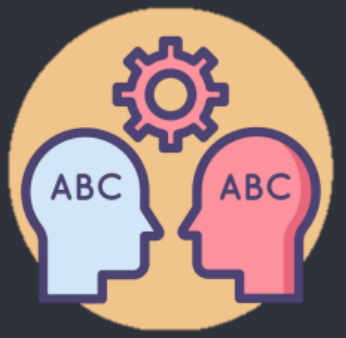
# Procesamiento de Lenguaje Natural{




[NLP]

<Aprendizaje No Supervisado | Reduccion de dimensiones | TSNE>

}

Ivan Rojas Gonzalez – Universidad Panamericana

# La Agenda de hoy {

01 Por que reducir dimensiones?

<Los limites del cerebro humano>

02 TSNE

<Para que sirve, que representa>

03 Implementando TSNE

<Reduciendo dimensiones en vectores de texto>

}

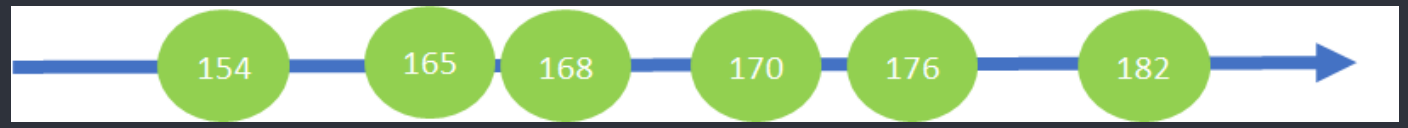
# {Por que reducir dimensiones?}

Ivan Rojas Gonzalez – Universidad Panamericana

# Algunos ejemplos de modelos{

<Supongamos que se desea graficar la estatura de la gente en el salon>

| Persona | Estatura |
|---|---|
| A | 1.70 |
| B | 1.68 |
| C | 1.65 |
| D | 1.54 |
| E | 1.76 |
| F | 1.82 |

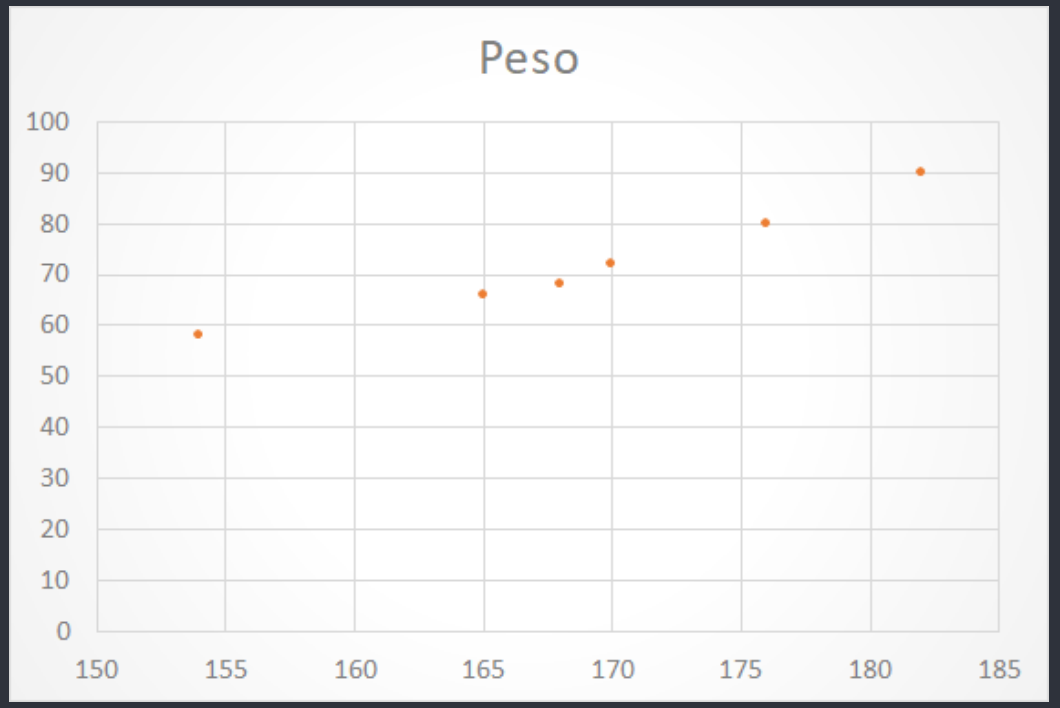



}

# Algunos ejemplos de modelos{

<Ahora consideremos tambien el peso>

| Persona | Estatura | Peso |
|---|---|---|
| A | 1.70 | 72 |
| B | 1.68 | 68 |
| C | 1.65 | 66 |
| D | 1.54 | 58 |
| E | 1.76 | 80 |
| F | 1.82 | 90 |

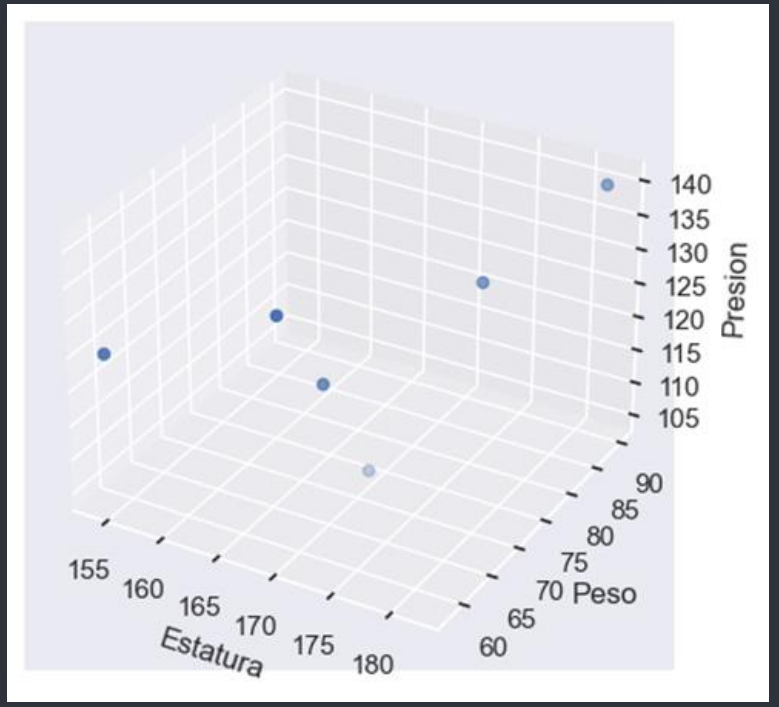



}

# Algunos ejemplos de modelos{

<Y si tambien consideramos la presion arterial?>

| Persona | Estatura | Peso | Presion |
|---|---|---|---|
| A | 1.70 | 72 | 105 |
| B | 1.68 | 68 | 120 |
| C | 1.65 | 66 | 130 |
| D | 1.54 | 58 | 125 |
| E | 1.76 | 80 | 130 |
| F | 1.82 | 90 | 140 |




}

# {T-Distributed Stochastic Neighbor Embeddings: TSNE }

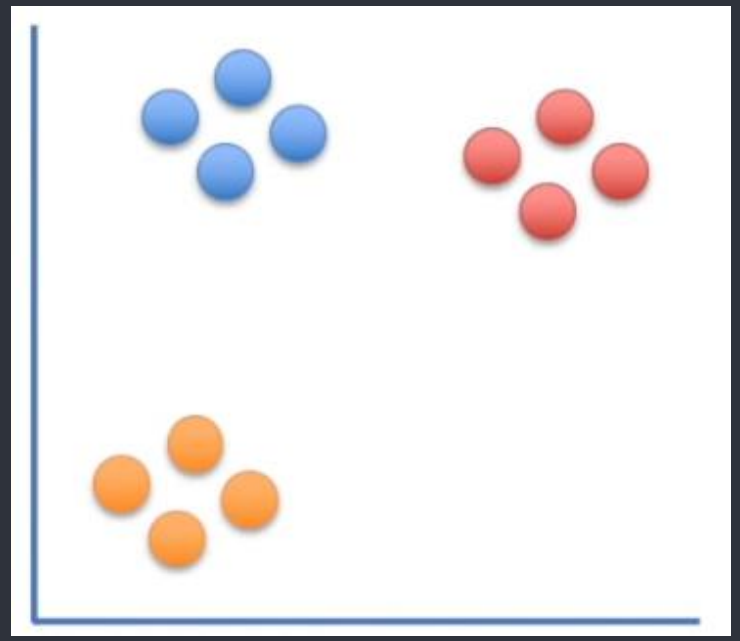
Ivan Rojas Gonzalez – Universidad Panamericana

```
TSNE{

}
```

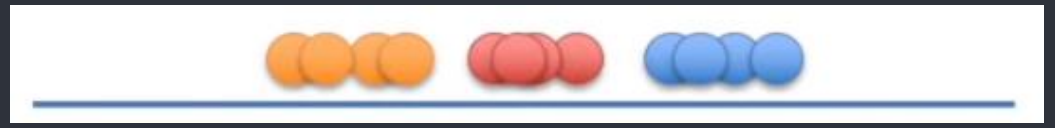
<Tomar un set de datos de alta dimensionalidad (un gran numero de columnas)>

<Reducirlo a X dimensiones (por lo general 2)>

```
TSNE{

}
```

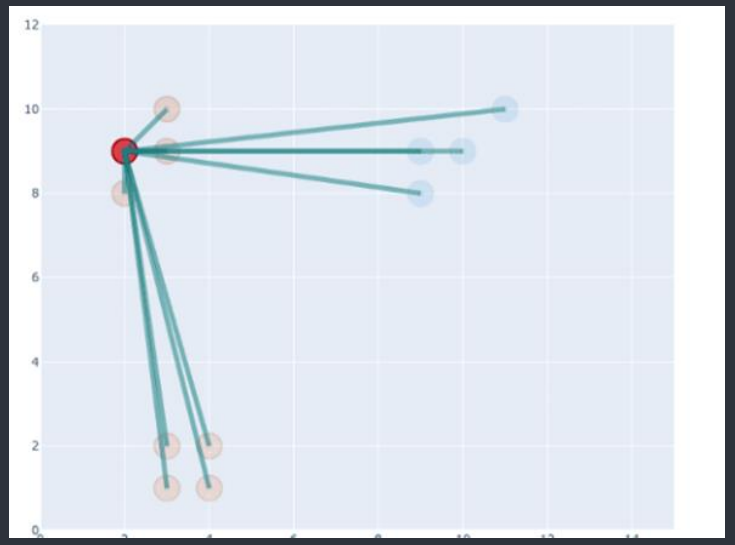


<Espacio original
en 2 dimensiones>

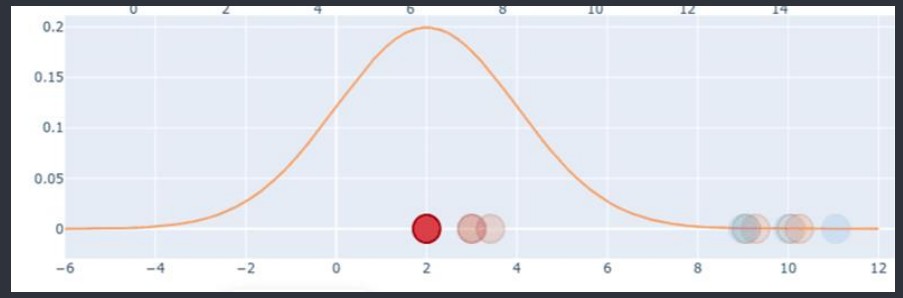


<Espacio final en 1
dimension>

# TSNE: Como funciona{



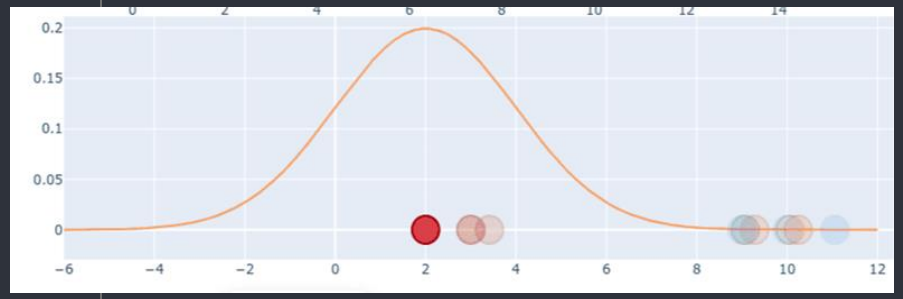
<Se compara la distancia entre cada punto del espacio>



<Y esa distancia se grafica dentro de una curva normal donde el centro es el punto observado>

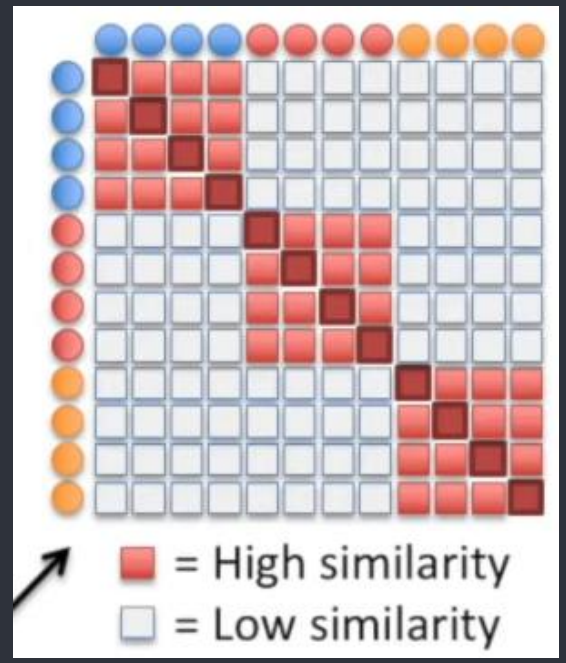
}

# TSNE: Como funciona{




<Con esto, obtenemos una matriz de "scores de similitud">

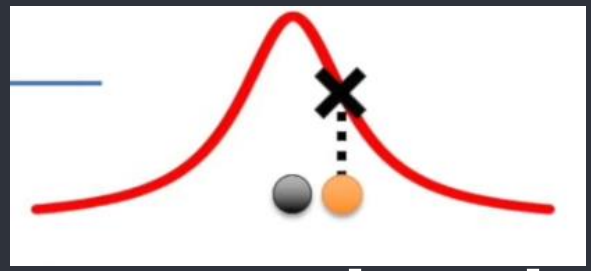
$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$




}

# TSNE: Como funciona{

<Ahora, se proyectan los puntos de manera aleatoria en un espacio con el numero de dimensiones deseado>

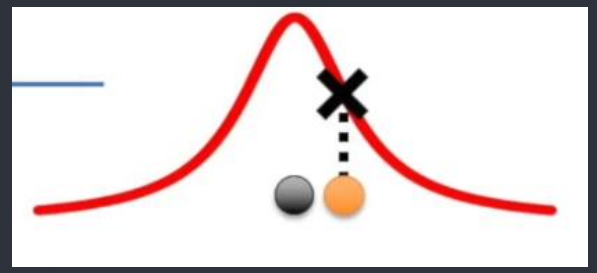


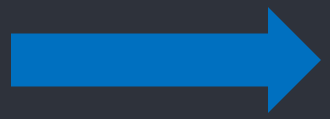
<Y se proyectan los valores en una curva de distribucion T de student, de aquí sale la T en "T-SNE">

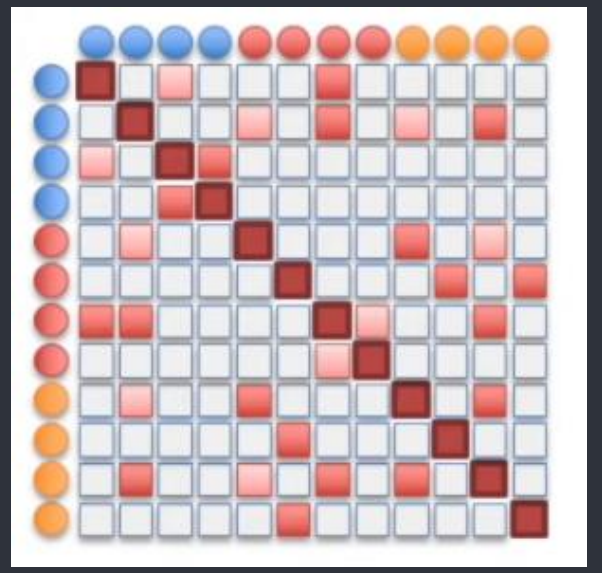
$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

}

# TSNE: Como funciona{



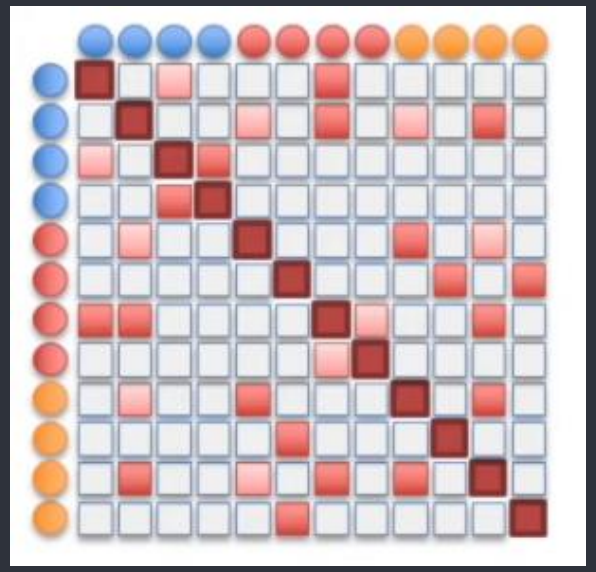


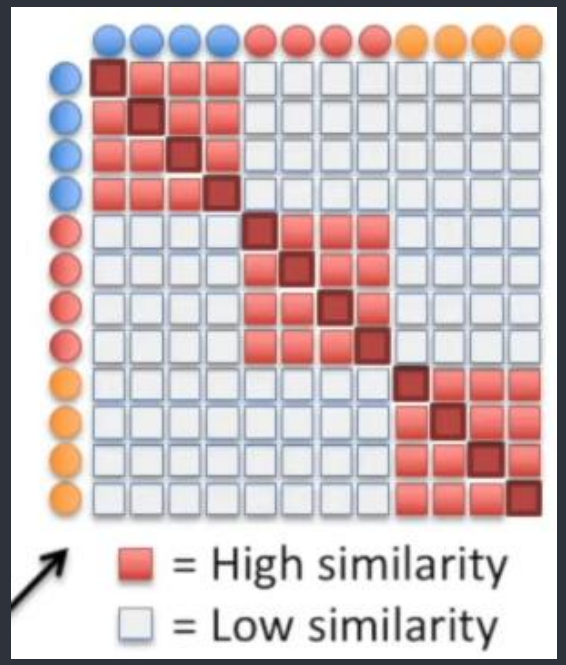


<Con esto obtenemos una nueva
matriz de similitud>

}

# TSNE: Como funciona{






<La idea ahora es hacer que la matriz generada con la distribucion T se parezca a la generada con la distribucion normal>

}

# TSNE: Como funciona{

$$C = D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left( \frac{P(x)}{Q(x)} \right)$$

<Esto se logra con la divergencia Kullback Leiber.
Puntos similares generan una penalizacion baja,
puntos disimilares una penalizacion alta>

}

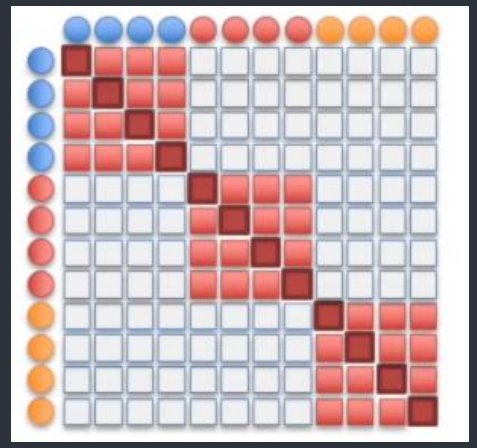
# TSNE: Como funciona{

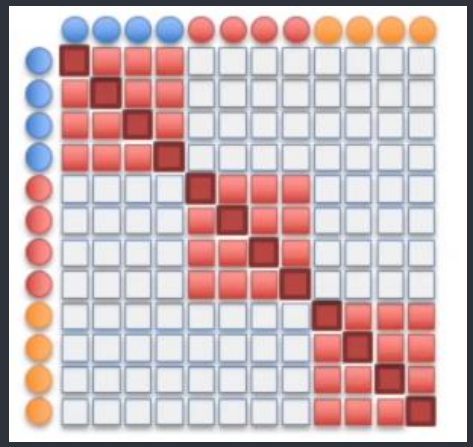
$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$$

<Para lograr optimizar, utilizamos gradiente descendente (derivar la función de optimización para encontrar valores minimos)>

}

# TSNE: Como funciona{

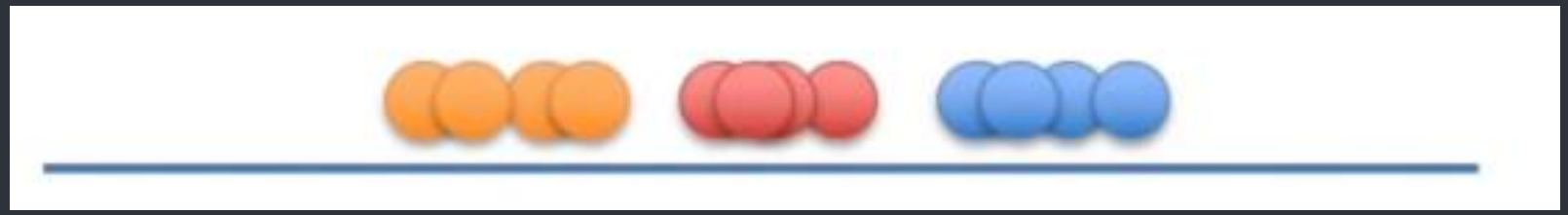




<Eventualmente, ambas matrices deberan ser iguales*>

}

# TSNE: Como funciona{



<Y con ello, preservar agrupamientos en un espacio dimensional reducido>

}

# {Implementando TSNE}

Ivan Rojas Gonzalez – Universidad Panamericana

# Implementando TSNE{

<Dado que estamos usando texto, primero tendremos que vecotrizarlo>

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(ngram_range=(1,1))
vectorized_text = vectorizer.fit_transform(df['tweet_clean'])
vectorized_text
```

}

# Implementando TSNE{

<Con el texto ya vectorizado, importamos la librería de TSNE>

Indicamos el numero de dimensiones que queremos obtener (por lo regular, 2)

Importamos la clase TSNE

```
from sklearn.manifold import TSNE
n_components = 2
tsne = TSNE(n_components, init='random')
tsne_result = tsne.fit_transform(vectorized_text)
```

Init es random porque estamos usando una "sparce matrix"

Pasamos el texto vecorizado como argumento al modelo

}

# Implementando TSNE{

<El resultado seran dos columnas con los valores de TSNE>

Creamos un dataframe con el resultado

```
tsne_result_df = pd.DataFrame({'tsne_1': tsne_result[:,0], 'tsne_2': tsne_result[:,1]})
tsne_result_df['label'] = df['intention_mapped']
tsne_result_df['text'] = df['tweet']
tsne_result_df.head()
```

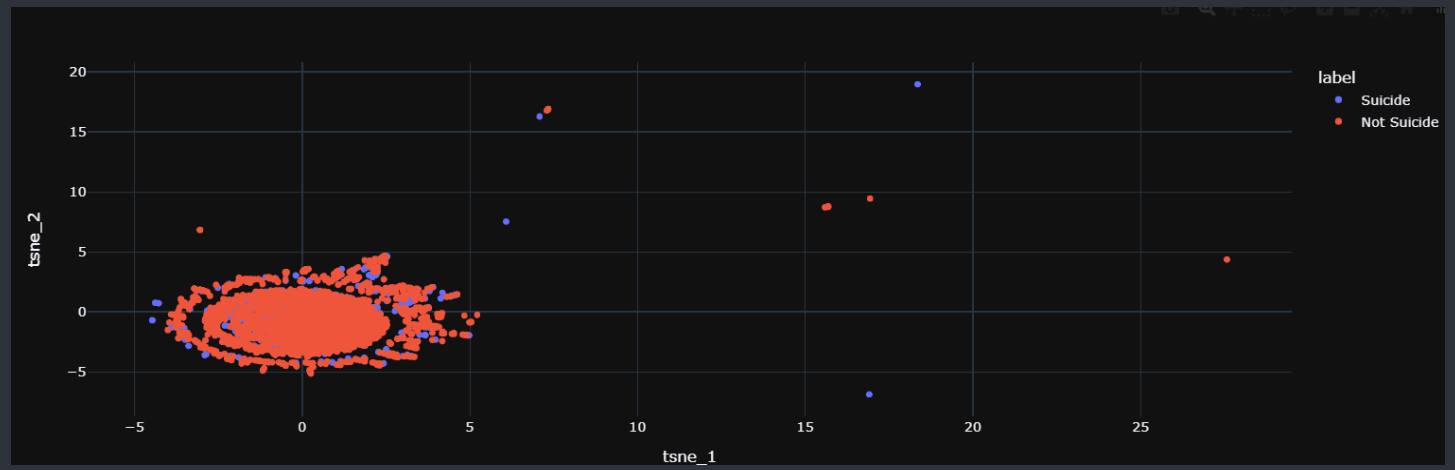
Unimos ese dataframe con las columnas que contienen el texto original y el intent

}

# Implementando TSNE{

<Y visualizamos el resultado>

```python
import plotly_express as px
fig = px.scatter(data_frame=tsne_result_df,
x=tsne_result_df['tsne_1'],
y=tsne_result_df['tsne_2'],
color=tsne_result_df['label'],
template='plotly_dark' ,
hover_data=['text'])

fig.show()
```




}

# Q&A