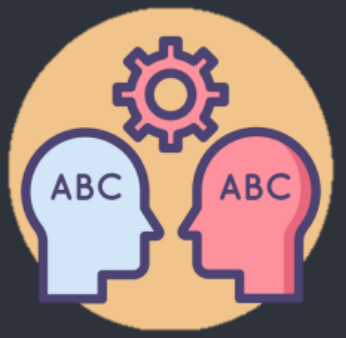
Procesamiento de Lenguaje Natural{



[NLP]

<Aprendizaje Supervisado | Clasificacion | Maquinas de Soporte Vectorial>

}

Ivan Rojas Gonzalez – Universidad Panamericana

La Agenda de hoy {

01 Intuicion sobre maquinas de soporte vectorial

<Cómo funciona>

02 Implementando Maquinas de Soporte Vectorial

<Llevando nuestro clasificador a la vida >

03 Evaluando el clasificador

<Que tan bueno es nuestro clasificador?>

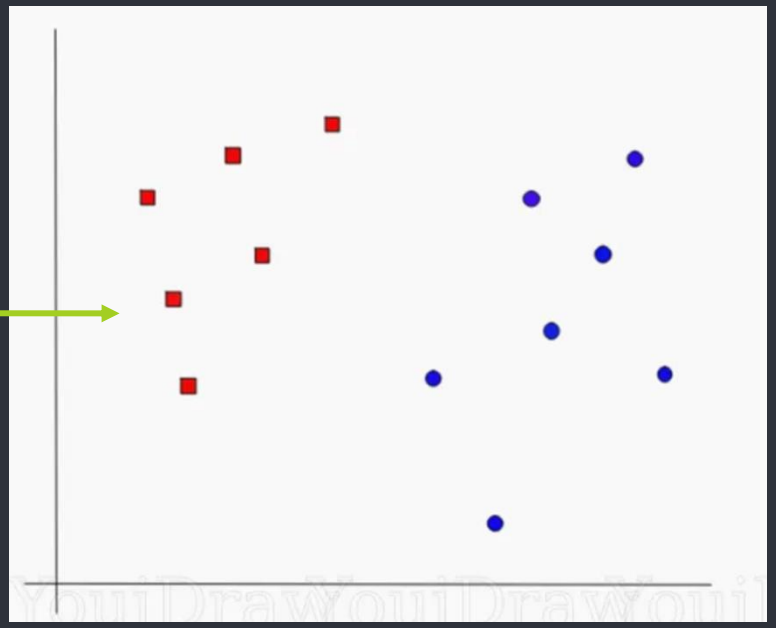
}

{Maquinas de Soporte Vectorial}

Ivan Rojas Gonzalez – Universidad Panamericana

Maquinas de Soporte Vectorial{

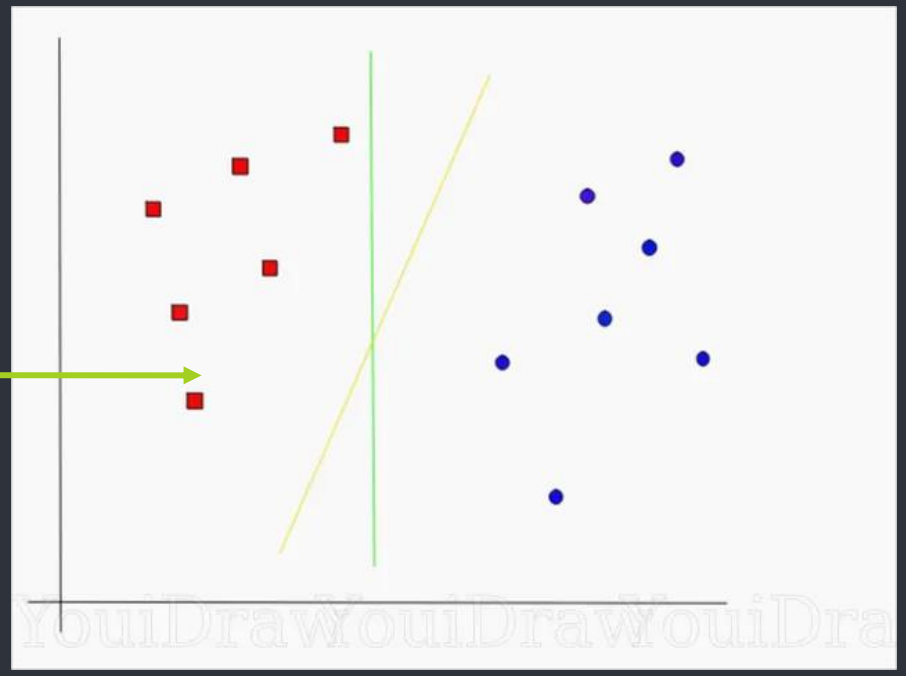
Imaginemos que queremos separar ambas clases usando una linea



}

Maquinas de Soporte Vectorial{

Cual de las dos lineas lo hace mejor?

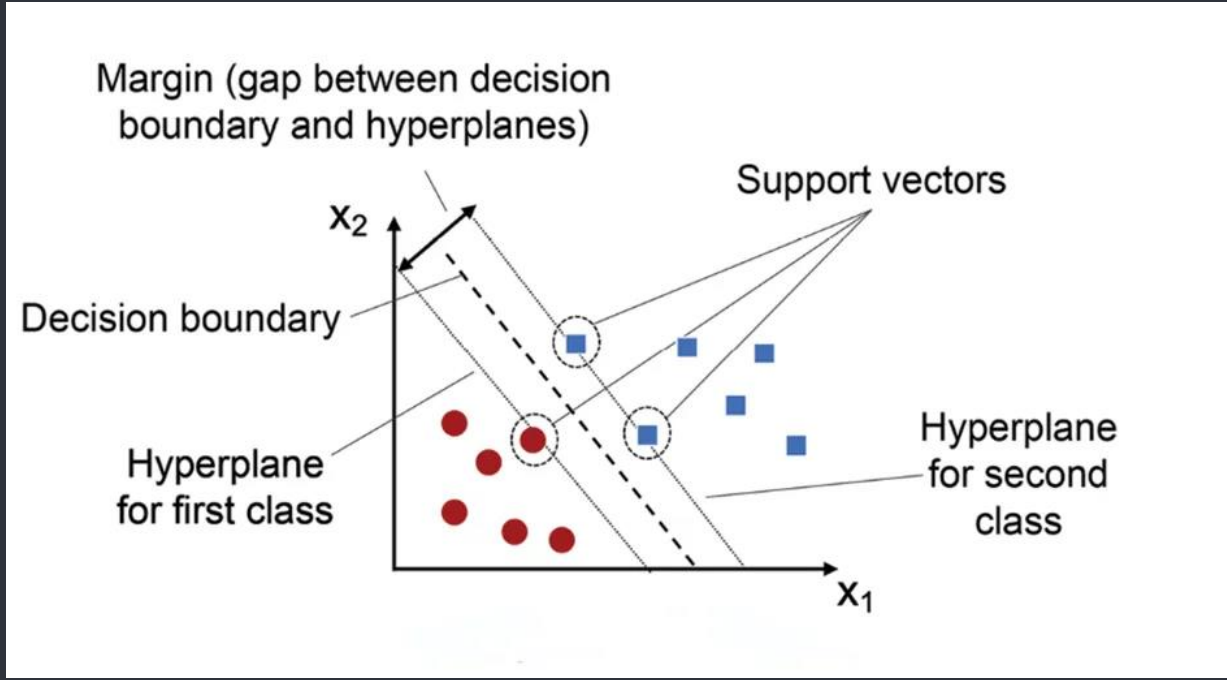


La linea amarilla permite generalizar mejor la separacion de clases

}

Maquinas de Soporte Vectorial{

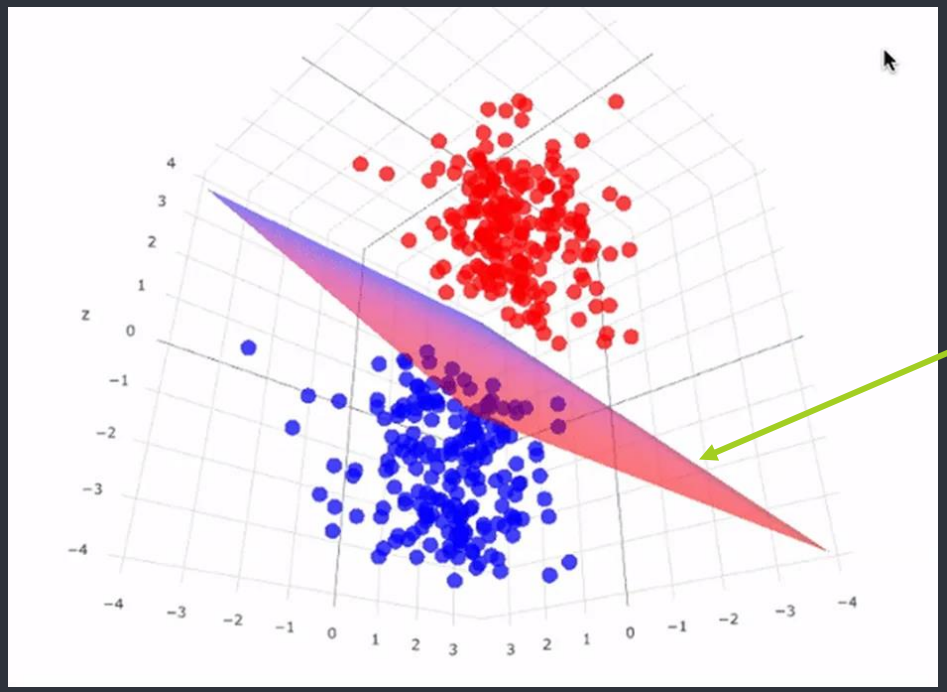
Una maquina de soporte vectorial busca maximizar el margen que separa a las clases



}

Maquinas de Soporte Vectorial{

Conforme el numero de dimensiones sube, las dimensiones del plano generado tambien lo hacen

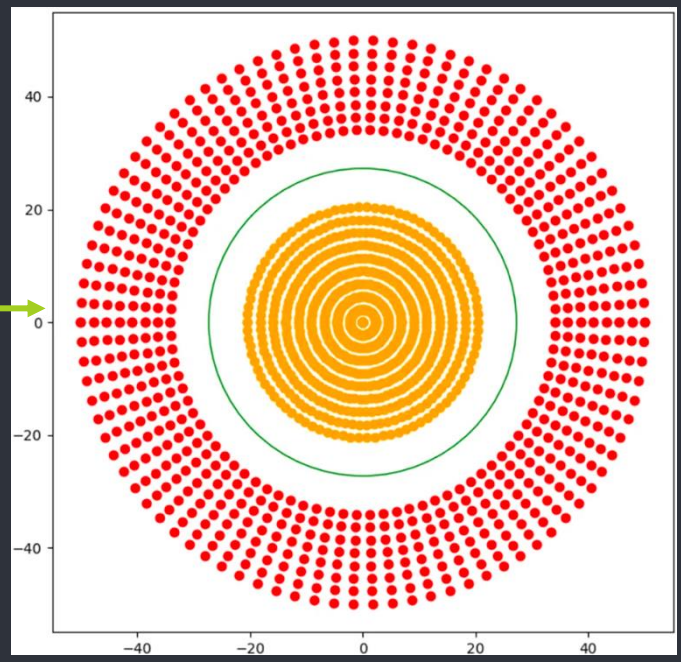


Hiperplano

}

Maquinas de Soporte Vectorial{

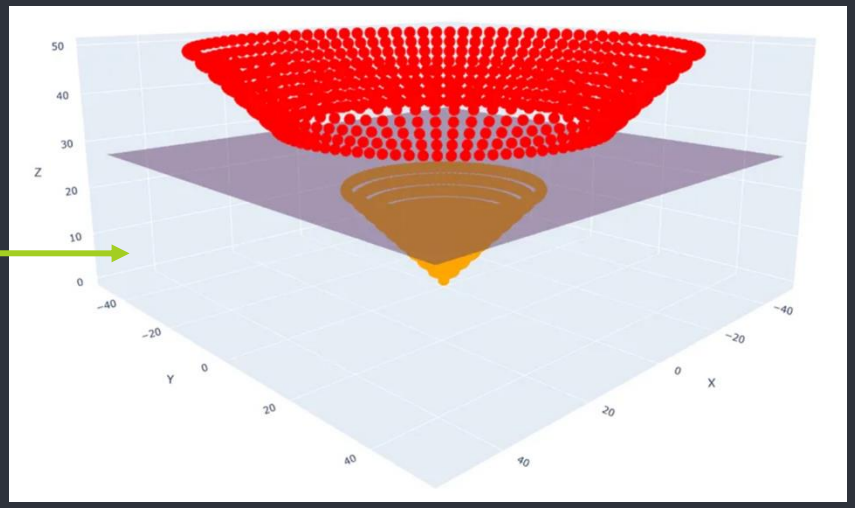
Que pasa si los datos no son linearmente separables?



}

Maquinas de Soporte Vectorial{

Las SVM's utilizan Kernels que les permiten computar la similitud de los datos en un espacio dimensional mas alto, buscando hacerlos separables



RBF Kernel

$$e^{-\gamma(a-b)^2}$$

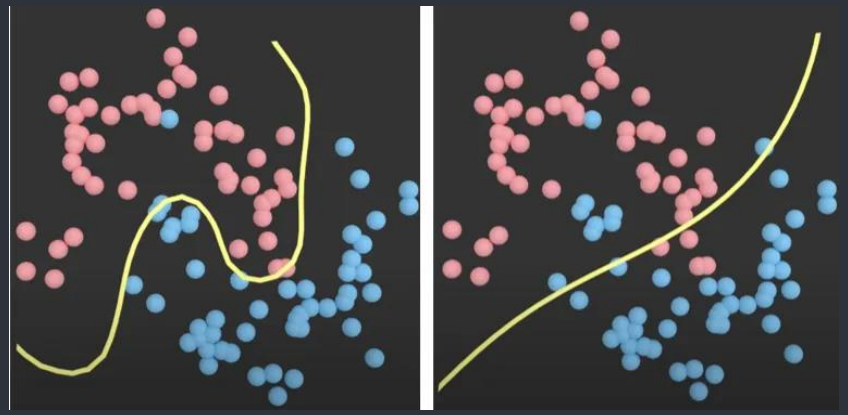
}

Maquinas de Soporte Vectorial{

Dependiendo del Kernel, habra distintos hyperparametros que podremos seleccionar

$$e^{-\gamma(a-b)^2}$$

Controla la linealidad del modelo



Gamma = 1

Gamma = 0.01

}

Maquinas de Soporte Vectorial{

Dependiendo del Kernel, habra distintos hyperparametros que podremos seleccionar

$$e^{-\gamma(a-b)^2}$$

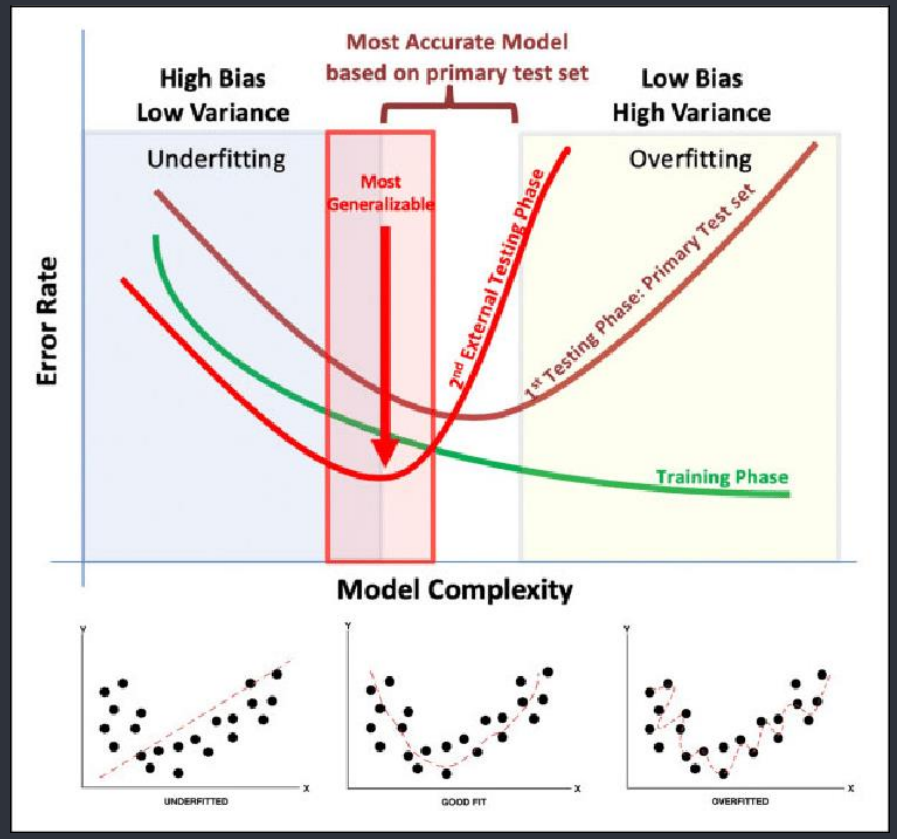
Controla la linealidad del modelo

Gamma = 1 | Gamma = 1 | Gamma = 1

}

Maquinas de Soporte Vectorial{

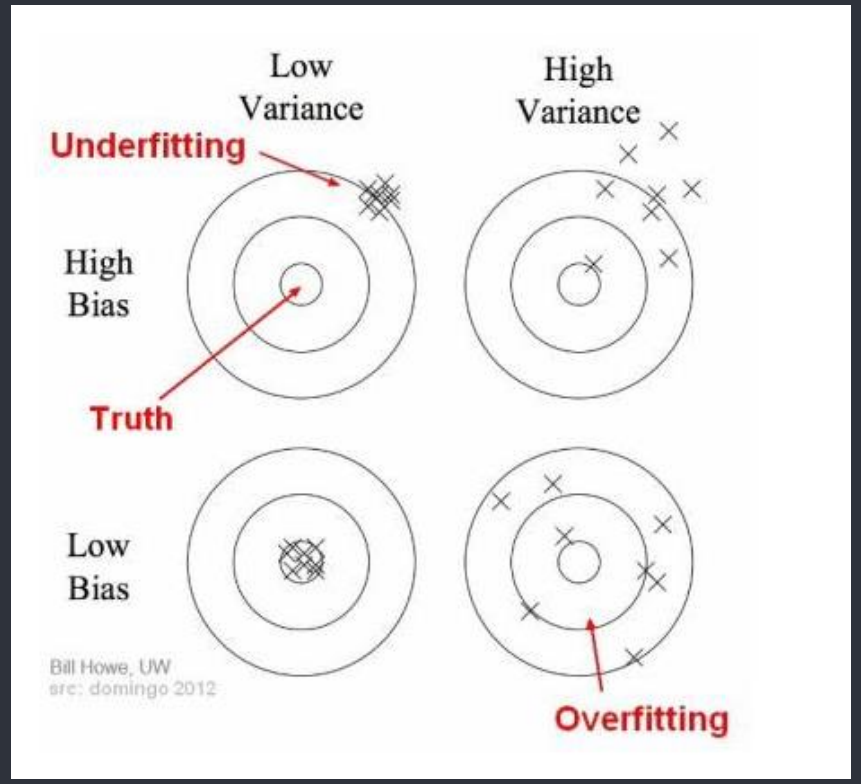
El Bias-Variance tradeoff



}

Maquinas de Soporte Vectorial{

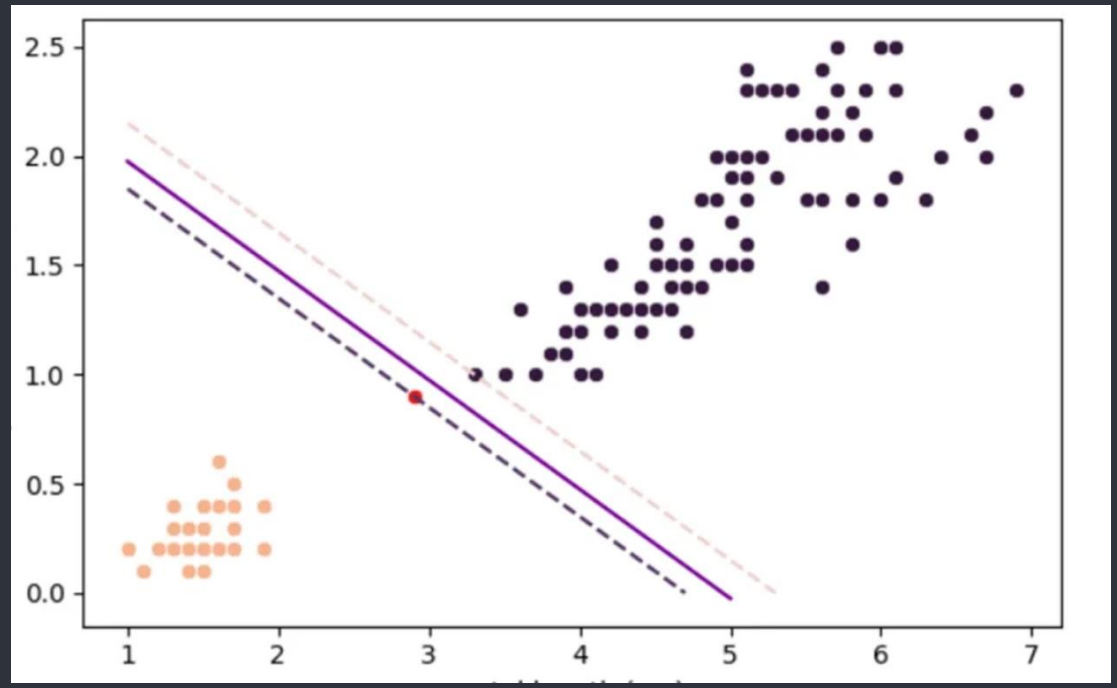
El Bias-Variance tradeoff



}

Maquinas de Soporte Vectorial{

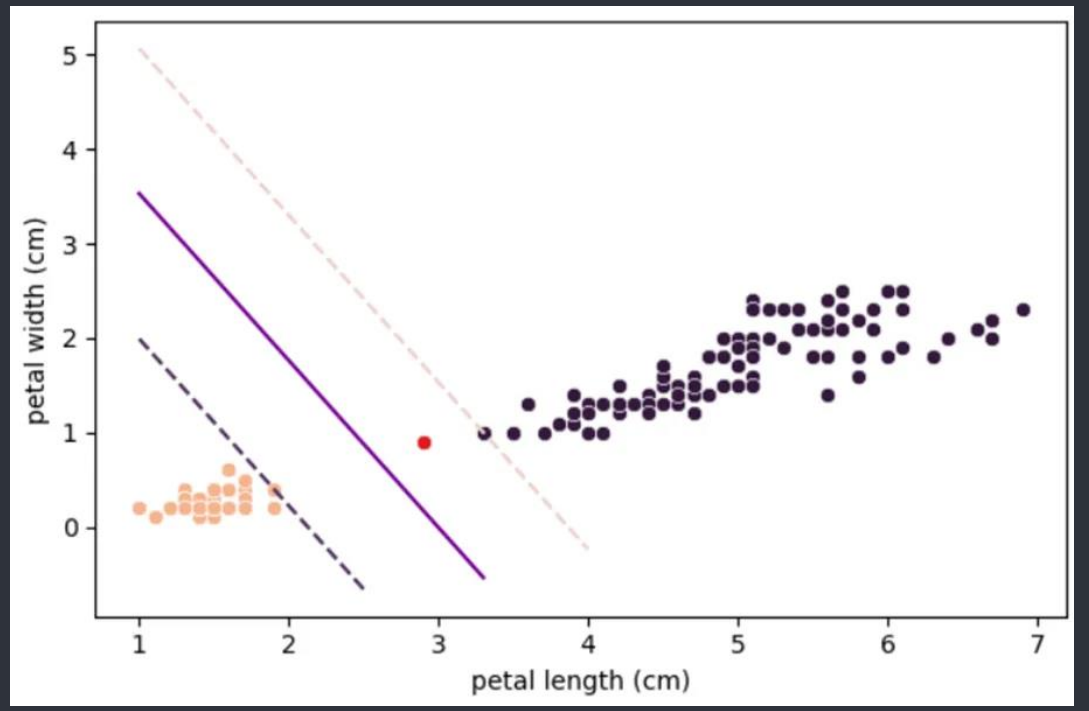
Outliers podrian resultar en un modelo sobre-ajustado (overfit)



}

Maquinas de Soporte Vectorial{

Usando soft margins, podemos tolerar algunas clasificaciones erroneas con el objetivo de mejorar la manera en que el modelo generaliza (bias-variance tradeoff)



Controlado por el parametro "C"

}

Valores mas grandes de C -> Hard margin

Valores mas chicos de C -> Soft margin

{Implementando Maquinas de Soporte Vectorial}

Ivan Rojas Gonzalez – Universidad Panamericana

Implementando Maquinas de Soporte Vectorial{

<Dado que estamos tratando con aprendizaje supervisado, tendremos que partir los datos en un set de entrenamiento y un set de prueba>

```python
from sklearn.model_selection import train_test_split

X = df['tweet_clean']
y = df['intention']

X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.85)
```

Se importa la libreria

Variable(s) de entrenamiento

Variable objetivo

% de los datos reservados para entrenamiento

}

Implementando Maquinas de Soporte Vectorial{

<Ahora procedemos a vectorizar el texto>

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(ngram_range=(1,3))

X_train_vectorized = tfidf.fit_transform(X_train)
X_test_vectorized = tfidf.transform(X_test)
```

Usamos fit_transform para los datos de entrenamiento

Y transform para los datos de prueba

}

Implementando Maquinas de Soporte Vectorial{

<Ahora, creamos nuestro modelo de maquinas de soporte vectorial>

```
from sklearn.svm import SVC
model = SVC()
model.fit(X_train_vectorized,y_train)

y_pred = model.predict(X_test_vectorized)
```

Importamos la libreria de maquinas de soporte vectorial para clasificacion

Instanciamos un modelo

Y lo entrenamos sobre los datos de entrenamiento

Obtenemos las predicciones sobre el set de prueba para sacar metricas de clasificacion

}

Implementando Maquinas de Soporte Vectorial{

<Finalmente, evaluamos el desempeño>

```
from sklearn.metrics import classification_report, confusion_matrix

print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
Non-Suicide	0.94	0.89	0.91	770
Suicide	0.86	0.93	0.90	598
accuracy			0.90	1368
macro avg	0.90	0.91	0.90	1368
weighted avg	0.91	0.90	0.91	1368

Importamos componentes de la libreria de metricas

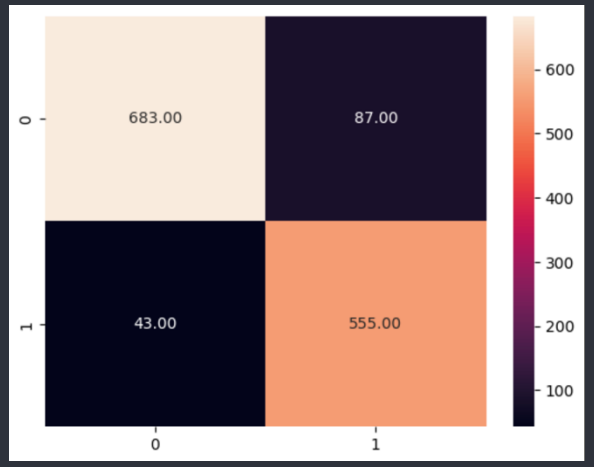
Obtenemos el reporte de clasificacion

}

Implementando Maquinas de Soporte Vectorial{

<De igual manera, podemos obtener la matriz de confusion>

```
import seaborn as sns
sns.heatmap(confusion_matrix(y_test,y_pred), annot=True, fmt='.2f')
```



}

Implementando Maquinas de Soporte Vectorial{

<Para predecir tweets nuevos, creamos una funcion>

```python
def nuevo_tweet(tweet, show_preprocess=False):
    resultado = procesar_tweet(tweet)
    resultado = [resultado]

    if show_preprocess==True:
        print(resultado)

    resultado = tfidf.transform(resultado)
    final = (model.predict(resultado))
    return final
```

Aplicamos la funcion de preprocesamiento sobre el nuevo tweet

Encerramos el tweet procesado en un array (requerimientos locos de el tfidf vectorizer)

Parametro booleano seteable en caso de querer imprimir como se ve el tweet ya preprocesado

Vectorizamos el tweet SOLO TRANSFORM, NO FIT_TRANSFORM

Y obtenemos la prediccion de clase

}

Implementando Maquinas de Soporte Vectorial{

<Ejemplos del llamado a nuestra funcion>

```
nuevo_tweet("""The cause of this problem is that
            input is just a string, but what is needed is a
            list (or an iterable) containing a single element.""")
```

✓ 0.0s

```
array(['Non-Suicide'], dtype=object)
```

```
nuevo_tweet("""I've been thinking about leaving home and
            killing myself. They will miss me one day!!
            """, show_preprocess=True)
```

✓ 0.0s

```
['ive thinking leaving home killing miss one day']

array(['Suicide'], dtype=object)
```

}

Q&A