

ENTREGA FINAL DE PROYECTO

POR:

ANDRES AFANADOR GUIZA

SEBASTIAN VILLEGAS ROJAS

MATERIA:

INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL

PROFESOR:

RAÚL RAMOS POLLÁN



FACULTAD DE INGENIERÍA

DEPARTAMENTO DE ELÉCTRICA

UNIVERSIDAD DE ANTIOQUIA

MEDELLÍN

2023-1

Introducción

Dada una señal de voltaje o corriente con distorsiones en la calidad de la energía vamos a clasificar los fenómenos, esto lo haremos mediante un método de extracción de características, bien se llama T.Wavelet, T.Fourier, T.Stockwel.

Y luego de haber extraído las características en el dominio de la frecuencia vamos a clasificar los fenómenos dentro de la onda.

A continuación, se muestramos imágenes para que sea más claro la clase de fenómenos a clasificar junto con un ejemplo generalizado de cómo funcionan este tipo de algoritmos:

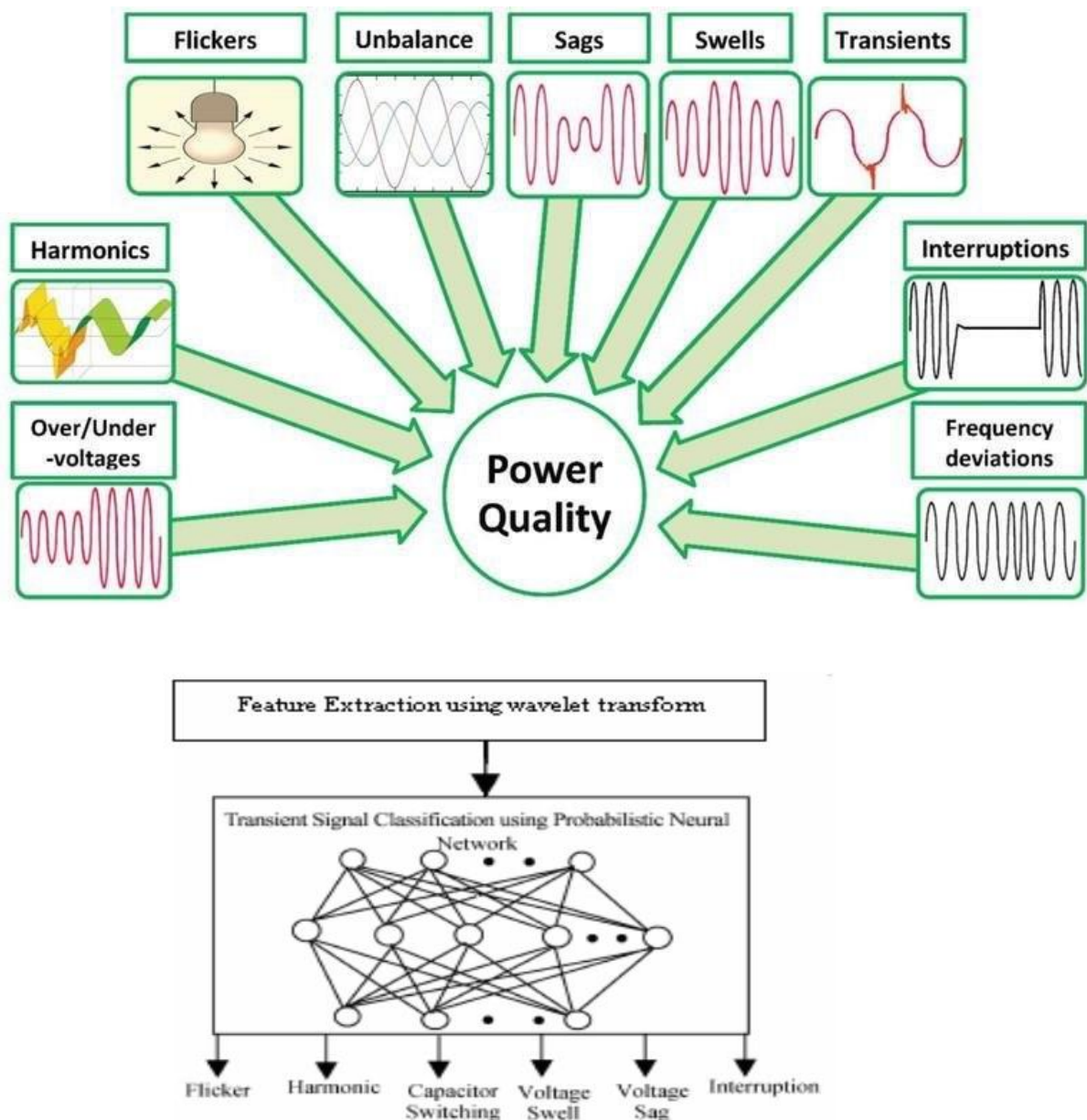


Figura 1. Fenómenos para clasificar y ejemplo de funcionamiento.

Métricas

El problema a trabajar es de una sola etiqueta a la vez ó sea solo vamos a clasificar si el fenómeno de la calidad es o no es sag, si el fenómeno de la calidad es o no es swell, y así con los demás fenómenos, por ende, se medirá la predicción de la IA con una sola etiqueta a la vez como:

$$Accuracy = (Number\ of\ correct\ prediction / (Total\ of\ prediction))$$

Aparte de esto miramos el coeficiente de determinación, error absoluto medio y el error promedio cuadrático:

$$R^2 = \frac{\sum_{t=1}^T (\hat{Y}_t - \bar{Y})^2}{\sum_{t=1}^T (Y_t - \bar{Y})^2}$$

$$EAM = \frac{\sum_{i=1}^n |y_i - x_i|}{n} = \frac{\sum_{i=1}^n |e_i|}{n}.$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$$

Las métricas de desempeño esperadas es tener una exactitud mayor o igual al 90%, en el tiempo que se demora en clasificar se espera un tiempo promedio menor a 1 segundo para cada fenómeno.

Exploración descriptiva del dataset:

La base de datos la vamos a generar mediante un algoritmo en Matlab que se encuentra liberado por un paper de investigación, esto lo hacemos para a la hora de entrenar la inteligencia artificial tengamos bien claro la clase de fenómeno que le introducimos a esta y de igual manera tener claro cuál fenómeno de la calidad de la energía estamos clasificando y poder dar un margen de error del programa en su predicción cuando el programa ya le estamos testeando su funcionamiento.

A continuación, se comparte el paper de investigación que nos permitirá generar la base de datos: [sciencedirect.com/science/article/pii/S0378779621001334](https://www.sciencedirect.com/science/article/pii/S0378779621001334).

El dataset utilizado en este proyecto consiste en conjuntos de datos etiquetados de ondas eléctricas, clasificadas en "Con impurezas" y "Onda sana". Se han descargado los conjuntos de datos de entrenamiento y prueba, y se ha realizado un análisis exploratorio para comprender mejor la estructura y las características de los datos.

La base de datos fue generada de manera artificial mediante un código que se obtuvo de un artículo de investigación, esta contiene para el 5000 vectores con sag y 5000 sin ninguna impureza, para luego tener en el periodo de testeo 10000 vectores con sag y 10000 sin ninguna impureza.

Estos vectores son todos de una onda de 60 Hz y de 10 ciclos, con valores normalizados para facilitar el programa, son subidos en un archivo .zip dado a que grandes tamaños de bases de datos, no nos dejaba correr en colab.

Extracción de características:

En el proceso de la IA por ahora llevamos hasta lo que sería la extracción de características de la base de datos, estos los definimos con las siguientes funciones:

```

def std(vecstd):
    #Funcion para la desviacion estandar
    return np.std(vecstd)
def mean(meanvec):
    #Funcion para el promedio
    return np.mean(meanvec)
def counth(vch):
    #Funcion para hacer la cuenta de los valores encima de 1
    vecn1=abs(vch)
    count1=0
    for i in vecn1:
        if abs(i)>1:
            count1=count1+1
    return count1
def countlow(vcl):
    #Funcion para contar las veces que el valor es menor a 0.1
    veclow=abs(vcl)
    clow=0
    for i in veclow:
        if abs(i)<=0.1:
            clow=clow+1
    return clow
def countinter(vint):
    #Funcion para contar las veces que se encuentra en el intervalo 0.1 y 1
    vinter=abs(vint)
    cinter=0
    for i in vinter:
        if abs(i)>0.1 and abs(i)<=1:
            cinter=cinter+1
    return cinter

```

Figura 2. Primeras funciones para extracción de características.

Las funciones std y mean, son la desviación estándar y el promedio respectivamente de una onda completa, ya que cuando la función es sana estos darán unos valores cercanos, pero cuando haya un bajón de tensión, estas tendrán un promedio lejano a cero y una desviación estándar por ende diferente a la de una función sana.

- Se extrajeron características como la desviación estándar, el promedio, el conteo de valores por encima de 1, el conteo de valores menores a 0.1 y el conteo de valores en el intervalo de 0.1 a 1. Las otras funciones son counth que cuenta las veces que el valor absoluto de la onda es mayor a 1, countlow que cuenta las veces que el valor absoluto de la onda está entre 0 y 0.1 , y finalmente countinter que cuenta las veces que la onda está entre 0.1 y 1.

Se calculó una característica llamada THD (Total Harmonic Distortion) para cada onda eléctrica. THD: El THD se define como la distorsión armónica de una onda, esta indica que tan distorsionada está la señal con respecto a su frecuencia fundamental que en nuestro caso siempre son 60 Hz, definiéndose matemáticamente así:

$$THD_v(\%) = \sqrt{\left(\frac{V_{rms}}{V_1}\right)^2 - 1} \times 100$$

Figura 3. THD de una función.

Donde V_1 es la magnitud original con la transformada de Fourier y V_{rms} es :

$$V_{rms} = \sqrt{\sum_{h=0}^{\infty} I_h^2}$$

1. Definición de V_{rms} .

```
[ ] def RMS(Vrms1):
    Vrms= np.abs(Vrms1)
    return sqrt(sum(np.array(Vrms)**2))
def THD(Vthd):
    Vthd= np.abs(Vthd)
    return sqrt((RMS(Vthd)/list(Vthd)[1])**2-1)*100
print(THD(list(X2)[0:15]))
print(list(np.abs(X2))[0:15])
```

Figura 4. Función RMS y THD.

Esta será importante puesto que nos ayudará a diferenciar una onda con problemas de una dañada, y en nuestro caso se hará para cada ciclo de los 10 ciclos de cada onda , pero tomaremos en cuenta sólo el valor mayor ,el menor, y el promedio del THD para los 10 ciclos devueltos en una lista con el siguiente código:

```
def THDIA(vTHDIA1):
    "Esta funcion me saca el THD promedio de todos los ciclos, el menor y el mayor"
    #Dividimos el vector en su numero de ciclos en una lista
    vTHDIA=list(vTHDIA1)
    interval=[]
    ffi=[]
    THDr=[]
    for i in range(1,m.floor(len(vTHDIA)/53.3)):
        num1=m.floor(53.3*(i-1))
        num2=m.floor(53.3*(i))
        interval=vTHDIA[num1:num2]
        #Hacemos la transformada de fourier y lo añadimos a una lista
        ffi=fftpack.fft(interval)

        #Le aplicamos el THD
        THDr.append(THD(list(ffi)[0:25]))
        #Le aplicamos el LHD que es THD pero considerando solo los primeros 7 frecuencias

    #Vamos a retornar una lista de 6 numeros
    mp=[np.mean(THDr),min(THDr),max(THDr)]
    return mp
```

Figura 5. Función de THD Max, mínimo y promedio.

- El conjunto de entrenamiento (Strain y Ntrain) contiene información sobre las ondas eléctricas con y sin impurezas, mientras que el conjunto de prueba (Stest y Ntest) se utiliza para evaluar el rendimiento de los modelos.
- Se realizó un preprocesamiento de los datos para extraer características estadísticas y aplicar la Transformada de Fourier (FFT) a las ondas eléctricas.

Iteraciones de desarrollo:

- Preprocesado de datos:
 - Se realizaron cálculos de características estadísticas y de transformada de Fourier para obtener una representación numérica de las ondas eléctricas.
- Modelos supervisados:
 - Se implementó un modelo de clasificación de Máquinas de Vectores de Soporte (SVC) utilizando las características extraídas.
- Resultados y métricas:
 - Se evaluó el desempeño del modelo SVC utilizando los conjuntos de datos de entrenamiento y prueba.
 - Se calcularon métricas como el coeficiente de determinación, el error absoluto medio y el error cuadrático medio.
 - Se realizó lo anterior con Random Forest y Logisticregression.
- Modelos no supervisados:
 - Se implementó el algoritmo de clustering K-Means (KMeans) para agrupar los datos de prueba según las características extraídas.
- Resultados y métricas:
 - Se visualizó la agrupación de los datos utilizando un gráfico de dispersión.
 - Se observó que las ondas eran todas iguales, un problema con la base de datos que no se puede cambiar ya que generadores de ondas sanas y con perturbaciones que nosotros necesitamos no abundan mucho.
 - Se exploraron los centroides y se analizó la distribución de los datos en los grupos.

Retos y consideraciones de despliegue:

Durante el desarrollo del proyecto, se encontraron varios retos y consideraciones importantes:

- Recolección y preparación de datos: Se requirió descargar los conjuntos de datos de repositorios de GitHub y realizar un preprocesamiento adecuado para extraer características relevantes para el modelado.
- Selección de modelos: Se exploraron modelos supervisados (SVC, RandomForestClassifier, LogisticRegression) y no supervisados (KMeans) para realizar tareas de clasificación y agrupación de datos.
- Evaluación y optimización de modelos: Se utilizaron diversas métricas para evaluar el rendimiento de los modelos y se buscó optimizar sus parámetros para obtener mejores resultados.
- Visualización de Curvas de aprendizaje:
 - Se realizaron evaluaciones de desempeño utilizando métricas como la exactitud, el coeficiente de determinación, el error absoluto medio y el error cuadrático medio.
 - Se generaron curvas de aprendizaje para analizar el rendimiento de los modelos en función del tamaño del conjunto de datos de entrenamiento.

Durante el desarrollo del proyecto, se utilizaron diversas técnicas de visualización para explorar y presentar los datos. A continuación, se describen algunas de las visualizaciones más relevantes utilizadas:

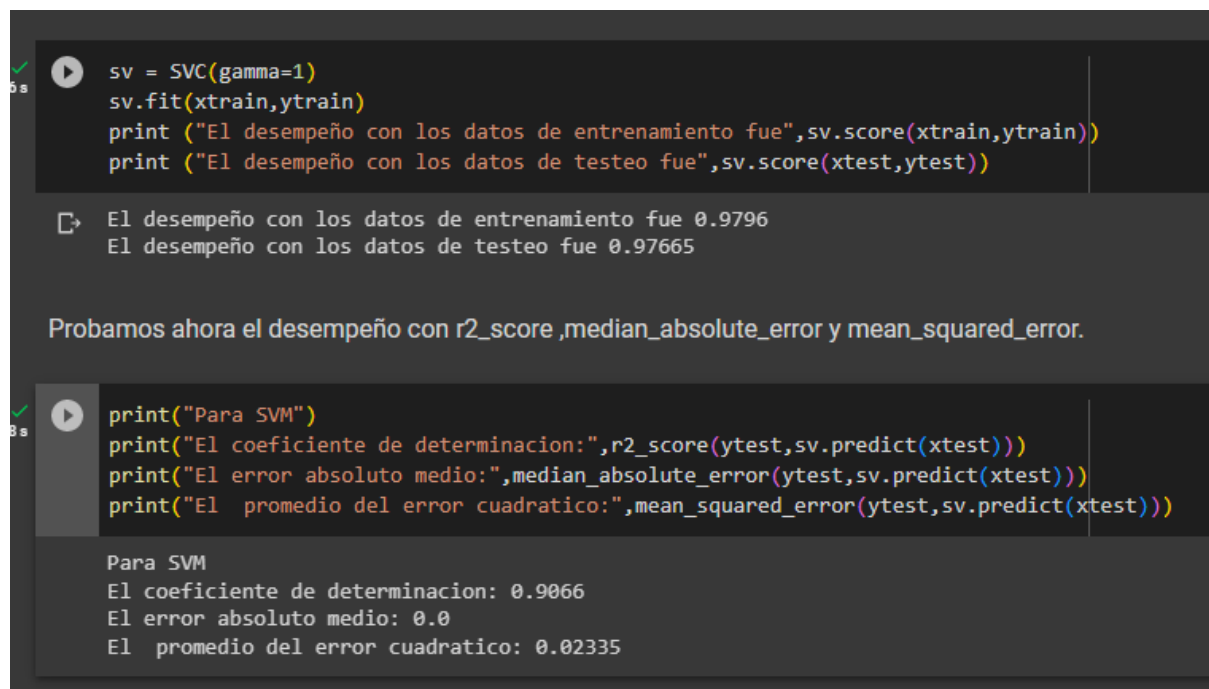
-Gráficos de dispersión: Se utilizaron gráficos de dispersión para visualizar la relación entre dos variables, como el promedio de la onda y el THD promedio. Estos gráficos permitieron identificar patrones y tendencias en los datos.

-Gráficos de barras: Se emplearon gráficos de barras para representar la cantidad de valores encima de 1, menores a 0.1 y en el intervalo entre 0.1 y 1. Estas visualizaciones ayudaron a comprender la distribución de los valores en las ondas eléctricas.

-Gráficos de líneas: Se generaron gráficos de líneas para visualizar las formas de onda de las señales eléctricas. Estos gráficos permitieron observar las variaciones en la amplitud y el tiempo de las ondas.

-Gráficos de centroides: Se realizaron gráficos de dispersión para mostrar los centroides obtenidos mediante el algoritmo de clustering KMeans. Estos gráficos ayudaron a visualizar la ubicación de los centroides y su relación con las instancias del conjunto de datos.

-Curvas de aprendizaje: Se observó que el tamaño la base de datos fue optimo para los sistemas supervisados dado a que las métricas de exactitud como `r2_score` , `median absolute error` y `mean square error` obtuvieron para `randomforest`, `svm` y `logisticregression` un bajo error y una exactitud encima del 80%.



```
sv = SVC(gamma=1)
sv.fit(xtrain,ytrain)
print ("El desempeño con los datos de entrenamiento fue",sv.score(xtrain,ytrain))
print ("El desempeño con los datos de testeo fue",sv.score(xtest,ytest))
```

El desempeño con los datos de entrenamiento fue 0.9796
El desempeño con los datos de testeo fue 0.97665

Probamos ahora el desempeño con `r2_score` ,`median_absolute_error` y `mean_squared_error`.

```
print("Para SVM")
print("El coeficiente de determinacion:",r2_score(ytest,sv.predict(xtest)))
print("El error absoluto medio:",median_absolute_error(ytest,sv.predict(xtest)))
print("El promedio del error cuadratico:",mean_squared_error(ytest,sv.predict(xtest)))
```

Para SVM
El coeficiente de determinacion: 0.9066
El error absoluto medio: 0.0
El promedio del error cuadratico: 0.02335

Figura 6.Resultados con SVM.


```
xtrain,ytrain=data(Strain,Ntrain)
xtest,ytest=data(Stest,Ntest)

rf = RandomForestClassifier(n_estimators=10, max_depth=10)
rf.fit(xtrain,ytrain)
print("Con Random Forest:")
print ("El desempeño con los datos de entrenamiento fue",rf.score(xtrain,ytrain))
print ("El desempeño con los datos de testeo fue",rf.score(xtest,ytest))
print("Para coeficiente de determinacion:",r2_score(ytest,rf.predict(xtest)))
print("Para el error absoluto medio:",median_absolute_error(ytest,rf.predict(xtest)))
print("Para el promedio del error cuadratico:",mean_squared_error(ytest,rf.predict(xtest)))
```

Con Random Forest:
El desempeño con los datos de entrenamiento fue 0.9989
El desempeño con los datos de testeo fue 0.99875
Para coeficiente de determinacion: 0.995
Para el error absoluto medio: 0.0
Para el promedio del error cuadratico: 0.00125

Figura 7. Resultados con Random Forest.

```
xtrain,ytrain=data(Strain,Ntrain)
xtest,ytest=data(Stest,Ntest)

lr = LogisticRegression()
lr.fit(xtrain,ytrain)
print("Para logistic regression:")
print ("El desempeño con los datos de entrenamiento fue",lr.score(xtrain,ytrain))
print ("El desempeño con los datos de testeo fue",lr.score(xtest,ytest))
print("Para coeficiente de determinacion:",r2_score(ytest,lr.predict(xtest)))
print("Para el error absoluto medio:",median_absolute_error(ytest,lr.predict(xtest)))
print("Para el promedio del error cuadratico:",mean_squared_error(ytest,lr.predict(xtest)))
```

El desempeño con los datos de entrenamiento fue 0.9789
El desempeño con los datos de testeo fue 0.9753
Para coeficiente de determinacion: 0.9012
Para el error absoluto medio: 0.0
Para el promedio del error cuadratico: 0.0247

Figura 7. Resultados con Logisticregression.

Por otro lado graficamos el mapa de los centroides del método no supervisado y obtuvimos:

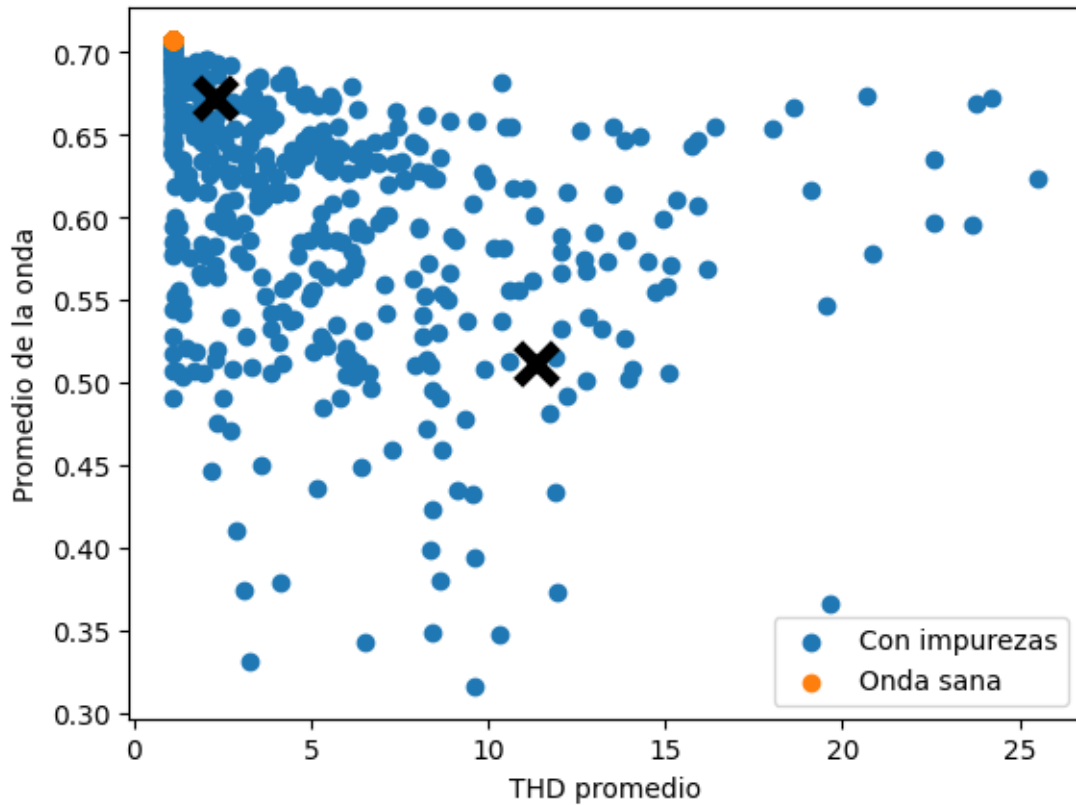


Figura 8. Graficas de centroides.

Nótese que todas ondas sanas tienen un problema y es que presentan un promedio de onda igual, esto ya no se puede modificar dado a que la base de datos fue sacada de un paper de investigación , y estas bases no son tan fáciles conseguir, por esto decidimos graficar las 100 primeras ondas sanas y obtuvimos :

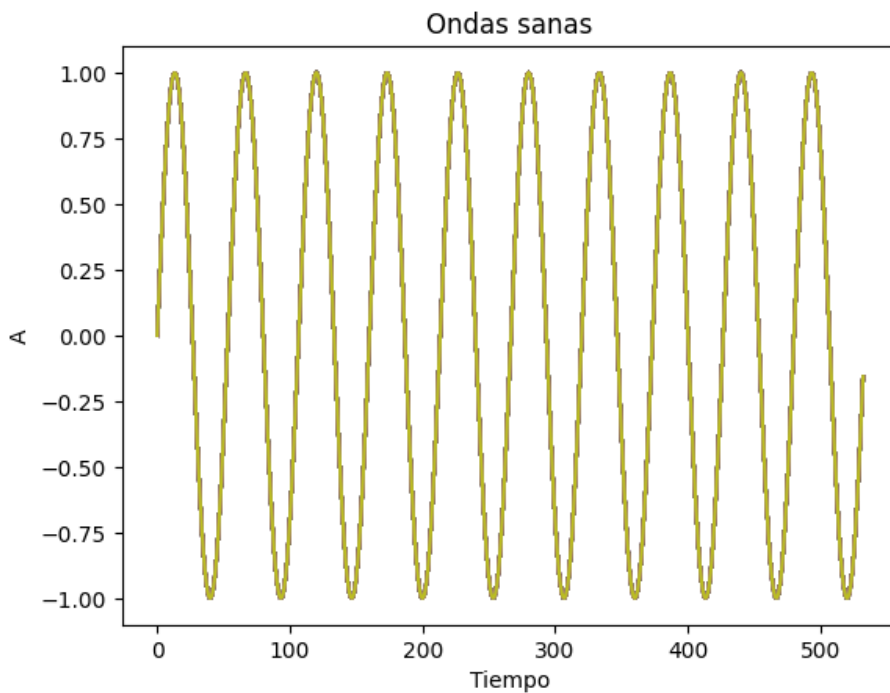


Figura 9. Graficas de ondas sanas.

A diferencia de las ondas con perturbación que si eran distintas entre sí:

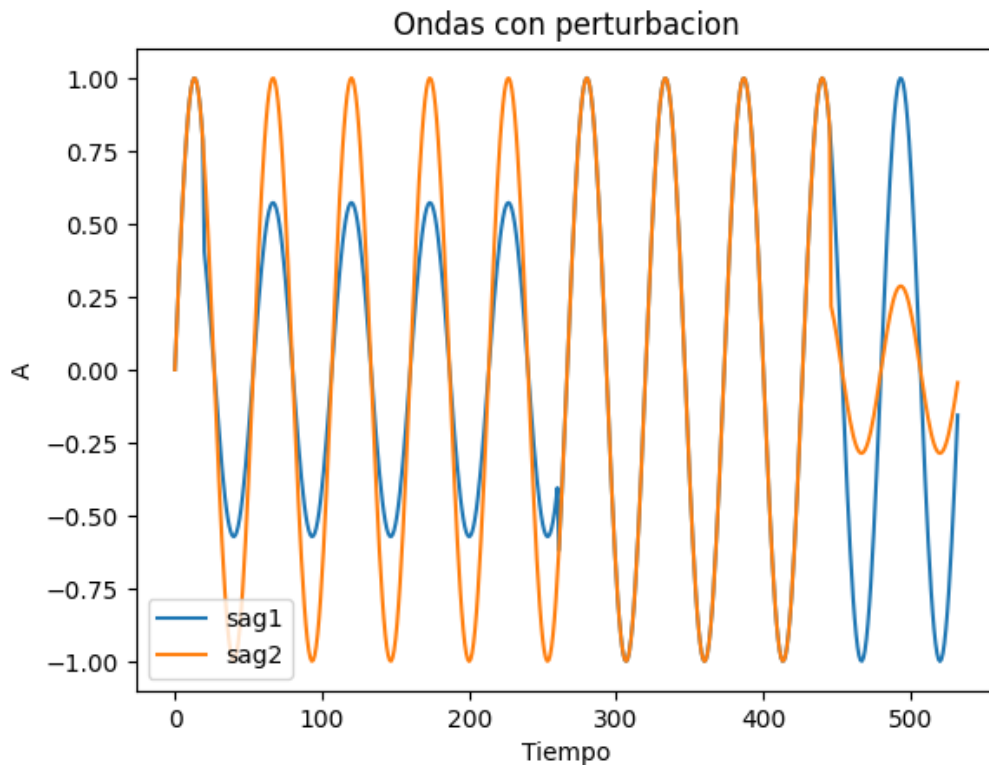


Figura 10. Graficas de ondas con sag.

Las visualizaciones mencionadas contribuyeron significativamente a la comprensión de los datos y los resultados obtenidos durante el proyecto. Proporcionaron una representación visual clara y concisa de las características de las ondas eléctricas, la clasificación realizada por los modelos y las relaciones entre las variables. Además, facilitaron la comunicación de los hallazgos y conclusiones del proyecto de manera efectiva.

Conclusiones:

Basado en el análisis realizado, se pueden obtener las siguientes conclusiones:

- Los modelos supervisados (SVC, RandomForestClassifier, LogisticRegression) mostraron un buen desempeño en la clasificación de las ondas eléctricas.
- El modelo SVC logró una alta precisión en la clasificación de las ondas eléctricas, tanto en los datos de entrenamiento como en los datos de prueba, no obstante, no fue el mejor método para clasificarlas.
- El método de RandomForest presento la mejor exactitud al etiquetar si la onda tenía problemas o no.
- La base de datos de las ondas sin distorsión tuvo el problema de que todas eran iguales, al no tener desfase entre ellas.
- El modelo de clustering KMeans tuvo problemas ya que había mucha cercanía entre las características de las ondas sanas y las ondas distorsionadas, esto es obvio ya que una pequeña distorsión o bajón en la onda es más difícil de detectar que un gran bajón.

- Se observó que las métricas de evaluación de los modelos varían dependiendo del modelo utilizado y de las características específicas del conjunto de datos.
- Las curvas de aprendizaje revelaron que el desempeño de los modelos mejoró a medida que se incrementaba el tamaño del conjunto de datos de entrenamiento.

En resumen, este proyecto exploró la clasificación y agrupación de ondas eléctricas utilizando técnicas de aprendizaje automático. Los modelos supervisados y no supervisados implementados mostraron resultados prometedores en la tarea de identificar ondas eléctricas con y sin impurezas. Sin embargo, se debe de realizar más investigación y ajustes en los modelos para obtener un desempeño aún más sólido. Además, se debe tener en cuenta que el despliegue de los modelos en un entorno real puede requerir consideraciones adicionales, como la escalabilidad y el tiempo de respuesta.