.UBAfiuba
FACULTAD DE INGENIERÍA

Organización de Datos - Ciencia de Datos
(9558 - TA047) Curso 02 - Rodriguez

Trabajo Práctico 2: Story Points

2º Cuatrimestre 2024
Grupo Nº 1

Integrantes:

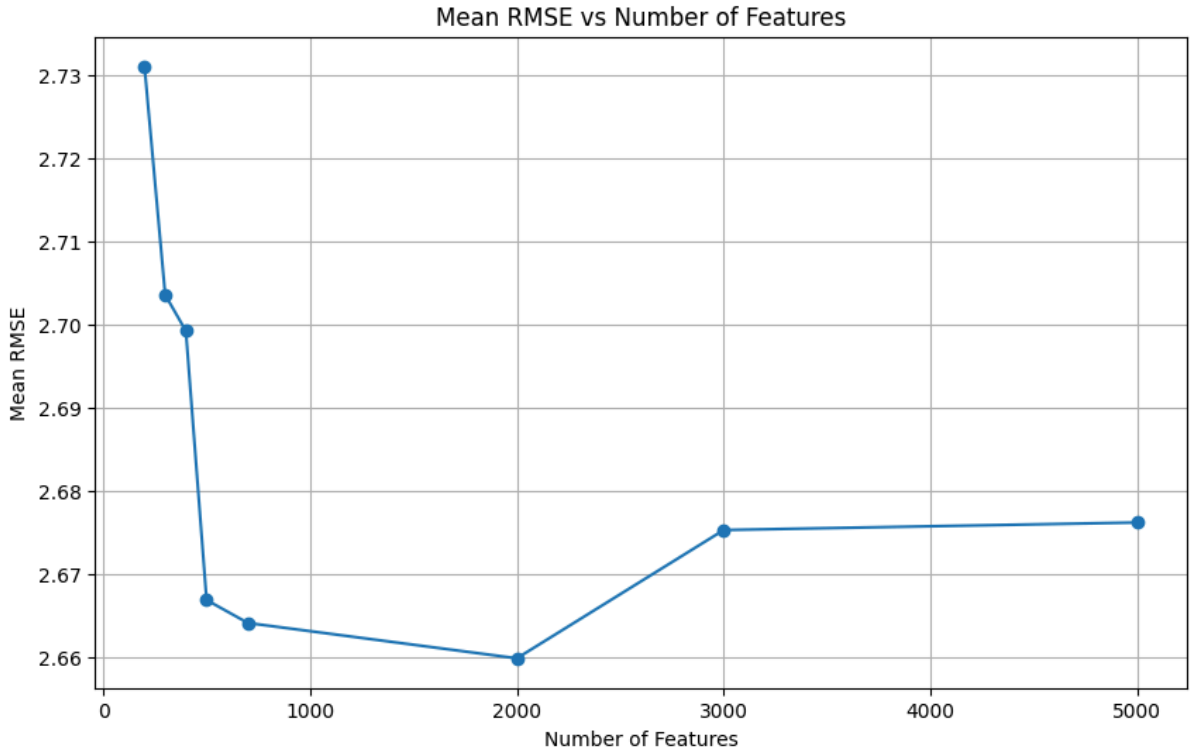
Beltrán Malbrán	**110036**
Florencia Dellisola	**109897**
Gian Luca Spagnolo	**108072**
Marcela Jazmín Cruz	**110066**

Índice

Introducción

En este Trabajo Práctico, disponemos de un conjunto de datos que contiene una serie de datos de uso (user stories) de distintos proyectos y el número de story points que tiene asignado cada uno. Los story points indican la complejidad de cada tarea. Nuestro objetivo será predecir el **story point** de cada **user story** dado el texto que lo representa. El dataset está compuesto por 7900 filas y 5 columnas, en las cuales no había ni duplicados ni nulos. El campo user story está dividido en las columnas título y descripción.

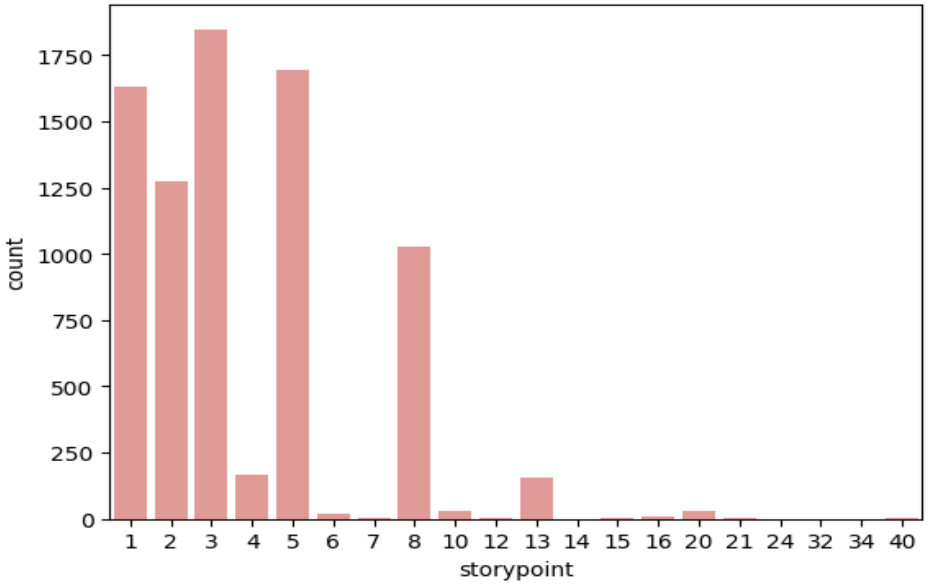
En primer lugar se realizó un preprocesamiento de los datos, donde se armó el atributo user story, el cual es la combinación del texto y la descripción, para luego poder eliminar los caracteres especiales, y por último se normalizo todo a minúsculas y se vectorizó. Una vez hecho el preprocesamiento del texto, se hizo el proceso de lematización sobre las palabras que aportan un significado, es decir las que no son stopwords. Este proceso consiste en llevar todas las palabras a su forma de diccionario o "lema". Luego se realizó una vectorización de este campo usando *TF-IDF*, el cual es una medida estadística utilizada para evaluar la importancia de una palabra dentro de un documento en relación con un conjunto de documentos. Por último se realizó un análisis de cuál es el número de características óptimo en relación al aumento del error. Tal como se puede ver en el siguiente gráfico se seleccionaron 2000 características.



Todo este procesamiento de datos se hizo para los siguientes modelos.

- Bayes Naïve
- Random Forest
- XGBoost
- Un modelo de red neuronal aplicando Keras y Tensor Flow.
- Un ensamble de, al menos, 3 modelos elegidos por el grupo.

También se analizó la distribución de la variable a predecir.



Cuadro de Resultados

Modelo	MSE	RMSE	Kaggle
Bayes Naive	12.481645569620254	3.5329372439402675	3.95117
Random Forest	6.1745978261270125	2.484873804869578	2.82918
XGBoost	5.620674809708	2.3707962396	2.81653
Red Neuronal	5.67374892038	2.38196324916	2.84119
Ensamble	5.8394464302973095	2.645590240773861	2.95222

El mejor predictor entre todos los modelos resultó ser **XGBoost**.

Descripción de los Modelos

En esta sección, se indica en qué consiste cada modelo de la tabla, junto a los hiperparámetros elegidos, técnicas de preprocesamiento de datos, modelos ensamblados y el entrenamiento correspondiente para poder detallar el caso del mejor modelo. Mencionamos los algoritmos y técnicas aplicadas y, para la Red Neuronal, se describe la arquitectura implementada.

XGBoost

Para este modelo, se comenzó haciendo el split de los *train* y *test sets* con una proporción de 80-20. Luego, se continuó con la búsqueda de los mejores hiperparámetros, los cuales fueron encontrados mediante haber aplicado **Grid search**. De esta forma, los mejores hiperparametros resultaron ser:

```
{'learning_rate': 0.1, 'max_depth': 9, 'n_estimators': 300, 'subsample': 0.8}
```

Una vez entrenado el modelo con estos hiperparametros, el RMSE obtenido de este modelo fue: `2.3707962396014364`

Random Forest

Para este modelo se realizó un *train test split* con una proporción de 80-20. Posteriormente, se buscaron los mejores hiperparametros al realizar **Random Search**, los cuales resultaron ser:

```
{'n_estimators': 150, 'min_samples_split': 13, 'min_samples_leaf': 10, 'max_depth': 30}
```

Una vez entrenado el modelo con estos hiperparametros, el RMSE obtenido de este modelo fue: `2.484873804869578`

Bayes Naïve

Este modelo se utilizó junto con el vectorizador **TF-IDF**. Luego, se realizó una búsqueda aleatoria de hiperparámetro utilizando **RandomizedSearchCV** con los siguientes valores probados: `'var_smoothing': [1e-09, 1e-06,1e-04]`

De esta forma, el mejor hiperparámetros encontrados han sido: `'var_smoothing':0.0001`

Una vez encontrados este hiper parámetro procedimos a entrenarlo, lo que resultó en un RMSE de `3.5329372439402675`

Red Neuronal

Para este modelo se realizó una selección de características con **SelectKBest**, seleccionando las 2000 características más relevantes. El entrenamiento de la red neuronal se realizó con **DropOut** y **penalización L2**. Las características de nuestra red neuronal implementada son las siguientes:

- **Capa de entrada**: 2000 (tamaño igual al número de características seleccionadas.
- **Capa oculta 1**: 64 neuronas, activación ReLU, regularización L2 (kernel_regularizer=0.01) y Dropout (30%).
- **Capa oculta 2**: 32 neuronas, activación ReLU, regularización L2 y Dropout (30%).
- **Capa de salida**: 1 neurona.

Teniendo estas características en cuenta, se decidió utilizar el *optimizador* **Adam** con una tasa de aprendizaje de 0.0002.

Se eligió esta arquitectura ya que la **regularización L2** ayuda a evitar el sobreajuste al penalizar los pesos más grandes, esto es especialmente útil teniendo en cuenta la diferencia entre los story points mostrada anteriormente. También se utiliza **Dropout** con el fin de evitar el sobreajuste. A la hora de entrenar, se utiliza **Early Stopping** para monitorear la pérdida en el conjunto donde, si no mejora después de 10 iteraciones, frena el entrenamiento y vuelve al mejor peso del modelo.

Modelo de Ensamble

Decidimos utilizar **Voting Regressor** utilizando los modelos **XGBoost**, **Random Forest** y la **Red Neuronal**. Esta elección se hizo ya que son tres modelos con diferentes mecanismos para aprender los datos. Por lo cual, al combinarlos, buscamos que se pueda ver relaciones y patrones entre los datos que individualmente no vieron. Mezclar árboles de decisión con el aprendizaje de una red neuronal vuelve el modelo más robusto a la hora de predecir el valor.

Conclusiones Generales

Consideramos que el análisis exploratorio de los datos resultó importante para comprender las características de las *User Stories* y su distribución de *Story Points*. Observamos que no había valores duplicados ni nulos, y continuamos con las tareas de preprocesamiento. La limpieza de texto, lematización y vectorización a través de TF-IDF contribuyeron favorablemente en el desempeño de los modelos.

El modelo XGBoost obtuvo el mejor RMSE en el set de test local (`2.3707962396014364`). El modelo con mejor desempeño en Kaggle también fue el XGBoost (`2.81653`).

Consideramos que el modelo XGBoost podría ser utilizado de manera productiva para predecir en un entorno real debido a su desempeño y capacidad para manejar grandes conjuntos de datos. Es un buen modelo para tener un panorama general de la variables a predecir donde, si bien tiene un error de 2.81653, permite ver las relaciones entre los datos y hacer un análisis general. Para obtener mejoras en los resultados, se podrían ajustar aún más hiperparámetros adicionales en los modelos y así optimizarlos.

Tiempo Dedicado

Integrante	Tarea	Prom. Hs Semana
Beltrán Malbrán	XGBoost Random Forest Red Neuronal Armado de Reporte	5
Florencia Dellisola	Red Neuronal Random Forest Preprocesamiento Armado de Reporte	5
Gian Luca Spagnolo	XGBoost Red Neuronal Armado de Reporte	5
Marcela Jazmín Cruz	Random Forest Bayes Neives Preprocesamiento	5