Prediction for final_test

In this section, we take the input information and use the appropriate machine that we tested in the previous section, we train it and predict the information.

Data Loading and Preprocessing

In this section, we load the training and testing data from CSV files and perform some preprocessing steps.

Load Train Data for Training

We load the training data from the 'train.csv' file and remove any rows with missing values.

- Load the training CSV and test CSV file.
- Remove rows with missing values.

Next, we split the training data into input features (X) and the target variable (y).

> - Drop the 'weather' column from the input features.
> - Preprocess the 'date' column by removing slashes and converting to float.

Load the testing CSV file. and do it same thing like Training

In [1]:
```
import pandas as pd

data = pd.read_csv('train.csv').dropna()
data2 = pd.read_csv('test.csv').dropna()

data = pd.concat([data, data2])

data.reset_index(drop=True, inplace=True)

X = data.drop('weather', axis=1)
X['date'] = pd.to_datetime(X['date'], format='%m/%d/%Y')
X['date'] = X['date'].dt.strftime('%Y%m%d').astype(float)
X= X.sort_values('date')

y = data['weather']
```

Data Loading and Preprocessing

In this section, we load the training data from the 'final_test.csv' file and perform some preprocessing steps.

Load Training Data

We load the training data from the CSV file and remove any rows with missing values.

> - Load the training CSV file.
> - Remove rows with missing values.
> - Preprocess the 'date' column by removing slashes and converting it to float.

In [5]:

```
# Load the training CSV file
infotmation = pd.read_csv('final_test.csv').dropna()

infotmation['date'] = pd.to_datetime(infotmation['date'], format='%m/%d/%Y
infotmation['date'] = infotmation['date'].dt.strftime('%Y%m%d').astype(flo
```

Model Training and Prediction

In this section, we use the Gradient Boosting Classifier from the scikit-learn library to train the model on the training data and make predictions on the testing data.

Model Training

We initialize a Gradient Boosting Classifier model and train it on the training data.

- Initialize a Gradient Boosting Classifier model.
- Fit the model to the training data.

Model Retraining

We retrain the model on the new training data.

- Retrain the model using the testing data.

Prediction

We make predictions on the testing data using the trained model.

- Make predictions on the testing data.

Save Predictions

We save the predictions to a CSV file called 'list.csv'.

- Create a DataFrame with the predicted values.
- Save the DataFrame to a CSV file.

Feel free to modify and expand upon the Markdown document based on your specific requirements.

In [6]:
```
from sklearn.ensemble import RandomForestClassifier , GradientBoostingClas
model = GradientBoostingClassifier()

model.fit(X, y)

# Make predictions on the testing data
y_pred = model.predict(infotmation)

df = pd.DataFrame(y_pred, columns=["weather"])

df.to_csv('list.csv', index=False)
# df.head()
```

In [ ]:
```
```