Introduction

In this project, we have the ambition to analyze a meteorological prediction solution from two models based on artificial intelligence. To do this, we will use an already existing database with 1461 days of meteorological analysis. This database is publicly available on the Kaggle site.
For this analysis, we will compare two machine learning methods which are Decision Tree and Support Machine Vector (SVM).

Why did you choose this subject

First of all, weather forecasts have a wide range and impact on many aspects of daily life. By knowing the upcoming weather conditions, we can plan our activities accordingly and make important decisions, such as whether to bring an umbrella or whether it is wise to go outside in the event of a storm. By using AI to improve the accuracy and timeliness of weather forecasts, we can help people better plan their day and make more informed decisions. AI can quickly process large amounts of weather data and provide forecasts at varying levels of accuracy, which can be useful for specific applications such as flight planning or agriculture.

In our analysis, we will use two models based on artificial intelligence to predict what type of weather will be like, more precisely to which category the future event will belong. Our categories will be: drizzle, rain, sun, snow and fog. Therefore, since we categorize our data and results, a classification model will be particularly suitable. Indeed, this one makes it possible to predict to which category an element will belong based on past data. Using a classification model, we can train our model on past weather data by telling it which category each event belonged to.

Additionally, classification models are often easier to set up and understand than other prediction models. They are also generally less sensitive to data errors than regression models, numerical models, which can be useful in the case of weather prediction, where data can sometimes be incomplete or imperfect.

There are other types of machine learning models that could also be used to predict weather, such as regression models or clustering models. However, these models are generally less suitable for this type of task, since they are designed to predict continuous values (like temperature or atmospheric pressure) rather than discrete categories (like sunshine or rain).

In our analysis, we will therefore focus on an approach by classification.

The database we will use is publicly available.

Classification models are machine learning algorithms used to predict whether a sample belongs to a certain class based on its characteristics. There are many different classification models, such as neural networks, k-nearest neighbors, etc. To solve a problem of weather forecast by categories, we will choose two popular models in this field, which we have also seen in progress and which seemed to us to be the most favorable and interesting for solving this problem. These two models are Support Vector Machine (SVM) and Decision Tree.

Materials and methods

Dataset

Source: https://www.kaggle.com/code/kamlesh0228/weather-prediction/data

This dataset contains comprehensive weather data for the city of Seattle from January 1, 2012 through December 31, 2015.

The number of days: 1461

Our database contains for each specific date:

- precipitation rate: All forms in which water falls on the land surface and open water bodies as rain, sleet, snow, hail, or drizzle
- maximum temperature: in degrees
- minimum temperature: in degrees
- wind speed: in km/h
- weather category: {drizzle, rain, sun, snow, fog}

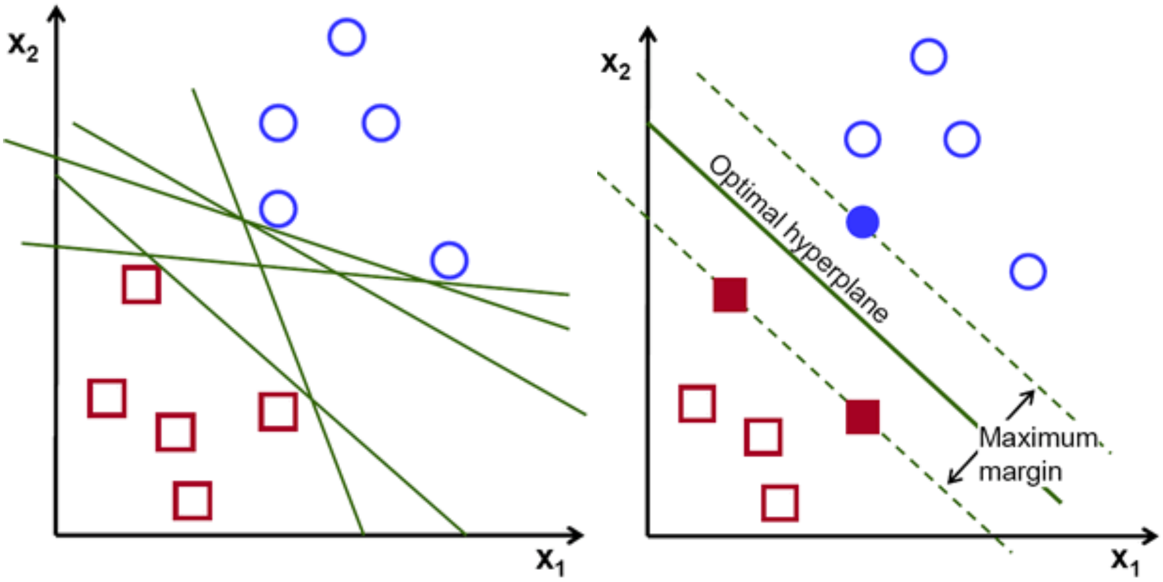
Models

For the Support Machine Vector model, we used the following python library:
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

For the Decision Tree model, we used the following python library:
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Support Machine Vector (SVM)

The goal of the SVM algorithm is to find a hyperplane that distinctly separates data points in an N-dimensional space (where N is the number of features).



Our goal is to find a hyperplane that optimally separates the two classes of data points by maximizing the distance between the closest points of the two classes, known as support vectors. This maximum distance, called the margin, builds confidence in the model's ability to correctly classify new data points in the future. There are several possible hyperplanes that could be chosen to separate the classes of data points.

Hyperplanes

Hyperplanes are boundaries used to classify data points based on their position relative to those boundaries. Points on either side of the hyperplane can be assigned to different classes. The dimension of the hyperplane depends on the number of features of the input data. If the number of features is 2, the hyperplane is a line. If the number of features is 3, the hyperplane is a two-dimensional plane.

Support Vectors

Support vectors are data points that are close to the hyperplane and influence its position and orientation. By using these support vectors, we maximize the margin of the classifier. By removing the support vectors, we change the position of the hyperplane. These are the points that allow us to build our SVM.

Linear SMV

The linear SVM algorithm can be used to solve binary classification problems, i.e. to predict whether an observation belongs to one class or another. For example, in the field of meteorology, we can use a linear SVM to predict whether tomorrow will be sunny or rainy depending on different meteorological variables, such as temperature, wind speed…

To use a linear SVM to predict weather, we first need to collect a historical data set that includes observations of different weather variables along with their corresponding weather state (sun or rain). We can then train a linear SVM on this data using the gradient descent algorithm to minimize the prediction error.

Once trained, our linear SVM will be able to predict the weather state using a spatially linear separation hyperplane of weather variables. Observations that fall on the other side of the hyperplane will be predicted as belonging to the other class. For example, if we trained our linear SVM to predict whether it will be sunny or rainy, observations that fall on the other side of the hyperplane will be predicted as sunny, while those that fall on the other side will be predicted as rainy.

There are several common kernel functions that can be used to optimize computations. The three common kernel functions are: sigmoid, RBF, polynomial.

- **Polynomial:** $K(x,x') = (x \cdot x' + c)^q$
- **RBF (radial basis function):** $K(x,x') = e^{-\frac{||x-x'||^2}{2\sigma^2}}$
- **Sigmoide:** $K(x,x') = tanh(\alpha x \cdot x' - b)$

Decision Trees

Decision Trees (Decision Trees) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

Decision trees are a form of predictive modeling, helping to map the different decisions or solutions to a given outcome. Decision trees are made up of different nodes. The root node is the start of the decision tree, which is usually the whole dataset within machine learning. Leaf nodes are the endpoint of a branch, or the final output of a series of decisions. The decision tree won’t branch any further from a leaf node. With decision trees in machine learning, the features of the data are internal nodes and the outcome is the leaf node.

They are an approach used in supervised machine learning, a technique which uses labeled input and output datasets to train models. The approach is used mainly to solve classification problems, which is the use of a model to categorize or classify an object.

Classification problems are the most common use of decision trees in machine learning. It is a supervised machine learning problem, in which the model is trained to classify whether data is a part of a known object class. Models are trained to assign class labels to processed data. The classes are learned by the model through processing labeled training data in the training part of the machine learning model lifecycle.

To solve a classification problem, a model must understand the features that categorize a datapoint into the different class labels. In practice, a classification problem can occur in a range of settings. Examples may include the classification of documents, image recognition software, or email spam detection.

A classification tree is a way of structuring a model to classify objects or data. The leaves or endpoint of the branches in a classification tree are the class labels, the point at which the branches stop splitting. The classification tree is generated incrementally, with the overall dataset being broken down into smaller subsets. It is used when the target variables are discrete or categorical, with branching happening usually through binary partitioning. For example, each node may branch on a yes or no answer. Classification trees are used when the target variable is categorical, or can be given a specific category such as yes or no. The endpoint of each branch is one category.

How to choose the best attribute at each node

While there are multiple ways to select the best attribute at each node, two methods, information gain and Gini impurity, act as popular splitting criterion for decision tree models. They help to evaluate the quality of each test condition and how well it will be able to classify samples into a class.

Entropy and Information Gain

It's difficult to explain information gain without first discussing entropy. Entropy is a concept that stems from information theory, which measures the impurity of the sample values. It is defined with by the following formula, where:

$$Entropy(S) = -\sum_{c \in C} p(c) log_2 p(c)$$

- S represents the data set that entropy is calculated
- c represents the classes in set, S
- p(c) represents the proportion of data points that belong to class c to the number of total data points in set, S

Entropy values can fall between 0 and 1. If all samples in the data set, S, belong to one class, then entropy will equal zero. If half of the samples are classified as one class and the other half are in another class, entropy will be at its highest at 1. In order to select the best feature to split on and find the optimal decision tree, the attribute with the smallest amount of entropy should be used. Information gain represents the difference in entropy before and after a split on a given attribute. The attribute with the highest information gain will produce the best split as it's doing the best job at classifying the training data according to its target classification. Information gain is usually represented with the following formula, where:

$$Information\ Gain(S, a) = Entropy(S) - \sum_{v \in vcalues(a)} \frac{|Sv|}{|S|} Entropy(Sv)$$

- a represents a specific attribute or class label

- Entropy(S) is the entropy of dataset, S
- |Sv|/ |S| represents the proportion of the values in Sv to the number of values in dataset, S
- Entropy(Sv) is the entropy of dataset, Sv

Then, repeat the calculation for information gain for each attribute in the table above, and select the attribute with the highest information gain to be the first split point in the decision tree. In this case, outlook produces the highest information gain. From there, the process is repeated for each subtree.

Comparison of the two methods

The SVM model uses a construction method based on support vectors, which consists of finding a hyperplane that separates the different classes with the largest possible margins. The Decision Tree model, on the other hand, uses a construction method based on decision trees, which consists of repetitively dividing the data into subsets using specific selection criteria.

On the other hand, in general, SVM models are more complex to build than Decision Tree models because they require more extensive parameter optimization. However, Decision Tree models can be more complex to interpret and understand because they can have complex decision tree structures.

Finally, SVM models generally perform better in terms of accuracy when used to solve classification problems. However, they may be less suitable for large datasets or datasets with lots of dimensions, as they can be slow to train and predict. Decision Tree models, on the other hand, are generally faster to train and predict, but they may perform worse in terms of accuracy than SVM models in some situations.

Presentation of the system

Link to the project on Github: https://github.com/Gadnadj/Weather-Prediction

SVM

1. The first step is to import all the packages that we will use
2. Read the database from a pre-downloaded CSV file
3. Parse and remove all irrelevant data (like database title) and table rows where null data is found
4. Divide data into dependent and independent components
5. Data balancing using SMOTE for more balanced data and better performance
6. Splitting Data into train and test, x_test, y_test, x_train, y_train (test to evaluate the performance of a model once trained, and TRAIN to adjust model parameters to minimize prediction error)
7. Scale the Data (training and launch) to normalize the data so that they all have roughly the same scale
8. Define function for Confusion Matrix in the form of Heat map
9. Define function for comparing different types of precision (precision, recall, f1-score…)
10. Apply Classification Models VMS
11. After launching the program, displaying the confusion matrix table
12. print the precision table
13. Print the accuracy of the SVM

Decision Tree

1. The first step is to import all the packages that we will use
2. Read the database from a pre-downloaded CSV file
3. Parse and remove all irrelevant data (like database title) and table rows where null data is found
4. Divide data into dependent and independent components
5. Data balancing using SMOTE for more balanced data and better performance
6. Splitting Data into train and test, x_test, y_test, x_train, y_train (test to evaluate the performance of a model once trained, and TRAIN to adjust model parameters to minimize prediction error)
7. Scale the Data (training and launch) to normalize the data so that they all have roughly the same scale
8. Define function for Confusion Matrix in the form of Heat map

9. Define function for comparing different types of precision (precision, recall, f1-score…)
10. Apply Classification Models Decision Tree
11. After launching the program, displaying the confusion matrix table
12. print the precision table
13. Print the accuracy of the Decision Tree

Programming dilemmas/problems we encountered

- Macbook computer with Mac OS Monterey as the operating system, does not include the pandas and sklearn libraries. We had to find another computer to be able to continue the project.
- The codes the algorithm that we found were difficult to understand, it earned us a lot of research to understand these

Results

Explanations of the different types of results

Accuracy: fraction of positive cases produced that are actually positive. It is calculated as the ratio of true positives to the total number of positive predictions made by the model. The higher the accuracy value, the more accurate the model is in its positive predictions.

Precision = (Number of true positives) / (Number of true positives + Number of false positives)

Recall: fraction of actual positive cases that are correctly predicted by the model. It is calculated as the ratio of true positives to the total number of true positives in the data set. The higher the recall value, the better the model is able to correctly detect a large number of positive cases in the data set.

Recall = true_positives / (true_positives + false_negatives)

f1-score: it is the harmonic mean of precision and recall. It is calculated as the harmonic mean of the precision and recall values, with a higher value indicating a better balance between precision and recall.

F1 = 2 * (precision * recall) / (precision + recall)

Support: This is the number of instances in the dataset that belong to a particular class. The support is used to calculate the weighted average of the precision, recall, and F1 score, where the weight of each metric is the support of the corresponding class.

Accuracy: ratio of the number of correct predictions made by the model to the total number of predictions made.

Accuracy = (number of correct predictions) / (total number of predictions)

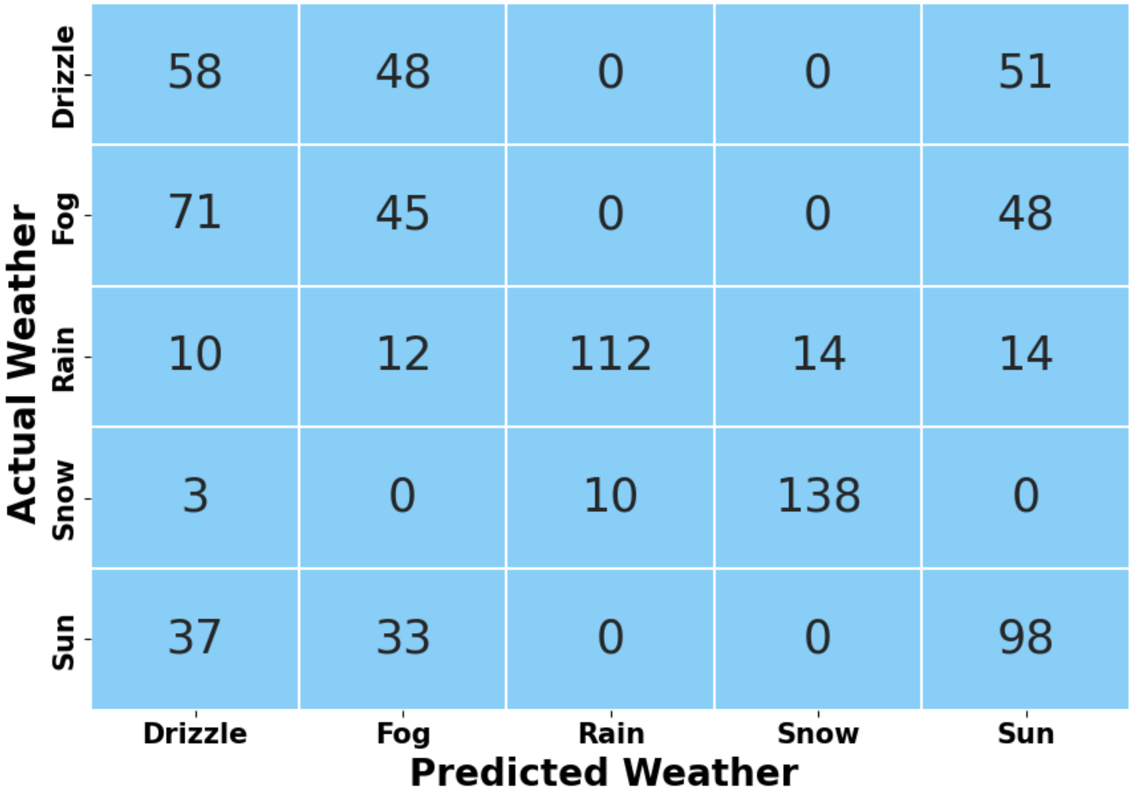
Macro Avg: method of calculating the average performance of a classification model across the classes of a multiclass classification model. It calculates the average of metric scores for each class, regardless of class balance.

Macro avg = (metric score for class 1 + metric score for class 2 + ... + metric score for class n) / n

Weighted Avg: Another method of averaging the performance of a classification model across classes in a multiclass classification model. It calculates the average of the metric scores for each class, taking into account the class balance.

Weighted avg = (metric score for class 1 * weight for class 1) + (metric score for class 2 * weight for class 2) + ... + (metric score for class n * weight for class n)

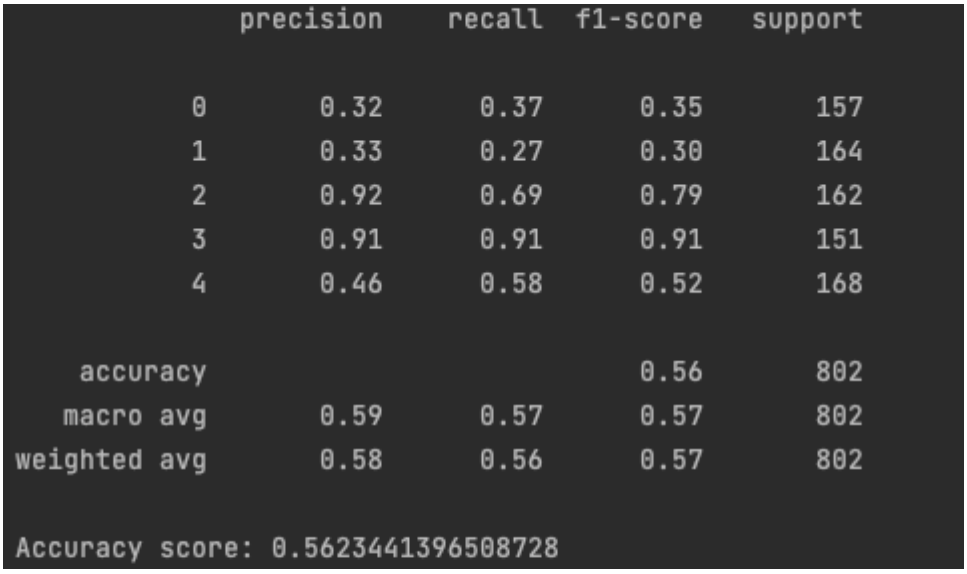
For the Drizzle class the model correctly predicted 65 drizzles out of 157 predictions



	Drizzle	Fog	Rain	Snow	Sun
Drizzle	58	48	0	0	51
Fog	71	45	0	0	48
Rain	10	12	112	14	14
Snow	3	0	10	138	0
Sun	37	33	0	0	98

SVM

- For the Drizzle class the model correctly predicted 58 fog out of 157 predictions
- For the Fog class the model correctly predicted 45 fog out of 164 predictions
- For the Rain class the model correctly predicted 112 rain out of 162 predictions
- For the Snow class the model correctly predicted 138 snow out of 151 predictions
- For the Sun class the model correctly predicted 98 sun out of 168 predictions



```
              precision    recall  f1-score   support

           0       0.32      0.37      0.35       157
           1       0.33      0.27      0.30       164
           2       0.92      0.69      0.79       162
           3       0.91      0.91      0.91       151
           4       0.46      0.58      0.52       168

    accuracy                           0.56       802
   macro avg       0.59      0.57      0.57       802
weighted avg       0.58      0.56      0.57       802

Accuracy score: 0.5623441396508728
```

0 - Represent Drizzle

1- Represent Fog

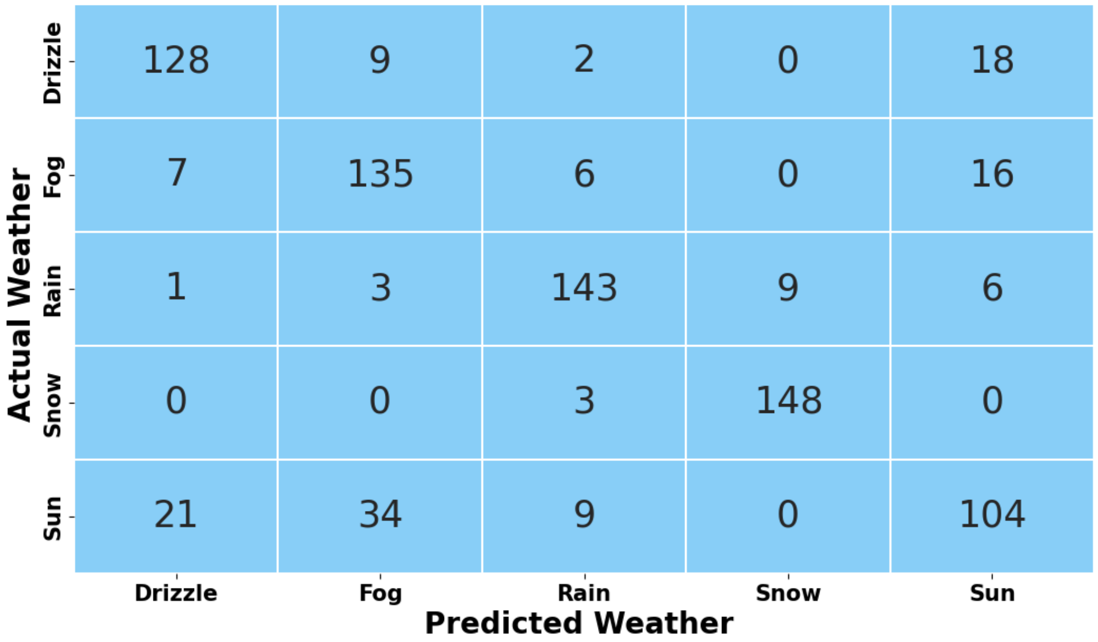
2- Represent Rain

3- Represent Snow

4- Represent Sun

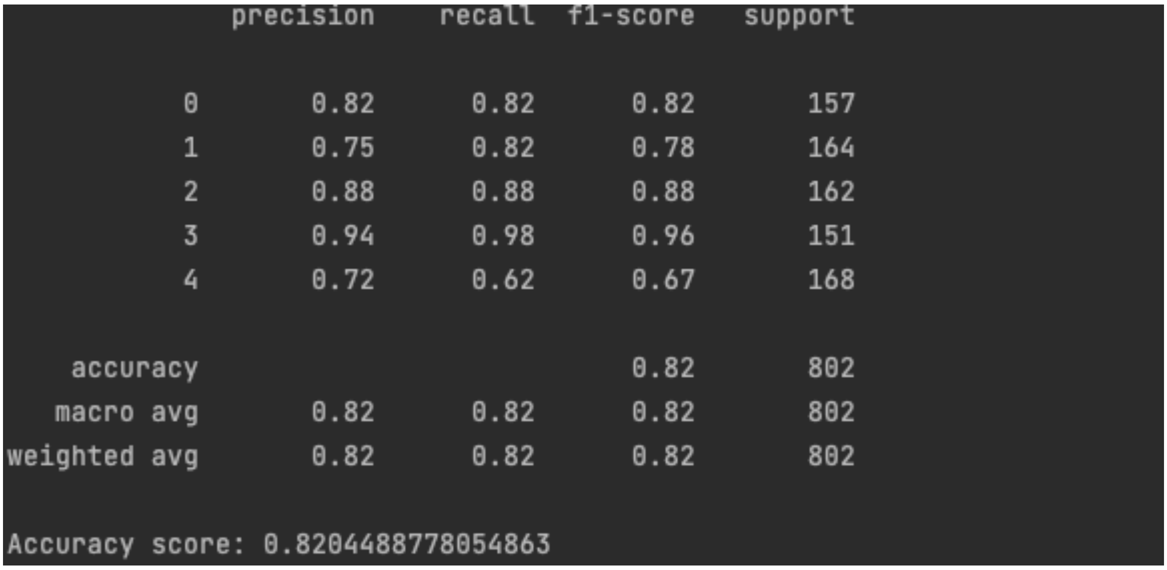
The accuracy rate is 0.562.

Decision Tree



Actual Weather \ Predicted Weather	Drizzle	Fog	Rain	Snow	Sun
Drizzle	128	9	2	0	18
Fog	7	135	6	0	16
Rain	1	3	143	9	6
Snow	0	0	3	148	0
Sun	21	34	9	0	104

- For the Drizzle class the model correctly predicted 128 drizzles out of 157 predictions
- For the Fog class the model correctly predicted 135 fog out of 164 predictions
- For the Rain class the model correctly predicted 143 rain out of 162 predictions
- For the Snow class the model correctly predicted 148 snow out of 151 predictions
- For the Sun class the model correctly predicted 104 sun out of 168 predictions



```
              precision    recall  f1-score   support

           0       0.82      0.82      0.82       157
           1       0.75      0.82      0.78       164
           2       0.88      0.88      0.88       162
           3       0.94      0.98      0.96       151
           4       0.72      0.62      0.67       168

    accuracy                           0.82       802
   macro avg       0.82      0.82      0.82       802
weighted avg       0.82      0.82      0.82       802

Accuracy score: 0.8204488778054863
```

0 - Represent Drizzle

1- Represent Fog

2- Represent Rain

3- Represent Snow

4- Represent Sun

The accuracy rate is 0.82.

We can see thanks to the github link below an image representing the decision tree (download the image if it is not displayable due to the size) https://github.com/Gadnadj/Weather-Prediction/blob/master/decison_tree.png

Comparison of results

Difference

As part of the performance evaluation of a weather prediction model, we used two different models: SVM (Support Vector Machine) and a decision tree. We compared the confusion matrices of these two models to assess their performance.

The accuracy rate of the SVM model was 0.56, which means that it correctly predicted about 56% of the predictions. The accuracy rate of the decision tree model was 0.80, which means that it correctly predicted about 80% of the predictions. Looking at the confusion matrices themselves, we found that the decision tree model produced a significantly higher number of correct predictions in each category, which explains why its accuracy rate is higher.

The Recall measures the proportion of real cases detected among all real cases. In the first table, the recall values range from 0.41 for drizzle to 0.69 for rain, while in the second table they range from 0.76 for drizzle to 0.88 for rain. This means that the Decision Tree model has higher recall values than the SVM model for each type of weather.

In the first table (SVM), the f1-score values range from 0.36 for drizzle to 0.79 for snow, while in the second table they range from 0.77 for drizzle to 0.89 for the snow. This means that the Decision Tree model has a better performance in terms of precision and recall than the SVM model, for each type of weather.

The differences between the two results are critical and important because they indicate that the decision tree model is significantly superior to the SVM model in predicting weather forecasts. This superiority is visible through several performance metrics, such as accuracy rate, accuracy for each class, recall for each class, and F1-score for each class.

This precision is important because it can have consequences on the decisions made based on these forecasts, such as travel planning or the organization of external events.

Similarities

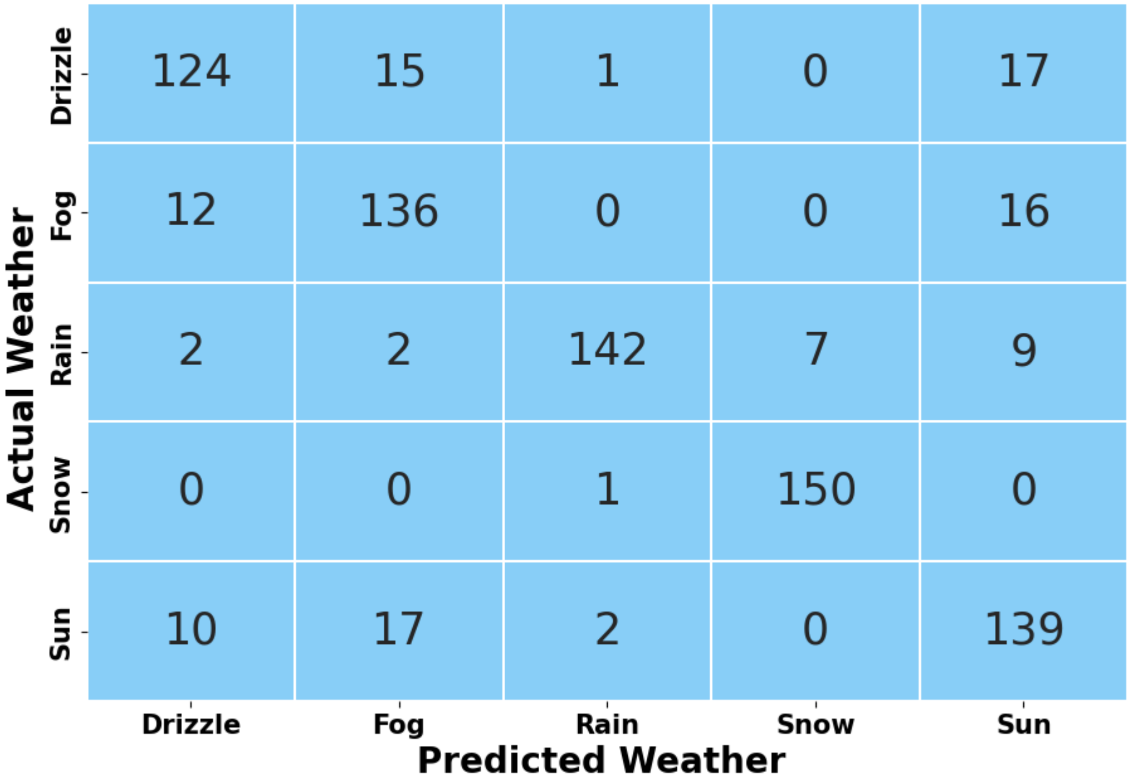
Both results show high accuracy rates for most weather classes, with values ranging from 0.69 to 0.95 for the SVM model and from 0.69 to 0.97 for the tree model. of decision.

Optimization of results

In order to optimize our results, we used the XGBoost approach. XGBoost (eXtreme Gradient Boosting) is an implementation of the gradient boosting algorithm that was developed to be very fast and efficient, which can be advantageous in an area like weather forecasting where data is often large. The gradient boosting algorithm is an advanced compilation technique that allows multiple models to be combined to create a more robust model. It involves training many models iteratively, using errors from previous models to improve the performance of the next model.

XGBoost adds many improvements to the gradient boosting algorithm, including advanced missing data handling, leaf count optimization, and pattern regularization. It also supports many advanced features, like parallelization and memory management.

The results obtained are more satisfactory and more precise.



	Drizzle	Fog	Rain	Snow	Sun
Drizzle	124	15	1	0	17
Fog	12	136	0	0	16
Rain	2	2	142	7	9
Snow	0	0	1	150	0
Sun	10	17	2	0	139

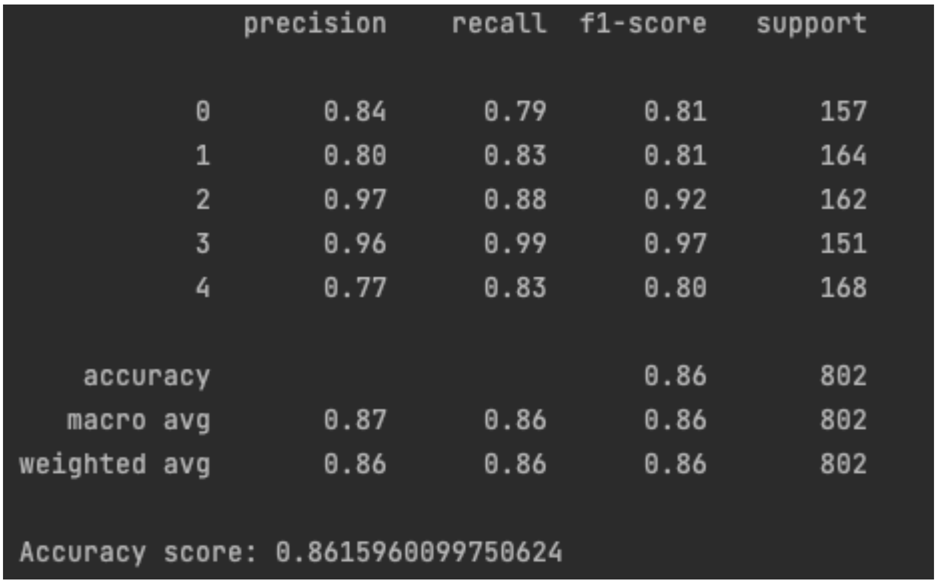
0 - Represent Drizzle

1- Represent Fog

2- Represent Rain

3- Represent Snow

4- Represent Sun

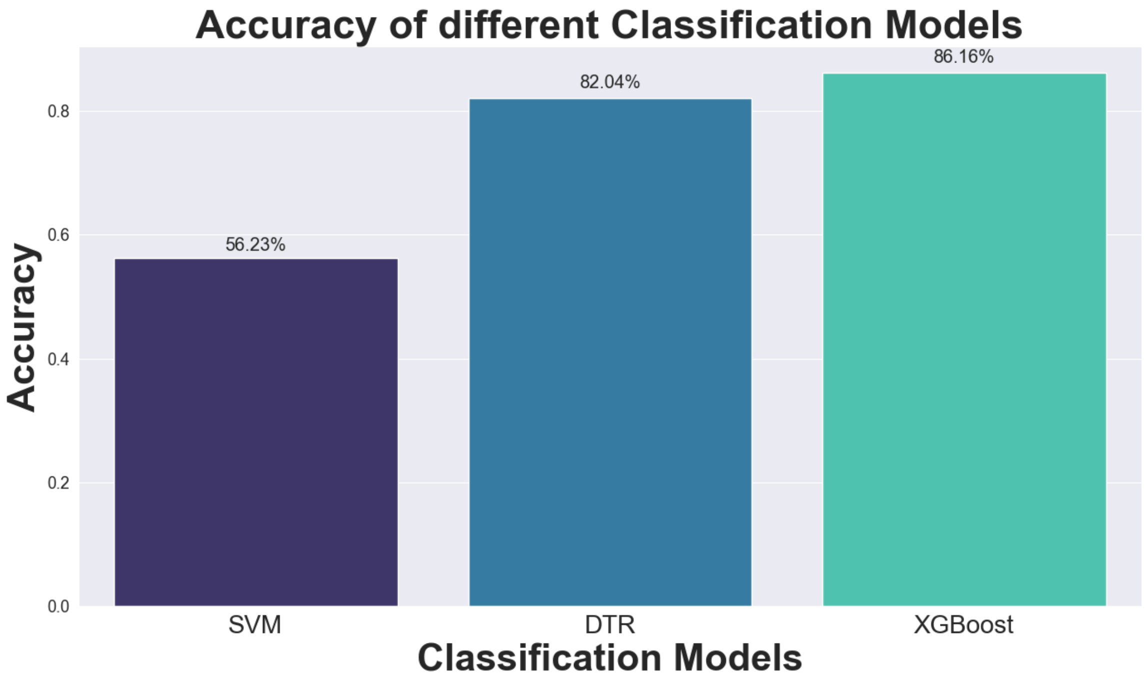


```
              precision    recall  f1-score   support

           0       0.84      0.79      0.81       157
           1       0.80      0.83      0.81       164
           2       0.97      0.88      0.92       162
           3       0.96      0.99      0.97       151
           4       0.77      0.83      0.80       168

    accuracy                           0.86       802
   macro avg       0.87      0.86      0.86       802
weighted avg       0.86      0.86      0.86       802

Accuracy score: 0.8615960099750624
```

- For the Drizzle class the model correctly predicted 124 fog out of 157 predictions
- For the Fog class the model correctly predicted 136 fog out of 164 predictions
- For the Rain class the model correctly predicted 142 rain out of 162 predictions
- For the Snow class the model correctly predicted 150 snow out of 151 predictions
- For the Sun class the model correctly predicted 139 sun out of 168 predictions

The accuracy rate is 0.86

Here is a graphical comparison of the three models:



We can see that the use of advanced compilation techniques, such as XGBoost, gradient boosting or, for example, Random Forest, can improve the accuracy of forecasts. These approaches allow multiple models to be combined to create a more robust model.

AI and algorithm knowledge gained from project

We learned how to use SVM algorithms and decision trees to predict the weather using historical data as training data. Moreover, we learned to evaluate the efficiency of these models using performance measures such as accuracy, recall and f1-score.

As part of our project on weather forecasting, we had the opportunity to expand our knowledge of python. Indeed, the Python library scikit-learn (sklearn) was unknown to us before this project and we were able to discover the use of machine learning algorithms such as SVMs and DTs, as well as methods to evaluate and optimize their performance. . We also learned how to work with weather data and how to use visualizations to understand the results of our models. Finally, we developed our programming skills using python to implement these algorithms and work with the data.

We have learned that artificial intelligence cannot be considered infallible, because there are possibilities for it to make errors or produce inaccurate results, as in the case of the SVM model whose accuracy is only 56%. We have gained in-depth knowledge of machine learning concepts and techniques, and we are able to use these skills to solve new problems. We understand that the quality of training data and its relevance have a huge influence on the effectiveness of machine learning models. We also learned that artificial intelligence can be complex to understand and explain, especially when based on complex models.

Conclusion

In conclusion of this project, despite a general idea perceived by the population, the use of algorithms to predict events will never be exact even if the data used is unlimited. The uncertainties of certain data, the lack of data or even unpredictable external elements can at any time generate a false prediction. In the field of weather forecasting, human activities can also have an impact on the results obtained and these are complicated to take into account. However, the use of AI can be useful to improve the accuracy of weather forecasts by using models such as SVMs and DTs to process data and evaluate model performance. By using these tools, we were able to make weather forecasts and thus help people make informed decisions. Although AI cannot predict the future with absolute accuracy, it can be a valuable tool for improving our understanding and decision-making in a wide range of areas, including weather forecasting.

References

1. https://www.kaggle.com/code/kamlesh0228/weather-prediction

2. Elia Georgiana Petre, Universitatea Petrol-Gaze din Ploieşti, A Decision Tree for Weather Prediction, 2009

3. Amy McGovern, Kimberly L. Elmore, David John Gagne II, Sue Ellen Haupt, Christopher D. Karstens, Ryan Lagerquist, Travis Smith, and John K. Williams, Using artificial intelligence to improve real-time decision-making for high-impact weather, 2017

4. Sayali D. Jadhav1 , H. P. Channe, Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques, Department of Computer Engineering, Pune Institute of Computer Technology, Savitribai Phule Pune University, Pune, India, 2016