Non normalized confusion matrix

As you can see from the shade of the plot, our model is predicting the label “N” much more than others.

A classification report has been prepared as follows giving the quality of predictions.

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.40      0.44      0.42         9
           1       0.33      0.21      0.26        52
           2       0.68      0.79      0.73       109

    accuracy                           0.59       170
   macro avg       0.47      0.48      0.47       170
weighted avg       0.56      0.59      0.57       170
```

Figure 13 Classification Report

There are three values for the overall F1-score 0.59, 0.47, 0.57 .In an imbalanced dataset where all classes are equally important, macro average is a good choice as it treats all classes equally.

Note: Up to now proper quality of predictions haven't been obtained using LGBM Classifier but the parameter will be tuned in order to make the model more accurate and able to predict rain.

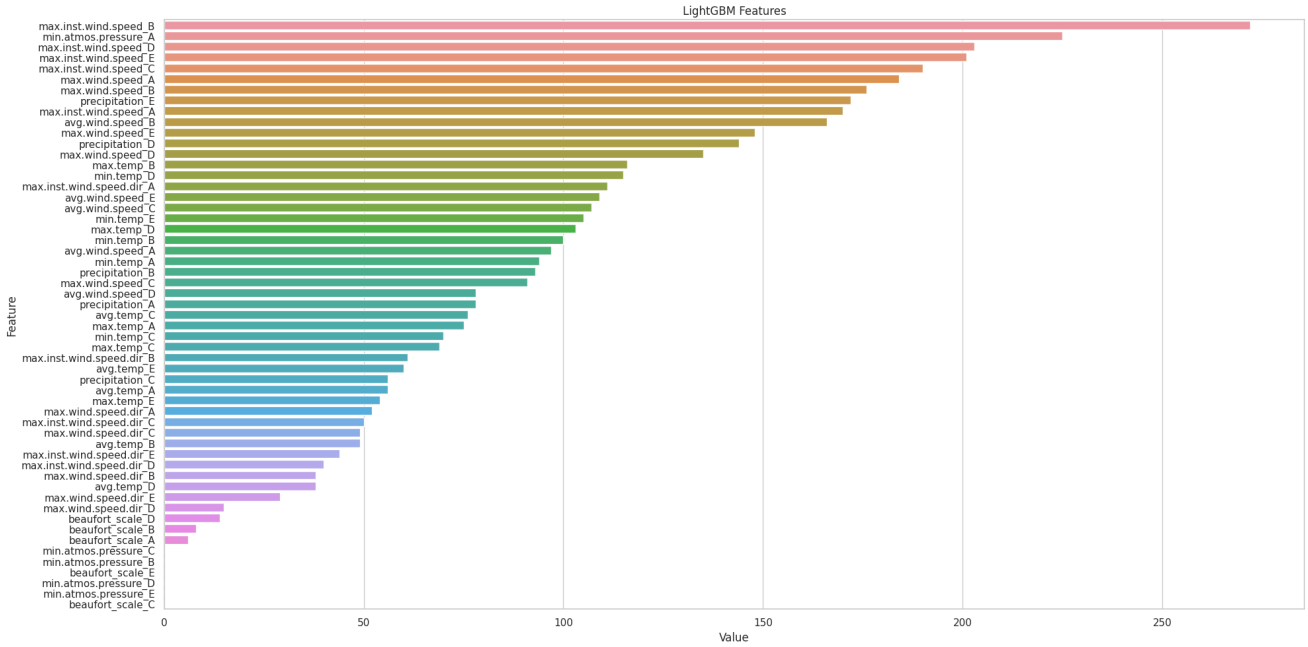


Figure 14 Feature Importance

From the plot above, the wind speed features and precipitation are the key features that are good predictors.The Beaufort scale feature created seems to have very low importance, so it might be better to remove it.

Interestingly, min.atmos.pressure in region A is the most important, whereas min.atmos.pressure in other regions are among the lowest in importance.

5.1.1 Predicting Test Data

The model is tested to predict rain prediction using test data as shown in Figure 15.

```
[ ] test_preds = clf.predict(X)
    submission_df = pd.concat([test['date'], pd.DataFrame(test_preds, columns=['label'])], axis=1)
    submission_df.head()
```

	date	label
0	0001f2fd	0
1	00177dc1	1
2	00b3a048	1
3	013f131b	2
4	01a1b150	2

Figure 15 Results of Rain Prediction on Test Data

The labels are still encoded as numeric values; let's bring the actual label names back. Since we already have the dictionary of the mapping of label names to numeric values, i.e. 'H' : 0, we can reverse the dictionary above to map the numbers to the names

```
submission_df['label'] = submission_df['label'].map(inv_map)
submission_df.head()
```

	date	label
0	0001f2fd	H
1	00177dc1	L
2	00b3a048	L
3	013f131b	N
4	01a1b150	N

Figure 16 Rain Prediction Results obtained based LGBM Model

5.2 XGB Classification

```
|: from xgboost import XGBClassifier

   xgb = XGBClassifier(booster = 'gbtree', learning_rate = 0.1, max_depth = 5, n_estimators = 180)
   xgb.fit(X_train_res, y_train_res);
```

```
|: XGBClassifierScore = xgb.score(X_test_res,y_test_res)
   print("Accuracy obtained by XGB Classifier model:", XGBClassifierScore*100)
```

```
Accuracy obtained by XGB Classifier model: 89.67136150234741
```

```
|: y_pred_xgb = xgb.predict(X_test_res)
   print(metrics.classification_report(y_test_res, y_pred_xgb))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        75
           1       0.81      0.93      0.86        67
           2       0.93      0.76      0.84        71

    accuracy                           0.90       213
   macro avg       0.90      0.90      0.89       213
weighted avg       0.90      0.90      0.90       213
```

Figure 17 XGB Classification Result

XGB Classification model technique after using the same shaped data that was splitting using train and test split, the accuracy obtained for the model was 89%.

6. Next Steps

The base model won't be enough to make a good prediction; here are some next steps to improve upon the given approach.

1. More feature preprocessing and engineering
2. Use cross-validation to have a better measure of the performance.
3. Tune class_weight parameter of LightGBM directly to handle imbalance classes
4. Balance the dataset by utilizing undersampling or resampling
5. Test out other algorithms like Random Forest Classifier, Optuna Stacking etc...