

# Pump it Up. Equipo CharlaTED

---

Ignacio Aguilera Martos (nacheteam, KNN), Luis Balderas Ruiz (luisbalru, RIP-PER), Francisco Luque Sánchez (fluque1995, C4.5), Iván Sevillano García (ise-ga24, SVM)

18 de febrero de 2020

Preprocesamiento y Clasificación

1. SVM
2. RIPPER
3. KNN
4. J48

# SVM

---

- Eliminación de variables.
  - *region, recorded\_by, num\_private,...*
  - Variables categóricas con más de 100 valores.

# Preprocesamiento para SVM

- Eliminación de variables.
  - *region*, *recorded\_by*, *num\_private*,...
  - Variables categóricas con más de 100 valores.
- Detección de valores perdidos(NA).
  - *population* = 0??
  - *construction\_year* = 0??
  - Valores vacíos.

# Preprocesamiento para SVM

- Eliminación de variables.
  - *region, recorded\_by, num\_private,...*
  - Variables categóricas con más de 100 valores.
- Detección de valores perdidos(NA).
  - *population = 0??*
  - *construction\_year = 0??*
  - Valores vacíos.
- Cambiar tipos de dato.
  - *region\_code/district\_code* a factor.
  - *date* a numérico.

# Limpieza de ruido e imputación de valores perdidos

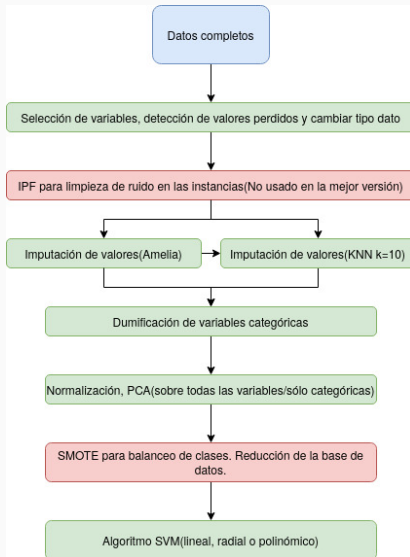
- Limpieza de ruido mediante IPF[SGLH15].
- Amelia [HKB<sup>+</sup>11] para valores perdidos numéricos(método iterativo de imputación).
- Imputación mediante KNN[MT12] de valores categóricos(modal).

## PCA para variables dumificadas. Selección de subespacios relevantes. Selección de instancias.

- One hot encoding. Para cada valor las variables categórica, creamos una variable que valdrá 1 si la instancia tiene este valor.
- Normalizamos las variables.
- Aplicamos PCA[JSV02] y nos quedamos con las variables cuya desviación típica sea mayor de un umbral(0.00001).
- Pasamos de 208 variables a 173.
- El método SMOTE[CBHK02] genera instancias de la clase minoritaria, sobrerrepresentandola.

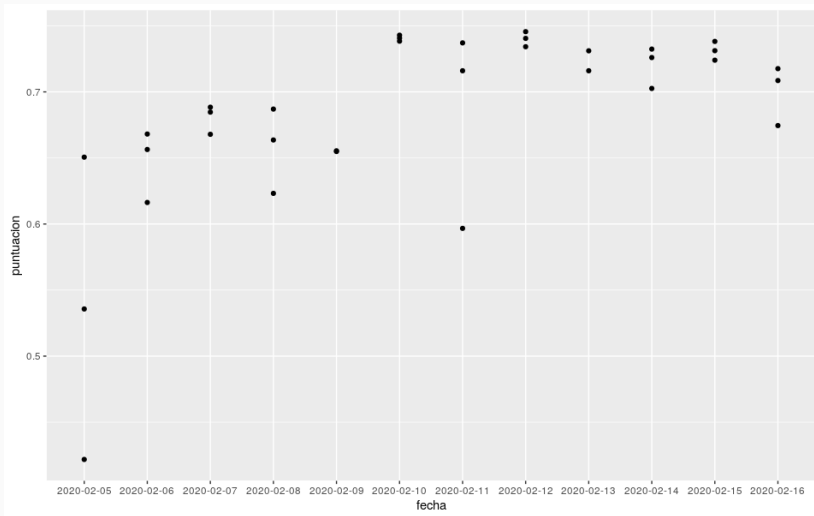


# Flujo de información



**Figura 1:** Verde: Utilizado en el mejor modelo

# Puntuación a lo largo del tiempo de SVM



**Figura 2:** Máxima puntuación: 74.52 %, 12 de febrero

# RIPPER

---

# Algoritmos y técnicas utilizados

- **Ingeniería de características. Selección y creación de características**
  - Selección de variables semánticamente representativas.
  - LasVegas Wrapper.
  - Imputación de valores perdidos con media o mediana.
  - Creación de características.

# Algoritmos y técnicas utilizados

- **Ingeniería de características. Selección y creación de características**
  - Selección de variables semánticamente representativas.
  - LasVegas Wrapper.
  - Imputación de valores perdidos con media o mediana.
  - Creación de características.
- **Técnicas basadas en instancias. Selección, eliminación de ruido, sobremuestreo**
  - ENN.
  - IPF.
  - SMOTE. Oversampling
  - Random Undersampling.

# Algoritmos y técnicas utilizados

- **Ingeniería de características. Selección y creación de características**
  - Selección de variables semánticamente representativas.
  - LasVegas Wrapper.
  - Imputación de valores perdidos con media o mediana.
  - Creación de características.
- **Técnicas basadas en instancias. Selección, eliminación de ruido, sobremuestreo**
  - ENN.
  - IPF.
  - SMOTE. Oversampling
  - Random Undersampling.
- **Ajuste de hiperparámetros.**
  - Gridsearch de parámetros F (número de folds), N (mínimo peso de instancias) y O (número de ejecución para optimizar).
  - 5-CV.

## LasVegas Wrapper

- Implementación en R (FSinR) que no consigue terminar una ejecución completa.

## LasVegas Wrapper

- Implementación en R (FSinR) que no consigue terminar una ejecución completa.

## ENN, IPF

- Limpieza de ruido empeora los resultados tanto en validación cruzada como test. Para ciertas configuraciones, incluso hacen desaparecer la clase minoritaria.



## SMOTE, Oversampling. Random Undersampling

- Necesidad de 'numerizar' los datos. Creación de variables dummies.

## **SMOTE, Oversampling. Random Undersampling**

- Necesidad de 'numerizar' los datos. Creación de variables dummies.
- Las técnicas de oversampling generan datos demasiado artificiales.  
En general, tienen accuracy cercano al 58 %.

## **Imputación de valores perdidos o mal escritos**

## SMOTE, Oversampling. Random Undersampling

- Necesidad de 'numerizar' los datos. Creación de variables dummies.
- Las técnicas de oversampling generan datos demasiado artificiales. En general, tienen accuracy cercano al 58 %.

## Imputación de valores perdidos o mal escritos

- *Funder*: factor de 2141 niveles.
- *Installer*: factor de 2411 niveles.
- Muchos de esos niveles son resultado de escribir mal o de distintas formas la misma palabra.
- Reducción de los niveles hasta aproximadamente 500 corrigiendo los nombres, con peor precisión como resultado.

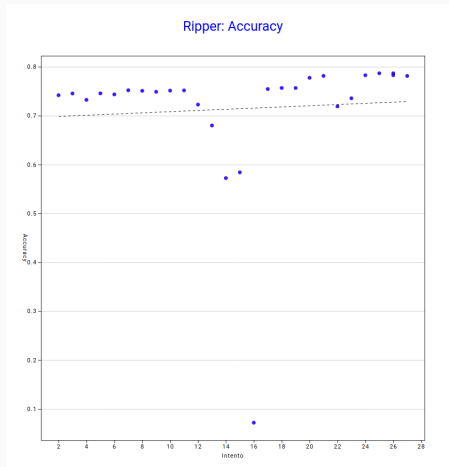
## Selección de características: Muchas columnas parecen inútiles

- Elimino las variables *wpt-name*, *subvillage*, *ward*, *recorded-by*, *scheme-name*, *num-private*, *region-code*, *quantity-group*, *source-type*, *waterpoint-type-group*, *payment-type* y *extraction-type-group*
- Imputo valores perdidos en *funder*, *installer*, *permit*, *scheme-management*, *public-meeting*, *gps-height*, *extraction-type*.
- Imputación de la variable *construction-year* con los resultados de la validación cruzada en entrenamiento del mejor algoritmo hasta el momento (KNN) para las instancias de test. Para training, para cada categoría, tomo la media de las instancias que pertenecen a dicha categoría y no tienen valor perdido ( $\neq 0$ ).

- Ripper (JRip de RWeka) con variables *latitude, longitude, date-recorded, basin, lga, funder, population, construction-year, gps-height, public-meeting, scheme-name, permit, extraction-type-class, management, management-group, payment, quality-group, quantity, source, source-type, source-class* y *waterpoint-type*.
- Hiperparámetros  $F = 2$ ,  $N = 3$ ,  $O = 29$ .

# Resultados

- 27 subidas. Mejor resultado: 0.7869.
- Mejor posición: 1752. Posición actual (17/02): 1834

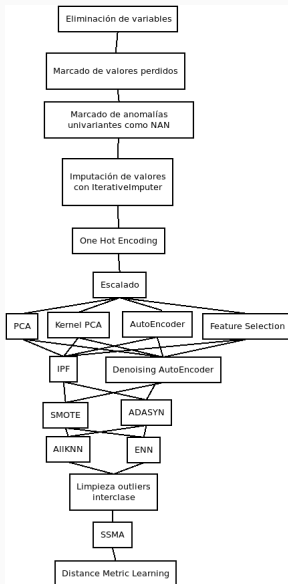


**Figura 3:** Gráfica de resultados

**KNN**

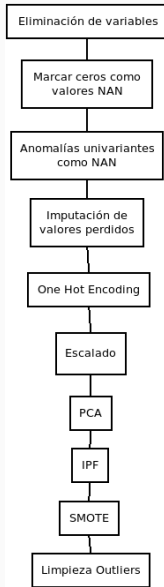
---

# Pipeline empleado





# Pipeline con mejor resultado





## **Eliminación de variables**

Elimino las variables wpt\_name, subvillage, scheme\_name, funder, installer, ward, amount\_tsh y num\_private.

# Explicación de las técnicas

## Eliminación de variables

Elimino las variables wpt\_name, subvillage, scheme\_name, funder, installer, ward, amount\_tsh y num\_private.

## Marcado de anomalías como valores perdidos

En cada columna se calcula la media y la desviación típica. Aquellos datos que se salgan del intervalo  $[media - 5std, media + 5std]$  se marcan como NAN.

# Explicación de las técnicas

## Eliminación de variables

Elimino las variables wpt\_name, subvillage, scheme\_name, funder, installer, ward, amount\_tsh y num\_private.

## Marcado de anomalías como valores perdidos

En cada columna se calcula la media y la desviación típica. Aquellos datos que se salgan del intervalo  $[media - 5std, media + 5std]$  se marcan como NAN.

## Imputación iterativa

Empleamos una imputación iterativa sobre los valores perdidos.

# Explicación de las técnicas

## Eliminación de variables

Elimino las variables `wpt_name`, `subvillage`, `scheme_name`, `funder`, `installer`, `ward`, `amount_tsh` y `num_private`.

## Marcado de anomalías como valores perdidos

En cada columna se calcula la media y la desviación típica. Aquellos datos que se salgan del intervalo  $[media - 5std, media + 5std]$  se marcan como NAN.

## Imputación iterativa

Empleamos una imputación iterativa sobre los valores perdidos.

## PCA

Aplicamos PCA pero sólo sobre las columnas categóricas. El objetivo es explicar las variables categóricas mejor que en su codificación original. Reducimos a 44 variables todas las categóricas.



## **IPF**

Ejecutamos un IPF para limpiar el ruido con 4 iteraciones.



## IPF

Ejecutamos un IPF para limpiar el ruido con 4 iteraciones.

## SMOTE

Hacemos un oversampling de las clases "functional needs repair" y "non functional" a 7500 y 22000 con respecto a 23500 de la clase "functional" con  $k = 7$ .

# Explicación de las técnicas

## IPF

Ejecutamos un IPF para limpiar el ruido con 4 iteraciones.

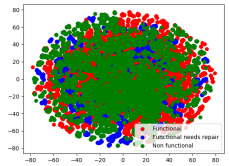
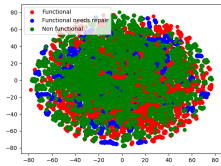
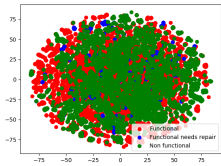
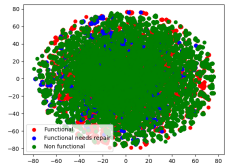
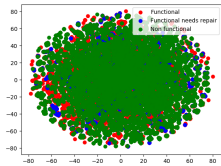
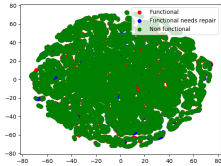
## SMOTE

Hacemos un oversampling de las clases "functional needs repair" y "non functional" a 7500 y 22000 con respecto a 23500 de la clase "functional" con  $k = 7$ .

## Limpieza Outliers

Hacemos una limpieza de anomalías por cada clase eliminando el 1 % más anómalo según KNN con  $k = 7$  y la métrica de la mayor distancia.

# Visualización de las técnicas





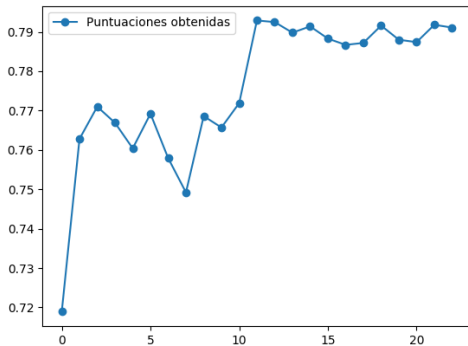
# Elección del K y dimensionalidad de PCA

## Parámetros del modelo

Se hace validación cruzada y se toma tanto el mejor valor de  $k$  para KNN como el mejor valor de dimensionalidad para PCA.

Estos parámetros son  $k = 1$  y  $dim = 44$ .

# Posición en DrivenData



Puntuación final obtenida: 79.29 %

Ranking final: 1729

Número de subidas: 23

**J48**

---

# Primera aproximación

## Eliminación de variables

- `scheme_name`: 1/2 de valores perdidos
- Variables categóricas con más de 100 características
- `recorded_by`: una única categoría
- `region_code`: Correlada con `district_code`
- `construction_year`: Correlada con `gps_height` y valores perdidos (valores 0 en esta columna no tienen sentido)

## Imputación de valores perdidos

- Variables numéricas: valor medio
- Variables categóricas: valor modal

## Resultado

0.7677



# Mejoras a dicha aproximación

## Eliminación de filas con missing values

- Quedan 49841 filas en el conjunto
- Precisión: 0.7328 → se descarta la vía

## Imputación en train por clase

- En lugar de la media y la moda globales, se imputa por media y mediana de la clase
- En test, seguimos imputando igual
- Precisión: 0.7677 → no mejora el resultado, se descarta

## Creación de una nueva categoría para los valores perdidos

- En las variables categóricas se añade una categoría nueva, que representa el valor perdido.
- Precisión: 0.7385 → se descarta la vía

## Segundo estudio - importancia de las variables

### Medición de la importancia de las variables

Para las variables numéricas con valores perdidos en entrenamiento, se computa el valor de dichas variables con la media de los valores de la clase, dejando el resto de variables imputadas de forma poco inteligente. Tras esto, se clasifica el conjunto de entrenamiento con validación cruzada.

### Intuición

Las variables que produzcan una mejora importante en el resultado contendrán información relevante para la solución del problema.

## Segundo estudio - importancia de las variables

### Resultados

Mejora escasa o nula en todas las variables.

Imputación correcta de `construction_year` → Mejora en la CV de 0.78 a 0.84.

La variable `construction_year` parece muy relevante a la hora de establecer la clasificación.

## Segundo estudio - importancia de las variables

### **Cómputo de** `construction_year`

Utilizamos el mejor modelo que tenemos hasta el momento.

En función de la clase que se le ha predicho a cada elemento del test, se imputa el valor en función de la media de dicha clase en el conjunto de entrenamiento.

### **Resultado**

0.7875

# Mejoras sobre este modelo (I)

## Edad de la fuente

Tenemos información de cuándo se hace la medida de la instancia (variable `date_recorded`).

No hay información perdida para dicha variable.

Calculamos la edad de la fuente como la diferencia entre el año de la medida y la fecha de construcción.

Se computa también el mes en el que se registra la fuente.

Precisión: 0.7925

## Eliminación de las variables de registro y antigüedad

Eliminamos las medidas previas, manteniendo sólo la edad.

Intentamos eliminar dependencia entre las variables.

Precisión: 0.7822 → descartamos esta vía, conservamos las tres variables.

## Mejoras sobre este modelo (II)

### **Eliminación de la variable `num_private`**

Esta variable, a pesar de ser numérica, tiene pocos valores distintos, y la mayoría de valores son 0.

No tenemos información sobre qué representa dicha variable, no podemos imputar nada sobre ella, por lo que decidimos prescindir de ella.

Precisión: 0.7945 (mejor modelo)

### **Eliminación del ruido de clase**

Hay filas repetidas (los valores de todas las columnas que nos hemos quedado son iguales), con clases distintas.

Nos quedamos con un solo ejemplo, cuya clase sea la que más se repite entre las filas repetidas.

Precisión: 0.7916 → descartamos el modelo

# Otras propuestas

## PCA

Las variables categóricas se transforman a variables binarias y se extraen las  $k = 28$  primeras características (tantas como variables categóricas teníamos de partida).

Precisión: 0.7822

## PCA + SMOTE

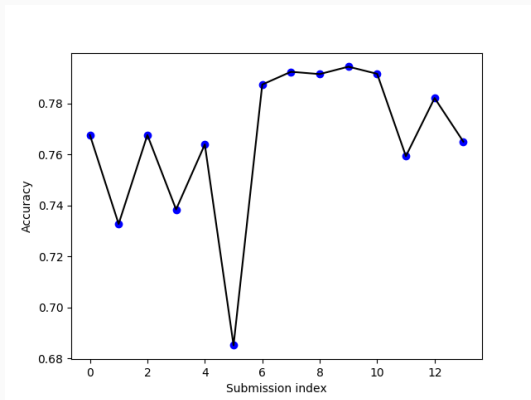
Sobre los datos anteriores, utilizamos SMOTE para intentar balancear las clases:

- La variable `functional needs repair` se iguala a la clase `functional`
- A la variable `non functional` se le duplican el número de ejemplos.

Precisión: 0.7651

Se genera demasiado ruido, y empeora la clasificación.





# Evolución de la puntuación









**Figura 4:** Evolución de la puntuación a lo largo de las subidas

Mejor puntuación final: 0.7945 - Posición actual: 1702



-  *An experiment with the edited nearest-neighbor rule*, IEEE Transactions on Systems, Man, and Cybernetics **SMC-6** (1976), no. 6, 448–452.
-  Shahrokh Asadi and Jamal Shahrabi, *Ripmc: Ripper for multiclass classification*, Neurocomputing **191** (2016), 19 – 33.
-  Nitesh Chawla, Kevin Bowyer, Lawrence Hall, and W. Kegelmeyer, *Smote: Synthetic minority over-sampling technique*, J. Artif. Intell. Res. (JAIR) **16** (2002), 321–357.
-  Salvador García María J del Jesus Francisco Herrera David Charte, Francisco Charte, *A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines*.

-  Haibo He, Yang Bai, Eduardo Garcia, and Shutao Li, *Adasyn: Adaptive synthetic sampling approach for imbalanced learning*, 07 2008, pp. 1322 – 1328.
-  James Honaker, Gary King, Matthew Blackwell, et al., *Amelia ii: A program for missing data*, Journal of statistical software **45** (2011), no. 7, 1–47.
-  I.T. Jolliffe and Springer-Verlag, *Principal component analysis*, Springer Series in Statistics, Springer, 2002.
-  R Malarvizhi and Antony Selvadoss Thanamani, *K-nearest neighbor in missing data imputation*, International Journal of Engineering Research and Development **5** (2012), no. 1, 5–7.
-  *Python distance metric learning*.
-  *Python outlier detection*.

-  Francisco Herrera Salvador García, José Ramón Cano, *A memetic algorithm for evolutionary prototype selection: A scaling up approach.*
-  José A. Sáez, Mikel Galar, Julián Luengo, and Francisco Herrera, *Inffc: An iterative class noise filter based on the fusion of classifiers with noise sensitivity control*, Information Fusion **27** (2015).
-  Kyuseok Shim Sridhar Ramaswamy, Rajeev Rastogi, *Efficient algorithms for mining outliers from large data sets.*
-  Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller, *Nonlinear component analysis as a kernel eigenvalue problem*, 1996.
-  Stef van Buuren and Karin Groothuis-Oudshoorn, *mice: Multivariate imputation by chained equations in r*, Journal of Statistical Software, Articles **45** (2011), no. 3, 1–67.



Dennis L. Wilson, *Asymptotic properties of nearest neighbor rules using edited data*, IEEE Trans. Systems, Man, and Cybernetics **2** (1972), 408–421.

¿Preguntas?