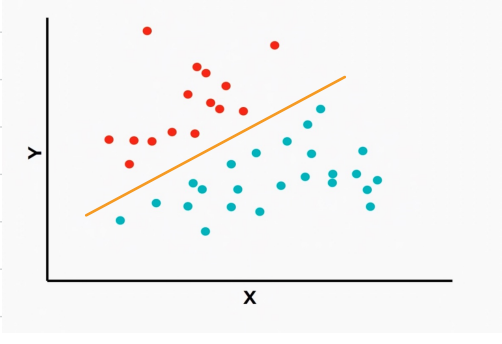
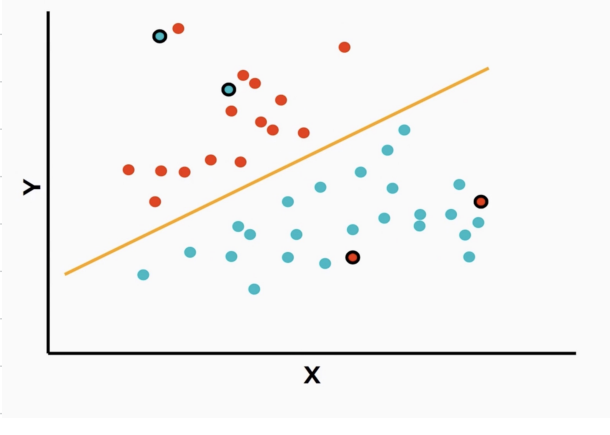
Maquinas de vectores de soporte o SVMs

Son otro algoritmo de aprendizaje supervisado y sirve para clasificación y regresión



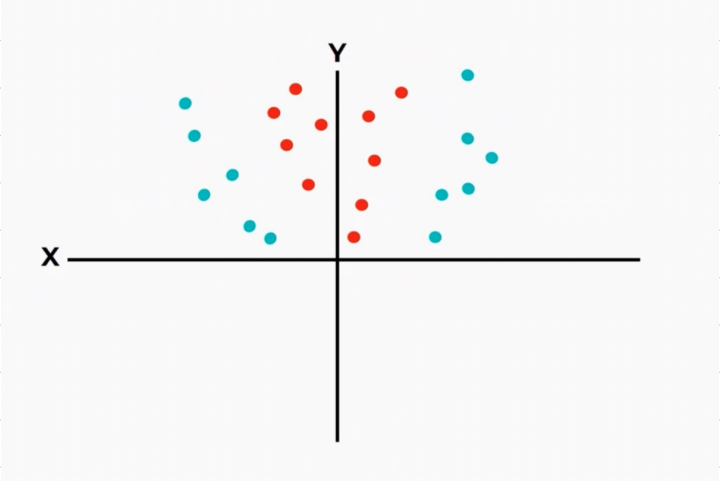
Es ideal para sets de datos con "pocos" registros de datos y multiples caracteristicas.

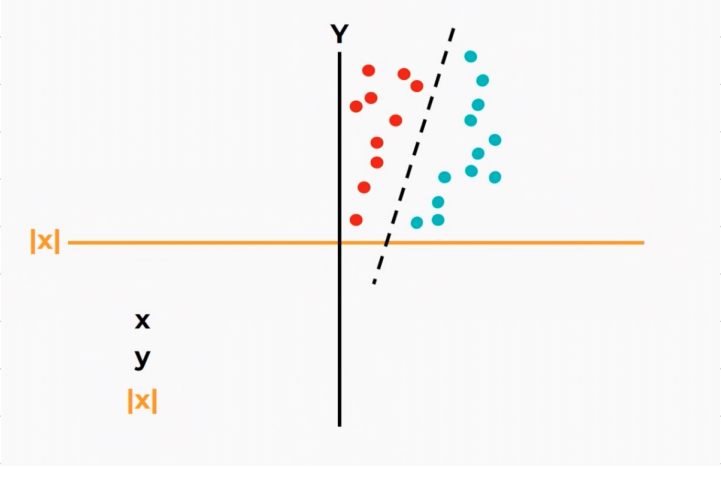
Siempre busca la mejor forma de separar los datos, una ventaja es que cuando hay datos atipico cuenta con parametros ajustables que los ignora para maximizar el margen entre los vectores de soporte



tiene la capacidad de transformar datos para encontrar la separación entre ellos

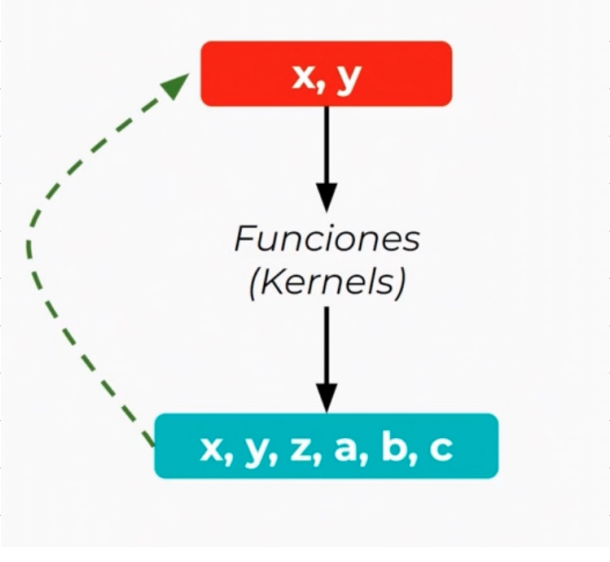
En este ejemplo no se pueden separar los datos con una linea recta





Los SVMs crean estas características por su cuenta, usando funciones (Kernels) y transforman

las dimensiones pequeñas (x, y) en dimensiones mas complejas (x, y, z, a, b, c) y devuelven los resultados con las características originales



Los SVMs cuentan con parametros ajustables como el Kernel, el dato "C" que controla que tan plana o ajustada debe quedar la linea de separacion y el dato Gamma que le dice si debe considerar los puntos más cercanos o lejanos a la linea.

Parte Practica

importación del set de datos de cancer de mama de sciKit-learn

```
from sklearn.datasets import load_breast_cancer

#tener una variable con la informacion
data = load_breast_cancer()

X = data.data
y = data.target
```

Se puede crear un DataFrame para evaluar y analizar los datos

```
import pandas as pd

x_df = pd.DataFrame(X,columns=data.feature_names)
```

metodos como: .info(), .describe(), .isna()

Para ver la composición de los datos y realizar analisis a partir de ahi.

Luego el proceso de rutina, separación de los datos en los diferentes conjuntos

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=.3)
```

Realizar la importación del modelo y metricas

```
from sklearn import svm
from sklearn import metrics

# support vector clasifier
modelo = svm.SVC()

modelo.fit(X_train, y_train)
predicciones = modelo.predict(X_test)
print(f"Exactitud: ", metrics.accuracy_score(y_test,predicciones))
```

Cambiando los parametros del modelo podemos tener mejores indices de resultados.

En el siguiente enlace se pueden ver las diferentes opciones:

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

```
kernels = ["linear", "rbf", "sigmoid", 'poly', 'precomputed']
gammas = [1, 0.1, 0.001, 0.0001, 0.00001]


for kernel in kernels:
  for gamma in gammas:
    modelo = svm.SVC(kernel=kernel, gamma = gamma)
    modelo.fit(X_train, y_train)
    predicciones = modelo.predict(X_test)
    print(f"{kernel} - {gamma} Exactitud: ", metrics.accuracy_score(y_test,predicciones))
```

Nos quedamos con los parametros que mejor se ajustan en el caso de "linear" el gamma no influye en el entrenamiento.

```
# Nos quedamos con el modelo y parametros que mejor resultados nos den

modelo = svm.SVC(kernel='linear')
modelo.fit(X_train, y_train)
predicciones = modelo.predict(X_test)
print(f"{kernel} - {gamma} Exactitud: ", metrics.accuracy_score(y_test,predicciones))
```

Una vez entrenado solamente restaria evaluar los resultados.

```
from sklearn.metrics import confusion_matrix
import pandas as pd

pd.DataFrame( confusion_matrix(y_pru, predicciones))
```