
TP2 - StoryPoint

Grupo 5 - DataPinto

Mejia Alan
Prieto Pablo
Flores Sosa Zoraida



Competición Kaggle

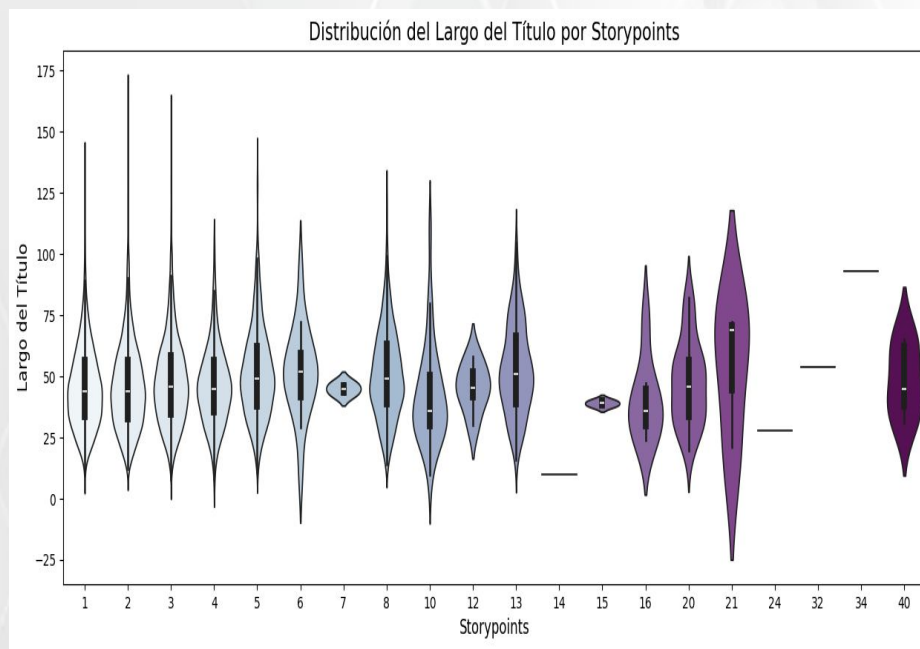
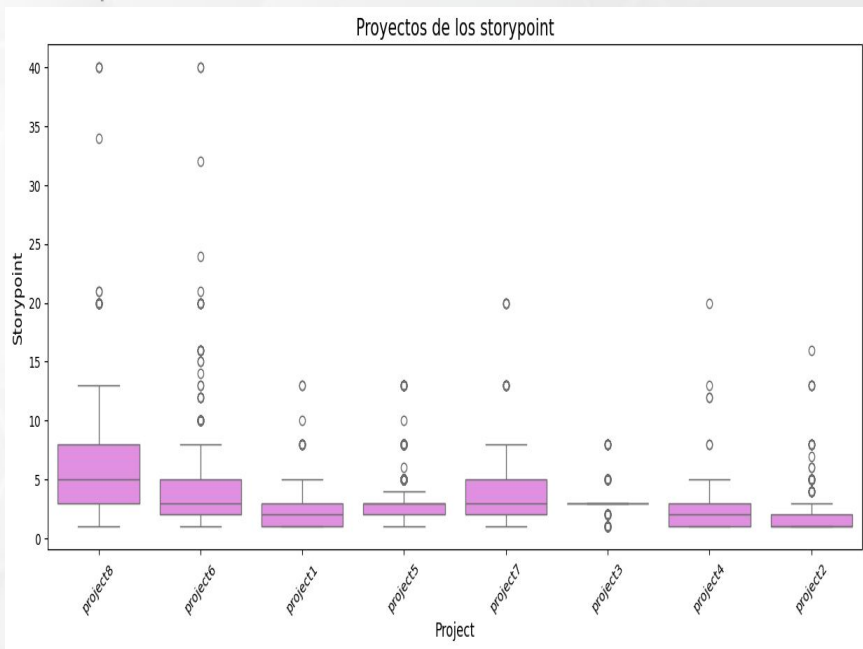


Objetivo

Dado conjunto de datos que contiene una serie de casos de uso (user stories) de distintos proyectos y el número de story points que tiene asignado cada uno, se desea predecir el valor de story point dado un user story.

Análisis Exploratorio

- ▶ El Set de train tiene 7900 datos no nulos y el test 1974 datos no nulos, no hay filas duplicadas
- ▶ **Boxplot:** observamos que hay más proyectos 8 con un promedio de 5 puntos de complejidad. Hay muy pocos de proyecto 3
- ▶ **ViolinPlot:** el largo del título con mayor densidad tiene un puntaje de 21 y su promedio es el más elevado a diferencia del resto



Modelos

- ▶ Random Forest
 - ▶ XGBoost
 - ▶ Bayes Naive
 - ▶ Ensemble
 - ▶ Red Neuronal
-

Random Forest



Preprocesamiento

- ▶ **Features:** Se agregaron Title_largo, Description_largo, Title_token_cant, Description_token_cant
 - ▶ **NLP:** minúsculas, caracteres especiales, stopwords
 - ▶ **División de datos:** Se realizó un split 80% a train y 20% validation y sacamos los storypoints de validation que no estén en el train
 - ▶ **Encodeo:** se realizo One Hot Encoded en project
-

Estructura del Modelo

Random Forest Regressor

- ▶ Random Forest Regressor y para convertir textos se uso Count vectorized: y optimización de búsqueda de hiperparámetros con Random-Search
 - ▶ **Pipeline** : con un preprocesador que transforma las columnas a count vectorized y tiene los hiperparametros de random forest
 - ▶ **Random-Search**: 10 iteraciones, cv=3, n-jobs=-1
 - ▶ **Count-vectorized**: max_features=1000, min_df=5, max_df=0.7
-

Métricas

▶ Hiperparámetros

Estimators: 40
min samples split: 10
min samples leaf: 2
max features: sqrt
max depth: None

▶ Errores

RMSE= 2.74

▶ Kaggle (Público / Privado)

Puntaje = 2.8028 / 2.61685

XGBoost



Preprocesamiento

- ▶ **NLP:** minúsculas, caracteres especiales, stopwords
 - ▶ **División de datos:** Se realizó un split 80% a train y 20% validation
-

Estructura del Modelo

XGBoost Regressor

- ▶ XGBoost con k-fold cross validation de 5 folds
 - ▶ **Cross-validation:** k-folds: num_boots_round = 1000, as_pandas_ true, earling_stopping_round = 10
-

Métricas

▶ Hiperparámetros

Max depth: 15

Alpha: 10

Estimator: 300

learning rate: 0.1

col sample bytree: 0.3

▶ Errores

RMSE= 2.5289

▶ Kaggle (Público / Privado)

Puntaje = 2.81617 / 2.58686

Bayes Naive



Preprocesamiento

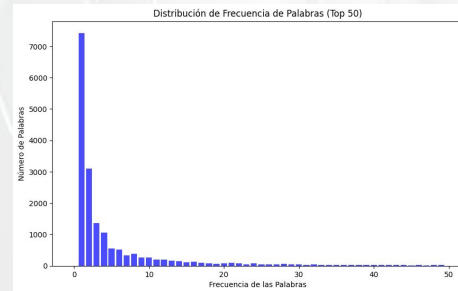
Análisis previo

```
storypoint
3      1465
5      1378
1      1298
2      1005
8       829
4       134
13      128
20       24
10       21
6         14
16         7
40         5
12         4
7          2
15         2
21         2
34         1
32         1
```

- Tamaño del vocabulario después del preprocesamiento
- Longitud promedio de los textos: 74
- Distribución de Frecuencia de Palabras
- Distribución de las etiquetas

Preprocesamiento

- Eliminar caracteres no alfabéticos
- Eliminar números
- Lemmatization
- Unir el título a la descripción
- Vectorización TfIdf
- Filtrado de stop words



Estructura del Modelo

Bayes Naive

- Modelo: **MultinomialNB**
 - Se optimizo los hiperparametros con **GridSearch** y usando **Cross Validation** de 5 folds.
 - Limitar el tamaño del vocabulario mediante **max_features**: [5000, 10000, 20000, 30000, 40000]
 - Incluir unigramas y bigramas **ngram_range**: [(1, 1), (1, 2)]
 - Filtrar palabras por tamaño mediante **min_df**: [3, 5, 10, 15, 20]
 - Dado al sobreajuste en relacion a Kaggle se regulariza mediante **alpha**: [0.5, 1.0, 2.0, 3.0, 5.0]
-

Métricas

► Hiperparámetros

TfidfVectorizer

'ngram_range': (1, 2)
'min_df': 3
'max_features': 20000

MultinomialNB

'alpha': 5.0

► Errores

RMSE= 2.6632

► Kaggle (Público / Privado)

Puntaje = 3.03537 / 2.88536

Ensamble



Preprocesamiento

- ▶ **División de datos:** Se realizó un split 70% train y 30% validation.
 - ▶ **TfidfVectorizer (NLP):** Lematización, minúsculas, caracteres especiales, stopwords.
 - ▶ **TruncatedSVD:** Reducción de dimensionalidad, manteniendo las más relevantes.
 - ▶ **Cross-Validation:** Búsqueda de hiperparámetros con KFold y Random Search.
-

Estructura del Modelo

Stacking Regressor

- ▶ Un ensamble híbrido con 3 modelos entrenando secuencialmente y 1 metamodelo para devolver la predicción final.
 - ▶ **Primer modelo:** XGBoostRegressor
 - ▶ **Segundo modelo:** RandomForestRegressor
 - ▶ **Tercer modelo:** ElasticNet
 - ▶ **Metamodelo:** GradientBoostingRegressor
-

Métricas

► Hiperparámetros

XGBRegressor

`'n_estimators': 100`
`'max_depth': 3`
`'learning_rate': 0.01`

RFRegressor

`'n_estimators': 300`
`'min_samples_split': 2`
`'min_samples_leaf': 1`
`'max_features': 'sqrt'`
`'max_depth': 7`
`'criterion': 'poisson'`

ElasticNet

`'l1_ratio': 0.1`
`'alpha': 1.0`

GradientBostRegressor

`'n_estimators': 200`
`'max_depth': 4`

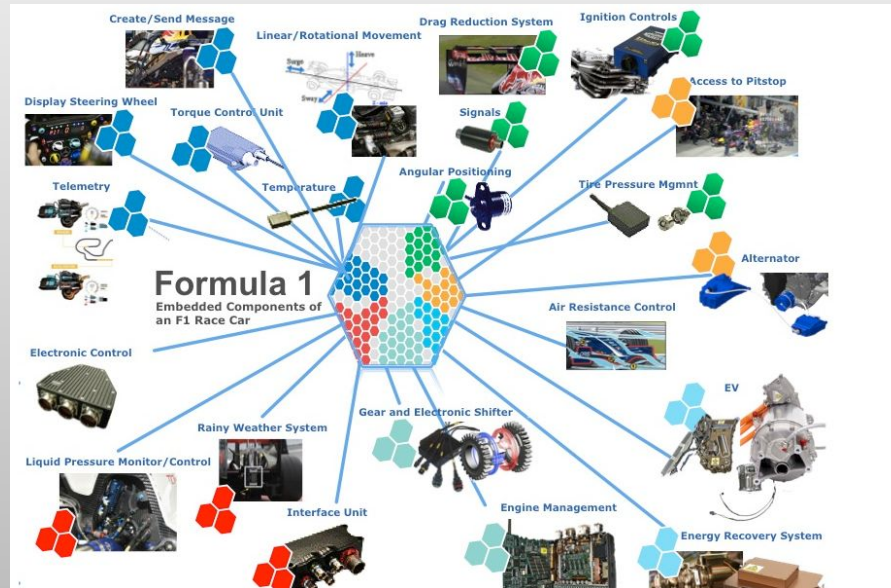
► Errores

RMSE= 2.9470

► Kaggle (Público / Privado)

Puntaje = 2.9733 / 2.85495

Red Neuronal



Preprocesamiento

- ▶ **División de datos:** Se realizó un split 70% train y 30% validation.
 - ▶ **TfidfVectorizer (NLP):** Lematización, minúsculas, caracteres especiales, stopwords.
 - ▶ **TruncatedSVD:** Reducción de dimensionalidad, manteniendo las más relevantes.
 - ▶ **StandardScaler:** Para estandarizar los features numéricos. Es útil para modelos que son sensibles a la escala, como redes neuronales.
 - ▶ **Cross-Validation:** Búsqueda de hiperparámetros con KFold y Random Search.
-

Estructura del Modelo

Keras Regressor (Tensor Flow)

- ▶ Red Neuronal de 4 capas secuenciales (capas decrecientes) y el optimizador parametrizado (con las opciones: Adam, Adamax, RMSprop, Nadam, SGD).
 - ▶ **Primera capa:** Densidad **128** y activación **ReLU** (Rectified Linear Unit), luego **BatchNormalization** (para normalizar luego de cada capa, estabilizar y acelerar el aprendizaje) y un **Dropout** de 30% (apaga 30% aleatoriamente las unidades) para reducir el sobreajuste.
 - ▶ **Segunda capa:** Densidad **64** y activación **ELU** (Exponential Linear Unit). Regularizando los pesos con **L2** (penalizando pesos grandes para evitar el sobreajuste), seguido **BatchNormalization** y **Dropout** 40%.
 - ▶ **Tercera capa:** Densidad **32** y activación **ReLU**.
 - ▶ **Última capa:** Densidad **1** y activación **Linear**, para regresión.
 - ▶ **Regulación Early Stopping:** Monitorear pérdidas y restaurando mejores pesos.
-

Métricas

► Hiperparámetros

TfidfVectorizer

'ngram_range': (1, 2)
'min_df': 3
'max_features': 3000
'max_df': 0.75

TruncatedSVD

'n_components': 100

Keras Regressor

'optimizer_name': 'SGD'
'learning_rate': 0.01
'epochs': 10
'batch_size': 256

► Errores

RMSE= 3.06748



















► Kaggle (Público / Privado)

Puntaje = 2.91698 / 2.74330

RESULTADOS

| | Random-F | XGBoost | Bayes-N | Ensamble | Red N. |
|---------------------------|----------|---------|---------|----------|---------|
| RMSE | 2.448 | 2.5289 | 2.6632 | 2.9470 | 3.0675 |
| Kaggle Público | 2.8028 | 2.8161 | 3.03537 | 2.97331 | 2.91698 |
| Kaggle Privado | 2.61685 | 2.58686 | 2.88536 | 2.85495 | 2.74330 |

Resultado Kaggle

| | | | | |
|---|-------------|---|---------|----|
| 7 | Random Four |    | 2.78301 | 58 |
| 8 | DataPinto |    | 2.80281 | 38 |
| 9 | Organico |    | 2.81653 | 32 |
| 7 | Organico |    | 2.54490 | 32 |
| 8 | DataPinto |    | 2.58686 | 38 |
| 9 | Random Four |    | 2.62991 | 58 |





¡Muchas Gracias!
