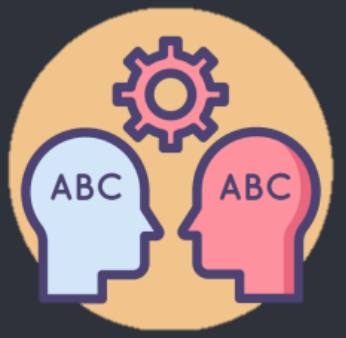
Procesamiento de Lenguaje Natural{



[NLP]

<Aprendizaje Supervisado | Clasificacion | Regresion Logistica>

}

Ivan Rojas Gonzalez – Universidad Panamericana

La Agenda de hoy {

01 Intuicion sobre regresion logistica

<Cómo funciona>

02 Implementando Regresion Logistica

<Llevando nuestro clasificador a la vida >

03 Evaluando el clasificador

<Que tan bueno es nuestro clasificador?>

}

{Regresion Logistica}

Ivan Rojas Gonzalez – Universidad Panamericana

Regresion Logistica{

Y en el otro eje, la probabilidad (recordemos que la probabilidad de cualquier evento se encuentra acotada entre 0 y 1)

En un eje, tenemos la variable independiente

}

Regresion Logistica{

La probabilidad de exito de un evento es igual a la probabilidad de que suceda, dividido entre la probabilidad de que no suceda

$$Odds\ (\theta) = \frac{Probability\ of\ an\ event\ happening}{Probability\ of\ an\ event\ not\ happening}$$

Esto puede ser expresado de la siguiente manera

$$Odds\ (\theta) = \frac{p}{1-p}$$

Dado que la probabilidad se modela como una funcion lineal, es posible expresar el logaritmo de la probabilidad de exito de la siguiente manera

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x$$

}

Regresion Logistica{

Exponenciando ambos lados, tenemos lo siguiente

$$e^{ln\left(\frac{p(x)}{1-p(x)}\right)} = e^{\beta_0+\beta_1 x}$$

Despejando, llegamos a:

$$p(x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$$

Simplificando la expresion, llegamos a la funcion logistica, tambien llamada sigmoide

$$p(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$$

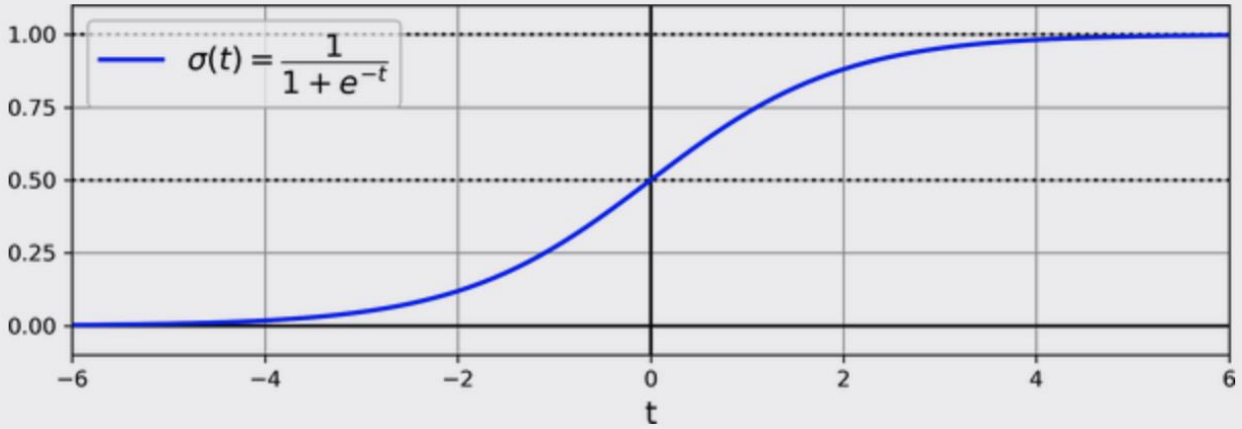
}

Regresion Logistica{

Simplificando la expresion, llegamos a la funcion logistica, tambien llamada sigmoide

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



}

Regresion Logistica{

Lo anterior visto aplica para un escenario donde se hace regresion logistica sobre una sola variable. Para regresión multivariada se realiza un ajuste

$$p = \frac{1}{1 + e^{-(b_0 + b_1 X_1 + b_2 X_2 + --- + b_k X_k)}}$$

}

{Implementando Regresion Logistica}

Ivan Rojas Gonzalez – Universidad Panamericana

Implementando Regresion Logistica{

<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 Regresion Logistica{

<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 Regresion Logistica{

<Ahora, creamos nuestro modelo de regresion logistica>

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train_vectorized,y_train)

y_pred = model.predict(X_test_vectorized)
```

Importamos la libreria de logistic regression

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 Regresion Logistica{

<Finalmente, evaluamos el desempeño>

```
from sklearn.metrics import classification_report, confusion_matrix

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

Importamos componentes de la libreria de metricas

Obtenemos el reporte de clasificacion

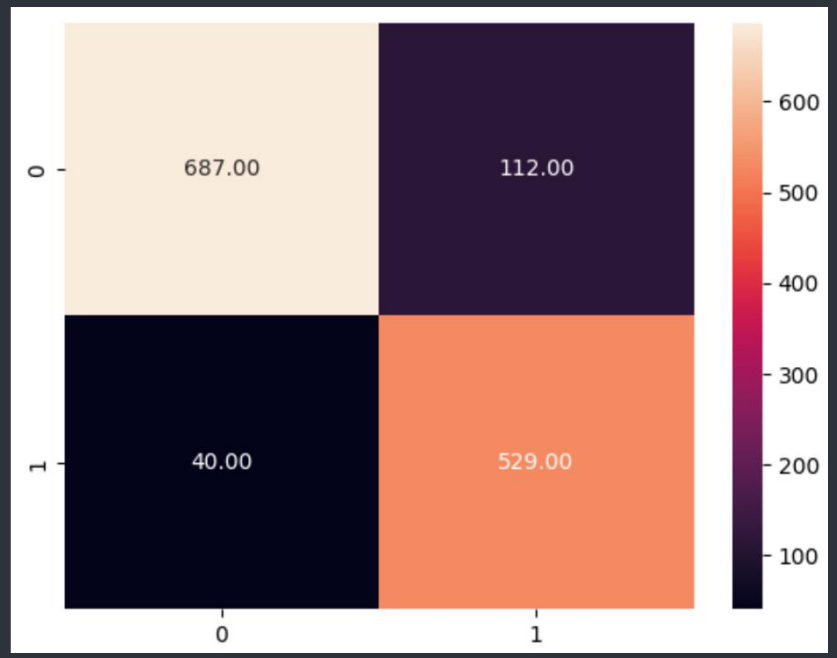
```
                precision    recall  f1-score   support

    Non-Suicide       0.94      0.86      0.90       799
        Suicide       0.83      0.93      0.87       569

       accuracy                           0.89      1368
      macro avg       0.89      0.89      0.89      1368
   weighted avg       0.90      0.89      0.89      1368
```

}

Implementando Regresion Logistica{

<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 Regresion Logistica{

<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),model.predict_proba(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 tanto la prediccion de clase como la probabilidad asociada

}

Implementando Regresion Logistica{

<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), array([[0.72859365, 0.27140635]]))
```

```
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), array([[0.26940604, 0.73059396]]))
```

}

Q&A