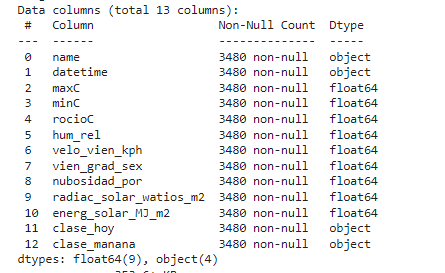
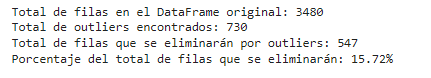
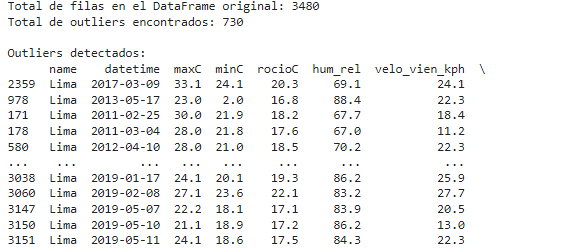
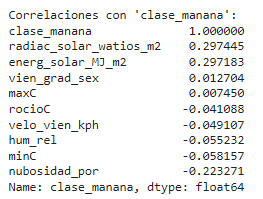
1)

* El dataframe tiene 13 columnas :
* La variable dependiente seria **clima\_ mañana .**
* El data\_Frame no tiene valores **NULOS**.
* El data\_frame no tiene datos **DUPLICADOS**
* Tenemos dos columnas categoricas , que se deben convertir a numericas : clase\_hoy y clase\_mañana : **(lluvioso 0 , nublado 1, parcialmente\_nublado2 )**
* Calculamos los outliers (MÉTODO DE CUARILES Q1,Q2,Q3,Q4) para nuestros datos y empezar a procesar : **730 DATOS OUTLIERS. QUE SON 547 FILAS ( 15.72% ) ESTE DATO ES MUY ALTO , SE PUEDE OPTAR POR REEMPLAZAR POR LA MEDIANA ,.**
* 



* Matriz de correlación: las variables con correlaciones **positivas mas altas son :Radiac\_solar\_watios\_m2 ; energ\_solar\_MJ\_m2**

**El resto de variables son minimas y/o negativas.**



2) Matriz de confusion:Para random forest tenemos

Clase 0:

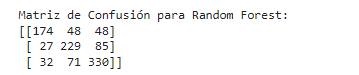
* 174 ejemplos de clase 0 fueron correctamente clasificados como clase 0.
* 48 ejemplos de clase 0 fueron incorrectamente clasificados como clase 1.
* 48 ejemplos de clase 0 fueron incorrectamente clasificados como clase 2.

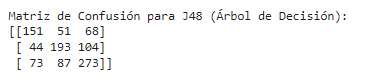
Clase 1:

* 27 ejemplos de clase 1 fueron incorrectamente clasificados como clase 0.
* 229 ejemplos de clase 1 fueron correctamente clasificados como clase 1.
* **85** ejemplos de clase 1 fueron incorrectamente clasificados como clase 2.

Clase 2:

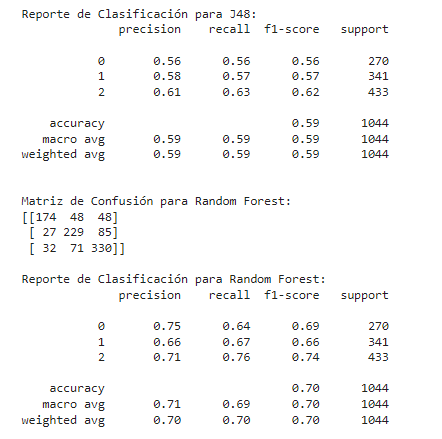
* **32** ejemplos de clase 2 fueron incorrectamente clasificados como clase 0.
* **71** ejemplos de clase 2 fueron incorrectamente clasificados como clase 1.
* **330** ejemplos de clase 2 fueron correctamente clasificados como clase 2.





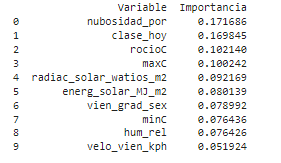
3) el modelo alcanzó una precision del 70% random forest y59% con j48 , esto se puede mejorar remplazando los outliers con la mediana , la media o la moda , o analizar bien los outliers , 15% es un porcentaje alto

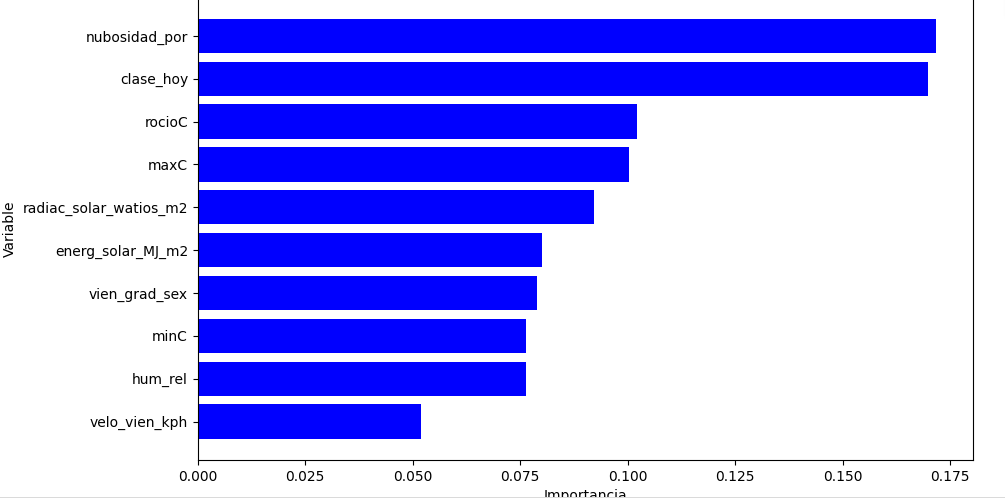
Ecplicación:



4) para evaluar los criterios en primer lugar nos guiamos de la precision , precision por clase , recall ,f1-score , matriz de confusion . **en nuestro caso random forest supera en TODO a J48.**

**5) Variables con mayore relavancia**





Codigo python – google colab

from google.colab import files

# Subir el archivo

#Se esta cargando directamente el archivo Clima\_Aeropuerto\_Lima

uploaded = files.upload()

#asignar nombre al archivo , el mismo .

archivo = "Clima\_Aeropuerto\_Lima.csv"

df = pd.read\_csv(archivo)

# Leer DataFrame

df = pd.read\_csv(file\_name)

# Exploración de datos

print(df.head())

print(df.info())

print(df.describe())

print(df.isnull().sum())

# Preprocesamiento de datos

df.dropna(inplace=True)  # Eliminar filas con valores nulos

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix, classification\_report

import matplotlib.pyplot as plt

import numpy as np

# Verificar las categorías originales en 'clase\_hoy' y 'clase\_manana'

print("\nCategorías originales de 'clase\_hoy':")

print(df['clase\_hoy'].unique())

print("\nCategorías originales de 'clase\_manana':")

print(df['clase\_manana'].unique())

# Convertir columnas categóricas a numéricas

df['clase\_hoy'] = df['clase\_hoy'].astype('category').cat.codes

df['clase\_manana'] = df['clase\_manana'].astype('category').cat.codes

# Mostrar las categorías convertidas

print("\nCategorías convertidas de 'clase\_hoy':")

print(df['clase\_hoy'].unique())

print("\nCategorías convertidas de 'clase\_manana':")

print(df['clase\_manana'].unique())

# Función para detectar outliers

def detectar\_outliers\_iqr(df):

    outliers = pd.DataFrame(columns=df.columns)

    # Calcular el IQR y los límites de outliers para cada columna numérica

    for column in df.select\_dtypes(include=['float64', 'int64']).columns:

        Q1 = df[column].quantile(0.25)

        Q3 = df[column].quantile(0.75)

        IQR = Q3 - Q1

        lower\_bound = Q1 - 1.5 \* IQR

        upper\_bound = Q3 + 1.5 \* IQR

        # Identificar outliers

        outliers\_column = df[(df[column] < lower\_bound) | (df[column] > upper\_bound)]

        outliers = pd.concat([outliers, outliers\_column])

    return outliers

# Detectar outliers

outliers = detectar\_outliers\_iqr(df)

# Calcular el número de filas que contienen outliers

filas\_con\_outliers = outliers.index.unique()

numero\_filas\_eliminar = len(filas\_con\_outliers)

# Calcular el porcentaje de filas que se eliminarían

porcentaje\_filas\_eliminar = (numero\_filas\_eliminar / df.shape[0]) \* 100

print(f"\nTotal de filas en el DataFrame original: {df.shape[0]}")

print(f"Total de outliers encontrados: {outliers.shape[0]}")

print(f"Total de filas que se eliminarán por outliers: {numero\_filas\_eliminar}")

print(f"Porcentaje del total de filas que se eliminarán: {porcentaje\_filas\_eliminar:.2f}%")

print("\nOutliers detectados:")

print(outliers)

# Definir variable dependiente

variable\_dependiente = 'clase\_manana'

variables\_independientes = df.columns.drop([variable\_dependiente, 'name', 'datetime'])

# Dividir los datos en conjuntos de entrenamiento (70%) y prueba (30%)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df[variables\_independientes], df[variable\_dependiente], test\_size=0.3, random\_state=42)

# Entrenar el modelo J48 (Árbol de Decisión)

modelo\_j48 = DecisionTreeClassifier(random\_state=42)

modelo\_j48.fit(X\_train, y\_train)

# Realizar predicciones con J48

y\_pred\_j48 = modelo\_j48.predict(X\_test)

# Calcular la matriz de confusión para J48

matriz\_confusion\_j48 = confusion\_matrix(y\_test, y\_pred\_j48)

# Mostrar la matriz de confusión y el reporte de clasificación para J48

print("\nMatriz de Confusión para J48 (Árbol de Decisión):")

print(matriz\_confusion\_j48)

print("\nReporte de Clasificación para J48:")

print(classification\_report(y\_test, y\_pred\_j48))

# Entrenar el modelo Random Forest

modelo\_rf = RandomForestClassifier(random\_state=42)

modelo\_rf.fit(X\_train, y\_train)

# Realizar predicciones con Random Forest

y\_pred\_rf = modelo\_rf.predict(X\_test)

# Calcular la matriz de confusión para Random Forest

matriz\_confusion\_rf = confusion\_matrix(y\_test, y\_pred\_rf)

# Mostrar la matriz de confusión y el reporte de clasificación para Random Forest

print("\nMatriz de Confusión para Random Forest:")

print(matriz\_confusion\_rf)

print("\nReporte de Clasificación para Random Forest:")

print(classification\_report(y\_test, y\_pred\_rf))

# Extraer las importancias de las características

importancias = modelo\_rf.feature\_importances\_

indices = np.argsort(importancias)[::-1]

# Crear un DataFrame con las importancias de las características

importancias\_df = pd.DataFrame({

    'Variable': X\_train.columns[indices],

    'Importancia': importancias[indices]

})

# Mostrar las importancias de las características

print(importancias\_df.head(10))

# Gráfico de barras de las importancias de las características

plt.figure(figsize=(10, 6))

plt.title("Importancia de las Características")

plt.barh(importancias\_df['Variable'], importancias\_df['Importancia'], color='b', align='center')

plt.xlabel('Importancia')

plt.ylabel('Variable')

plt.gca().invert\_yaxis()

plt.show()