Hola José Luis,   
  
Muchas gracias por su atención y guía. Respondo a sus preguntas:

1. **¿Podrías describirnos en más detalle cómo has realizado la limpieza y preparación de datos y que esfuerzo del total del proyecto ha sido?**

Como se comenta en el proyecto, este tiempo se calculó inicialmente mucho más reducido porque se esperaba que con el volumen de estudios NLP las librerías de reconocimiento de lenguaje tóxico tuvieran mejores resultados.

Ha sido un 40% del tiempo del proyecto (numerosos estudios indican que un científico de datos utiliza el 60% de su tiempo en ETLs, y se ha estado cerca). Los datos de Kaggle (en inglés) ya se daban estructurados y no presentaron ningún problema a la hora de realizar el análisis.

El primer análisis de recopilación de insultos con la intención de tener una visión general de la situación fue sencillo aunque lento, por tener que ir copiando cada uno de ellos a mano. Se encontró un artículo en el que había una recopilación que más tarde se iría ampliando.

El problema comenzó con los datos de twitter. La extracción se realiza desde jupyter que abre una comunicación con la API (código disponible en git), al indicarle el lenguaje, introduces ESP e incluye muchos tweets en mexicano y otros dialectos. Cuyos insultos son diferentes a los del castellano y que estaban fuera de nuestro estudio, por lo que debían de ser eliminamos a mano. Además, necesitábamos que algunos de ellos tuvieran la intención de agredir verbalmente utilizando determinados términos para poder comprobar si eran o no reconocidos, en caso de que no lo fueran poder entrenar con esos datos un modelo robusto. Una vez extraídos y filtrado los tweets, para una buena ingeniería de características las contracciones, abreviaturas, …que se utilizan actualmente en redes sociales, debían de ser sustituidas por su palabra completa (a modo de ejemplo ‘xq’ debía ser convertido con porque) se buscó alguna fuente que hubiera recopilado estas contracciones y sólo se encontró en inglés por lo que la librería de sustitución se tuvo que hacer mediante la inspección de tweets en general, tarea que volvió a ser tediosa pero necesaria. Adicionalmente había que eliminar terminología propias de twitter como @, \_ suprimir páginas web y demás contenido que no nos aportara información.

1. **En el apartado "4.1.3 Análisis de los datos" das a entender que la Matriz de confusión es una muy buena forma de presentar los resultados, podrías presentarnos las matrices de confusión de Multinomial Naive Bayes y Support Vector Machine.**

Aunque es cierto que no ha sido presentado en el documento por estar implícito en los resultados, si que puede ser encontrada la matriz de confusión en el código del Git. Para poder verlo de forma visual se estructura a modo de dibujo:

1)

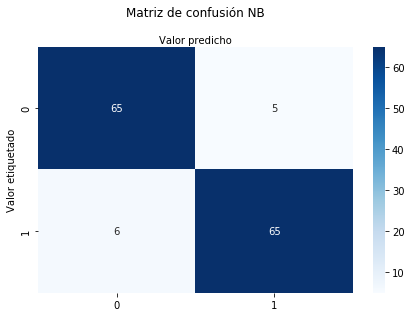


Ilustración 1 Matriz confusión Naive Bayes

Solo encontramos de los 141, once falsos positivos sobre la muestra, cinco fueron reconocidos como tóxicos cuando no lo eran y 6 como no tóxicos cuando si que lo eran.

2)

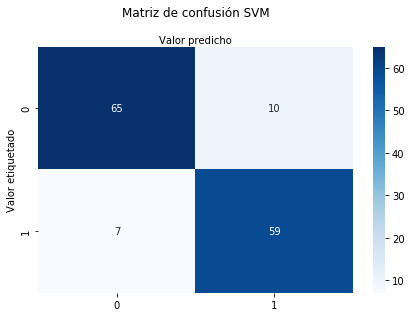


Ilustración 2Matriz de confusión SVM

El modelo SVM presenta un comportamiento peor incrementando el número de falsos positivos (en 5) y falsos negativos (en 1).

1. **Los modelos que presentas tienen sus propias debilidades y fortalezas sobre el conjunto de datos. Podrías explicarnos cuales son, y, por tanto, un modelo que convine ambos modelos (usar como atributos las predicciones hechas por otros clasificadores en lugar de los datos originales de entrada) en qué grado mejoraría el "Accuracy"**

**SVM**

|  |  |
| --- | --- |
| Fortalezas | Debilidades |
| Exactitud | Dependiendo de los parámetros elegidos incluido el kernel puede llevar a casos de sobre entrenamiento. |
| Funciona bien con conjuntos de datos pequeños | En caso de datasets grandes el tiempo de entrenamiento puede ser demasiado grande |
| Permite a separación de datos incluso cuando no es una línea lo que divide el grupo de datos. Esto está permitido gracias a la elección del kernel | No funciona bien con datos que contienen ruido |

**Naive Bayes**

|  |  |
| --- | --- |
| Fortalezas | Debilidades |
| Simplicidad al ser implementado debido a la simplicidad del modelo | Da por supuesto la independencia entre variables, cosa que es muy difícil en la vida real |
| Computacionalmente rápido | La precision y el ‘recall’ con volúmenes de datos pequeños se mantienen bajos. |
| Funciona bien con grandes volúmenes de datos |  |

Con la intención de introducir un tercer modelo que desempatara a la hora de tener por ejemplo SVM=1 y NB=0. Se hace el cálculo con el modelo Random Forest cuyo ‘accuracy’ es 84.39 y su matriz de confusión:

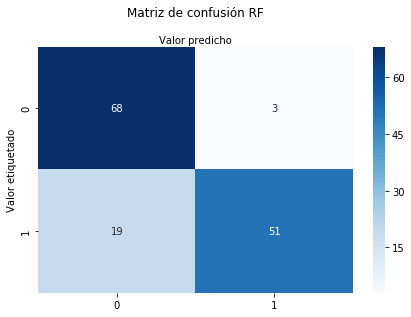


Ilustración Matriz de confusión Random Forest

Utilizamos la función ensemble de sklearn. Una vez colocados los tres modelos en una matriz llamada estimadores creamos un clasificador de votación, al poner el parámetro como hard estamos indicando que haga predicciones por mayoría de votos.

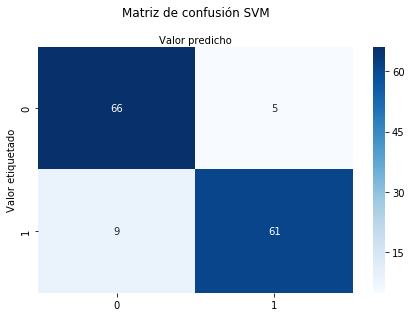


Ilustración Resultado ensemble RF, SVM y NB

Donde comprobamos que el modelo tiene un desempeño prácticamente igual que el Naive Bayes porque los nuevos acertado hacen incrementar también los falsos positivos.