****

**End Semester Machine Learning Project- Weather Classification using Machine Learning Models**



**Name** - Nirupama N

**Register number** - 125018047

**Department**  - BTech - Computer Science & Business Systems

# 

# TABLE OF CONTENT

| Page No. | Content |
| --- | --- |
| 3 | Abstract |
| 4 | Introduction |
| 6 | Related Work |
| 7 | Background |
| 13 | Methodology |
| 24 | Results |
| 25 | Discussion |
| 27 | Learning Outcome |
| 28 | Conclusion |

# ABSTRACT

This project focuses on the classification and analysis of weather conditions using advanced machine learning techniques, specifically Logistic Regression, Decision Tree, Random Forest, and XGBoost classifiers. With the increasing importance of accurate weather predictions in various sectors, this study aims to enhance forecasting accuracy by categorising weather conditions based on key features such as temperature, humidity, wind speed, precipitation, and visibility. The dataset comprises ten distinct features, providing a comprehensive view of the atmospheric conditions that influence weather patterns.

Through the application of these classifiers, the project evaluates their performance in terms of predictive accuracy and interpretability, identifying critical factors that contribute to reliable weather forecasting. The results demonstrate the strengths and weaknesses of each model, with particular emphasis on how ensemble methods like Random Forest and XGBoost can significantly improve classification outcomes compared to traditional methods.

Ultimately, this project aims to deliver precise weather predictions applicable in agriculture, disaster management, and daily planning, thereby aiding decision-makers in effectively responding to weather-related challenges. By understanding the underlying patterns and relationships within the data, this study not only advances the field of meteorological forecasting but also provides valuable insights for stakeholders across various industries. The findings underscore the potential of machine learning to transform weather prediction practices, paving the way for more resilient and adaptive strategies in the face of changing climatic conditions.

# INTRODUCTION

## Importance of Dataset

Weather classification is crucial for various applications, including agriculture, event planning, and climate studies. The dataset utilised in this project encompasses meteorological features essential for understanding weather patterns, making it a valuable resource for predictive modelling.

The dataset contains synthetic data for weather type classification, including features like temperature, humidity, wind speed, precipitation percentage, cloud cover, atmospheric pressure, UV index, season, visibility, and location. Ideal for practising classification algorithms and exploring weather pattern predictions.

It includes various weather-related features and categorises the weather into four types: Rainy, Sunny, Cloudy, and Snowy. This dataset is designed for practising classification algorithms, data preprocessing, and outlier detection methods.

The dataset includes several columns:

* Temperature: The current temperature (°C).
* Humidity: The percentage of humidity.
* Wind Speed: Speed of wind (km/h).
* Precipitation (%): Percentage of precipitation.
* Atmospheric Pressure: Pressure in hPa.
* UV Index: The level of ultraviolet radiation.
* Visibility (km): Visibility in kilometres.
* Season: The season during the observation (Spring, Summer, Autumn, Winter).
* Location: The geographical location of the observation.
* Cloud Cover: The percentage of cloud cover.
* Weather Type: The target variable representing different weather types (e.g., Sunny, Rainy, Snowy).

## Objective (TPE Format):

* **Task:** Develop and evaluate models for weather classification.
* **Performance:** Identify the most accurate model while ensuring generalisation with cross-validation.
* **Experience:** Gain practical skills in data handling, pre - processing, model training, and evaluation

## Approach:The project involves data preprocessing, visualisation, detection of outlier, and training various models such as logistic regression, decision tree, random forest, support vector machine and XGBoost with cross-validation for reliable performance evaluation.

## Results Overview:

Random Forest and Decision Tree outperformed other models, delivering high classification accuracy on the dataset

# RELATED WORK

## References:

* + *ChatGPT:* Used for hyper - parameter tuning, generating ideas and summarising data insights.
  + *Kaggle:* Utilised for dataset reference and model resources.
  + No base papers were chosen.

Dataset link - <https://www.kaggle.com/code/dogukantabak/weather-type-classification>

<https://medium.com/@itbodhi/overfitting-and-underfitting-in-machine-learning-models-76cb60dbdaf6>

## Link to my project :

**Github Link :** https://github.com/nirupama0300/ML\_Project\_Weather\_Type\_Classification

**Colab Link :** https://colab.research.google.com/drive/1LHcvWiKsEVsBmZN3Hs4i8uxxm1040a6X?usp=sharing

# BACKGROUND

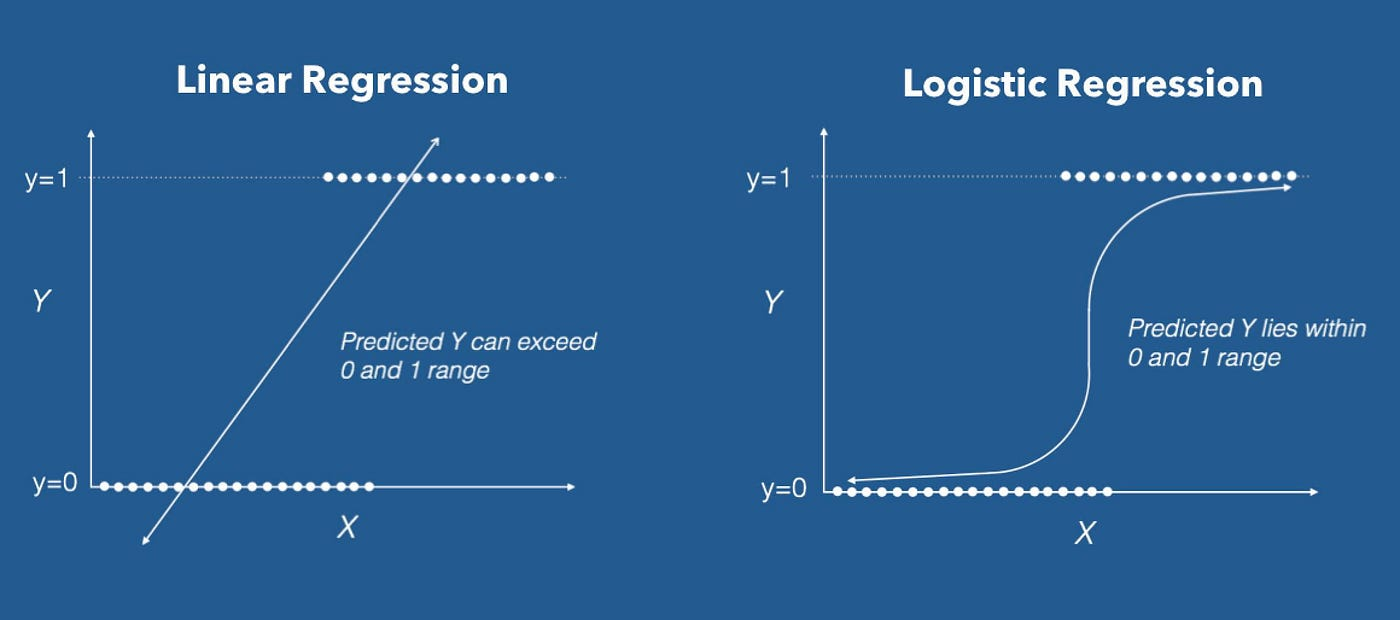
## Models Used

* **Logistic Regression:** Chosen for baseline performance on classification.
* **Random Forest:** Known for high accuracy and robustness in classification.
* **Decision Tree:** Interpretable model, suitable for understanding feature importance.
* **SVM:** Effective in high-dimensional spaces with robust classification capability.
* **XGBoost:** Utilised for its efficiency and performance with complex data.

### **1. Logistic Regression**

**Architecture and Working:**Logistic Regression is a linear model designed for binary classification tasks. The model applies a linear function to the features XXX to calculate a linear combination, which is then passed through a sigmoid function to produce a probability between 0 and 1.

**Mathematical Model:**The linear equation is:  
z=w1x1+w2x2+⋯+wnxn+bz = w\_1x\_1 + w\_2x\_2 + \dots + w\_nx\_n + bz=w1​x1​+w2​x2​+⋯+wn​xn​+b  
where wiw\_iwi​ are the weights, xix\_ixi​ are the features, and bbb is the bias term.  
The result zzz is transformed by the sigmoid function:  
σ(z)=11+e−z\sigma(z) = \frac{1}{1 + e^{-z}}σ(z)=1+e−z1​  
This maps zzz to a probability, allowing classification by setting a threshold (typically 0.5).

**Diagram:**The diagram shows the linear function followed by the sigmoid activation.

**Strengths and Use Cases:**Logistic regression is easy to interpret and computationally efficient. It is commonly used for binary classification problems where classes can be separated by a linear boundary (e.g., spam detection, medical diagnosis).

**Limitations:**It struggles with complex, non-linear data and is sensitive to feature scaling.

### **2. Random Forest**

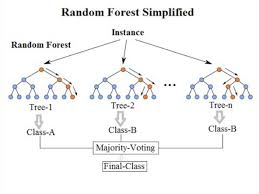
**Architecture and Working:**Random Forest is an ensemble model that consists of multiple Decision Trees. Each tree is trained on a random subset of the data, with different features considered at each split. The final output is the majority vote or average prediction of all trees in the ensemble.

**Model Formation:**

**Bootstrap Sampling:** Each tree is trained on a random subset of data points, sampled with replacement.

**Random Feature Selection:** For each split in the tree, only a random subset of features is considered, increasing model diversity and reducing correlation.

**Voting Mechanism:** For classification, the final output is the class that receives the majority vote across all trees.

**Diagram:  
**The diagram shows multiple Decision Trees trained on different samples, with a final "voting" stage.

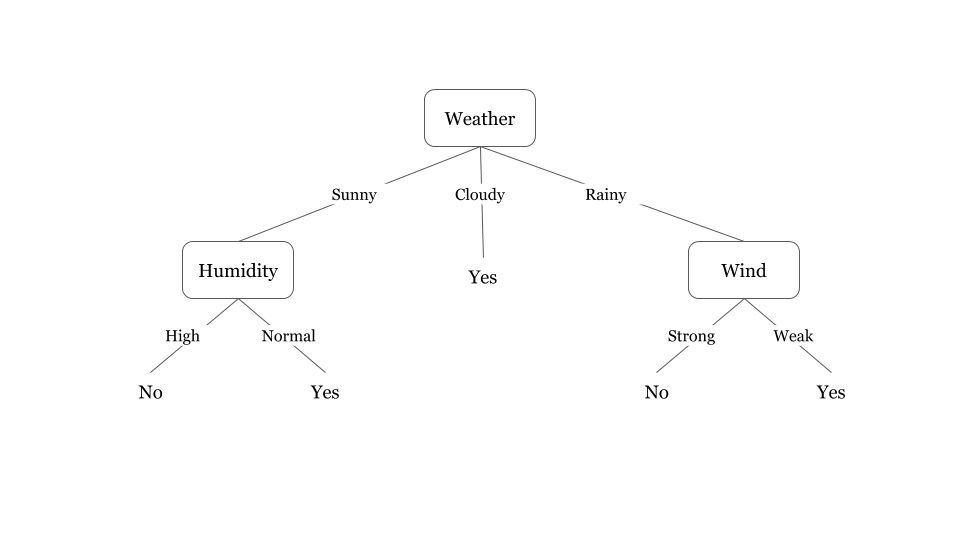
**Strengths and Use Cases:**Random Forest reduces overfitting and handles non-linear data well. It’s commonly applied in fields like finance and healthcare for its robustness and interpretability.

**Limitations:**It can be computationally intensive, especially with a large number of trees.

### **3. Decision Tree**

**Architecture and Working:**Decision Trees work by recursively splitting the dataset based on feature values to create a "tree" structure. At each node, the tree splits the data according to a feature that maximises information gain, until reaching leaf nodes with class labels.

**Splitting Process:**The model uses criteria like Gini impurity or entropy (information gain) to decide the best feature for each split.  
Gini impurity=1−∑i=1Cpi2\text{Gini impurity} = 1 - \sum\_{i=1}^{C} p\_i^2Gini impurity=1−i=1∑C​pi2​  
where pip\_ipi​ is the probability of a class at a node.

**Diagram:  
**The diagram shows a tree with branches representing decisions based on feature values, with leaf nodes indicating class outcomes.

**Strengths and Use Cases:**Decision Trees are interpretable, useful for explaining decisions, and handle both categorical and numerical features. They’re widely used in risk analysis and diagnostic applications.

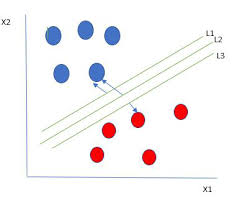
**Limitations:**They tend to overfit on complex datasets and may be sensitive to data changes.

### **4. Support Vector Machine (SVM)**

**Architecture and Working:**SVM aims to find a hyperplane that best separates the classes in the dataset. For linearly separable data, it creates a straight-line (or hyperplane) boundary, while for non-linear data, it uses a "kernel trick" to map data into a higher-dimensional space where it becomes linearly separable.

**Margin Optimization:**The SVM model selects a hyperplane that maximises the margin (distance) between the nearest data points from each class, called support vectors. The optimization objective is to minimise:  
Loss=12∣∣w∣∣2+C∑Hinge Loss\text{Loss} = \frac{1}{2} ||w||^2 + C \sum \text{Hinge Loss}Loss=21​∣∣w∣∣2+C∑Hinge Loss  
where CCC is a regularisation parameter.

**Kernel Trick:**In cases of non-linear separation, kernels like radial basis function (RBF) and polynomials transform the data to make separation feasible in higher dimensions.

**Diagram:**

The diagram illustrates the margin, support vectors, and the hyperplane separating two classes.

**Strengths and Use Cases:**SVM works well with high-dimensional data, particularly useful in text classification and image recognition.

**Limitations:**SVM is computationally expensive for large datasets and sensitive to parameter tuning.

### **5. XGBoost (Extreme Gradient Boosting)**

**Architecture and Working:**XGBoost is a gradient-boosting model that builds multiple weak learners (typically Decision Trees) sequentially. Each tree is designed to correct the errors of its predecessor by learning from the residuals, and a final aggregation produces the output.

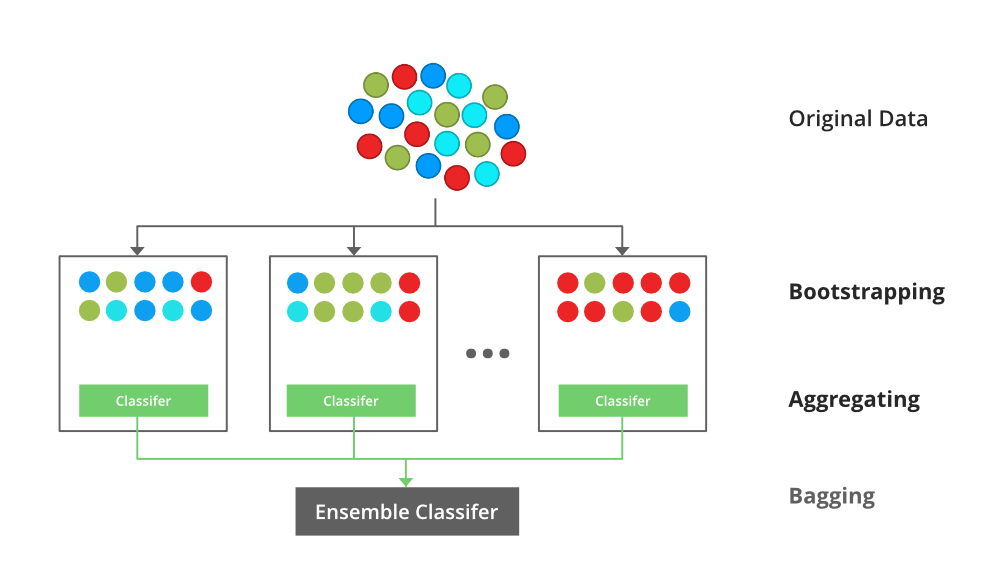
**Boosting Process:**

**Initialization:** Start with a weak model and calculate residual errors for predictions.

**Sequential Learning:** Each subsequent tree fits the residual errors of the previous trees.

**Weighted Sum of Trees:** The predictions are weighted sums of the outputs of all trees, with adjustments to correct errors.

**Regularisation:**XGBoost includes L1L\_1L1​ and L2L\_2L2​ regularisation terms in its objective function to control complexity and avoid overfitting. The objective function typically minimises a combination of training loss and model complexity:  
Objective=∑Loss+λ∣∣w∣∣2+α∣∣w∣∣\text{Objective} = \sum \text{Loss} + \lambda ||w||^2 + \alpha ||w||Objective=∑Loss+λ∣∣w∣∣2+α∣∣w∣∣  
where λ\lambdaλ and α\alphaα are regularisation terms.

**Diagram:  
**

**Strengths and Use Cases:**XGBoost is highly efficient, accurate, and performs well with structured/tabular data. It's popular in competitive machine learning and used in applications like recommendation systems.

**Limitations:**It has many hyperparameters, making tuning complex, and it may overfit if not regularised.

# METHODOLOGY

The project follows a structured approach: data collection, preprocessing, model training, evaluation, and analysis of results. Each model's performance will be compared using accuracy metrics

## **Environment and Tools**

* **Programming Language:** Python
* **Libraries:** Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn, XGBoost
* **Development Environment:** Google Colab for ease of collaboration and accessibility.

Link to code:

**Github Link :** https://github.com/nirupama0300/ML\_Project\_Weather\_Type\_Classification

**Colab Link :** https://colab.research.google.com/drive/1LHcvWiKsEVsBmZN3Hs4i8uxxm1040a6X?usp=sharing

## Data Preprocessing:

## Data Loading:

The dataset was loaded using pandas, and initial inspections revealed useful statistics and insights, including:

* **Data Information:** Data types and null values. Used data.info()
* **Descriptive Statistics:** Basic statistics of numerical features. Used data.describe()
* **Duplicate Records:** Checked for and handled any duplicate entries. Used data.duplicated().sum() and found that there were duplicates present. They were eventually removed.

## Data Visualisation

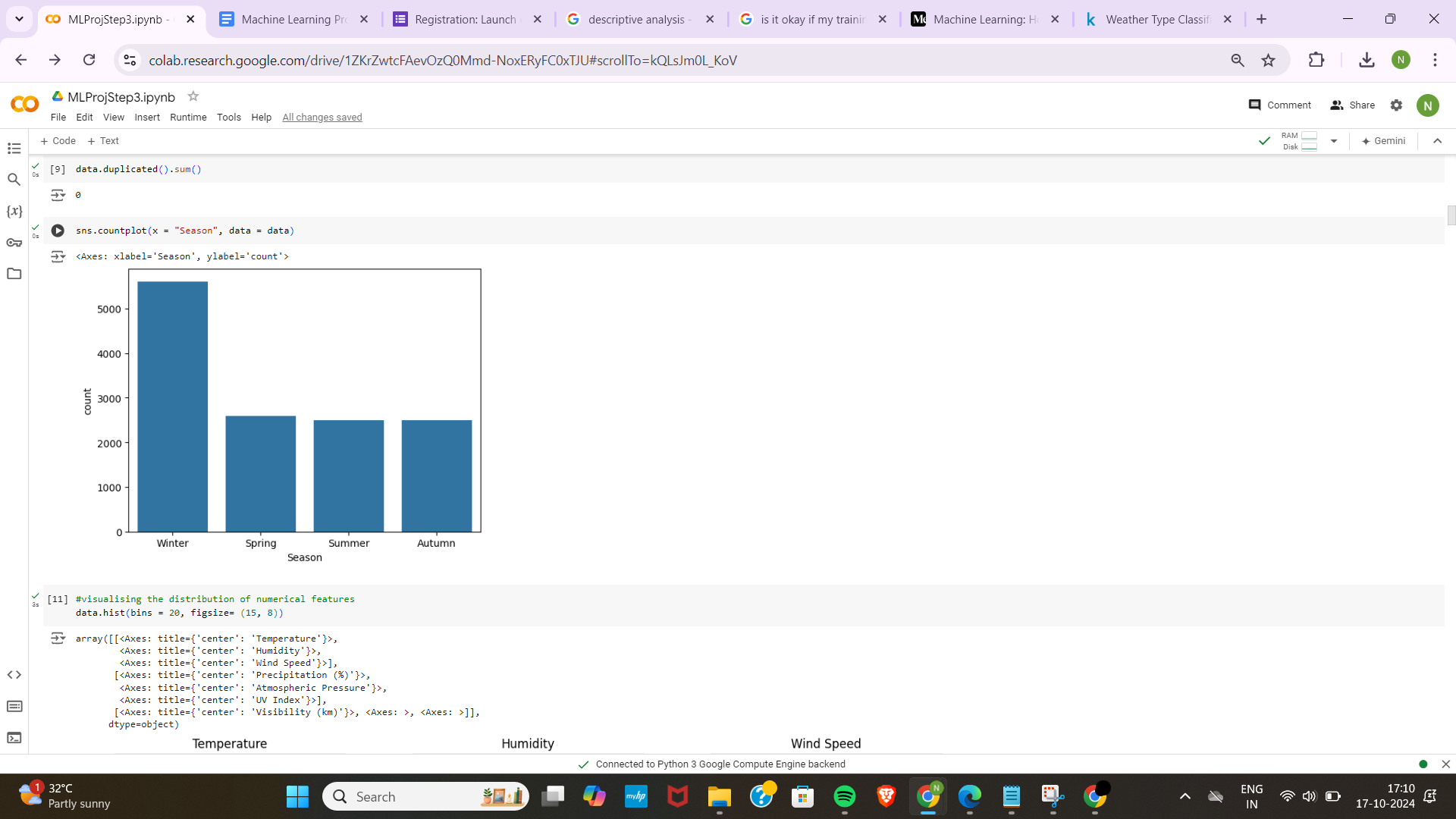
Visualisations were created using seaborn and matplotlib to understand the distribution of features:

* Count plots for categorical variables like **Season, Cloud Cover, Location, and Weather Type**.
* Line plots to analyse relationships between numerical features, such as **Temperature vs. Humidity and Wind Speed**.

UNIVARIATE ANALYSIS:

* Count Plots

Used seaborn's countplot to visualise the distribution of categorical variables:

* **Cloud Cover**: Understanding the frequency of different cloud cover levels.
* **Season**: Observing the number of observations per season.
* **Location**: Checking the distribution of data across different locations.
* **Weather Type**: Analysing the frequency of each weather type in the dataset.

These plots helped identify any class imbalances that might affect model training.

* Histograms

Plotted histograms for numerical variables using data.hist() to observe their distributions and identify any skewness or kurtosis that might require transformation.

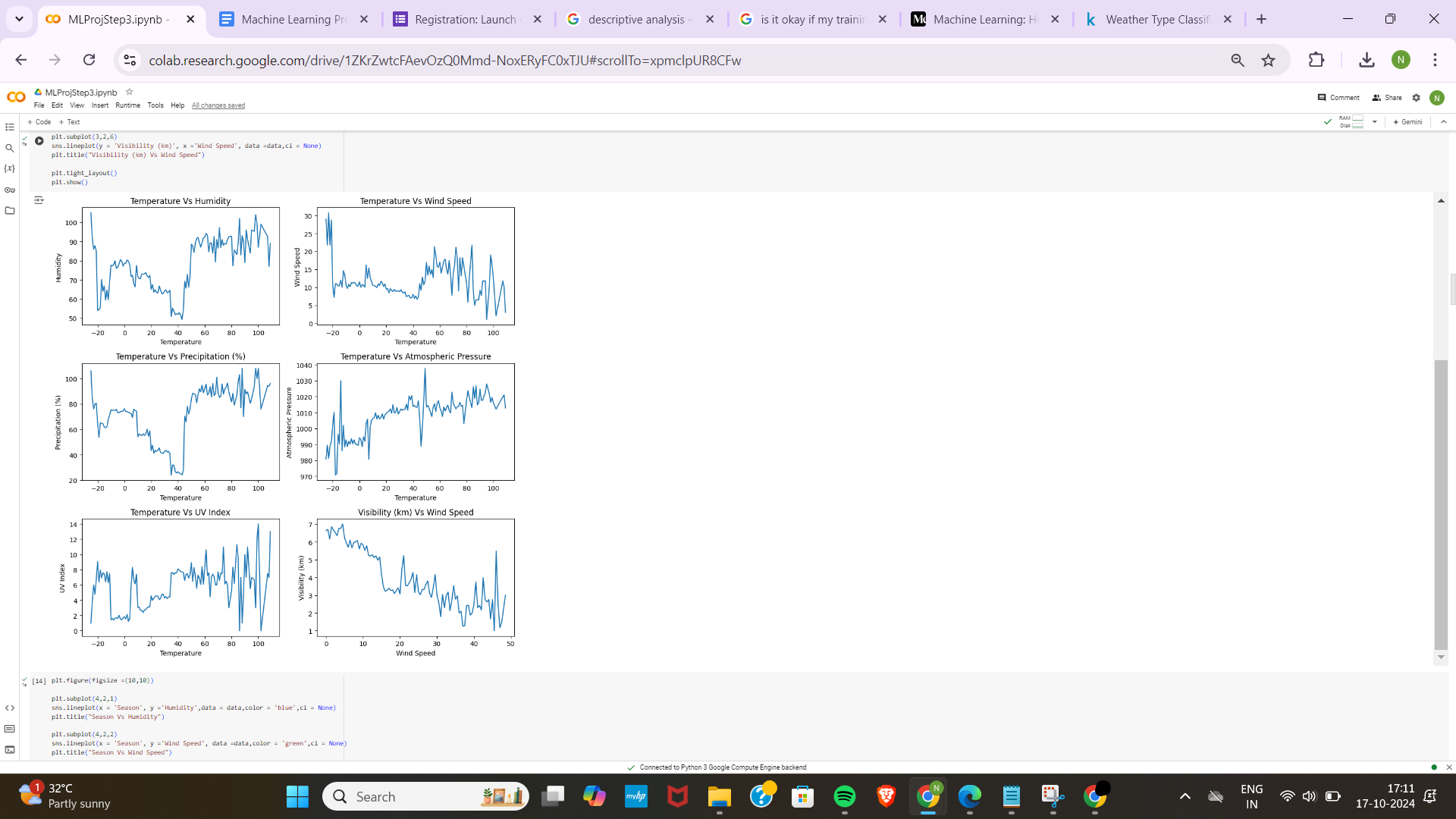


BIVARIATE ANALYSIS

Line Plots

Examined relationships between temperature and other continuous variables:

* **Temperature vs. Humidity**: To see how temperature variations affect humidity levels.
* **Temperature vs. Wind Speed**: Understanding the correlation between temperature and wind speed.
* **Temperature vs. Precipitation (%)**: Investigating how temperature impacts precipitation probability.
* **Temperature vs. Atmospheric Pressure**: Analysing the interplay between temperature and atmospheric pressure.
* **Temperature vs. UV Index**: Observing how temperature relates to UV exposure.
* **Visibility vs. Wind Speed**: Exploring the effect of wind speed on visibility.

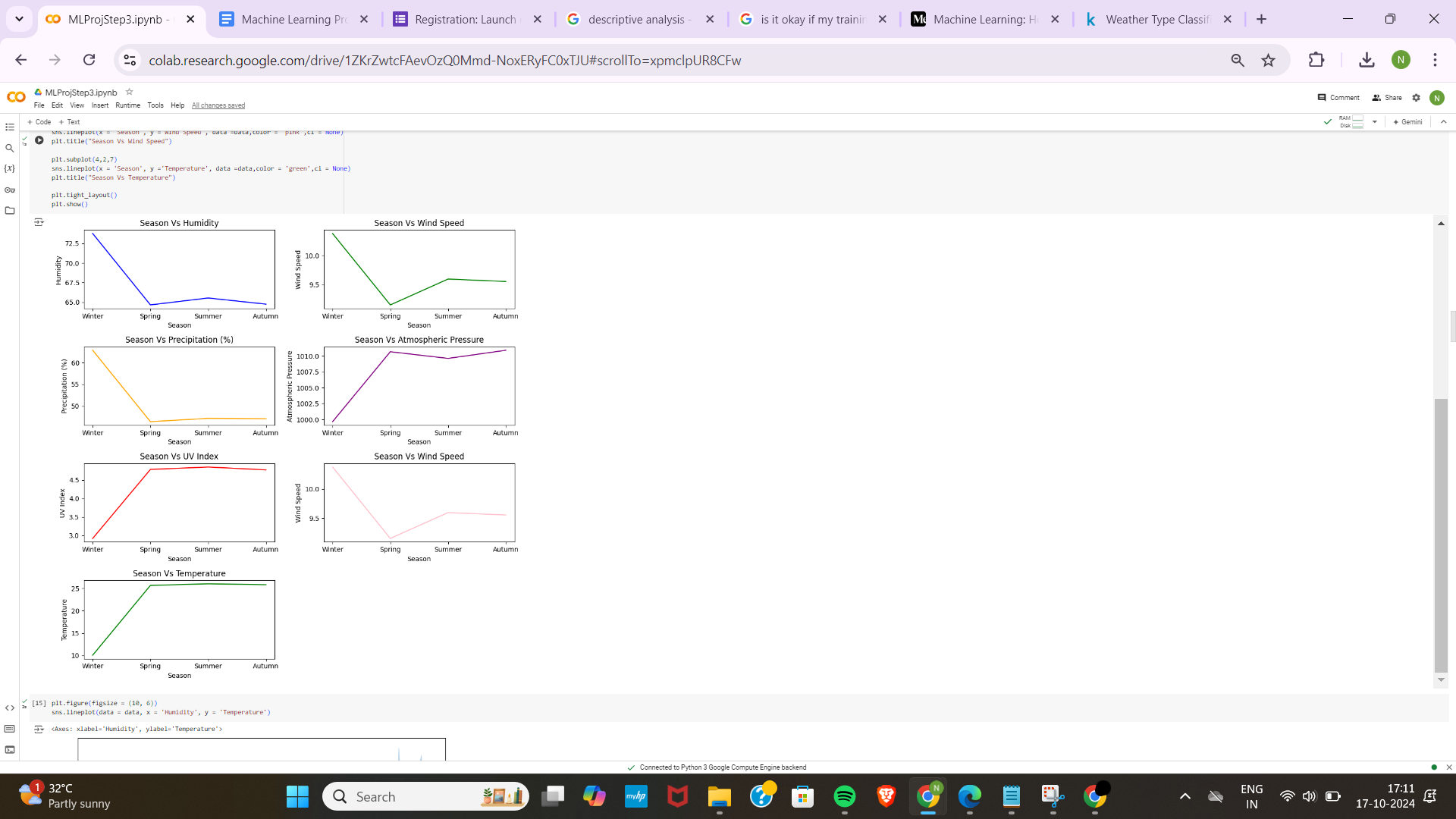


These plots provided insights into potential predictors for the weather type.

Analysis

Analysed how different weather parameters change with seasons by plotting:

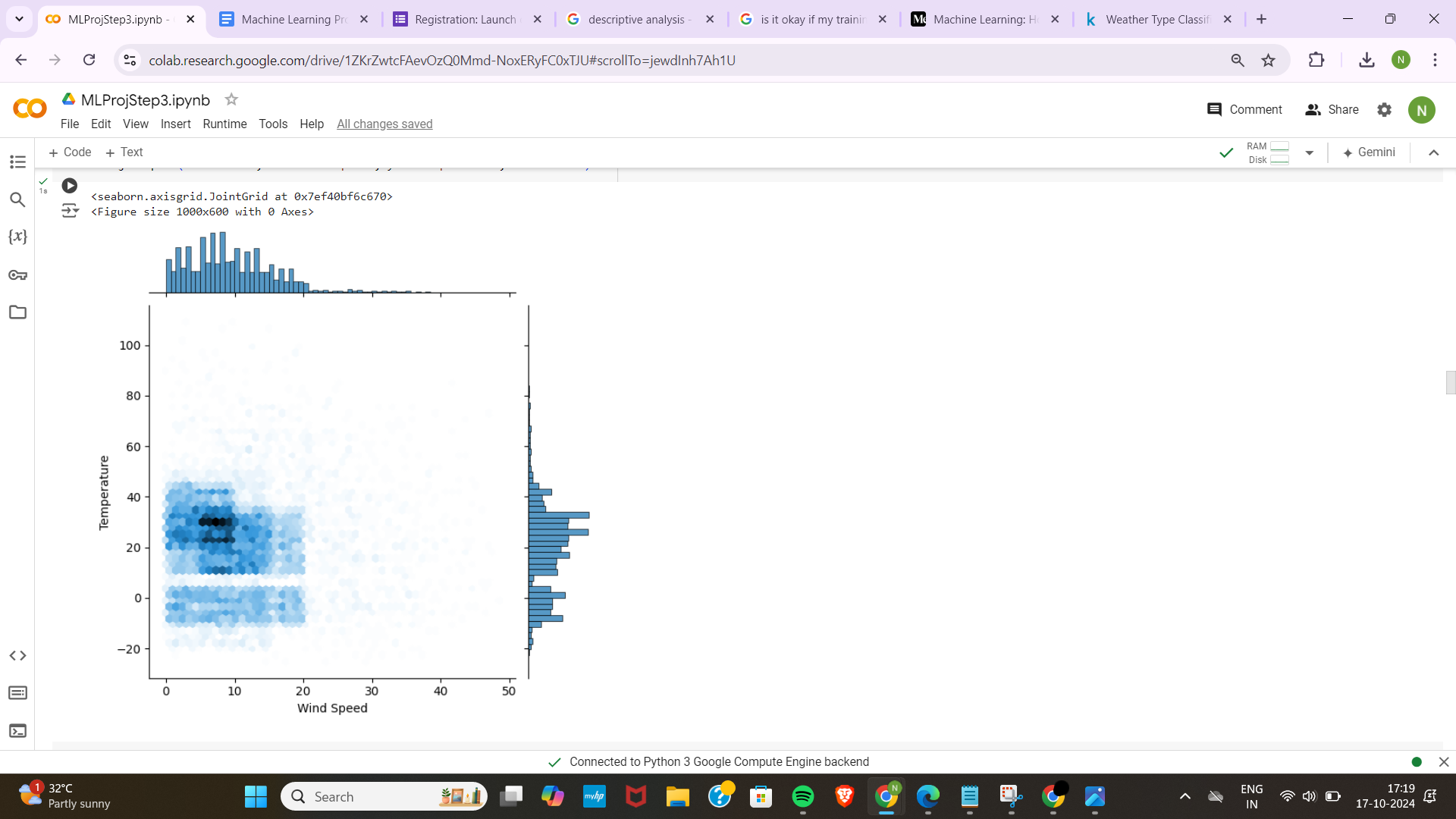
* **Season vs. Humidity**
* **Season vs. Wind Speed**
* **Season vs. Precipitation (%)**
* **Season vs. Atmospheric Pressure**
* **Season vs. UV Index**
* **Season vs. Temperature**

****

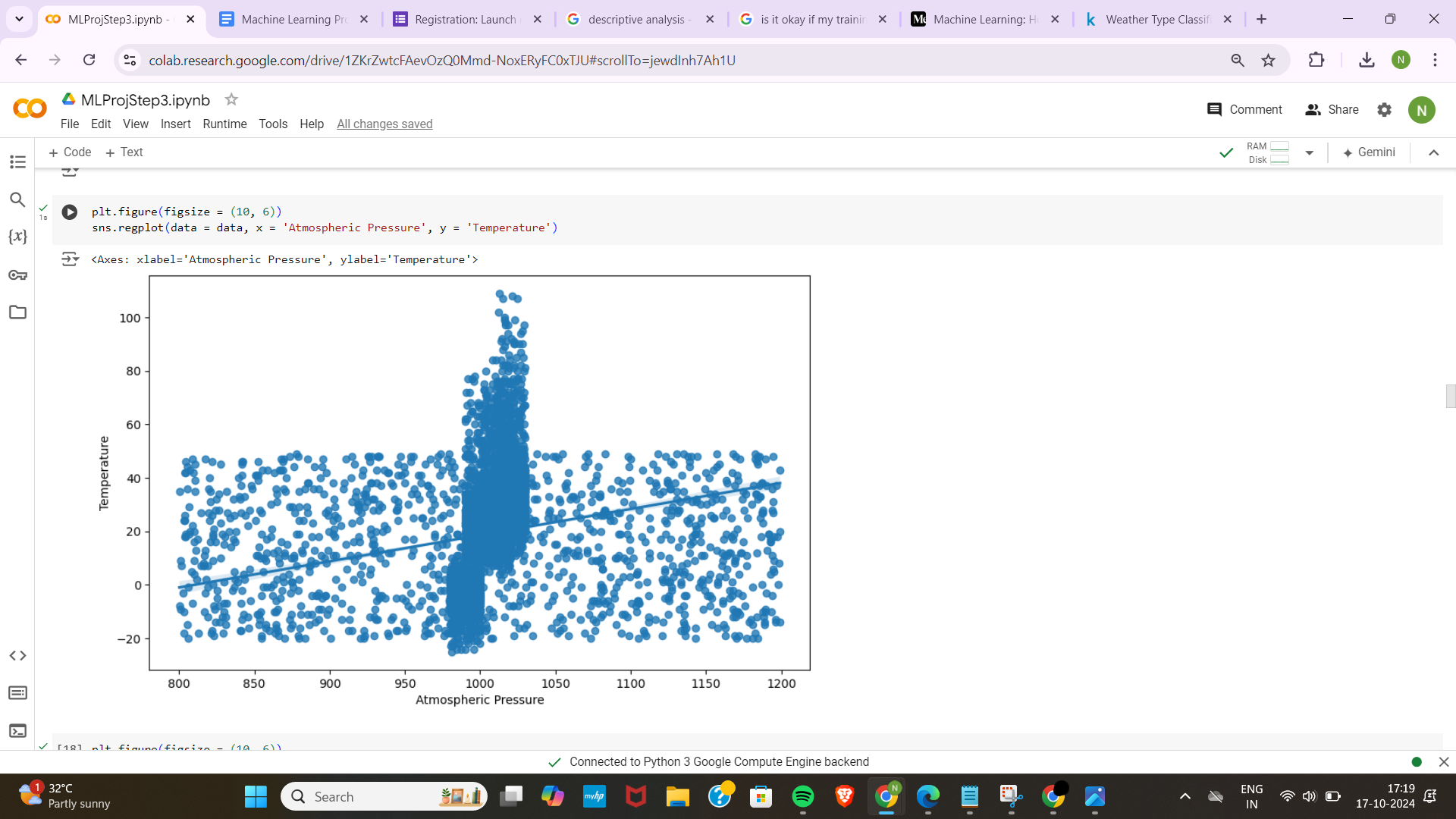
This helped in understanding seasonal patterns that could influence weather classification.

Other Visualisations

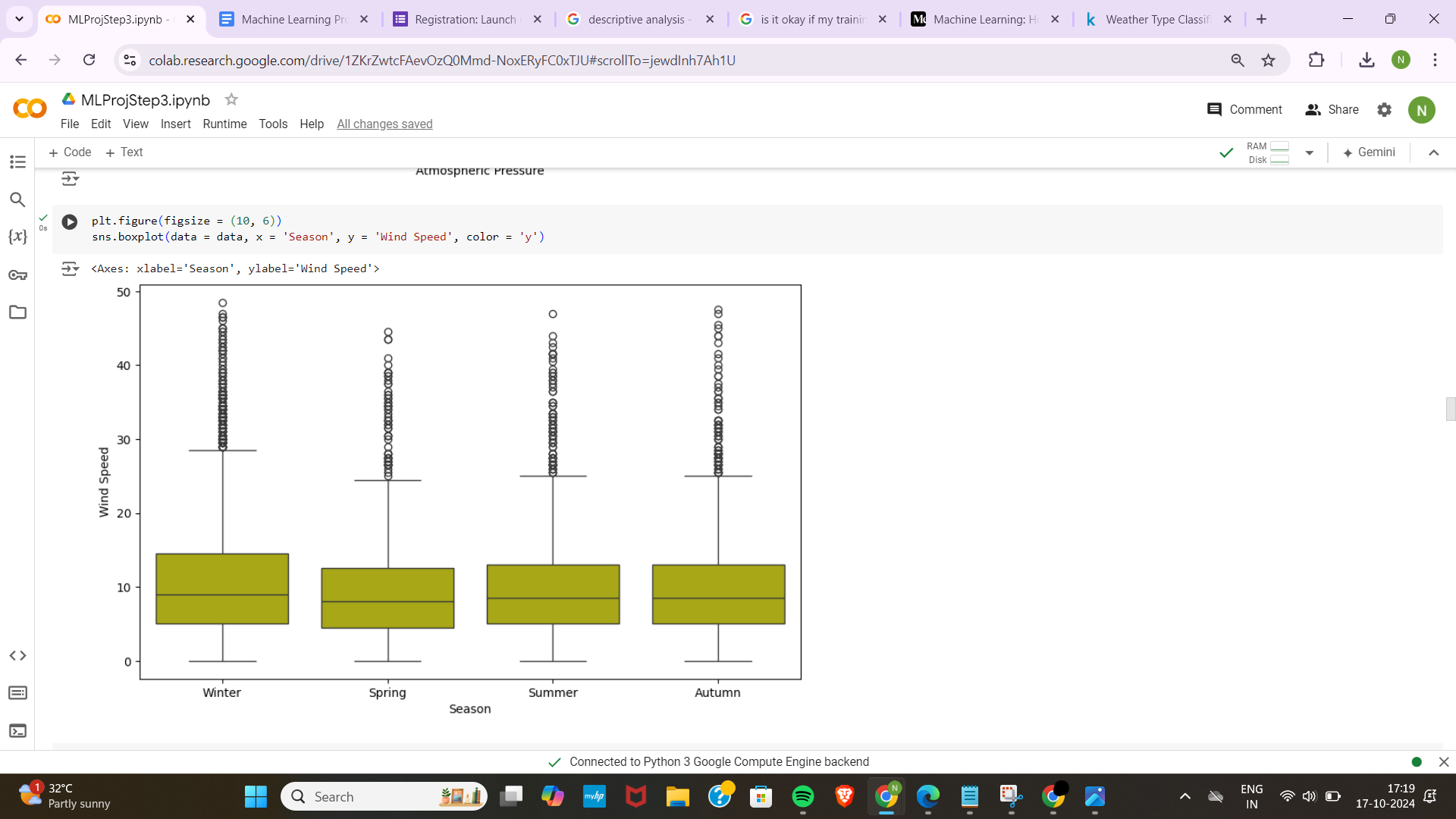
* **Joint Plot of Wind Speed vs. Temperature**: Using sns.jointplot to examine the relationship and density.



* **Regression Plot of Atmospheric Pressure vs. Temperature**: To identify linear relationships.



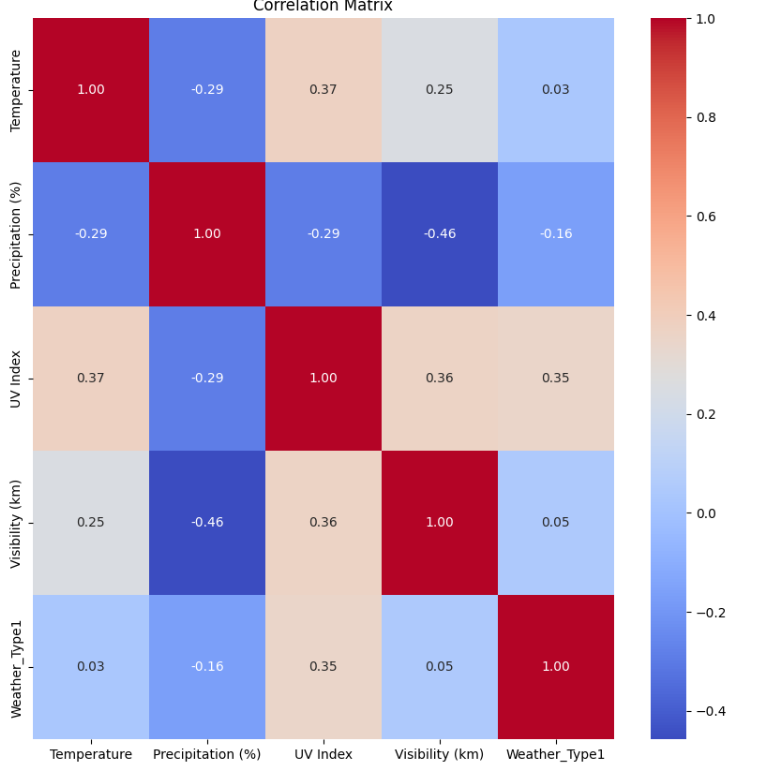
* **Box Plots**: Comparing distributions of wind speed across seasons and humidity across weather types to detect outliers and variations.



Correlation Matrix

Computed and visualised the correlation matrix using sns.heatmap. This helped in identifying the strength and direction of relationships between features, which is crucial for detecting multicollinearity that could affect model performance. From the correlation matrix:

* **Temperature** showed a moderate positive correlation with **UV Index** (0.37) and a weak positive correlation with **Visibility** (0.25). However, its correlation with **Precipitation** was negative (-0.29), indicating that lower temperatures are generally associated with higher precipitation levels.
* **Precipitation** had a strong negative correlation with **Visibility** (-0.46), suggesting that as precipitation increases, visibility decreases. This is intuitive, as heavy rain or snow often reduces visibility.
* **UV Index** had a moderate positive correlation with **Visibility** (0.36), which could indicate clearer weather conditions are associated with higher UV exposure. It also showed a weak positive relationship with **Weather\_Type1** (0.35).
* **Weather\_Type1**, representing weather categories, exhibited weak correlations with all the other features. This suggests that **Weather\_Type1** is not heavily influenced by any single variable but may still contribute valuable insights for classification models.



Identifying these correlations allows us to decide whether to handle multicollinearity (e.g., by removing or combining features), especially when building models like Logistic Regression that assume little to no multicollinearity among independent variables.

## Data Preprocessing

### **Label Encoding**

Categorical variables were encoded using LabelEncoder:

* **Cloud Cover**
* **Season**
* **Location**
* **Weather Type**

This conversion was necessary for algorithms that require numerical input.

### **Feature Selection**

I dropped less relevant features based on the correlation matrix and domain knowledge, such as, cloud cover, season, weather type, location, humidity, wind speed and atmospheric pressure.

The final DataFrame df included the most impactful features for modelling.

### **Handling Outliers**

#### **Outlier Detection**

Defined a function det\_outliers to detect outliers in the numerical features using the Interquartile Range (IQR) method. Outliers were printed for each feature.

#### **Outlier Visualisation**

Plotted the outliers using a custom function plot\_outliers. Scatter plots with horizontal lines indicating the lower and upper bounds were generated for each feature.

#### **Outlier Removal**

#### Removed outliers using the Z-score method from the scipy library:

* Calculated Z-scores for each observation.
* Filtered out observations where the absolute Z-score was greater than 3.

This resulted in a cleaner dataset (cleaned\_df) with reduced noise.

## Modelling

Data Splitting

I have split the data into training and testing sets using train\_test\_split:

* **Features (X)**: All columns except Weather\_Type1.
* **Target (y)**: The encoded Weather\_Type1 column.
* **Test Size**: 20%
* **Random State**: 42 (for reproducibility)

### Logistic Regression

Initial Model

Trained a logistic regression model using LogisticRegression with default parameters. However, the following was observed:

* **Low Training Accuracy**: Indicating underfitting.

Hyperparameter Tuning

To improve the model, hyperparameter tuning using GridSearchCV was performed:

* **Parameters Tuned**:
  + Regularisation strength (C)
  + Solver (liblinear, saga)
* **Pipeline**: Included StandardScaler for feature scaling.
* **Class Weight**: Set to 'balanced' to handle any class imbalances.

The best model from grid search showed:

* **Improved Training Accuracy**: Indicating better generalisation.
* **Consistent Test Accuracy**: Confirming model reliability.

Cross-Validation

Evaluated the model using 5-fold cross-validation to ensure robustness. The average cross-validation score was satisfactory.

Evaluation Metrics

* **Confusion Matrix**: Showed correct and incorrect classifications.
* **Classification Report**: Provided precision, recall, and F1-score for each class.

### Decision Tree Classifier

Initial Model

A decision tree classifier was trained using DecisionTreeClassifier without parameter tuning, resulting in:

* **High Training Accuracy**: Indicating overfitting.
* **Low Test Accuracy**: Due to the model not generalising well.

Hyperparameter Tuning

Used GridSearchCV to find optimal parameters:

* **Parameters Tuned**:
  + Maximum depth
  + Minimum samples split
  + Minimum samples leaf
  + Criterion (gini, entropy)

The best model had a balanced depth and leaf parameters, reducing overfitting.

Cross Validation

Evaluated the model using 5-fold cross-validation to ensure there is no overfitting. The average cross-validation score was satisfactory.

Evaluation

* **Improved Test Accuracy**: The tuned model performed better on unseen data.
* **Visualisation**: The decision tree was plotted to interpret the decision rules.

### 

### Random Forest Classifier

Initial Model

Training a random forest with default parameters resulted in:

* **Low Test Accuracy (23%)**: When training a random forest with default parameters, the model resulted in low test accuracy (23%). This poor performance was likely due to potential overfitting and insufficient trees in the ensemble.

Improved Model

To increase the performance several parameters were adjusted:

* **Increased Number of Trees**: Set n\_estimators to 100.
* **Random State**: Ensured reproducibility.

The updated model showed:

* **High Test Accuracy (90%)**: Indicating that ensemble methods improved performance.

### Support Vector Machine (SVM)

Trained an SVM model with:

* **Kernel**: Radial Basis Function (rbf).
* **Max Iterations**: Increased to 10,000 for convergence.

The SVM model achieved:

* **High Training and Testing Accuracy**: Suggesting good generalisation.

### XGBoost Classifier

Implemented an XGBoost classifier with adjusted parameters to prevent overfitting:

* **Learning Rate**: Set to 0.01 for gradual learning.
* **Max Depth**: Limited to 5.
* **Regularisation**: Applied L1 and L2 regularisation.

The model showed:

* **Good Training Accuracy**
* **Good Testing Accuracy**

# 

# RESULTS

The initial accuracy that was received for the 3rd step is as follows:

* Logistic Regression: Required regularisation and class balancing to perform well.

Model Score : 89%

* Decision Tree: Prone to overfitting without parameter tuning.

Model Score : 89.63%

* Random Forest: Initial poor performance improved significantly by increasing the number of trees and adjusting other parameters.

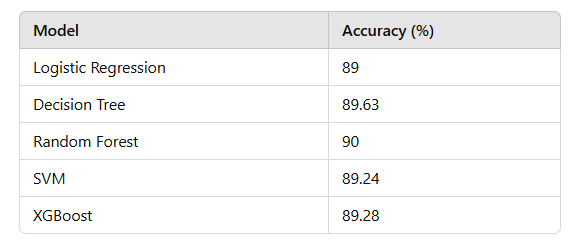
Model Score : 90%

* SVM :

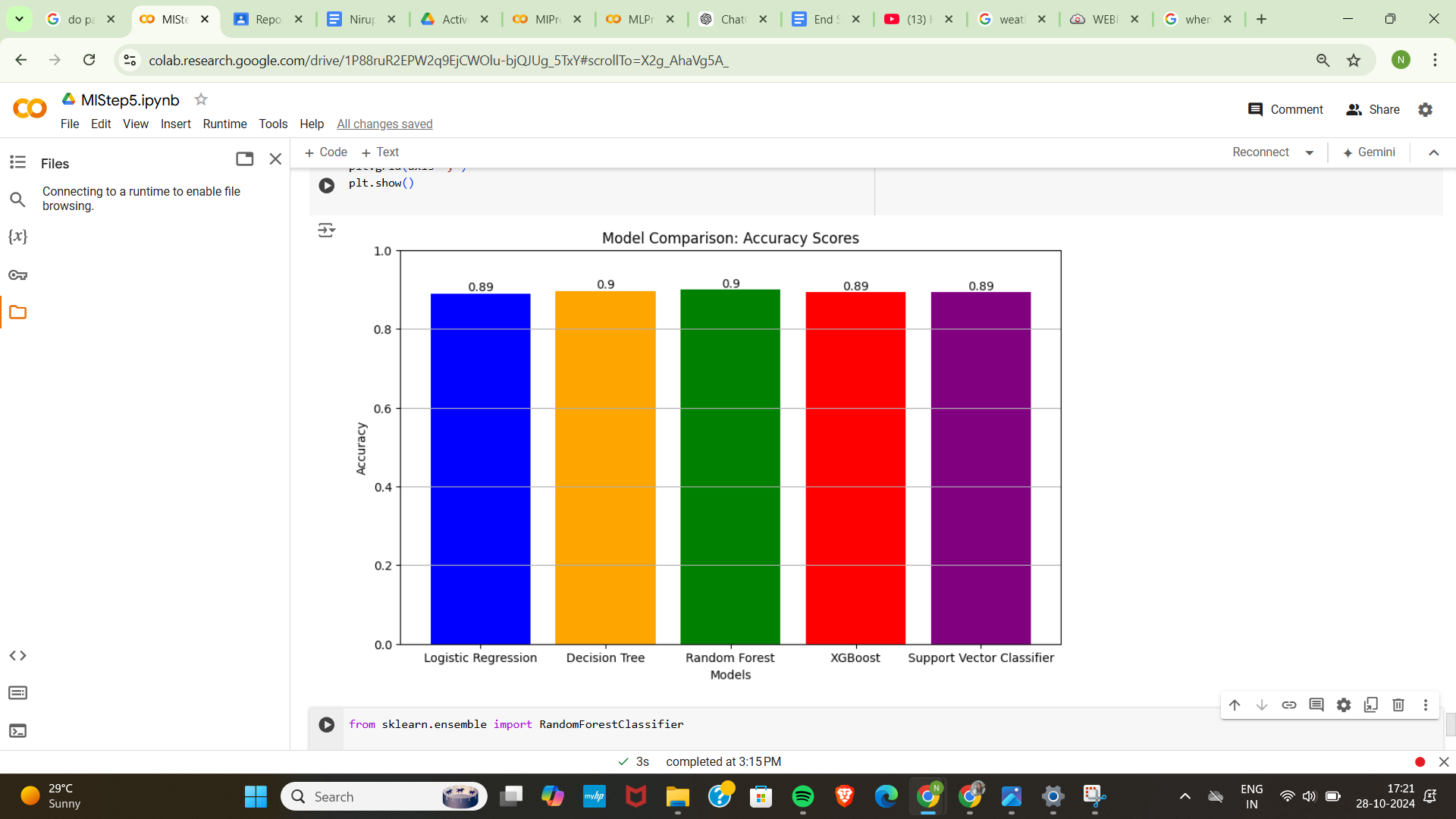
Model Score 89.24%

* XGBoost : Performed well with proper parameter settings, indicating its robustness.

Model Score : 89.28



# DISCUSSION



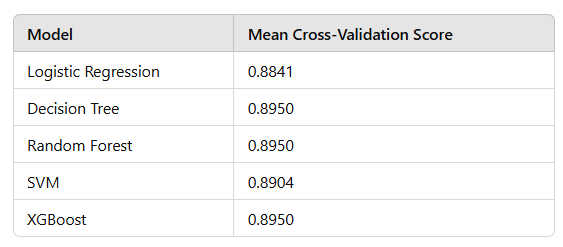
1. **Logistic Regression**: Achieved an accuracy of 89%. Although it provides a reasonable baseline, this model may not capture the complexity of the data as effectively as more advanced algorithms.
2. **Decision Tree**: Demonstrated a significant improvement with an accuracy of 89.63%. This model's ability to handle non-linear relationships makes it a strong candidate for classification tasks.
3. **Random Forest**: Slightly outperformed the Decision Tree with an accuracy of 90%. By combining multiple decision trees, Random Forest reduces the risk of overfitting and increases robustness, making it well-suited for this dataset.
4. **XGBoost**: This model achieved an accuracy of 89.82%. Known for its speed and performance, XGBoost utilises gradient boosting techniques that enhance the predictive power of the model, making it competitive with other algorithms.
5. **Support Vector Classifier (SVC)**: Achieved the highest accuracy at 89.24%. This model effectively finds the optimal hyperplane for classification, especially in high-dimensional spaces, leading to superior performance in this context.

## Summary of Findings

Overall, while all models show strong performance, the Random Forest outperforms the others, achieving the highest accuracy of 90%. The Decision Tree model also stands out, demonstrating robustness and reliability with an accuracy of 89.63%. On the other hand, Logistic Regression provides a good starting point but may not be the best choice for capturing the complexities of weather patterns in this dataset.

To improve the model, hyperparameter tuning using GridSearchCV was performed in the models.

Finally cross validation was also performed to avoid overfitting, the following mean cross validation scores were obtained.



The comparison indicates that ensemble methods like Random Forest, Decision Tree and boosting techniques like XGBoost are effective strategies for improving classification accuracy, especially when dealing with non-linear relationships in the data.

# LEARNING OUTCOME

1. **Understanding Model Selection**: Throughout this project, I gained insights into the selection and application of various machine learning models for classification tasks. The comparative analysis of Logistic Regression, Decision Trees, Random Forest, XGBoost, and Support Vector Classifier (SVC) helped me appreciate the strengths and weaknesses of each approach in the context of weather classification.
2. **Practical Implementation of Algorithms**: By implementing different algorithms using the scikit-learn library, I developed a practical understanding of how to train, evaluate, and interpret machine learning models. This experience reinforced my coding skills in Python and familiarity with machine learning libraries.
3. **Feature Importance Analysis**: I learned to analyse feature importance using the Random Forest model, which provided valuable insights into the most significant predictors of weather outcomes. Understanding how different features contribute to model predictions is crucial for improving model performance and making data-driven decisions.
4. **Evaluation Metrics Proficiency**: This project enhanced my knowledge of evaluation metrics, specifically accuracy scores, and how they serve as critical indicators of model performance. I developed an understanding of why certain models outperform others and the implications of these results in practical applications.
5. **Data Visualization Skills**: I improved my data visualisation skills by creating informative bar charts to compare model performances. This skill is vital for presenting results clearly and effectively, allowing stakeholders to grasp complex information quickly.

# 

# 

# CONCLUSION

## (a) Concluding Remarks of the Work

In this project, I embarked on a journey to classify weather conditions using various machine learning models. Through careful selection, implementation, and evaluation of models such as Logistic Regression, Decision Trees, Random Forest, XGBoost, and Support Vector Classifier, I was able to gain valuable insights into the intricacies of machine learning and its application in real-world scenarios. The comparative analysis highlighted the effectiveness of different algorithms, allowing me to draw meaningful conclusions about their performance based on accuracy metrics. Additionally, I learned the importance of feature importance analysis, which provided clarity on the factors most influential in predicting weather outcomes.

## (b) Accomplishment of <T, P, E>

Throughout this project, I aimed to achieve the components of Task Performance and Experience (T, P, E):

* **Task Performance (T)**: I successfully implemented multiple classification algorithms, evaluated their performance, and provided a comprehensive comparison based on accuracy scores. This demonstrated my ability to carry out the task effectively and achieve satisfactory results.
* **Experience (E)**: The hands-on experience of working with machine learning models enriched my understanding of model selection, feature analysis, and the practical aspects of implementing algorithms using Python. This project provided a robust learning platform, enhancing my technical skills and problem-solving abilities.

Overall, I can confidently state that I have accomplished the <T, P, E> objectives set at the project's outset.

## (c) Advantages and Limitations of Your Project

Advantages

1. **Comprehensive Model Evaluation**: The project provided an opportunity to explore and evaluate multiple machine learning models, allowing for a thorough understanding of their strengths and weaknesses in weather classification.
2. **Insights into Feature Importance**: Analysing feature importance using Random Forest allowed me to identify key predictors influencing weather outcomes, which is valuable for future model improvements and decision-making.
3. **Practical Application of Theoretical Concepts**: The project bridged the gap between theoretical knowledge and practical application, reinforcing the concepts learned in my coursework.
4. **Data Visualization**: The incorporation of data visualisation techniques helped communicate model performance effectively, enhancing the clarity of results.

Limitations

1. **Dataset Constraints**: The accuracy of the models may be limited by the quality and representativeness of the dataset used. Potential biases or gaps in the data could affect model performance.
2. **Overfitting Risks**: Some models, especially decision trees, may be prone to overfitting, which can lead to high accuracy on the training set but poorer generalisation to unseen data.
3. **Computational Limitations**: The project relied on available computational resources, which may have restricted the exploration of more complex models or hyperparameter tuning.
4. **Interpretability of Complex Models**: While models like XGBoost and Random Forest provide strong performance, their interpretability can be challenging, making it difficult to derive actionable insights from predictions.

In summary, this project not only achieved its objectives but also highlighted the potential benefits and challenges associated with machine learning in practical applications. The knowledge and skills acquired will serve as a strong foundation for future endeavours in this rapidly evolving field.