María Camila Terán — 201822000

Juan Diego Cardona — 201819447

Nicolás Ortega — 201814515

Inteligencia de negocios
Proyecto 1

Tabla de Contenidos

Comprensión del negocio y enfoque analítico

Para iniciar, es necesario resaltar la importancia de la analítica de texto en el campo empresarial. Hoy en día aproximadamente el 55% de la población mundial interactúa diariamente con las redes sociales. Es allí donde los usuarios comparten opiniones, noticias, experiencias, gustos, descripciones e inclusive sentimientos.

Oportunidad/problema del negocio	Teniendo en cuenta lo mencionado anteriormente, se observa que un gran número de personas interactúan y se expresan en las redes sociales. En ese sentido, el objetivo del negocio es clasificar aquellas emociones basándose en los diferentes textos que los usuarios de las redes sociales publican. Esto es de interés para el negocio ya que se busca saber cuáles son las emociones que más se reflejan en estos medios con el propósito de tener una idea que es lo que a las personas les gusta y qué no. Por otra parte, podría permitir conocer fácilmente la reacción general de las personas a ciertos eventos, productos o situaciones, información que podría utilizar un negocio para enfocar mejor sus esfuerzos y optimizar sus campañas de mercadeo.	
Descripción del requerimiento desde el punto de vista de aprendizaje de la máquina	Los datos previstos por la organización proporcionan información para implementar algoritmos de aprendizaje supervisado. Esto teniendo en cuenta que son textos que expresan alguna emoción por lo que tienen etiquetas. Se cuenta con unos datos base para que posteriormente se puedan relacionar palabras con los sentimientos que las personas desean expresar.	
Detalles de la actividad de minería de datos		
Tarea	Técnica	Algoritmo e hiper-parámetros utilizados
Clasificación	Árbol de decisión	**Algoritmo**: DecissionTreeClassifier() de sklearn.tree **Hiperparámetros**: 'criterion': 'entropy', 'max_depth': 20, 'min_samples_split': 4
Clasificación	Random Forest (implementa varios árboles de decisión)	**Algoritmo**: RandomForestClassifier() de sklearn.ensemble **Hiperparámetros**: 'criterion': 'gini', 'max_depth': None, 'min_samples_split': 4
Clasificación	Support Vector Machine (SVM)	**Algoritmo**: SVC() de sklearn.svm **Hiperparámetros**: 'C': 2.154434690031882
Clasificación	Naive Bayes	**Algoritmo**: MultinomialNB() de sklearn **Hiperparámetros**: 'alpha': 0, 'fit_prior': True
Clasificación	K Nearest Neighbors	**Algoritmo**: KNeighborsClassifier () de sklearn.neighbors **Hiperparámetros**: 'n_neighbors': 1, 'p': 1

Perfilamiento de datos y limpieza

El primer paso para realizar fue hacer un estudio desde el mismo archivo txt. En este se pudo encontrar ciertas frases en inglés junto con el sentimiento al cual

se relacionaba. Esto se tomó como los datos de entrenamiento dado que se utilizarán técnicas de aprendizaje supervisado. Para iniciar, se utilizó una técnica llamada “tokenización” la cual consiste en eliminar del texto todo aquello que no aporte información relevante (Amat, párr. 6) para luego dividir las palabras. En está ocasión, al tratarse de textos que los usuarios publican en las redes sociales, es necesario tener en cuenta que algunos usan abreviaturas, signos de puntuación incorrectos y demás. Luego, de esto, se dividió el texto en las distintas palabras. Ahora bien, ya con esto realizado se obtuvo que en los datos se utilizan 34073 palabras siendo solo 4778 distintas. Esto indica un alto nivel de palabras comunes que los usuarios utilizan para expresar sus sentimientos. Posteriormente, se vio necesario excluir las palabras más usadas en inglés dado que estas no necesariamente se relacionan con los sentimientos que puede expresar una persona. Estas, conocidas como “stop words” fueron ‘i’, ‘me’, ‘myself’, entre otras. Ahora bien, sin contar estas palabras, se decidió obtener las 30 palabras más comunes en los datos junto con su frecuencia. Los resultados fueron los siguientes:

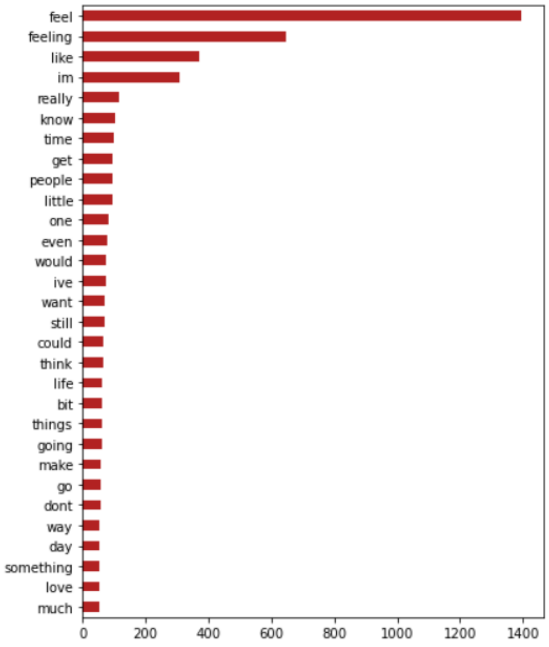


Ilustración 1 30 palabras más comunes junto con su frecuencia

Los resultados tienen sentido en el sentido de que las palabras más utilizadas son “sentir” o “sintiendo” lo que concuerda con el objetivo de negocio. No obstante, es necesario realizar la búsqueda para relacionar estas palabras con sentimientos pues, en esta lista, solo aparece una palabra relacionada con estos y sería “amor” e inclusive, tiene una frecuencia de uso menor a 50. Se utilizó de igual manera la técnica de “frecuencia de términos” con el propósito de medir la importancia de un término dentro del texto. La importancia de la palabra no se mide teniendo en cuenta el número de veces que aparece, sino por el peso. Por ejemplo, como se mencionó anteriormente, en esta ocasión “love” es una palabra significativa pero la palabra “would” aparece más veces. Se obtiene el estadístico

tf-idf que “mide que tan informativo es un término” al comprarlo con su aparición en otros textos. Se utilizan las siguientes ecuaciones:

$$Term\ frequency = \frac{n_t}{longitud\ d}$$

Siento t la palabra y d el documento.

$$Inverse\ document\ frequency = \log\left(\frac{n_d}{n_{d,t}}\right)$$

Siendo n_d el número total de documentos y $n_{d,t}$ el número de documentos que contienen el término t.

Y por último se tiene el estadístico:

$$idf\ (t) = \log\frac{1 + n_d}{1 + n_{d,t}} + 1$$

Se tiene la siguiente tabla con algunos estadísticos calculados:

	token	count	total_count	tf	n_documentos	idf	tf_idf
0	aaaah	1	1	1.0	1	8.443331	8.443331
1	abandoned	3	3	1.0	3	7.344719	7.344719
2	abandoning	1	1	1.0	1	8.443331	8.443331
3	abandonment	1	1	1.0	1	8.443331	8.443331
4	abba	1	1	1.0	1	8.443331	8.443331

Ilustración 2 Estadísticas de las frecuencias

Algunos términos son similares, más se ve la importancia de otros con la corrección realizada en tf_idf. Como se puede observar, estos son sentimientos como la expresión “aaaah” o el término “abandoned”, por lo que tiene sentido dado que el estudio son sentimientos. Ahora bien, será necesario clasificar los mensajes. Para esto se utilizó la técnica “vectorización”, en el cual, de acuerdo a la palabra, se forma una columna y se asigna un valor, en este caso sería el estadístico de frecuencia hallado anteriormente. Finalmente, se tiene la vectorización donde se crea una matriz con las columnas representando los términos y las filas un documento por lo que intersección sería el valor tf-idf.

Finalmente se reviso la distribución de los valores de la variable objetivo y se pudo identificar que había un desbalanceo con los registros de “sadness”y “joy” de manera que procedimos a balancearlo con la técnica de SMOTE . Para la realización de los modelos de clasificación se separaron los datos en conjuntos de entrenamiento y prueba para los modelos.

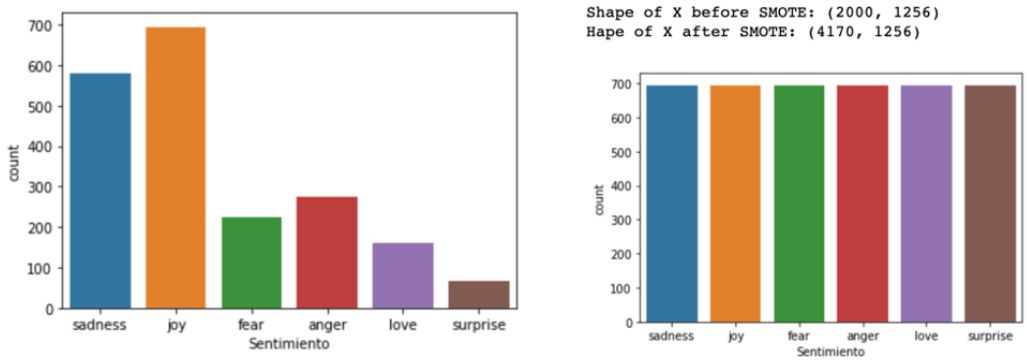


Ilustración 3: Antes y despúes del balanceo

Modelado y Evaluación

Modelo SVM lineal

Como primer algoritmo se implementó el de SVM Lineal (*Support Vector Machines)* el cual es un algoritmo de aprendizaje supervisado relacionado con problemas de clasificación y regresión. Dado que el conjunto de datos que tenemos posee ejemplos de mensajes para entrenamiento con etiquetas de sentimientos, se puede construir un modelo que sirva para predecir el sentimiento que tendrán las siguientes muestras.

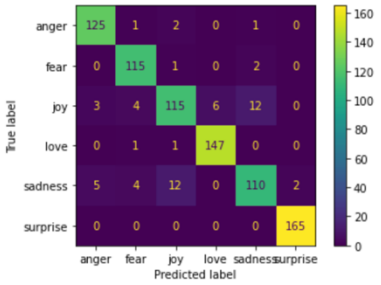
Se inició con la búsqueda de hiperparámetros utilizando la búsqueda con Kfold y GridSearch del cual se pudo obtener que el mejor parámetro es:

```
{'C': 2.154434690031882}
```

El desempeño sobre el conjunto de prueba fue el siguiente:

```
Exactitud: 0.93
Recall: 0.9316546762589928
Precisión: 0.9303623257835066
Puntuación F1: 0.9305370395511813
```

La matriz de confusión fue la siguiente:



Podemos observar que presenta unas métricas **bastante buenas** para ambos conjuntos.

Árboles de decisión

La siguiente técnica a implementar fue un árbol de decisión. Es una de las técnicas clásicas y, de acuerdo con el blog de *data scientist* de MonkeyLearn, es de las más populares cuando se trata de una tarea de clasificación. Funciona como un diagrama de flujo, separando los puntos de datos en dos categorías similares a la vez, desde el "tronco del árbol" hasta las "ramas" y las "hojas", donde las categorías se vuelven más finamente similares. Esto crea categorías dentro de categorías, lo que permite la clasificación orgánica con supervisión humana limitada.

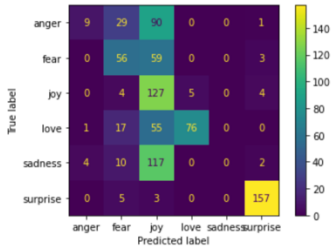
Se inició con la búsqueda de hiperparámetros utilizando la búsqueda con Kfold y GridSearch del cual se pudo obtener que el mejor conjunto de parámetros es:

```
{'criterion': 'entropy', 'max_depth': 20, 'min_samples_split': 4}
```

El desempeño del modelo sobre el conjunto de prueba fue el siguiente:

```
Exactitud: 0.51
Recall: 0.5095923261390888
Precisión: 0.5658104012197747
Puntuación F1: 0.4631029019065507
```

La matriz de confusión fue la siguiente:



En general observamos que no presenta unas métricas muy buenas. En realidad no es un buen modelo, porque no tiene buena exactitud, sensibilidad o precisión. El modelo no es capaz de generalizar apropiadamente sobre un conjunto de datos. No se recomendaría utilizarlo

Random Forest

Dado que el desempeño del árbol de decisión no fue el esperado, se decidió implementar la técnica de "bosques aleatorios". Esta técnica crea árboles de decisión sobre muestras de datos seleccionadas al azar, obtiene predicciones de cada árbol y selecciona la mejor solución mediante votación. Una de sus principales ventajas es que no sufre el problema de sobreajuste. La razón principal es que toma el promedio de todas las predicciones, lo que anula los sesgos. Como implementa árboles, maneja los mismos parámetros que en el caso anterior.

Se inició con la búsqueda de hiperparámetros utilizando la búsqueda con Kfold y GridSearch del cual se pudo obtener que el mejor conjunto de parámetros es:

```
{'criterion': 'gini', 'max_depth': None, 'min_samples_split': 4}
```

En este caso se cambia el criterio de decisión en los nodos, y adicionalmente se establece que la profundidad del árbol sea “None”, lo que le indicará al algoritmo que llegue hasta las que las hojas sean muy “limpias” o ya no tengan el número suficiente de muestras para dividirse (min_samples_split)

El desempeño del modelo sobre el conjunto de prueba fue el siguiente:

```
Exactitud: 0.87
Recall: 0.8657074340527577
Precisión: 0.8769139897764142
Puntuación F1: 0.8642293155486366
```

La matriz de confusión fue la siguiente:

Podemos observar que presenta unas métricas **bastante buenas**. En comparación con el caso anterior el desempeño mejora considerablemente. El resultado es un buen modelo, que es preciso y no comete tantos errores de falsos positvios o falsos negativos. Podría utilizarse en un contexto organizacional de negocio.

KNN

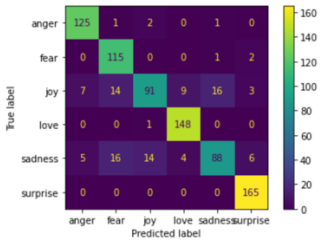
Posteriormente se utilizó la técnica de K-nearest-neighbbors, un algoritmo de aprendizaje supervisado que se basa en el concepto de proximidad, es decir, clasifica los elementos similares de acuerdo a la distancia Euclidiana. El concepto se basa en que los elementos parecidos deben estar cerca. A diferencia de otros modelos, el aprendizaje del KNN ocurre al probar los datos del test. Este algoritmo es clave al momento de realizar recomendaciones ya que su método de clasificación permite observar elementos relacionados dadas sus características. Se inició con la búsqueda de hiperparámetros utilizando la búsqueda con Kfold y GridSearch del cual se pudo obtener que el mejor parámetro es:

```
{'n_neighbors': 1, 'p': 1}
```

El desempeño sobre el conjunto de prueba fue el siguiente:

```
Exactitud: 0.88
Recall: 0.8776978417266187
Precisión: 0.8761156316647769
Puntuación F1: 0.8713134016565672
```

La matriz de confusión fue la siguiente:



Teniendo en cuenta la matriz de confusión, se observan buenos resultados. Tanto la exactitud, el recall, la precisión y la puntuación F1 dan valores superiores a 0.8 y son parecidos por lo que no se evidencian problemas de overfitting ni otros. Las métricas obtenidas demuestran que es un buen modelo y que se podría utilizar para resolver los objetivos del negocio.

Modelo Naive-Baynes

Se intentó realizar un último modelo con el algoritmo Naive-Baynes ya que este algoritmo se basa en las probabilidades y es útil para obtener clasificaciones precisas. Se aplica la ecuación del teorema de Baynes para predecir una instancia de prueba x:

$$p(C = c|X = x) = \frac{P(C = c)p(C = c|X = x)}{p(X = x)}$$

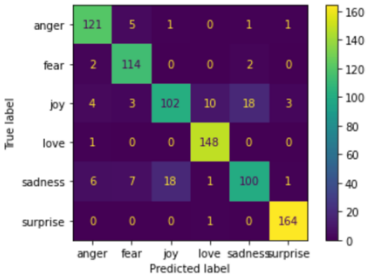
Siendo C la variable aleatoria, X un vector de variables aleatorias, c una etiqueta de clase particular y x un vector de valor de un atributo observado particular. Luego de calcular la probabilidad previa para los sentimientos dados, es decir, las etiquetas, se calcula la probabilidad con cada variable para cada clase. Estos son los valores utilizados en el teorema de Baynes con el fin de calcular la probabilidad posterior para elegir la más alta. Una ventaja de este algoritmo es que admite variables categóricas, más dada la limpieza de datos previa, se utilizaron datos numéricos.

Se inició con la búsqueda de hiperparámetros utilizando la búsqueda con Kfold y GridSearch del cual se pudo obtener que el mejor parámetro es: `{'alpha': 0, 'fit_prior': True}`

El desempeño sobre el conjunto de prueba fue el siguiente:

```
Exactitud: 0.90
Recall: 0.8980815347721822
Precisión: 0.8952534435097056
Puntuación F1: 0.8951315506349566
```

La matriz de confusión fue la siguiente:



Como se puede observar, los resultados son bastante buenos dado que, tanto la precisión, como la exactitud, como el recall y el F1 Socre dan cercanos a 0.9. Esto indica que no se cometen tantos errores en cuanto a falsos positivos y falsos negativos por lo que es un excelente modelo y puede ser utilizado para cumplir con los objetivos propuestos.

Resultados

Tras realizar los diferentes modelos se obtuvieron los siguientes resultados:

	Modelo	Params	Accuracy	Recall	Precission	F1 Score
0	Árbol de clasificación	{'criterion': 'entropy', 'max_depth': 20, 'min...	0.509592	0.509592	0.565810	0.463103
1	Random Forest	{'criterion': 'gini', 'max_depth': None, 'min_...	0.865707	0.865707	0.876914	0.864229
2	Modelo SVM	{'C': 2.154434690031882}	0.931655	0.931655	0.930362	0.930537
3	K Nearest Neighbors	{'n_neighbors': 1, 'p': 1}	0.877698	0.877698	0.876116	0.871313
4	Naive Baynes	{'alpha': 0, 'fit_prior': True}	0.898082	0.898082	0.895253	0.895132

De los cuales se puede concluir que los mejores modelos en cuanto a las métricas seleccionadas fueron el modelo SVM y el de Naive-Baynes. Estos modelos son muy precisos y tienen muy pocos falsos positivos, lo que indica que estos son capaces de generalizar. Al tratar con etiquetas, esto será relevante para el futuro pues el modelo ya tuvo unos datos de entrenamiento de los cuales se basa para realizar las clasificaciones. En general, los algoritmos utilizados pueden servir para clasificar aquellas emociones basándose en los diferentes textos que los usuarios de las redes sociales publica a excepción del modelo de árbol de clasificación, ya que este no tuvo muy buenas métricas de desempeño y no es recomendable utilizarlo. Esto dado que puede clasificar falsos negativos y falsos positivos y la idea es poder obtener un modelo preciso y exacto para automatizar procesos de manera correcta.

Finalmente, en cuanto a la minería de texto, se utilizaron técnicas precisas que permitieron obtener los buenos resultados de las métricas. Lo anterior teniendo en cuenta que las palabras claves se relacionaban con el contexto de negocio y no eran las palabras comunes del idioma. Esto permitió relacionar los sentimientos con estos términos. Asimismo, se realizó la corrección del estadístico tf-idf lo que permitió una cuantificación más exacta lo que será útil en un futuro para realizar la clasificación.

Referencias

Amat, J. (diciembre 2020). Análisis de texto con Python. Recuperado de https://www.cienciadedatos.net/documentos/py25-text-mining-python.html

AprendeIA. (2021). Naive-Baynes Teoría [Blog virutal]. Recuperado de https://aprendeia.com/naive-bayes-teoria-machine-learning/

Mosquera, R., Castrillón, O., Parra, L.(2018). Máquinas de Soporte Vectorial, Clasificador Naïve Bayes y Algoritmos Genéticos para la Predicción de Riesgos Psicosociales en Docentes de Colegios Públicos Colombianos. Recuperado de https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-07642018000600153&lang=pt

Anexos

Repositorio de Github

Link del repositorio: https://github.com/juandiegocardona/Proyecto1

Trabajo en equipo

Estudiante	Roles	Algoritmos	Tareas Realizadas	Horas	Repartición de Puntos
Juan Diego Cardona	Líder de datos Líder de Analítica	Modelo SVM Lineal	• Lectura de Datos • Limpieza y Tokenización • Análisis y Exploración • Vectorización • Edición Documento • Modelado y evaluación algoritmos • Edición ppt y video	12	33,3
María Camila Terán	Líder de Negocio Líder de Analítica	KNN Modelo Naive-Baynes	• Edición Documento • Perfilamiento de datos • Comprensión del Negocio • Enfoque Analítico • Modelado y evaluación algoritmos • Edición ppt y video	12	33,3
Nicolás Ortega	Líder de Proyecto Líder de Analítica	Arboles de Decisión Random Forest	• Vectorización y Balanceo • Edición mayoritaria del Notebook • Edición Documento • Modelado y evaluación algoritmos • Resultados y Conclusiones	12	33,3