Asignatura	Datos del alumno	Fecha
Técnicas de inteligencia Artificial	Apellidos: González Rojas	Enero 2022
	Nombre: Samara Jocelyn	

Apartado A (Regresión)

En las siguientes imágenes se muestran los puntos solicitados en la actividad

```
import pandas as pd
data_train = pd.read_csv('C:/Users/52553/Desktop/Mestria UNIR/2° semestre/Tecnicas de Inteligencia Artificial/weatherHistory.csv'
```

```
#Tamaño del conjunto de datos
#En el cual si cumplimos con tener mas de 1000 muestras y 6 caracteristicas
data_train.shape
```

```
(96453, 12)
```

```
#Primeras cinco muestras
data_train.head()
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

Lo anterior supone un problema ya que los algoritmos de regresión **no trabajan con valores categóricos**. Hay un alcance importante denominado **Label encoding** donde cada categoría se convierte en una etiqueta discreta. Mediante el método `astype('category').cat.codes` de -pandas se pueden convertir de categoría a números

```
data_train["Daily Summary"] = data_train["Daily Summary"].astype('category').cat.codes
data_train["Precip Type"] = data_train["Precip Type"].astype('category').cat.codes
# Esta será nuestro valor objetivo a predecir
data_train["Summary"] = data_train["Summary"].astype('category').cat.codes
#Este timestamp no los nececitamos
data_train = data_train.drop(columns=["Formatted Date"])
```

Mediante Python y las librerías que consideres, analiza el dataset proporcionando una caracterización del mismo, mostrando **al menos** algunas de sus características en modo texto (mediante tablas o prosa) y **al menos** algunas de ellas en modo gráfico (p.ej., histogramas, diagramas de dispersión, diagramas de cajas y bigotes, etc.). **Las características y las gráficas incluidas han de provenir de la ejecución del código en Python que se aporte como respuesta.**

```
#1° Prosa con la cual describimos las estadisticas del dataset
import matplotlib.pyplot as plt
import seaborn as sns
#Descibir el dataset, estadísticamente
data_train.describe()
```

	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
count	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.0	96453.000000	96453.000000
mean	16.006024	0.105699	11.932678	10.855029	0.734899	10.810640	187.509232	10.347325	0.0	1003.235956	129.308057
std	4.361497	0.324420	9.551546	10.696847	0.195473	6.913571	107.383428	4.192123	0.0	116.969906	56.275398
min	0.000000	-1.000000	-21.822222	-27.716667	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
25%	17.000000	0.000000	4.688889	2.311111	0.600000	5.828200	116.000000	8.339800	0.0	1011.900000	95.000000
50%	18.000000	0.000000	12.000000	12.000000	0.780000	9.965900	180.000000	10.046400	0.0	1016.450000	118.000000
75%	19.000000	0.000000	18.838889	18.838889	0.890000	14.135800	290.000000	14.812000	0.0	1021.090000	181.000000
max	26.000000	1.000000	39.905556	39.344444	1.000000	63.852600	359.000000	16.100000	0.0	1046.380000	213.000000

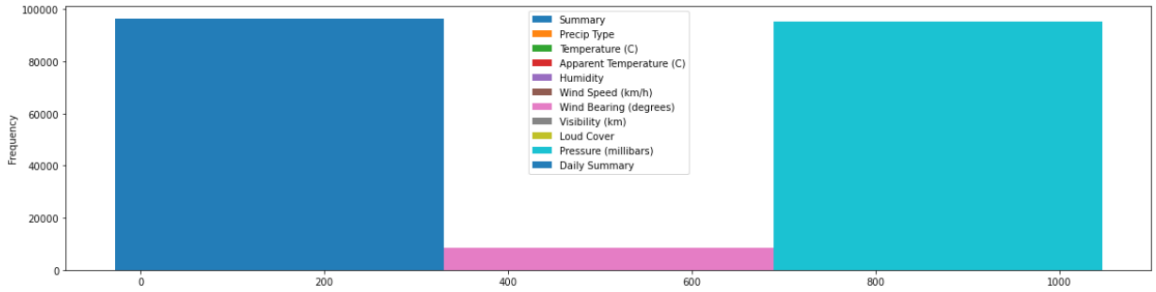
Asignatura	Datos del alumno	Fecha
Técnicas de inteligencia Artificial	Apellidos: González Rojas Nombre: Samara Jocelyn	Enero 2022

```
#Tipos de datos
#Con esto validamos que ya no tenemos datos
data_train.dtypes
```

```
Summary                       int8
Precip Type                   int8
Temperature (C)            float64
Apparent Temperature (C)   float64
Humidity                   float64
Wind Speed (km/h)          float64
Wind Bearing (degrees)     float64
Visibility (km)            float64
Loud Cover                 float64
Pressure (millibars)       float64
Daily Summary                int16
dtype: object
```

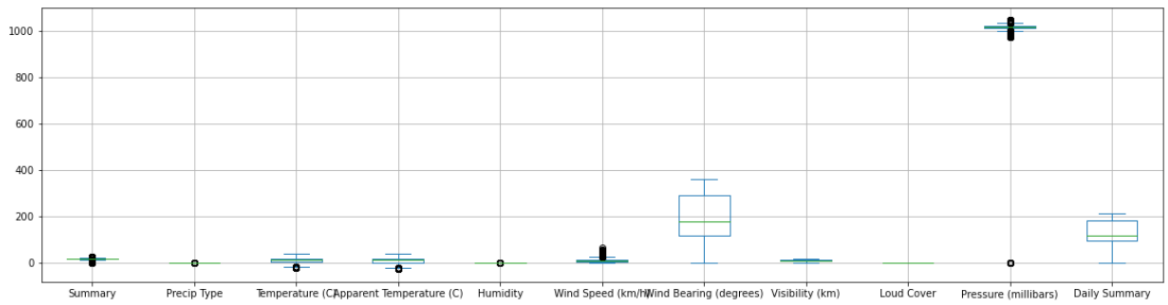
```
#explorar la frecuencia de datos, mediante un histograma
data_train.plot.hist(bins=3,figsize=(20,5))
# Solo se muestran 3 porque son las que mas ponderan
```

```
<AxesSubplot:ylabel='Frequency'>
```




```
#Graficos de cajas, para observar valores atípicos
data_train.plot.box(figsize=(20,5),grid=True)
#Podemos ver que la variable Presion tiene muchos valores muy separados
```

```
<AxesSubplot:>
```



Dividir muestras de entrenamiento X y salidas y. Normalizar el conjunto de datos.

```
y = data_train['Daily Summary']
X= data_train.drop(columns=['Daily Summary'])
```

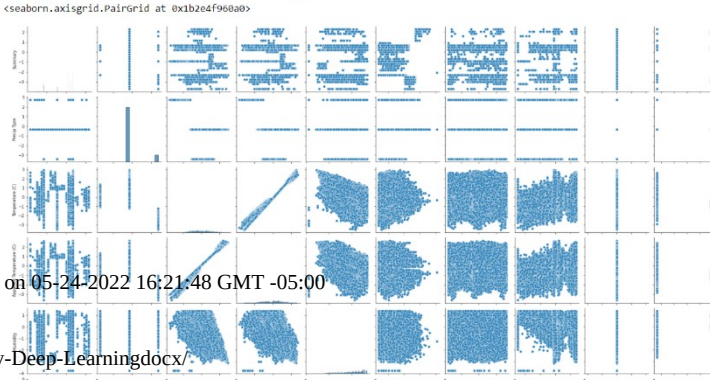
StandardScaler estandariza una característica restando la media y luego escalando a la varianza unitaria. La varianza unitaria significa dividir todos los valores por la desviación estándar.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaler = scaler.fit_transform(X)
print(X_scaler)
```

```
[[ 0.68645963 -0.32581153 -0.25759902 ...  1.30697578  0.
   0.10168518]
 [ 0.68645963 -0.32581153 -0.26981351 ...  1.30697578  0.
   0.1059598 ]
 [ 0.22789914 -0.32581153 -0.26748694 ...  1.09958576  0.
   0.10861007]
 ...
 [ 0.68645963 -0.32581153  1.0580761  ...  1.37226523  0.
   0.10621628]
 [ 0.68645963 -0.32581153  1.06398335 ...  1.37226523  0.
   0.10869557]
 [ 0.68645963 -0.32581153  0.89056308 ...  1.23400522  0.
   0.11049091]]
```

```
#La siguiente grafica nos muestra la dispersión que hay entre los datos comparandolos unos con otros
#y esto nos dice que tan agrupados o dispersos estan los datos
sns.pairplot(pd.DataFrame(X_scaler,columns=X.columns))
```

```
<seaborn.axisgrid.PairGrid at 0x1b2e4f960a0>
```



Asignatura	Datos del alumno	Fecha
Técnicas de inteligencia Artificial	Apellidos: González Rojas	Enero 2022
	Nombre: Samara Jocelyn	

Elige al menos un método de regresión no basado en redes neuronales (p.ej. regresión lineal, regresión polinómica, regresión logarítmica, SVR, random forest regression, etc.)

¿Qué es la Regresión?: Es una medida estadística que ayuda a determinar la relación entre una variable dependiente $\hat{Y}$, con respecto a una serie variables independientes X

La **regresión Random Forest** es un algoritmo de aprendizaje supervisado que utiliza el método de aprendizaje conjunto para la regresión. El método de aprendizaje por conjuntos es una técnica que combina predicciones de múltiples algoritmos de aprendizaje automático para realizar una predicción más precisa que la de un solo modelo.

Incluye en el informe una explicación de los parámetros que consideres relevantes en cada ejecución.

- `max_depth` : La profundidad máxima del árbol. Si es None, los nodos se expanden hasta que todas las hojas sean puras o hasta que todas las hojas contengan menos que min_samples_split samples.
- `min_samples_split` :El número mínimo de muestras necesarias para dividir un nodo interno:
  - Si es int, entonces considera min_samples_split como el número mínimo.
  - Si es float, entonces min_samples_split es una fracción y ceil(min_samples_split * n_samples) son el número mínimo de muestras para cada división.

```
from sklearn.ensemble import RandomForestRegressor
```

```
rds = RandomForestRegressor(max_depth=5,min_samples_split=3)
```

Mediante Python y las librerías que consideres, entrena el modelo o modelos escogidos, dividiendo el dataset en datos de entrenamiento y datos de test previamente en base a tu criterio y pruébalos frente a dichos datos de test.

```
# Generar conjuntos de entrenamiento y prueba
from sklearn.model_selection import train_test_split

X_train, X_validation, Y_train, Y_validation = train_test_split(X_scaler, y, test_size=0.20, random_state=1, shuffle=True)
```

```
#Entrenamiento de RandomForestRegressor
rds.fit(X_train,Y_train)
```

```
RandomForestRegressor(max_depth=5, min_samples_split=3)
```

```
# Predicción mediante red RandomForestRegressor
y_pred_rs = rds.predict(X_validation)
```

Elige al menos una arquitectura de red neuronal (**describe en el informe** las neuronas en la capa de entrada, las capas intermedias – **al menos dos capas intermedias** – y capa de salida, funciones de activación en cada caso) que permita realizar una **regresión**.

Se utiliza un modelo secuencial con dos capas ocultas densamente $H_{1,2} \in \mathbb{R}^{64}$ conectadas y una capa de salida $O \in \mathbb{R}$ que devuelve un único valor continuo, para la regresión.

```
#Tuve que modificar este codigo ya que meracba error e instale nuevamente tensorflow
# TensorFlow y tf.keras
import tensorflow as tf
from tensorflow.keras import *
from keras.models import Sequential
```

```
#Igual que arriba hice algunas modificaciones para que corriera
model = Sequential(
    [layers.Dense(64, activation='relu', input_shape=[len(X_train[0])]), #capa de entrada
     layers.Dense(32, activation='relu'),
     layers.Dense(32, activation='relu'),
     layers.Dense(1)]) #capa oculta
optimizer = tf.keras.optimizers.RMSprop(0.001)
model.compile(loss='mse',optimizer=optimizer,metrics=['mae', 'mse'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 64)                704

 dense_5 (Dense)             (None, 32)                2080

 dense_6 (Dense)             (None, 32)                1056

 dense_7 (Dense)             (None, 1)                 33

=================================================================
Total params: 3,873
Trainable params: 3,873
Non-trainable params: 0
```

Asignatura	Datos del alumno	Fecha
Técnicas de inteligencia Artificial	Apellidos: González Rojas	Enero 2022
	Nombre: Samara Jocelyn	

Mediante Python y utilizando **al menos Keras sobre TensorFlow 2.0 (tensorflow.keras)**, entrena el modelo o modelos de red neuronal escogidos, dividiendo el dataset en datos de entrenamiento y datos de test previamente en base a tu criterio y pruébalos frente a dichos datos de test.

```
# Entrenamienro de la red neuronal con 10 EPOCHS
model.fit(X_train, Y_train, epochs=10,validation_split = 0.2)
```

```
Epoch 1/10
1930/1930 [==============================] - 6s 3ms/step - loss: 3022.7256 - mae: 42.0722 - mse: 3022.7256 - val_loss: 2101.698
2 - val_mae: 36.1824 - val_mse: 2101.6982
Epoch 2/10
1930/1930 [==============================] - 5s 2ms/step - loss: 2078.9224 - mae: 36.1144 - mse: 2078.9224 - val_loss: 2103.202
1 - val_mae: 36.0389 - val_mse: 2103.2021
Epoch 3/10
1930/1930 [==============================] - 4s 2ms/step - loss: 2052.5652 - mae: 35.8368 - mse: 2052.5652 - val_loss: 2057.897
7 - val_mae: 35.8224 - val_mse: 2057.8977
Epoch 4/10
1930/1930 [==============================] - 4s 2ms/step - loss: 2034.3732 - mae: 35.5242 - mse: 2034.3732 - val_loss: 2023.955
2 - val_mae: 35.1695 - val_mse: 2023.9552
Epoch 5/10
1930/1930 [==============================] - 5s 3ms/step - loss: 2024.2976 - mae: 35.3301 - mse: 2024.2976 - val_loss: 2059.295
9 - val_mae: 35.3626 - val_mse: 2059.2959
Epoch 6/10
1930/1930 [==============================] - 5s 3ms/step - loss: 2016.8792 - mae: 35.2356 - mse: 2016.8792 - val_loss: 2083.858
2 - val_mae: 36.0076 - val_mse: 2083.8582
Epoch 7/10
1930/1930 [==============================] - 5s 3ms/step - loss: 2009.3063 - mae: 35.1034 - mse: 2009.3063 - val_loss: 2018.029
1 - val_mae: 35.0804 - val_mse: 2018.0291
Epoch 8/10
1930/1930 [==============================] - 5s 2ms/step - loss: 2003.5017 - mae: 35.0018 - mse: 2003.5017 - val_loss: 2007.879
3 - val_mae: 34.9806 - val_mse: 2007.8793
Epoch 9/10
1930/1930 [==============================] - 5s 2ms/step - loss: 1999.0471 - mae: 34.9385 - mse: 1999.0471 - val_loss: 2008.764
8 - val_mae: 34.9564 - val_mse: 2008.7648
Epoch 10/10
1930/1930 [==============================] - 5s 2ms/step - loss: 1993.1090 - mae: 34.8287 - mse: 1993.1090 - val_loss: 2068.678
5 - val_mae: 35.6681 - val_mse: 2068.6785
```

```
# Predicción mediante red neuronal
#Se importo numpy ya que hasta este punto no estaba declarado
import numpy as np
y_pred_dnn = np.floor(model.predict(X_validation))
```

Mediante Python y las librerías que consideres, muestra los resultados obtenidos por los diferentes algoritmos escogidos de forma gráfica y comparada/superpuesta. Existen diferentes medidas de variación para poder calcular el error predictivo en $\hat{Y}$.

- Suma total de los cuadrados (del inglés MAE): mida la variación de la variable dependiente Y (punto actual) y el valor predicho $\hat{Y}$
- Suma de los errores cuadrados (del inglés MSE): variación atribuida a los factores predictivos
- $\mathbf{R}^2$:La interpretación más habitual de la $\mathbf{R}^2$ es el grado de ajuste del modelo de regresión a los datos observados. Entre más tienda a 1, mejor será el modelo predictivo.

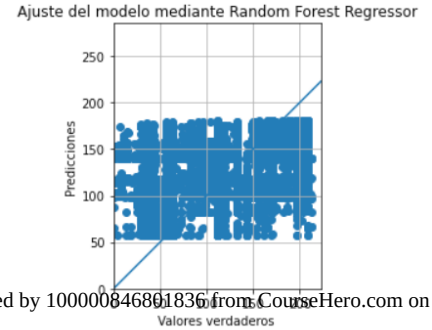
No hay un valor correcto para el MSEy el MAE. Cuanto más bajo sea el valor, mejor, y 0 significa que el modelo es perfecto.

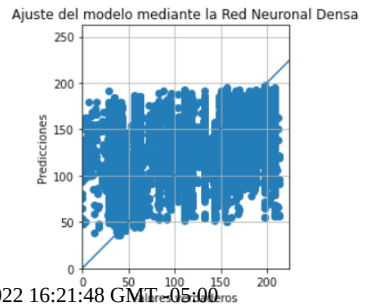
```
from sklearn import metrics
import numpy as np

print('Mean Absolute Error (MAE)  - Random Forest Regressor:', metrics.mean_absolute_error(y_pred_rs, Y_validation))
print('Mean Squared Error (MSE) - Random Forest Regressor:', metrics.mean_squared_error(y_pred_rs, Y_validation))
print('R^2:', metrics.r2_score(y_pred_rs, Y_validation))
print('Mean Absolute Error (MAE) -  Red Neurnoal Densa:', metrics.mean_absolute_error(y_pred_dnn, Y_validation))
print('Mean Squared Error (MSE) -  Red Neurnoal Densa:', metrics.mean_squared_error(y_pred_dnn, Y_validation))
print('R^2:', metrics.r2_score(y_pred_dnn, Y_validation))
```

```
Mean Absolute Error (MAE)  - Random Forest Regressor: 34.27449832148013
Mean Squared Error (MSE) - Random Forest Regressor: 1985.5188569915554
R^2: -0.7183753794427938
Mean Absolute Error (MAE) -  Red Neurnoal Densa: 33.968327
Mean Squared Error (MSE) -  Red Neurnoal Densa: 1974.3379
R^2: -0.5033767220011804
```

En las metricas anteriores nos dicen que el conjunto de datos es no se ve bien para ningun modelo pero la metrica es poco mejor para red neuronal, ya que los R^2 son negativos pero el de Red Neuronal es un poco mejor. Esto podria mejorarse con las capas y la funcion de activación y en mi caso aun cambiado estas metricas no mejora

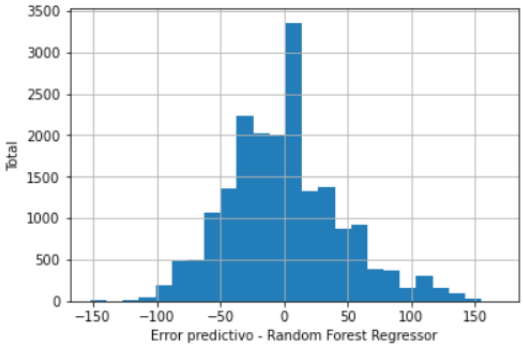
En las siguientes gráficas se observa el ajuste del modelo, valores deseados vs. verdaderos, a una línea de regresión. Entre más se ajusten a ella, será mejor el modelo.

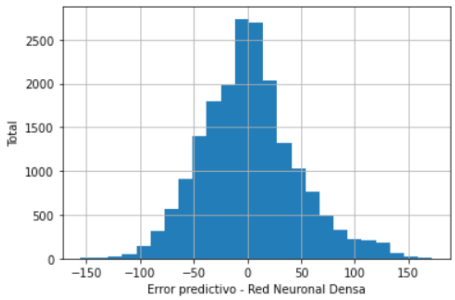




Asignatura	Datos del alumno	Fecha
Técnicas de inteligencia Artificial	Apellidos: González Rojas	Enero 2022
	Nombre: Samara Jocelyn	

En las siguientes gráficas se observa el patrón de error predictivo acumulado, entre más se visualice como una distribución normal (campana de Gauss), mejor será la interoperabilidad del modelo.





Conclusiones: Por lo que podemos ver en los resultados obtenidos ninguno de los modelos se ve muy optimo para el conjunto de datos y considero que si se debería de estar iterando en varios parámetros y ver con cuales se logra mejorar un poco.

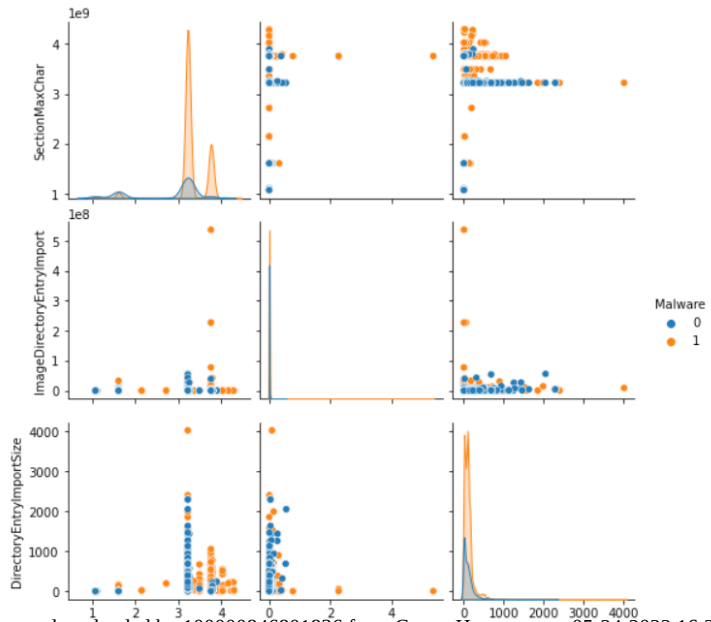
Apartado B (Clasificación)

```
import pandas as pd
data_malware = pd.read_csv('C:/Users/52553/Desktop/Mestria UNIR/2° semestre/Tecnicas de Inteligencia Artificial/dataset_malwares.
data_malware = data_malware.drop(columns=['Name'])
```

```
data_malware.head()
```

	e_magic	e_cblp	e_cp	e_crlc	e_cparhdr	e_minalloc	e_maxalloc	e_ss	e_sp	e_csum	...	SectionMaxChar	SectionMainChar	DirectoryEntryImport	Directo
0	23117	144	3	0	4	0	65535	0	184	0	...	3758096608	0	7	
1	23117	144	3	0	4	0	65535	0	184	0	...	3791650880	0	16	
2	23117	144	3	0	4	0	65535	0	184	0	...	3221225536	0	6	
3	23117	144	3	0	4	0	65535	0	184	0	...	3224371328	0	8	
4	23117	144	3	0	4	0	65535	0	184	0	...	3227516992	0	2	

5 rows × 78 columns



A diferencia del apartado A en esta grafica podemos observar que otro color adicional al azul y es porque nos dice como se agrupan los datos por clase, de igual manera se ve la distribución en la diagonal y en los cuadros adicionales se ve como se agrupan por clase.

Asignatura	Datos del alumno	Fecha
Técnicas de inteligencia Artificial	Apellidos: González Rojas	Enero 2022
	Nombre: Samara Jocelyn	

Elige al menos un método de clasificación no basado en redes neuronales (p.ej. regresión logística, árboles de decisión, reglas de clasificación, random forest, SVM, etc.).

```
#Máquina de soporte vectorial
from sklearn.svm import SVC
```

Incluye en el informe una explicación de los parámetros que consideres relevantes en cada ejecución.

- C: Parámetro de regularización. La fuerza de la regularización es inversamente proporcional a C. Compensa el efecto del sesgos, varianza.
- kernel $'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'$. Especifica el tipo de núcleo que se utilizará en el algoritmo, es decir, como se proyectan las muestras si no se pueden resolver en un problema lineal.

```
svm = SVC(C=10,kernel='rbf')
```

Mediante Python y las librerías que consideres, entrena el modelo o modelos escogidos, dividiendo el dataset en datos de entrenamiento y datos de test previamente en base a tu criterio y pruébalos frente a dichos datos de test.

```
#Etiquetas y muestras
y = data_malware['Malware']
X = data_malware.drop(columns = ['Malware'])
```

```
#Pre-procesamiento y estandarización
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaler = scaler.fit_transform(X)
print(X_scaler)
```

```
[[ 0.         -0.03506542 -0.04751096 ... -0.07054894 -0.0198525
  -0.04066791]
 [ 0.         -0.03506542 -0.04751096 ... -0.03849221 -0.02110877
  -0.02469983]
 [ 0.         -0.03506542 -0.04751096 ... -0.07599254 -0.02110877
  -0.04066791]
 ...
 [ 0.         -0.18093613 -0.04958686 ... -0.07296832 -0.02110877
  -0.04066791]
 [ 0.         -0.03506542 -0.04751096 ... -0.06691988 -0.02110877
  -0.04066791]
 [ 0.         -0.03506542 -0.04751096 ...  0.00021781 -0.02110877
  -0.04066791]]
```

```
from sklearn.model_selection import train_test_split
X_train, X_validation, Y_train, Y_validation = train_test_split(X_scaler, y, test_size=0.20, random_state=1, shuffle=True)
```

```
# Entrenamiento
svm.fit(X_train,Y_train)
```

```
SVC(C=10)
```

```
#Predicción
y_pred_svm = svm.predict(X_validation)
```

Elige al menos una arquitectura de red neuronal (**describe en el informe** las neuronas en la capa de entrada, las capas intermedias $H_1 \in \mathbb{R}^{32}$ y $H_2 \in \mathbb{R}^{24}$ **al menos dos capas intermedias** y capa de salida $O \in \mathbb{R}$, funciones de activación en cada caso, **al menos utiliza relu en algunas de las**

```
import tensorflow as tf
from tensorflow.keras import *
from keras.models import Sequential
model = Sequential([
    layers.Dense(64, activation='relu', input_shape=[len(X_train[0])]), #entrada
    layers.Dense(32, activation='relu'), #intermedia
    layers.Dense(1,activation='tanh')]) #salida tangente hiperbólica, de la familia sigmoide
model.compile(loss='binary_crossentropy',optimizer='ADAM',metrics=['accuracy'])
model.summary()
#Le quitamos una capa ya que esto ayuda a mejorar el funcionamiento de la Red Neuronal
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 64)                4992

 dense_4 (Dense)             (None, 32)                2080

 dense_5 (Dense)             (None, 1)                 33

=================================================================
Total params: 7,105
Trainable params: 7,105
Non-trainable params: 0
_________________________________________________________________
```

Mediante Python y utilizando **al menos Keras sobre TensorFlow 2.0 (tensorflow.keras)**, entrena el modelo o modelos de red neuronal escogidos, dividiendo el dataset en datos de entrenamiento y datos de test previamente en base a tu criterio y pruébalos frente a dichos datos de test.

```
#entrenamiento
modelfit(X_train,Y_train,epochs=10)
```

Asignatura	Datos del alumno	Fecha
Técnicas de inteligencia Artificial	Apellidos: González Rojas	Enero 2022
	Nombre: Samara Jocelyn	

```
Epoch 6/10
491/491 [==============================] - 1s 3ms/step - loss: 0.0889 - accuracy: 0.9772
Epoch 7/10
491/491 [==============================] - 1s 3ms/step - loss: 0.0983 - accuracy: 0.9735
Epoch 8/10
491/491 [==============================] - 1s 3ms/step - loss: 0.0838 - accuracy: 0.9792
Epoch 9/10
491/491 [==============================] - 1s 3ms/step - loss: 0.0841 - accuracy: 0.9788
Epoch 10/10
491/491 [==============================] - 1s 3ms/step - loss: 0.0815 - accuracy: 0.9788
```

```
#Predicción
import numpy as np
y_pred_dnn = np.ceil(model.predict(X_validation).flatten())
#Transformar a entero las predicciones
y_pred_dnn_int = [abs(y) for y in y_pred_dnn]
```

Mediante Python y las librerías que consideres, muestra los resultados obtenidos por los diferentes algoritmos escogidos de forma gráfica y comparada/superpuesta

```
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import classification_report
```

```
#Reporte de clasificación svm
print(classification_report(y_pred_svm,Y_validation))
```

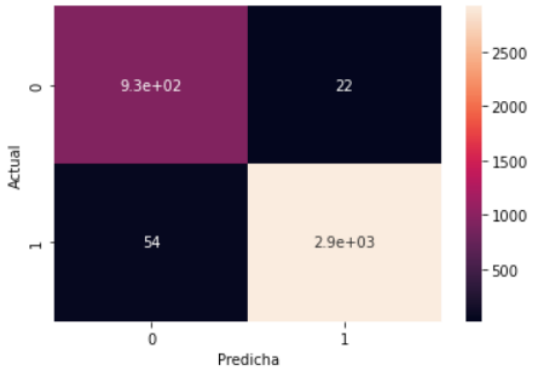
	precision	recall	f1-score	support
0	0.95	0.98	0.96	951
1	0.99	0.98	0.99	2972
accuracy			0.98	3923
macro avg	0.97	0.98	0.97	3923
weighted avg	0.98	0.98	0.98	3923

```
#Reporte de clasificación Red Neuronal Densa
print(classification_report(y_pred_dnn_int,Y_validation))
```

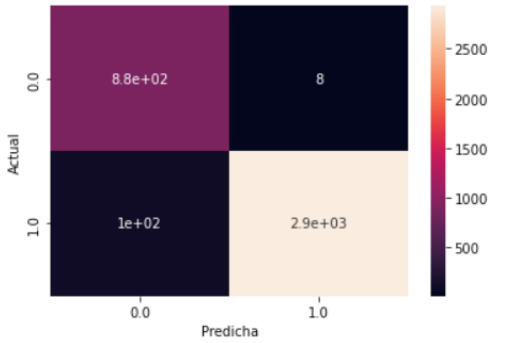
	precision	recall	f1-score	support
0.0	0.90	0.99	0.94	890
1.0	1.00	0.97	0.98	3033
accuracy			0.97	3923
macro avg	0.95	0.98	0.96	3923
weighted avg	0.97	0.97	0.97	3923

Lo que nos dicen estos reportes es que nos conviene mas SVC porque sus números son mas cercanos a 100 y la Red Neuronal aunque no salio mal el reporte de SVC se ve mejor y esto es porque la Red Neuronal es un método muy complejo para un dataset muy pequeño. Y las siguientes gráficas confirman lo dicho

```
df_svm.columns=['y_Actual','y_Predicha']
confusion_matrix = pd.crosstab(df_svm['y_Actual'], df_s
sns.heatmap(confusion_matrix, annot=True)
plt.show()
```




```
df_dnn = pd.DataFrame([y_pred_dnn_int,Y_validatio
df_dnn.columns=['y_Actual','y_Predicha']
confusion_matrix = pd.crosstab(df_dnn['y_Actual']
sns.heatmap(confusion_matrix, annot=True)
plt.show()
```



Las graficas confirman los antes mencionado