Prueba - Modelos avanzados y redes neuronales

En esta prueba validaremos nuestros conocimientos en modelos avanzados y redes neuronales. La proliferación de noticias falsas en plataformas digitales ha alcanzado proporciones alarmantes, socavando la confianza pública en la información y distorsionando la percepción de la realidad. Ante este desafío, la aplicación de técnicas avanzadas de inteligencia artificial, como las redes neuronales recurrentes, emerge como una herramienta crucial para combatir este fenómeno. Al entrenar modelos de aprendizaje profundo con conjuntos de datos vastos y variados, podemos potenciar su capacidad para discernir entre información veraz y engañosa, contribuyendo así a preservar la integridad del discurso público y fortalecer la salud democrática. Una evaluación rigurosa de estos modelos, con un enfoque en su precisión, robustez y capacidad para adaptarse a la evolución de las tácticas de desinformación, resulta fundamental para garantizar su efectividad y relevancia en la lucha contra las noticias falsas en la era digital.

Lee todo el documento antes de comenzar el desarrollo individual, para asegurarte de tener el máximo de puntaje y enfocar bien los esfuerzos.

Tiempo asociado: 3 horas cronológicas.

Descripción

En esta ocasión trabajaremos con dos bases de datos. La primera de ellas corresponde a textos de noticias clasificadas como Verdaderas o Falsas; y partir de la información que tenemos de ellas entrenaremos un modelo de red neuronal recurrente que permita clasificarlas noticias como tales.

La segunda base de datos posee información de artículos compartidos por la compañía de informaciones Mashable en un periodo de dos años, para la que tenemos múltiples características que describen los textos junto con la cantidad de veces que cada uno de estos artículos fue compartido. Nuestra misión respecto de esta segunda base de datos será entrenar múltiples modelos para predecir la cantidad de veces que una noticia se compartirá. Las variables presentes en esta base de datos son:

1. **url**: URL del artículo
2. **timedelta:** Días transcurridos entre la publicación del artículo y la adquisición del conjunto de datos
3. **n_tokens_title:** Número de palabras del título
4. **n_tokens_content:** Número de palabras del contenido
5. **n_unique_tokens:** Tasa de palabras únicas en el contenido
6. **n_non_stop_words:** Tasa de palabras que no se detienen en el contenido

7. **n_non_stop_unique_tokens:** Tasa de palabras únicas sin parada en el contenido
8. **num_hrefs:** Número de enlaces
9. **num_self_hrefs:** Número de enlaces a otros artículos publicados por Mashable
10. **num_imgs:** Número de imágenes
11. **num_videos:** Número de vídeos
12. **average_token_length:** Longitud media de las palabras del contenido
13. **num_keywords:** Número de palabras clave en los metadatos
14. **data_channel_is_lifestyle:** ¿Es el canal de datos 'Estilo de vida'?
15. **data_channel_is_entertainment:** ¿Es el canal de datos 'Entretenimiento'?
16. **data_channel_is_bus:** ¿Es el canal de datos 'Business'?
17. **data_channel_is_socmed:** ¿Es el canal de datos 'Social Media'?
18. **data_channel_is_tech:** ¿Es el canal de datos 'Tech'?
19. **data_channel_is_world:** ¿Es el canal de datos 'Mundo'?
20. **kw_min_min:** Peor palabra clave (cuotas mín.)
21. **kw_max_min:** Peor palabra clave (participaciones máx.)
22. **kw_avg_min:** Peor palabra clave (acciones medias)
23. **kw_min_max:** Mejor palabra clave (acciones mín.)
24. **kw_max_max:** Mejor palabra clave (acciones máx.)
25. **kw_avg_max:** Mejor palabra clave (acciones medias)
26. **kw_min_avg:** Palabra clave media (participaciones mínimas)
27. **kw_max_avg:** Palabra clave media (participaciones máx.)
28. **kw_avg_avg:** Palabra clave media (acciones medias)
29. **self_reference_min_shares:** Cuotas mínimas de los artículos referenciados en Mashable
30. **self_reference_max_shares:** Máx. de shares de artículos referenciados en Mashable.
31. **self_reference_avg_sharess:** Acciones medias de artículos referenciados en Mashable.
32. **weekday_is_monday:** ¿El artículo se publicó un lunes?
33. **weekday_is_tuesday:** ¿El artículo se publicó un martes?
34. **weekday_is_wednesday:** ¿El artículo se publicó un miércoles?
35. **weekday_is_thursday:** ¿El artículo se publicó un jueves?
36. **weekday_is_friday:** ¿El artículo se publicó un viernes?
37. **weekday_is_saturday:** ¿El artículo se publicó un sábado?
38. **weekday_is_sunday:** ¿El artículo se publicó un domingo?
39. **is_weekend:** ¿El artículo se publicó en fin de semana?
40. **LDA_00:** Cercanía al tema 0 del LDA
41. **LDA_01:** Cercanía al tema 1 del LDA
42. **LDA_02:** Cercanía al tema 2 del LDA
43. **LDA_03:** Cercanía al tema 3 del LDA
44. **LDA_04:** Cercanía al tema 4 del LDA
45. **global_subjectivity:** Subjetividad del texto
46. **global_sentiment_polarity:** Polaridad del sentimiento del texto
47. **global_rate_positive_words:** Tasa de palabras positivas en el contenido
48. **global_rate_negative_words:** Tasa de palabras negativas en el contenido

49. **rate_positive_words:** Tasa de palabras positivas entre los tokens no neutrales
50. **rate_negative_words:** Tasa de palabras negativas entre los tokens no neutrales
51. **avg_positive_polarity:** Polaridad media de las palabras positivas
52. **min_positive_polarity:** Polaridad mínima de las palabras positivas
53. **max_positive_polarity:** Polaridad máxima de las palabras positivas
54. **avg_negative_polarity:** Polaridad media de las palabras negativas
55. **min_negative_polarity:** Polaridad mínima de las palabras negativas
56. **max_negative_polarity:** Polaridad máxima de las palabras negativas
57. **title_subjectivity:** Subjetividad del título
58. **title_sentiment_polarity:** Polaridad del título
59. **abs_title_subjectivity:** Nivel absoluto de subjetividad
60. **abs_title_sentiment_polarity:** Nivel de polaridad absoluta
61. **shares:** Número de veces en que se compartio el artículo (objetivo)

Aplicando los conceptos y herramientas aprendidas hasta ahora deberás realizar las siguientes tareas:

1. Carga la base de datos **news1.csv** y realiza un conteo por clase (columna label). Divide el conjunto de datos en entrenamiento y test (33%) y realiza una exploración de frecuencias de palabras.

2. Para nuestro análisis usaremos un Word Embedding ya entrenado, en particular utilizaremos word2Vec entrenado con corpus Google News con 3 millones de palabras, cada una representada con 300 dimensiones. Para esto, carga Word Embedding de Google como se muestra a continuación

```
url = 'https://drive.google.com/uc?id=191stTi4bltaYgZX5l-i2mcxjcxjuMNPK'

# Nombre del archivo descargado
nombre_archivo = 'archivo.bin'

# Descargar el archivo desde Google Drive
gdown.download(url, nombre_archivo, quiet=False)

# Cargar el archivo con KeyedVectors
word2vec_model_google =
KeyedVectors.load_word2vec_format(nombre_archivo, binary=True)
```

   Tokeniza las palabras del conjunto de entrenamiento con un máximo de 80.000 palabras para el vocabulario, y realiza padding para cada secuencia con largo máximo de 80. Finalmente, crea la matriz de embedding usando el Word2Vec pre entrenado.

3. Implementa un modelo de red neuronal recurrente LSTM, con un mínimo de tres capas, empleando regularización Dropout. (Considera la capa de Embedding con los

pesos pre-entrenados). El modelo debe lograr un accuracy superior al 80% y la cantidad de épocas no debe superar las 20. Muestra las métricas apropiadas para medir el rendimiento del modelo, incluyendo la curva ROC.

4. Realiza predicciones para el conjunto de noticias del archivo "**news_pred.csv**", que contiene siete noticias. Analiza una noticia en la que el modelo se equivoque; si no la hay, una en la que acierte el modelo. Comenta tu resultado.

5. Carga la base de datos de artículos publicados por Mashable llamada "**OnlineNewsPopularity.csv**", quita los espacios en blanco que existen en los nombres de las características y elimina columnas que no aporten información a modelos de predicción de la cantidad de veces que el artículo se compartirá. Revisa la existencia o no de valores ausentes y decide qué hacer con ellos en caso que corresponda. Finalmente, calcula los principales indicadores estadísticos para las variables numéricas y comenta.

6. Realice una búsqueda de outliers para la variable objetivo ***shares*** y elimina estos valores. Muestra gráficamente la distribución de la variable con outliers y sin outliers. Calcula y muestra gráficamente las correlaciones entre las variables numéricas y lista aquellas cuyas correlaciones son mayores a 0.7 en términos absolutos; luego elimina una variable de cada par de ellas con correlaciones altas. Debes mostrar gráficamente la situación antes de la eliminación de variables con alta correlación y después de haber realizado la eliminación.

7. Escoge cuatro variables numéricas y analiza su distribución. Luego normaliza todas las variables numéricas y divide los datos en entrenamiento y test (33%).

8. Entrena un modelo de Random Forest usando búsqueda de grilla. La búsqueda debe considerar:

    - para **n_estimators,** valores entre 100 y 300 para 10 valores
    - para **max_depth,** valores entres 2 y 15 para 10 valores
    - para **max_features,** buscar para 'sqrt' y 'log2'.

    En la búsqueda de grilla considera cinco fold. Muestra los mejores hiper parámetros encontrados por la búsqueda de grilla. Calcula dos métricas sobre el conjunto de test, para evaluar el modelo.

9. Entrena un modelo de red neuronal Fully Connected con al menos dos capas ocultas y aplicando regularización Dropout. Utiliza en cada capa una cantidad de neuronas mayor a 300. Calcula las mismas métricas usadas en el modelo anterior, sobre el conjunto de test, para evaluar el modelo.

10. Implementa y entrena un modelo Extreme Gradient Boosting, usando búsqueda de grilla con 5 fold. Para ello considera:
    - **n_estimators**: [100, 120, 150, 200, 300]
    - **learning_rate**: [0.008, 0.07, 0.009, 0.01, 0.02]
    - **subsample**: np.linspace(0.05, 1, 5)
    - **scale_pos_weight**: np.linspace(0.8, 0.9, 2)

Muestra los mejores hiper parámetros encontrados, calcula y muestra gráficamente la importancia de las características para la predicción en el modelo. Calcula las mismas métricas usadas en el modelo anterior, sobre el conjunto de test, para evaluar el modelo.

11. Construye un modelo de Bagging usando al menos tres modelos heterogéneos. Para ello utiliza la siguiente función (utilizando el archivo entregado):

```
import util_bagging as ubagging
ubagging.bagging_het
```

    Calcula las mismas métricas usadas en el modelo anterior, sobre el conjunto de test, para evaluar el modelo.

    Muestra en un DataFrame los resultados de todos los modelos y concluye.

Requerimientos

1. Implementa un modelo de red neuronal recurrente LSTM para el procesamiento de textos, aplicando tokenización, padding y creando una matriz de embedding; utiliza el modelo para realizar predicciones. **(3 puntos)**

2. Analiza variables considerando su distribución y correlación, y prepara los datos de acuerdo con ello para implementar modelos de ensamble. **(1 punto)**

3. Implementa modelos de ensamble Random Forest y Bagging, considerando los requerimientos necesarios y evaluando su desempeño. **(3 puntos)**

4. Implementa diversos modelos de redes neuronales, realiza predicciones con ellos y los evalúa, comparando con otros modelos predictivos. **(3 puntos)**

😊 ¡Mucho éxito!

Consideraciones y recomendaciones
• Recuerda añadir todos los comentarios y descripciones que consideres

importantes para comprender tu desarrollo.

- Es posible que, si no cuentas con los recursos adecuados en tu computador, tengas algunos problemas en la ejecución local de Keras y Tensorflow, además de algunos problemas para cargar las bases de datos (que son muy pesadas). Puedes optar por cargar los datos online directamente para trabajar el Google Colab; consulta con tu tutor respecto de la manera de hacerlo.