BRAC UNIVERSITY

Inspiring Excellence

CSE422 : Artificial Intelligence
Project Report
Project Title : Rain or snow prediction in Hungary

Group: 13, Lab Section : 09, Summer 2023	
ID	**Name**
21301610	Shihab Muhtasim
21301648	Raiyan Wasi Siddiky

Table of Contents

Introduction: Weather Prediction for Rain or Snow in Hungary

Weather prediction is an extremely critical aspect of modern life, influencing various sectors ranging from agriculture to transportation. This is especially true in the case of extreme weather conditions such as rain or snow, which can bring commercial life to a halt. It is therefore essential to have accurate weather forecasts, as they can help individuals and organizations make informed decisions and plan their activities accordingly. This project, "Weather Prediction: Rain or Snow in Hungary," aims to address the challenge of predicting whether it will rain or snow in Hungary based on various meteorological variables.

Project Objectives:
The primary objective of this project is to develop a predictive model that can forecast whether a specific location in Hungary will experience rain or snow within a given time frame. By utilizing historical weather data and applying machine learning techniques, we seek to create a model that offers reliable predictions with a high degree of accuracy. This project will contribute to enhancing the precision of weather forecasts, providing valuable insights for individuals, businesses, and authorities in Hungary.

Problem Statement:
Hungary, like many regions, experiences unpredictable weather conditions that can have significant implications for day-to-day activities and planning. Traditional forecasting methods may fall short in providing accurate predictions for rapidly changing weather patterns, particularly in cases of rain and snow. This project addresses the challenge of improving the accuracy of weather predictions for rain and snow, enabling residents, farmers, transportation services, and other stakeholders to make well-informed choices based on reliable forecasts.

Motivation:
The motivation behind this project lies in the potential positive impact on individuals' lives and societal activities. Accurate weather predictions can empower people to optimize their schedules, mitigate risks associated with adverse weather, and even enhance agricultural practices. Additionally, sectors such as transportation and emergency services can benefit from timely information about rain and snowfall, ensuring public safety and efficient resource allocation. By harnessing the power of data and machine learning, this project aims to contribute to the well-being and productivity of Hungary's population.

Dataset description

Source: Kaggle
Link: weather prediction ,regression , neural model | Kaggle
Reference: Salma Maamouri

Dataset description

1. **How many features:**

There are 10 features initially. Such as: Summary , Apparent Temperature , Humidity , Wind Speed (km/h) ,Wind Bearing (degrees), Visibility (km), Loud Cover , Pressure (millibars), Daily Summary, Temperature (C) .
1 class: Precip type with two unique values rain, snow
After pre- processing: 35 features

2. **Is this a classification or regression problem ? Why do you think so?**

In this project, we are addressing a classification problem. Classification involves the task of predicting categories or classes based on given input features. In the context of weather prediction in Hungary, our aim is to predict whether the weather conditions will lead to the occurrence of rain or snow. These outcomes, "rain" and "snow," are distinct categories that we are trying to predict based on various weather-related features. Therefore, the nature of our task, which involves predicting a categorical label (rain or snow), signifies that our project falls within the domain of classification problems.

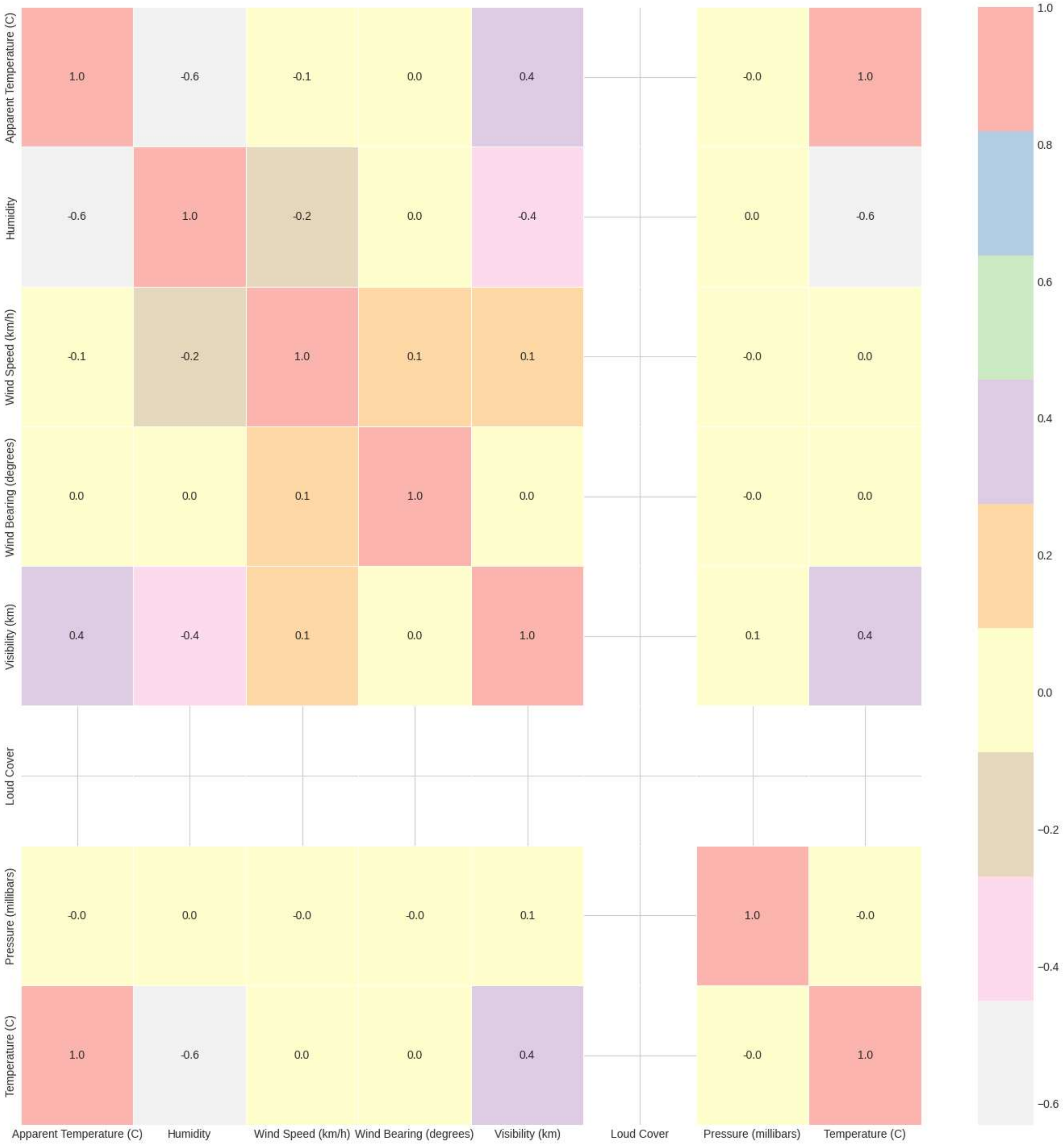
3. **Data points:** 96453

4. **What kind of features are in your dataset? Quantitative or categorical?**

The dataset contains a mixture of both quantitative and categorical features. Quantitative features are those that involve numerical measurements, such as temperature, humidity, wind speed, and pressure. These features have values that can be quantified and compared.

On the other hand, categorical features represent different categories or labels. For instance, features like "Summary" and "Daily Summary" provide descriptions of the weather conditions in categorical terms. Similarly, "Precip Type" is a categorical feature indicating whether it's raining or snowing. These features help classify weather conditions into specific categories based on their descriptions.

5. Correlation of all features:

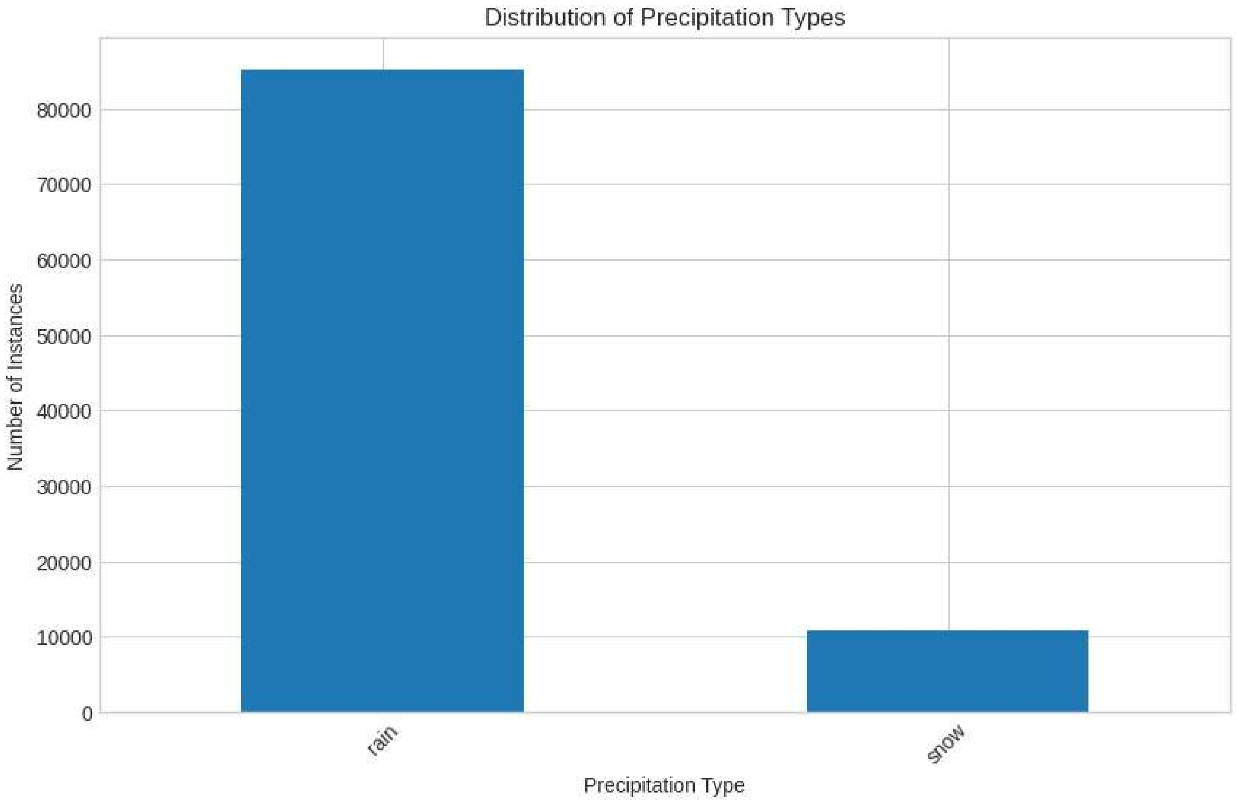


Imbalanced Dataset

1. For the output feature, do all unique classes have an equal number of instances or not?

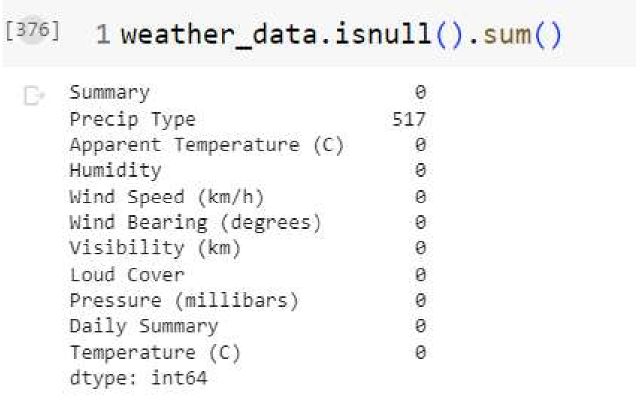
- No

2. Represent using a bar chart of N classes (N=number of classes you have in your dataset).



Dataset Preprocessing

Faults: Null values



```
[376] weather_data.isnull().sum()
```

```
Summary                       0
Precip Type                 517
Apparent Temperature (C)      0
Humidity                      0
Wind Speed (km/h)             0
Wind Bearing (degrees)        0
Visibility (km)               0
Loud Cover                    0
Pressure (millibars)          0
Daily Summary                 0
Temperature (C)               0
dtype: int64
```

Drop null values rows:

```
print("Shape of dataframe before dropping:", weather_data.shape)
weather_data = weather_data.dropna(axis = 0, subset = ['Precip Type'])
print("Shape after dropping:", weather_data.shape)
```

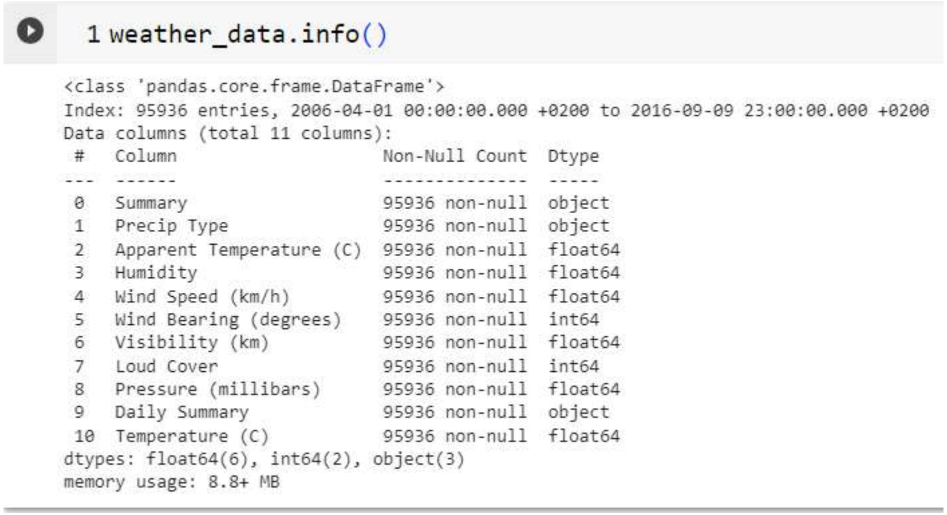
Drop null value column:

```
weather_data.drop('Loud Cover', axis=1, inplace=True)
weather_data.head()
```

Impute values:

```
imputer = SimpleImputer(strategy='mean')
X = weather_data.drop('Precip Type', axis=1)
X_imputed = imputer.fit_transform(X)
```

Fault: Categorical values:



```
weather_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 95936 entries, 2006-04-01 00:00:00.000 +0200 to 2016-09-09 23:00:00.000 +0200
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Summary                   95936 non-null  object 
 1   Precip Type               95936 non-null  object 
 2   Apparent Temperature (C)  95936 non-null  float64
 3   Humidity                  95936 non-null  float64
 4   Wind Speed (km/h)         95936 non-null  float64
 5   Wind Bearing (degrees)    95936 non-null  int64  
 6   Visibility (km)           95936 non-null  float64
 7   Loud Cover                95936 non-null  int64  
 8   Pressure (millibars)      95936 non-null  float64
 9   Daily Summary             95936 non-null  object 
 10  Temperature (C)           95936 non-null  float64
dtypes: float64(6), int64(2), object(3)
memory usage: 8.8+ MB
```

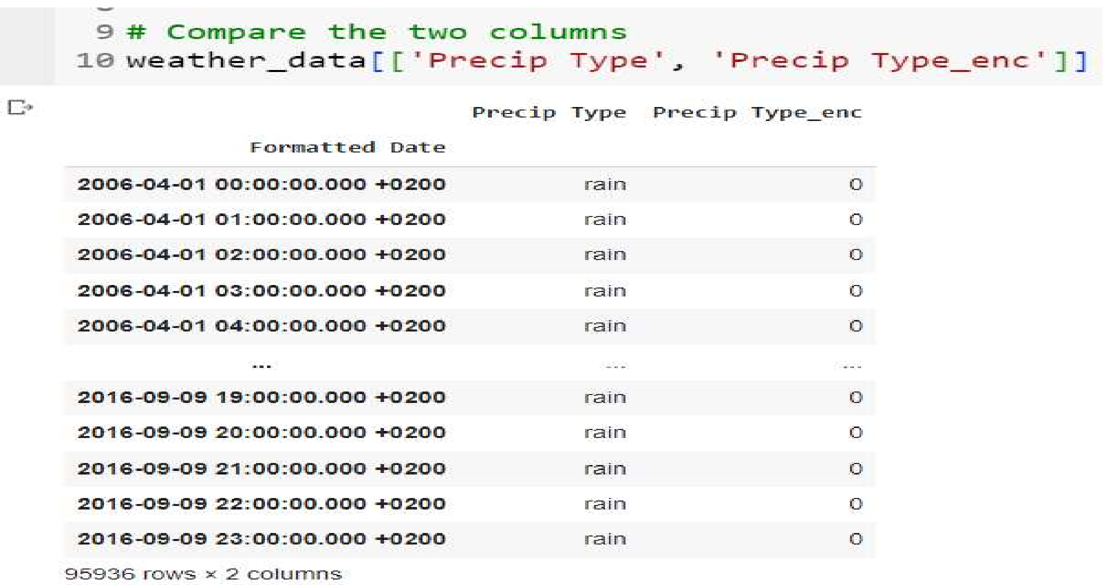
The object types are qualitative data while the float and int type data are quantitative.

Encoding

We need to replace the categorical data with numerical data so that the computer can understand and notice a pattern so that it can make a prediction. We do this via:

1. **Binary encoding:**

```
from sklearn.preprocessing import LabelEncoder

# Setting up the LabelEncoder object
enc = LabelEncoder()

# Apply the encoding to the "Precip Type" column
weather_data['Precip Type_enc'] =
enc.fit_transform(weather_data['Precip Type'])
```



```
# Compare the two columns
weather_data[['Precip Type', 'Precip Type_enc']]
```

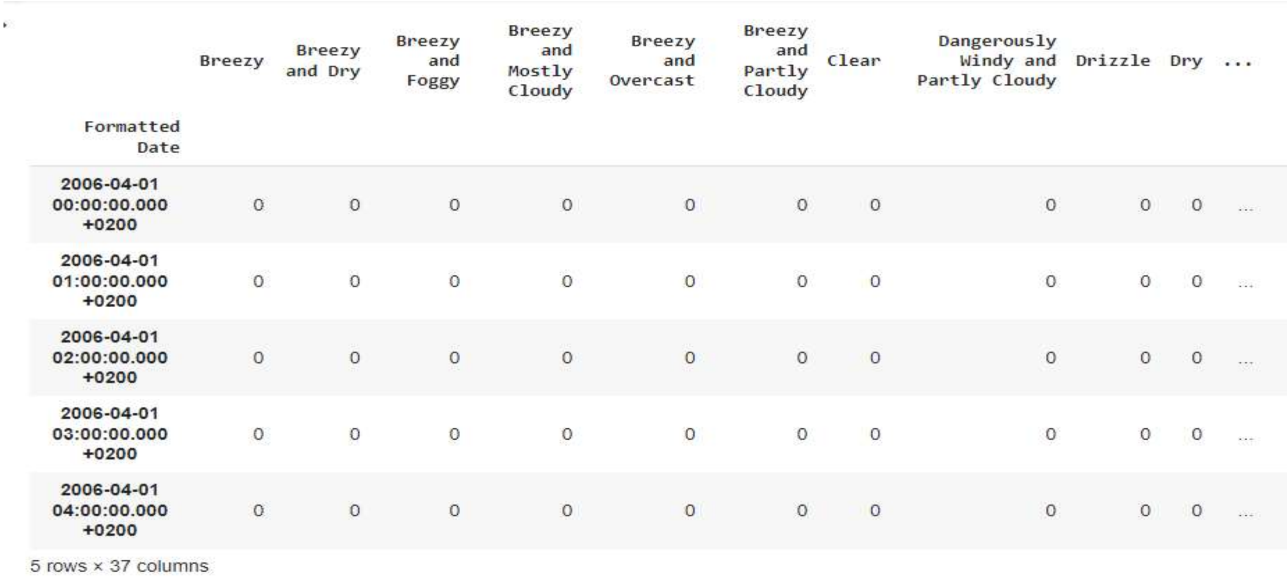
```
                                Precip Type   Precip Type_enc
             Formatted Date
2006-04-01 00:00:00.000 +0200          rain                 0
2006-04-01 01:00:00.000 +0200          rain                 0
2006-04-01 02:00:00.000 +0200          rain                 0
2006-04-01 03:00:00.000 +0200          rain                 0
2006-04-01 04:00:00.000 +0200          rain                 0
             ...                        ...               ...
2016-09-09 19:00:00.000 +0200          rain                 0
2016-09-09 20:00:00.000 +0200          rain                 0
2016-09-09 21:00:00.000 +0200          rain                 0
2016-09-09 22:00:00.000 +0200          rain                 0
2016-09-09 23:00:00.000 +0200          rain                 0
95936 rows × 2 columns
```

2. **Mapping:**

```
weather_data['Daily Summary'] = weather_data['Daily 
Summary'].map({
    'Partly cloudy throughout the day.': 1,
    'Mostly cloudy throughout the day.': 2,
    'Foggy in the evening.': 3,
    'Foggy overnight and breezy in the morning.': 4,
    'Overcast throughout the day.': 5,
    'Partly cloudy until night.': 6,
    'Mostly cloudy until night.': 7,
    'Foggy starting overnight continuing until morning.': 8,
    'Foggy in the morning.': 9,
    'Partly cloudy until evening.': 10,
    'Partly cloudy starting in the morning.': 11,
    'Mostly cloudy starting overnight continuing until night.': 
12,
    'Partly cloudy starting in the afternoon.': 13,
    'Partly cloudy starting overnight.': 14,
    'Mostly cloudy starting overnight.': 15,
    'Mostly cloudy until night and breezy in the afternoon.': 
16,
    'Mostly cloudy until evening.': 17,
    'Foggy throughout the day.': 18,
    'Partly cloudy starting in the morning.': 19,
    'Partly cloudy starting in the morning continuing until
evening.': 20,
    'Foggy until morning.': 21,
    'Partly cloudy starting in the morning continuing until
night.': 22,
    'Mostly cloudy starting in the morning.': 23,
    'Foggy starting in the evening.': 24,
    'Partly cloudy starting in the afternoon continuing until
evening.': 25,
    'Foggy overnight.': 26,
    'Clear throughout the day.': 27,
    'Partly cloudy starting overnight continuing until night.':
28,
    'Partly cloudy overnight.': 29,
    'Partly cloudy starting overnight continuing until
evening.': 30,
    'Foggy until night.': 31,
    'Partly cloudy in the morning.': 32,
    'Foggy starting overnight continuing until afternoon.': 33,
    'Foggy until afternoon.': 34,
    'Breezy and mostly cloudy overnight.': 35,
    'Partly cloudy overnight and breezy starting in the morning
continuing until afternoon.': 36,
    'Breezy in the morning and foggy in the evening.': 37,
    'Mostly cloudy until evening and breezy in the evening.':
38,
    'Mostly cloudy starting in the evening.': 39,
    'Mostly cloudy throughout the day and breezy starting
overnight continuing until afternoon.': 40,
    'Breezy starting in the morning continuing until night.':
41,
    'Overcast throughout the day and breezy starting overnight
continuing until morning.': 42,
    'Breezy starting overnight continuing until morning and
foggy in the evening.': 43,
    'Light rain until morning.': 44,
    'Mostly cloudy until night and breezy starting in the
afternoon continuing until night.': 45,
    'Mostly cloudy starting in the morning continuing until
afternoon.': 46,
    'Breezy until afternoon and overcast throughout the day.':
47,
    'Partly cloudy until evening and breezy in the afternoon.':
48,
    'Breezy starting overnight continuing until morning and
partly cloudy starting overnight continuing until evening.': 49,
    'Light rain starting overnight.': 50,
    'Partly cloudy starting overnight continuing until evening
and breezy starting in the morning continuing until evening.':
51,
    'Foggy starting in the morning continuing until evening and
breezy in the evening.': 52,
    'Partly cloudy throughout the day and breezy in the
afternoon.': 53,
    'Mostly cloudy starting overnight continuing until evening
and breezy starting overnight continuing until morning.': 54,
    'Partly cloudy starting overnight continuing until evening
and breezy in the morning.': 55,
    'Overcast throughout the day and breezy overnight.': 56,
    'Light rain in the morning.': 57,
    'Rain until morning.': 58,
    'Breezy in the morning and mostly cloudy starting in the
evening.': 59,
    'Mostly cloudy starting in the morning and breezy
overnight.': 60,
    'Partly cloudy starting overnight and breezy starting in the
morning continuing until afternoon.': 61,
    'Partly cloudy starting in the morning and breezy starting
in the afternoon continuing until evening.': 62,
    'Partly cloudy starting in the morning continuing until
evening and breezy in the afternoon.': 63,
    'Foggy starting overnight continuing until morning and
breezy in the afternoon.': 64
})
```

3. One hot encoding:

```
summary_enc = pd.get_dummies(weather_data['Summary'])

# Drop original 'Summary' column
weather_data.drop('Summary', axis=1, inplace=True)

# Concatenate with the original dataframe
weather_data = pd.concat([summary_enc, weather_data], axis=1)


weather_data.head()
```



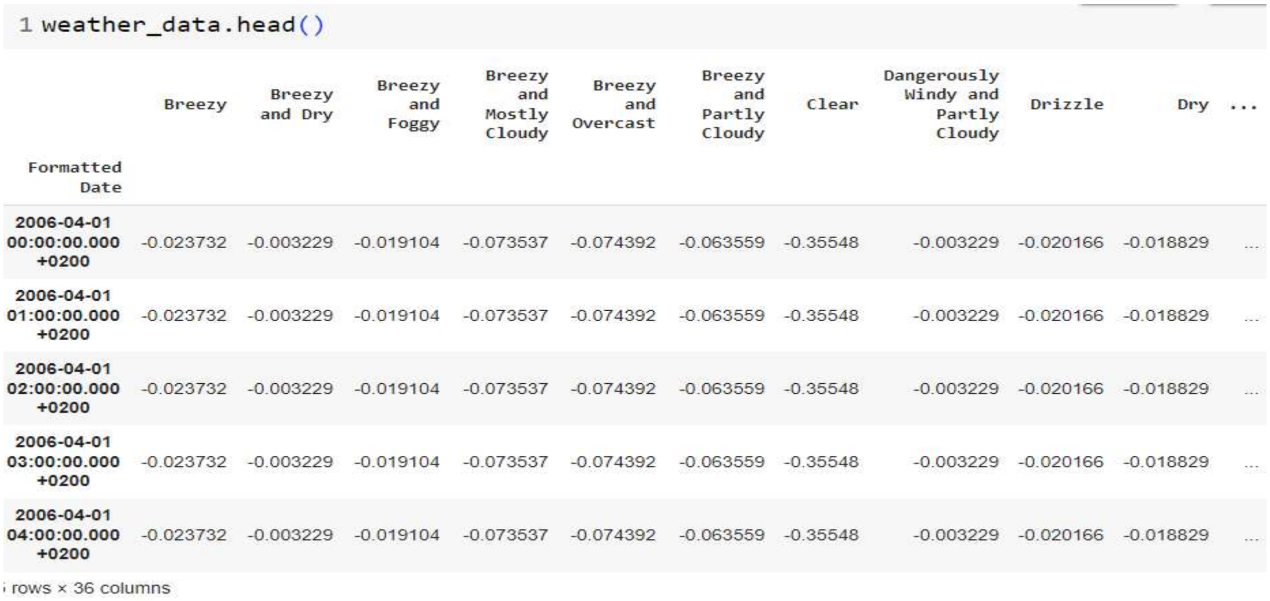
	Breezy	Breezy and Dry	Breezy and Foggy	Breezy and Mostly Cloudy	Breezy and Overcast	Breezy and Partly Cloudy	Clear	Dangerously Windy and Partly Cloudy	Drizzle	Dry	...
Formatted Date											
2006-04-01 00:00:00.000 +0200	0	0	0	0	0	0	0	0	0	0	...
2006-04-01 01:00:00.000 +0200	0	0	0	0	0	0	0	0	0	0	...
2006-04-01 02:00:00.000 +0200	0	0	0	0	0	0	0	0	0	0	...
2006-04-01 03:00:00.000 +0200	0	0	0	0	0	0	0	0	0	0	...
2006-04-01 04:00:00.000 +0200	0	0	0	0	0	0	0	0	0	0	...

5 rows × 37 columns

4. Feature scaling:

One of the problems that the algorithm often faces is that it gets biased towards larger values. To avoid this problem, we scale the numerical data to bring it in a small range.

```
from sklearn.preprocessing import StandardScaler

# Separate the target variable
a = weather_data.drop('Precip Type', axis=1)

# Initialize the StandardScaler
scaler = StandardScaler()

scaled_features = scaler.fit_transform(a)

# Update the original weather_data with the scaled
weather_data[a.columns] = scaled_features
```



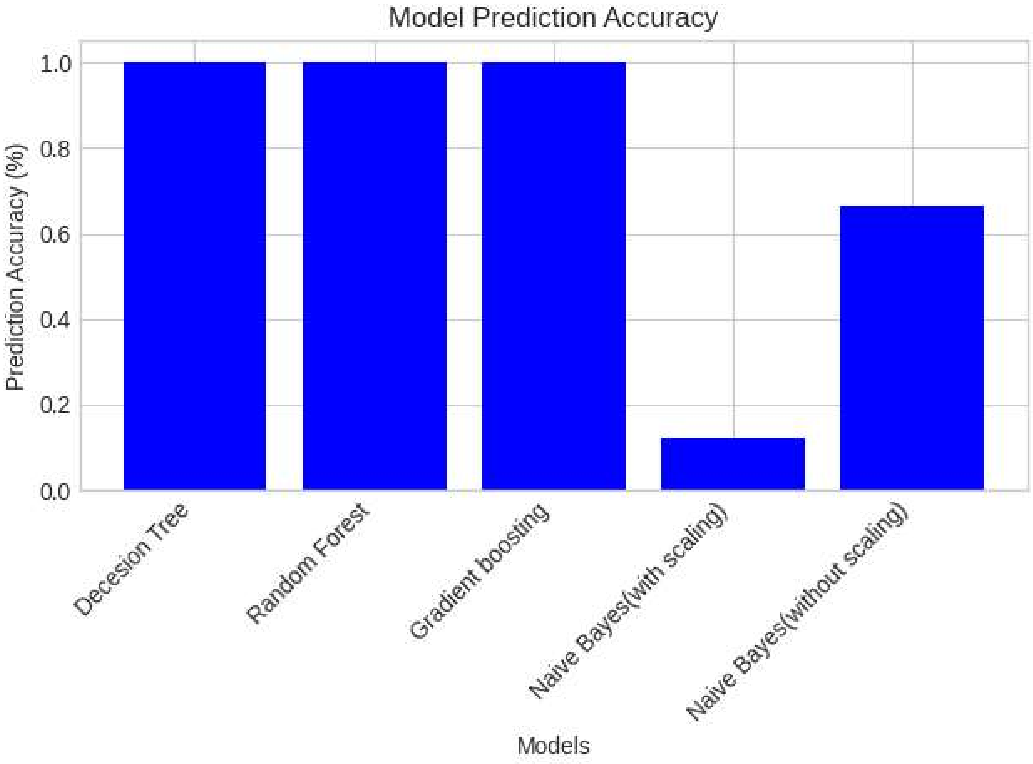
```
weather_data.head()
```

Formatted Date	Breezy	Breezy and Dry	Breezy and Foggy	Breezy and Mostly Cloudy	Breezy and Overcast	Breezy and Partly Cloudy	Clear	Dangerously Windy and Partly Cloudy	Drizzle	Dry	...
2006-04-01 00:00:00.000 +0200	-0.023732	-0.003229	-0.019104	-0.073537	-0.074392	-0.063559	-0.35548	-0.003229	-0.020166	-0.018829	...
2006-04-01 01:00:00.000 +0200	-0.023732	-0.003229	-0.019104	-0.073537	-0.074392	-0.063559	-0.35548	-0.003229	-0.020166	-0.018829	...
2006-04-01 02:00:00.000 +0200	-0.023732	-0.003229	-0.019104	-0.073537	-0.074392	-0.063559	-0.35548	-0.003229	-0.020166	-0.018829	...
2006-04-01 03:00:00.000 +0200	-0.023732	-0.003229	-0.019104	-0.073537	-0.074392	-0.063559	-0.35548	-0.003229	-0.020166	-0.018829	...
2006-04-01 04:00:00.000 +0200	-0.023732	-0.003229	-0.019104	-0.073537	-0.074392	-0.063559	-0.35548	-0.003229	-0.020166	-0.018829	...

5 rows × 36 columns

7. Model Selection Comparison Analysis:

Bar chart showcasing prediction accuracy of all models:



Precision, recall comparison of each model:

Precision emphasizes the accuracy of positive predictions among all positive predictions. Recall emphasizes the model's ability to correctly identify all positive instances among all actual positive instances.

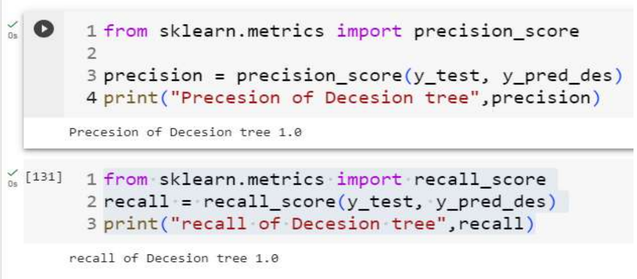
Precision = (True Positive)/(True Positive + False Positive)

Recall = (True Positive)/(True Positive + False Negative)

Decision Tree:

Precision = 1

Recall = 1



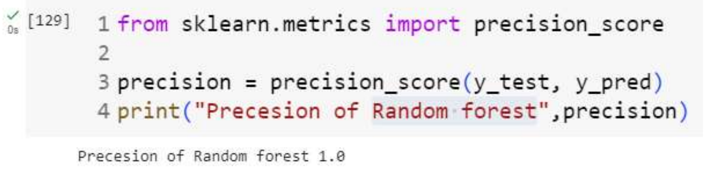
```
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_pred_des)
print("Precesion of Decesion tree",precision)
```

```
Precesion of Decesion tree 1.0
```

```
from sklearn.metrics import recall_score
recall = recall_score(y_test, y_pred_des)
print("recall of Decesion tree",recall)
```

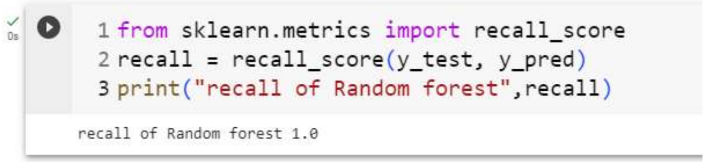
```
recall of Decesion tree 1.0
```

Random Forest:
Precision = 1
Recall = 1



```
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_pred)
print("Precesion of Random forest",precision)
```

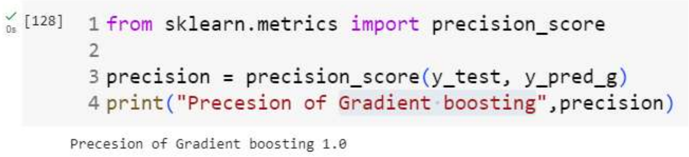
```
Precesion of Random forest 1.0
```



```
from sklearn.metrics import recall_score
recall = recall_score(y_test, y_pred)
print("recall of Random forest",recall)
```

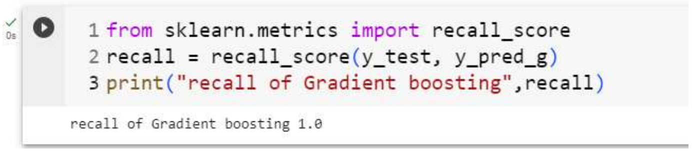
```
recall of Random forest 1.0
```

Gradient Boosting:
Precision = 1
Recall = 1



```
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_pred_g)
print("Precesion of Gradient boosting",precision)
```

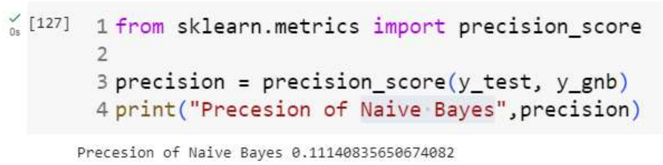
```
Precesion of Gradient boosting 1.0
```



```
from sklearn.metrics import recall_score
recall = recall_score(y_test, y_pred_g)
print("recall of Gradient boosting",recall)
```

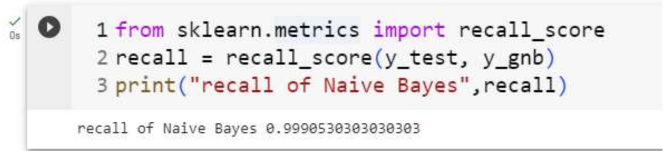
```
recall of Gradient boosting 1.0
```

Naive Bayes(with Scaling):
Precision = 0.111
Recall = 0.999



```
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_gnb)
print("Precesion of Naive Bayes",precision)
```

```
Precesion of Naive Bayes 0.11140835650674082
```

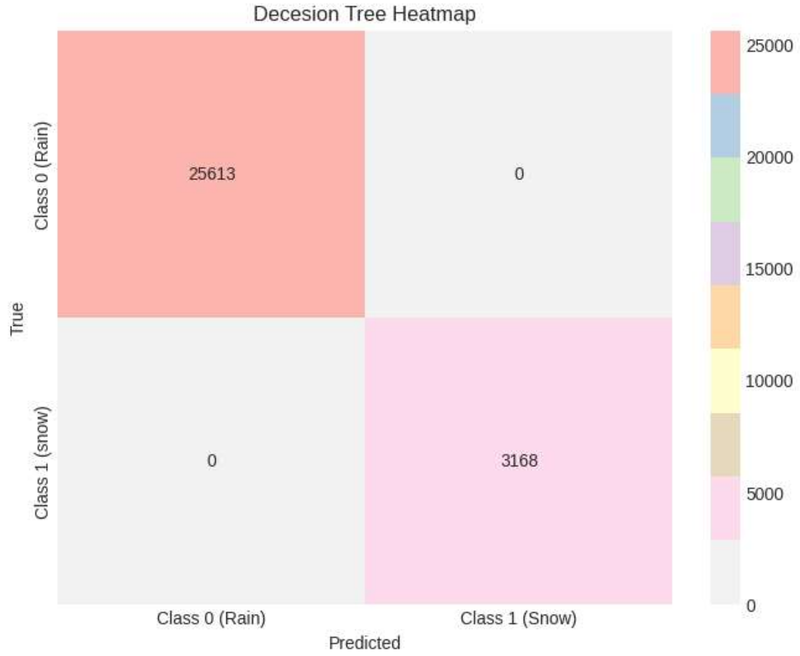


```
from sklearn.metrics import recall_score
recall = recall_score(y_test, y_gnb)
print("recall of Naive Bayes",recall)
```

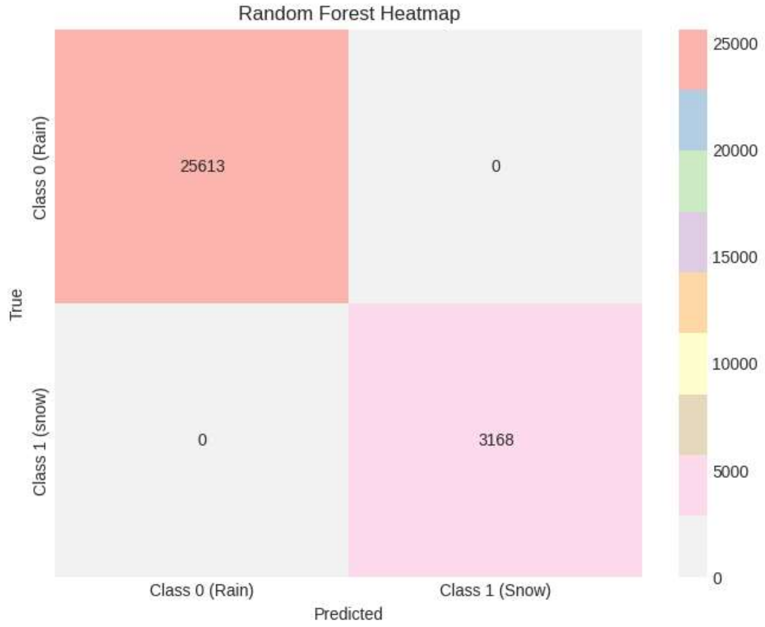
```
recall of Naive Bayes 0.9990530303030303
```

As we can see, the first three models all give extremely accurate predictions, with 0 error so both precision and recall are 1, meaning there are no false positives or false negatives. However, the Naive Bayes model is extremely inaccurate in this case, with a very low precision meaning that there is a huge amount of false positives, which for our data means that it is extremely likely to predict that it is going to rain on days when it will snow, which is likely to inconvenience anyone following this model. This shows that the naive bayes model is not a good algorithm for this particular dataset.

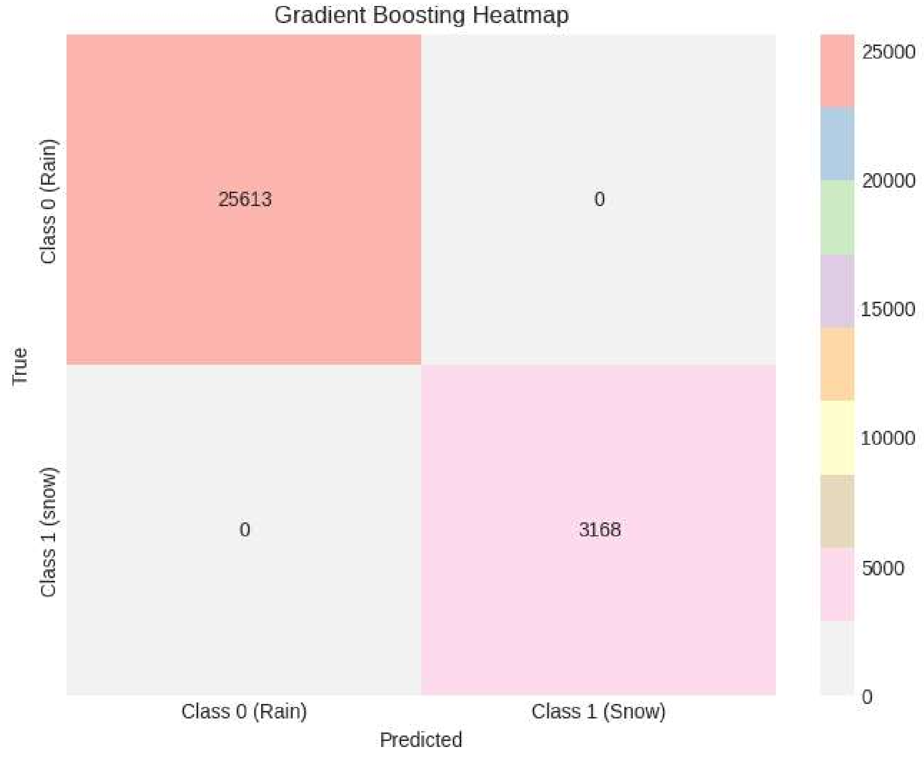
Confusion matrix:
Decision tree:



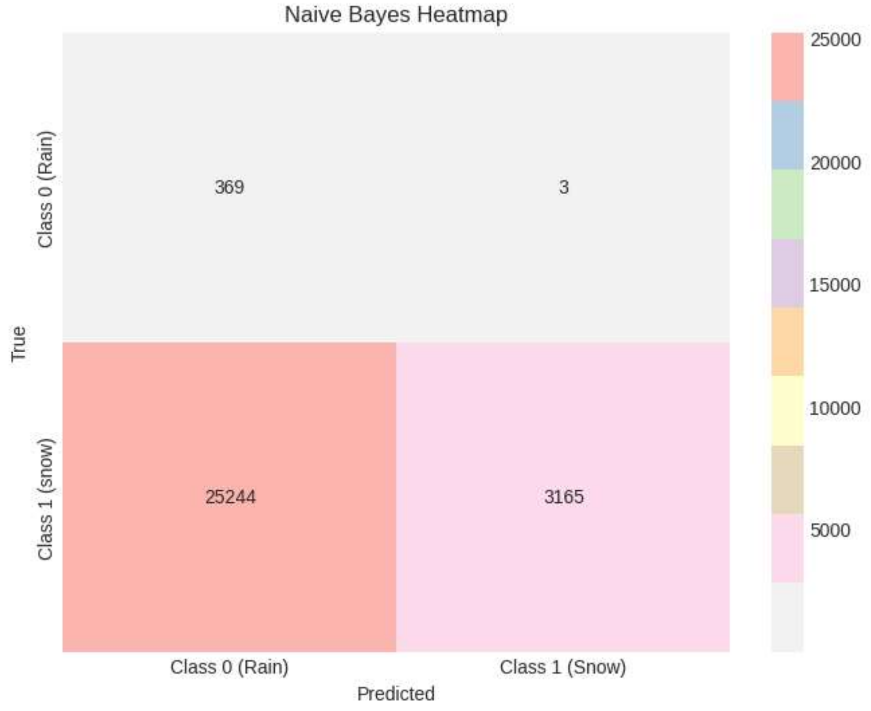
Random Forest:



Gradient Boosting:



Naive Bayes(with scaling):



Github Link:
https://github.com/shihabmuhtasim/Machinearning-Model-Weather-Prediction-Rain-Snow-

Models used

Random Forest:

- Create a multitude of decision trees, each with its perspective.
- Gather predictions from all trees and select the most common prediction.
- This ensemble approach helps balance out individual weaknesses and provides a robust prediction.

Gradient Boosting:

- Begin with an initial decision tree that may have errors.
- Focus on correcting the mistakes made by the initial model.
- Build subsequent models to fix errors of the previous ones, step by step.
- This iterative process improves accuracy over time by refining predictions.

Decision Tree Classifier:

- Construct a tree-like model of decisions and their possible consequences.
- Split data into subsets based on conditions that maximize information gain.
- Reach a leaf node that represents a decision or classification.
- Easy to interpret, useful for visualizing decision-making processes.

Naive Bayes Classifier:

- Based on Bayes' theorem, a probabilistic approach to classification.
- Assumes features are conditionally independent given the class label.
- Calculates probabilities for different classes based on features.
- Efficient and effective for text classification and simple datasets.