Machine Learning Classification Project : Weather Classification

Team Name : Code_Wars
Dhairya Gupta - IMT2021005
Sheikh Muteeb - IMT2021008
Suyash Ajit Chavan - IMT2021048

October 23, 2023

1 Introduction

In this report, **Code_wars** tackles the task of building a classification model to predict whether average humidity will surpass 50% the next day. Utilizing a diverse meteorological dataset, our focus is on creating a robust model that precisely classifies instances where humidity crosses this critical threshold. The dataset, ranging from temperature metrics to atmospheric conditions, serves as a valuable resource for training and evaluating our model. As we navigate the intricacies of the data, our goal is to uncover meaningful patterns essential for accurate predictions.

2 Exploratory Data analysis :

2.1 Statistical analysis of feature Columns:

Date Analysis:

- **Total Observations:** 116,368
- **Unique Dates:** 3,414
- **Most Frequent Date:** March 1, 2014, observed 47 times.

Location Analysis:

- **Total Observations:** 116,368
- **Unique Dates:** 49
- **Most Frequent Location:** Indore, observed 2,778 times.

Temperature Analysis:

- **Minimum Temperature (MinTemp):**
  - **Mean:** 12.20°C
  - **Range:** −8.50°C to 31.90°C
  - **Median:** 12.00°C
  - **Standard Deviation:** 6.40°C
- **Maximum Temperature (MaxTemp):**
  - **Mean:** 23.22°C
  - **Range:** −4.80°C to 47.30°C
  - **Median:** 22.60°C
  - **Standard Deviation:** 7.12°C

Precipitation and Evaporation Analysis:

- **Rainfall:**
  - **Mean:** 2.38 mm
  - **Range:** 0 mm to 371 mm
  - **Median:** 0 mm
  - **Standard Deviation:** 8.54 mm
- **Evaporation:**
  - **Mean:** 5.47 mm
  - **Range:** 0 mm to 145 mm
  - **Median:** 4.80 mm
  - **Standard Deviation:** 4.20 mm

Sunlight Analysis*(Sunshine Duration)*:

- **Mean:** 7.61 hours
- **Range:** 0 hours to 14.5 hours
- **Median:** 8.40 hours
- **Standard Deviation:** 3.79 hours

Wind Analysis Analysis:

- **Wind Gust Direction (WindGustDir):**
  - **Most Frequent Direction:** West(W) observed 7983 times.
- **Wind Direction at 9am (WindDir9am):**
  - **Most Frequent Direction:** North(N) observed 9426 times.

- **Wind Direction at 3pm (WindDir3pm):**
  - **Most Frequent Direction:** Souteast(SE) observed 8677 times.
- **Wind Gust Speed:**
  - **Mean:** 40.04 km/h
  - **Range:** 6 km/h to 135 km/h
  - **Median:** 39 km/h
  - **Standard Deviation:** 13.63 km/h

Pressure Analysis:

- **Pressure at 9am:**
  - **Mean:** 1017.64 hPa
  - **Range:** 980.50 hPa to 1041.00 hPa
  - **Median:** 1017.60 hPa
  - **Standard Deviation:** 7.12 hPa
- **Evaporation:**
  - **Mean:** 1015.25 hPa
  - **Range:** 978.20 hPa to 1039.60 hPa
  - **Median:** 1015.20 hPa
  - **Standard Deviation:** 7.05 hPa

Cloud Coverage Analysis:

- **Cloud Coverage at 9am:**
  - **Mean:** 4.45 oktas
  - **Range:** 0.00 oktas to 9.00 oktas
  - **Median:** 5.00 oktas
  - **Standard Deviation:** 2.89 oktas
- **Cloud Coverage at 3pm:**
  - **Mean:** 4.51 oktas
  - **Range:** 0.00 oktas to 9.00 oktas
  - **Median:** 5.00 oktas
  - **Standard Deviation:** 2.72 oktas

Task*(Target Variable Analysis)*:

- **Mean:** 0.224
- **Range:** 0.00 to 1.00
- **Median:** 0.00
- **Standard Deviation:** 0.417

2.2 Correlation matrix :

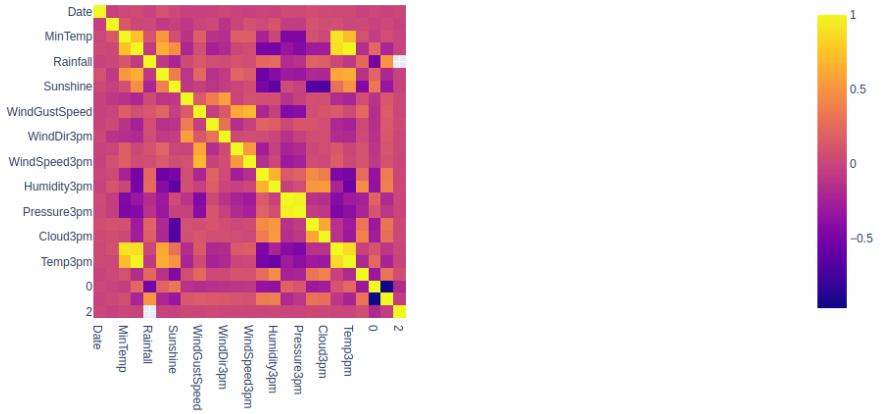

Figure 1: Correlation Matrix.

In the correlation matrix we looked for correlation across different feature and on basis of that dropped some features which had a high correlation with other features, which may be implying the same message. The correlation we found in the matrix and the columns we dropped accordingly:

- *Sunshine* showed strong negative correlation with *Cloud Cover* and *Humidity.*
- *MaxTemp, MinTemp* showed strong positive correlation with Temparature , and also *Temp9am* and *Temp3pm* had a strong positive correlation.
- Based on this correlation we tried dropping different combinations of columns to achieve maximum accuracy.

The target variable *Task* had:

- Strong negative correlations with:
  - *Sunshine*
  - *MaxTemp*
  - *Pressure9am and Pressure3pm*
- Strong positive correlations with:
  - *Rainfall*
  - *Humidity both at 3pm and 9am*
  - *Cloud3pm and Cloud9am*

2.3 Visualizations for feature distribution :

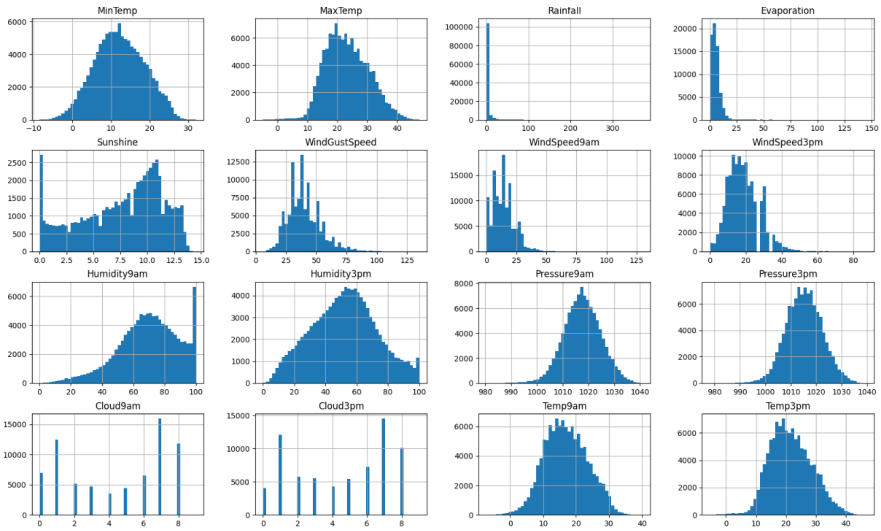

Figure 2: Histogram for feature columns.

We used histograms for the features to visualize their distribution , and also used for identifying outliers:
The following insights were gained from it:

- Feature columns *MinTemp, MaxTemp, Pressure3pm, Pressure9am, Temp9am, Temp3pm, Humidity3pm* roughly followed a normal distribution with very small amount of outliers.
- Feature columns *Rainfall, Evaporation, Wind9am, WindGustSpeed* followed a very skewed distribution with a large amount of outliers.
- Majority values of the *Rainfall* feature were 0.

2.4 Box plots to identify outliers :

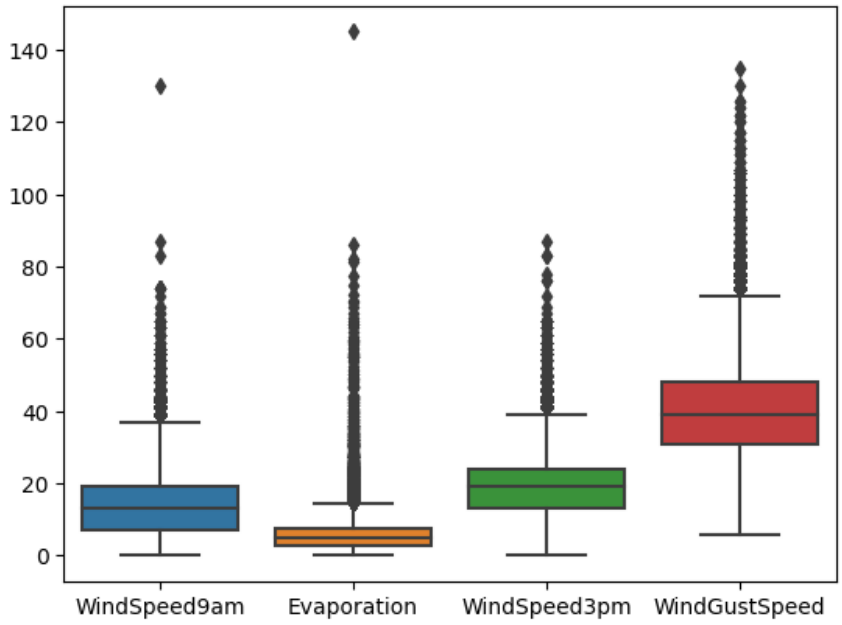

Figure 3: Box Plots.

Box Plots were used to identify outliers in the feature columns:
The following is the analysis of the box plots:

- The feature columns *WindSpeed9am, WindSpeed3pm, Evaporation, WindGustSpeed* had very big outliers and were handled using the IQR method
- Features like *Pressure , Temperature , Min/Max Temperatures, Humidity* showed very small to almost negligible amount of outliers.
- Rest of the feature columns showed some amount of outliers and were handeled using the IQR method.

3 Preprocessing of data :

3.1 Encoding:

A combination of one hot and label encoding was employed for categorical columns. Specifically:

- One hot encoding was used for low cardinality columns.
- Label encoding was applied to high cardinality columns.

This approach ensured efficient representation of categorical data without an explosion in the number of columns, especially for high cardinality features.

3.2 Feature Engineering :

Purpose: We decided to replace the Date feature with three new features: 'Day,' 'Month,' and 'Year.' The idea behind this change was to facilitate the imputation of numerical features such as Sunshine and Evaporation. These two columns had a substantial number of missing values, ranging from 40% to 50%. Initially, we employed an iterative imputer to fill in all the missing values. However, we later experimented with a different approach, grouping the values based on the Month and Location columns. Our most successful results came from grouping solely by Month.

3.3 Imputing:

Methods: We experimented with various imputation methods, including KNNImputer and SimpleImputer with different strategies. Ultimately, Iterative Imputer proved to be the most effective in obtaining the desired results.

Method Descriptions:

- **Iterative Imputer:** Iterative Imputer is an advanced method for filling missing values in data. It iteratively estimates missing values based on observed data, considering relationships between features for more accurate imputation in complex datasets.
- **Simple Imputer:** SimpleImputer is a basic imputation technique that fills missing values with a constant or statistical measure, such as the mean, median, or most frequent value of the observed data. It provides a straightforward way to handle missing values but may not capture complex relationships within the dataset.
- **KNNImputer:** As it's name suggests KNNImputer is an imputation technique that uses the k-nearest neighbors algorithm to fill missing values. It estimates missing data points based on the values of their nearest neighbors in the dataset, making it a more adaptive method that considers local patterns in the data.

After determining Iterative Imputer as the most effective approach, we sought to enhance accuracy further by implementing a custom imputation strategy for specific columns, such as "Evaporation" and "Sunshine." Considering the significance of temporal patterns, we introduced "Months" and "Year" columns. Our rationale was that data entries sharing the same location and month should exhibit comparable values for the specified columns. Consequently, we grouped the data by both place and month, implementing a tailored imputation method that considered the contextual relevance of entries. Notably, the most substantial improvement in accuracy was observed when grouping solely based on months, resulting in a marginal yet noteworthy enhancement in our overall results.

3.4 Sampling :

Purpose: We employed sampling techniques to address the imbalance in the classified data for both classes, which initially had a ratio of about 2:8 for each class. To rectify this, we utilized the imblearn.over_sampling library. Additionally, instead of balancing the dataset directly, we explored an alternative approach by adjusting the hyperparameter "scale_pos_weight" in the XGBoost-Classifier. Interestingly, both methods yielded the same score on Kaggle, which happened to be our best-performing result.

3.5 Standardization and normalization :

Methods: We tried using "Standardisation" and "Normalization":
Standardisation: It is a data preprocessing technique that scales and centers features, making their mean zero and standard deviation one. It ensures consistent scales across features, improving the performance of machine learning algorithms.
Normalization: Normalization is a data preprocessing method that scales feature values to a common range, typically between 0 and 1, ensuring equal contribution from different features in machine learning models.

We used "StandardScaler" for Stnadardisation and "Normalizer" for Normalization. But both them gave a suboptimal answer after appliying it on data.

4 Model Training :

4.1 Principal component analysis (PCA) :

Purpose: Weather classification aims to classify test data into 2 categories based on whether humidity crosses 50% on following day. PCA can be employed to simplify the analysis of weather data by reducing the dimensionality while preserving essential information.

Advantages of PCA in Weather Classification:

- **Dimensionality Reduction:** PCA reduces the number of variables, making it easier to visualize and analyze weather data.
- **Noise Reduction:** By focusing on the most significant sources of variation, PCA can help reduce noise in the data.
- **Computational Efficiency:** The reduced dataset is computationally more efficient for classification algorithms, speeding up the analysis process.

Observations after using PCA : If we choose to select first K Principal directions, following results are obtained :

1. $10 \leq k \leq 16$: score = 0.62765 (same as original data)
2. $4 < k < 10$: score = 0.61895 (decrease in performance)
3. $k < 4$: score = 0.61611

4.2 Metrics :

Following machine learning metrics are used to identify best model :

- **1. Accuracy :** Measures the proportion of correctly classified instances (both true positives and true negatives) to the total number of instances. It provides an overall view of the model's performance.
  Formula : (TP + TN) / (TP + TN + FP + FN)
- **2. Precision :** Focuses on the accuracy of positive predictions. It measures the ratio of true positive predictions to all positive predictions made by the model.
  Formula : TP / (TP + FP)
- **3. Recall :** Evaluates the model's ability to identify all relevant instances of the positive class. It measures the ratio of true positives to all actual positive instances.
  Formula : P / (TP + FN)
- **4. F1Score :** Combines precision and recall into a single metric, providing a balance between them. It is particularly useful when you want to consider both false positives and false negatives.
  Formula : 2 * (Precision * Recall) / (Precision + Recall)

4.3 Training various Classification models :

Following classification models are tried out for training process :
1. Decision Tree classifier
2. Random Forest classifier
3. Gradient Boosting classifier
4. Logistic regression
5. Stochastic gradient descent classifier
6. XGBoost classifier
7. K-nearest neighbors classifier
8. ligthGbm classifier

Results :
1. We got best performance from XGBoost model after tuning its hyper-parameters.
2. Second best scores are observed from Random forests with hyper-parameters tuning.

4.4 Decision Tree classifier :

Description : Decision trees are a supervised learning method used for classification. They make predictions by recursively partitioning the input space into regions and assigning a class label to each region. Each node in the tree represents a decision or a test on an input feature.

Results : 1. Following scores are observed on validation data:

- a. Accuracy : 0.8371
- b. Precision : 0.8324
- c. Recall : 0.8448
- d. F1 Score : 0.8385

4.5 Random Forest classifier :

Description : Random Forest is an ensemble learning method that consists of multiple decision trees. It aggregates the predictions of these trees to make more robust and accurate classifications. It helps reduce overfitting and improve generalization.

Results : 1. Following scores are observed on validation data:

- a. Accuracy : 0.9084
- b. Precision : 0.9098
- c. Recall : 0.9071
- d. F1 Score : 0.9085

2. **Hyperparameter tuning :** We manually tuned following hyperparameters by doing normal binary search. Grid-Search was also tried, but it took long time to train and gave poor performance.

- a. n_estimators : 500
- b. min_sample_split : 2
- c. max_features : "log2"
- d. class_weight : "balanced"

4.6 Gradient Boosting classifier :

Description : Gradient Boosting is another ensemble method that builds decision trees sequentially. It corrects the errors of the previous trees, focusing on

the instances that were misclassified, and combines their predictions to create a strong classifier.

Results : 1. Following scores are observed on validation data:

- a. Accuracy : 0.8695
- b. Precision : 0.8796
- c. Recall : 0.8562
- d. F1 Score : 0.8677

4.7 Logistic regression :

Description : Logistic regression is a statistical model used for binary classification. It models the probability of an instance belonging to a particular class and makes predictions based on a logistic function.

Results : 1. Following scores are observed on validation data:

- a. Accuracy : 0.7895
- b. Precision : 0.7954
- c. Recall : 0.7796
- d. F1 Score : 0.7874

4.8 Stochastic gradient descent classifier :

Description : SGD is an optimization technique used with various machine learning models, including logistic regression. It updates model parameters iteratively to minimize the loss function and improve classification accuracy.

Results : 1. Following scores are observed on validation data:

- a. Accuracy : 0.7429
- b. Precision : 0.8841
- c. Recall : 0.5590
- d. F1 Score : 0685

4.9 XGBoost classifier :

Description : XGBoost (Extreme Gradient Boosting) is an optimized implementation of gradient boosting that is known for its speed and performance. It uses a regularized objective function and incorporates techniques to prevent overfitting.

Results : 1. Following scores are observed on validation data:

- a. Accuracy : 0.9157
- b. Precision : 0.9375
- c. Recall :0.8912
- d. F1 Score : 0.9138

2. **Hyperparameter tuning :** We manually tuned following hyperparameters by doing normal binary search. Grid-Search was also tried, but it took long time to train and gave poor performance.

- a. n_estimators : 800
- b. max_depth : 13
- c. learning_rate : 0.08
- d. gamma : 0.5
- e. reg_lambda : 10
- f. min_child_weight : 7
- g. objective : "binary:logistic"
- h. eval_metric : "logloss"
- i. scale_pos_weight : 1

4.10 K-nearest neighbors classifier :

Description : KNN is a non-parametric and instance-based learning algorithm used for classification. It assigns a class label to an instance based on the majority class among its 'k' nearest neighbors in the feature space.

Results : 1. Following scores are observed on validation data:

- a. Accuracy : 0.8657
- b. Precision : 0.7997
- c. Recall : 0.9758
- d. F1 Score : 0.8790

4.11 ligthGbm classifier :

Description : LightGBM is a gradient boosting framework that uses a histogram-based learning method. It is designed for efficiency and can handle large datasets with high dimensionality. LightGBM uses a leaf-wise growth strategy to build decision trees.

Results : 1. Following scores are observed on validation data:

- a. Accuracy : 0.9049
- b. Precision : 0.9354
- c. Recall : 0.8699
- d. F1 Score : 0.9014