UNIVERSIDAD DE GRANADA

# PRÁCTICA 3: COMPETICIÓN

ANA BUENDÍA RUIZ-AZUAGA

Práctica 3: Competición

**Correo electrónico**
anabuenrua@correo.ugr.es
**Grupo de prácticas**: A (Jorge Casillas)

E.T.S. INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

*Granada, a 5 de enero de 2022*

SUBMISSIONS

| Score | Submitted by | Timestamp |
|---|---|---|
| 0.8513 | Ana_Buendia_UGR_IN | 2021-12-23 21:35:56 UTC |
| 0.8483 | Ana_Buendia_UGR_IN | 2021-12-26 20:52:55 UTC |
| 0.8448 | Ana_Buendia_UGR_IN | 2021-12-26 23:04:49 UTC |
| 0.8576 | Ana_Buendia_UGR_IN | 2021-12-26 23:21:57 UTC |
| 0.8554 | Ana_Buendia_UGR_IN | 2021-12-27 16:18:11 UTC |
| 0.8581 | Ana_Buendia_UGR_IN | 2021-12-27 17:38:58 UTC |
| 0.8515 | Ana_Buendia_UGR_IN | 2021-12-27 23:50:18 UTC |
| 0.8609 | Ana_Buendia_UGR_IN | 2021-12-28 16:59:01 UTC |
| 0.8601 | Ana_Buendia_UGR_IN | 2021-12-28 19:25:49 UTC |
| 0.8609 | Ana_Buendia_UGR_IN | 2021-12-28 23:00:35 UTC |
| 0.8630 | Ana_Buendia_UGR_IN | 2021-12-29 19:06:45 UTC |
| 0.8630 | Ana_Buendia_UGR_IN | 2021-12-29 22:44:24 UTC |
| 0.8628 | Ana_Buendia_UGR_IN | 2021-12-29 23:47:09 UTC |
| 0.8629 | Ana_Buendia_UGR_IN | 2021-12-30 23:33:54 UTC |
| 0.8620 | Ana_Buendia_UGR_IN | 2021-12-31 20:20:59 UTC |
| 0.8630 | Ana_Buendia_UGR_IN | 2021-12-31 20:25:44 UTC |

Figura 1: Tabla de submissions de DrivenData

# ÍNDICE GENERAL

# 1

# PRÁCTICA 3

## 1.1 INTRODUCCIÓN

Para la realización de esta práctica se ha participado en la competición de DrivenData Flu Shot Learning: Predict H1N1 and Seasonal Flu Vaccines.

El objetivo de la competición es predecir si una persona se ha vacunado con la vacuna de H1N1 o la vacuna de la gripe estacional usando 36 atributos distintos, siendo estos tanto categóricos como ordinales. Cabe destacar que algunos atributos, como respondent_id, solo sirven para identificar el ejemplo.

El conjunto de entrenamiento consta de 26707 instancias con sus etiquetas:

- **h1n1_vaccine**: Indica si la persona está vacunada contra el H1N1 (1) o no (0).
- **seasonal_vaccine**: Indica si la persona está vacunada contra la gripe estacional (1) o no (0).

Como método de evaluación se usará el área bajo la curva ROC (AUC) para cada una de las variables, siendo la media de estas la puntuación obtenida.

## 1.2 ESTRUCTURA

El primer archivo de subida es *ejemplo_flu.py*, proporcionado por los profesores, con su correspondiente submission *submission_ejemplo_flu_rf.csv*. Cada uno de ellos se encuentra en su carpeta correspondiente: los archivos fuente se encuentran en *src* (contiene archivos numerados de submissions) y sus correspondientes resultados se encuentran análogamente numerados en la carpeta *submission*.

Además, en el directorio raíz se encuentra este documento de memoria de la práctica, así como el archivo excel editable y su exportación a pdf de la tabla resumen de subidas y la captura de pantalla de las submissions en DrivenData. ´

## 1.3 EXPLORACIÓN DE LOS DATOS

El primer paso en la commpetición es analizar el conjunto de datos y como está distribuido para entender mejor el problema y poder abordarlo de forma eficaz.

### 1.3.1 *Balanceo de clases*

Comenzamos comparando si las clases de nuestro problema están balanceadas o no, ya que si no lo están puede influir mucho en los clasificadores.

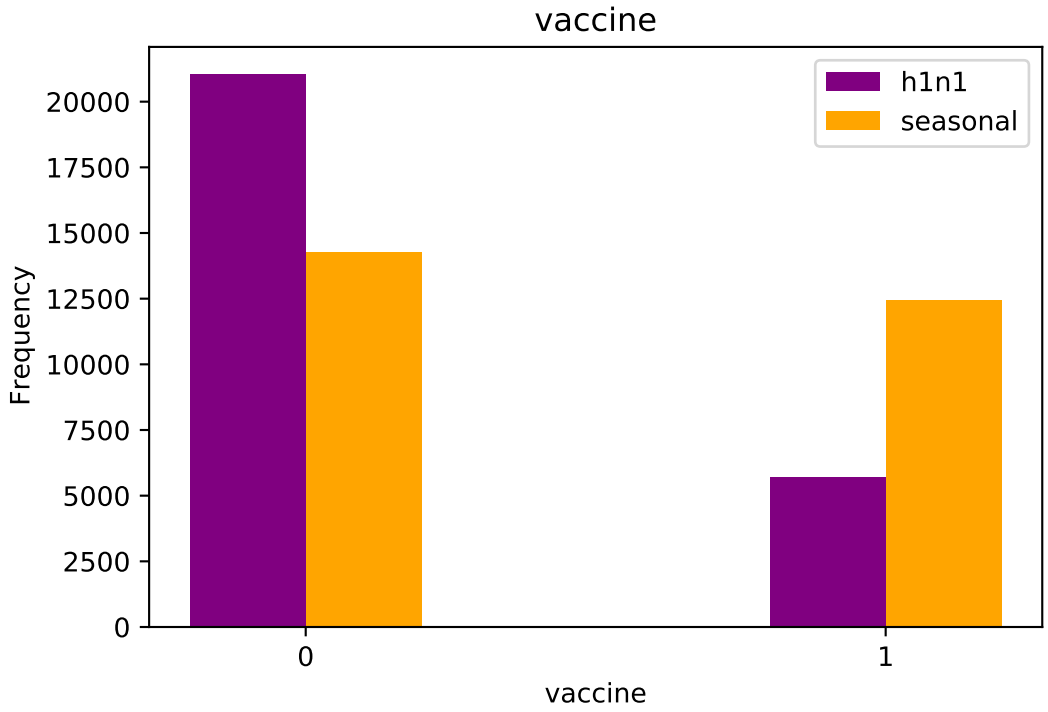


Figura 2: Desbalanceo entre las vacunas

Observamos en (2) que la clase de las vacunas estacionales está más o menos balanceada, mientras que muy poca gente se ha vacunado de h1n1.

### 1.3.2 *Correlaciones*

Vamos a representar también un heatmap que muestre las correlaciones entre las distintas variables y las vacunas, para así tener una idea aproximada de qué variables proporcionan información similar y cuáles pueden ser mejores predictores de las vacunas.

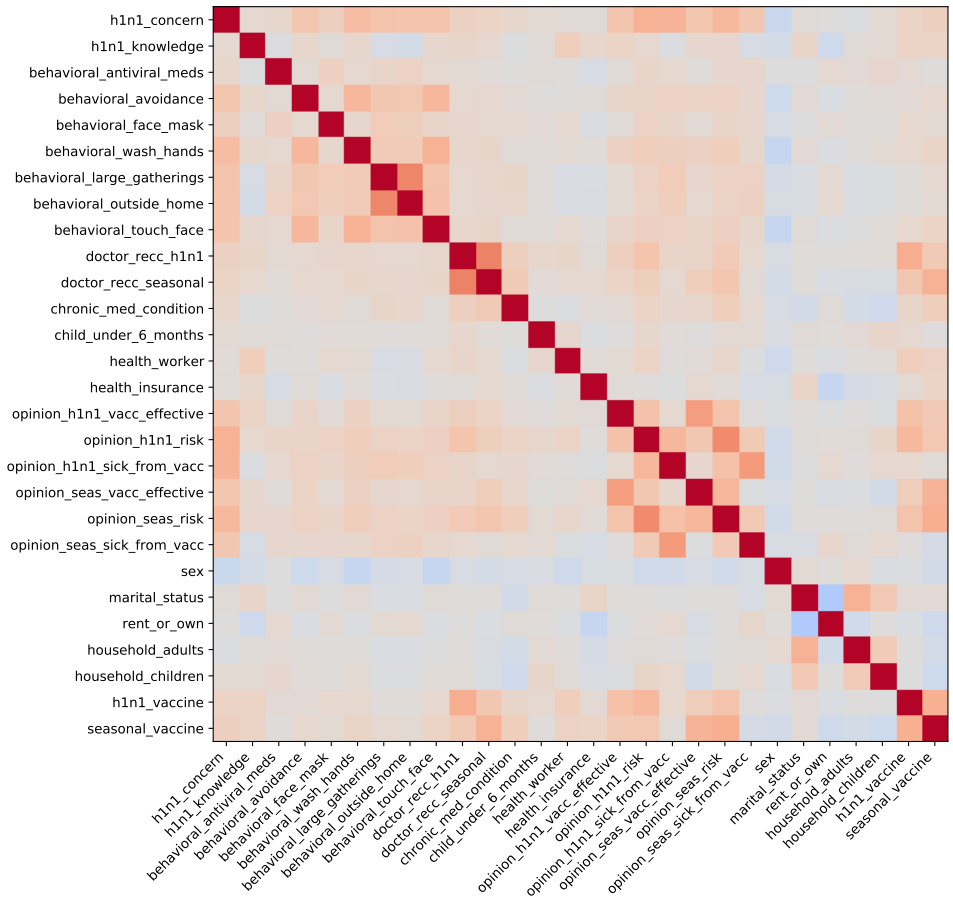


Figura 3: Correlaciones entre las variables y las clases

En (3) lo primero que llama la atención es como ambas vacunas están correlacionadas.

Además, vemos varias agrupaciones con correlación bastante alta, como por ejemplo todo el cúmulo de variables de las opiniones, las recomendaciones de los médicos sobre si ponerse o no una vacuna o todas las variables de comportamientos (lavarse las manos, evitar aglomeraciones etc).

Nos fijamos en que las recomendaciones de los doctores de vacunarse están muy relacionadas con ponerse las vacunas, al igual que las opiniones de los riesgos.

### 1.3.3 *Características*

Mirando el conjunto de características es claro que hay muchos valores perdidos, y, de cara al preprocesado, es necesario comprender la distribución de estas variables para realizar una buena imputacion de valores.

Comenzamos analizando qué códigos de empleo e industria se corresponden con trabajadores de la salud, con el fin de hacer una imputación más correcta.

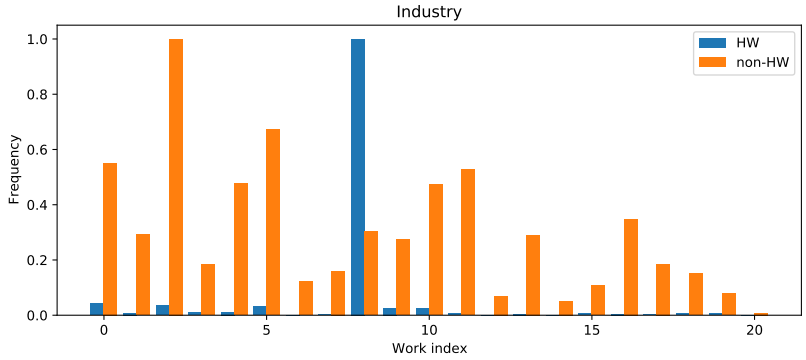


Figura 4: Industrias en las que trabajan sanitarios y no sanitarios

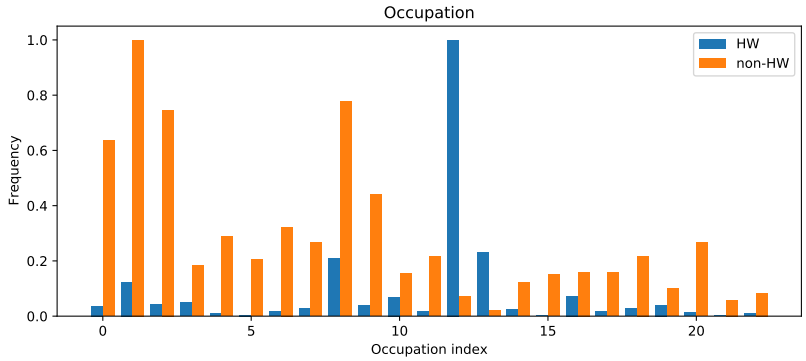


Figura 5: Ocupación de sanitarios y no sanitarios

Como vemos en (4) y (5), los sanitarios están en muchas industrias, pero sobre todo están en una de ellas, al igual que sucede con la ocupación. Por tanto, cada vez que tengamos un sanitario con empleo desconocido, parece razonable asignar esas etiquetas.

Además, muchos de los valores perdidos en industria y ocupación se encuentran en personas en desempleo o que no están en busca de empleo. Vamos a analizar también la correlación entre no estar trabajando, así como su edad y ser sanitario:

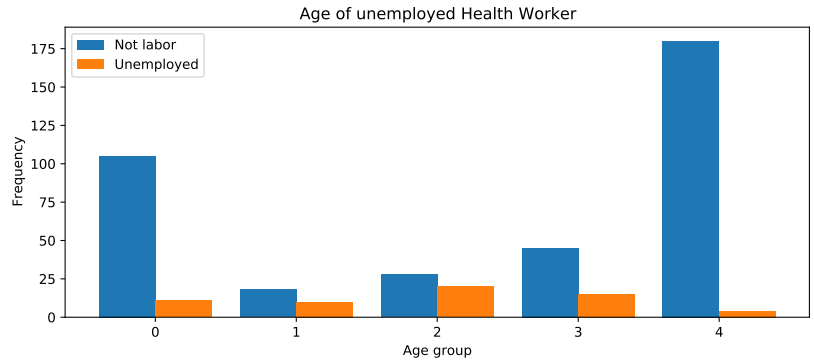


Figura 6: Estado de desempleados o sin buscar trabajo según edad de los sanitarios

Y comparamos los resultados con los de la población general:

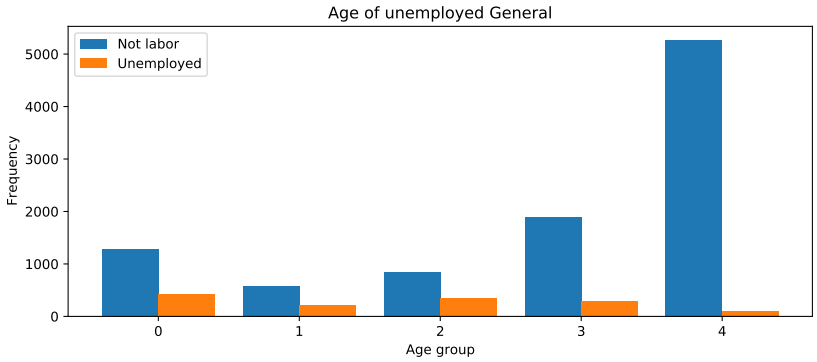


Figura 7: Estado de desempleados o sin buscar trabajo según edad de toda la población

En (6) y (7) se ve más o menos la misma tendencia, con quizá maás estudiantes en medicina que en general.

Estudiamos ahora el estado de empleo según la edad:

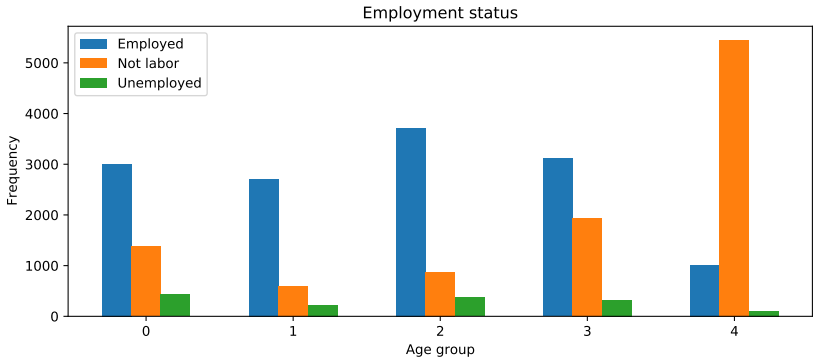


Figura 8: Estado de empleo según edad de toda la población

Observando (8) vemos que el desempleo es bajo para edades menores de 55 años y después sube. También hay muchos estudiantes en el primer grupo. ¿Estos grupos tienen ingresos? Realizamos otra representación para comprobarlo, ya que en las correlaciones parece una variable relevante.

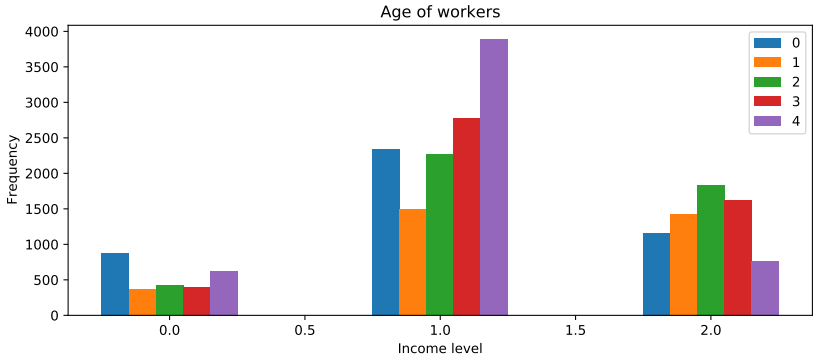


Figura 9: Ingresos según la edad

En (9) se confirma como las personas mayores tienen más ingresos quelas personas jóvenes, pero no parece aportar demasiada información relevante que podamos usar.

## 1.4 PROGRESO

### 1.4.1 *Tabla de progresos*

| Fecha | Hora(CET) | Posición | Sc. Train. | Sc. Test | Preprocesado | Algoritmos | Parámetros | Comentarios |
|---|---|---|---|---|---|---|---|---|
| 23/12/21 | 22:35:00 | 344 | 0.8543 | 0.8513 | Imputación de valores | Random forest | Por defecto | Fichero de prueba del profesor |
| 26/12/21 | 20:52:00 | 344 | 0.8589 | 0.8483 | Eliminando instancias con muchos nulos e imputación de valores simple | Random forest | Por defecto | |
| 27/12/21 | 00:04:00 | 344 | 0.8502 | 0.8448 | Imputación de valores simple | CatboostClassifier | Iterations 10, depth 2, learning_rate 1 | Probando el catboost |
| 27/12/21 | 00:21:00 | 322 | 0.8624 | 0.8576 | Imputación de valores simple | CatboostClassifier | Iterations 45, depth 6 | Ajustados parámetros de catboost para resultado óptimo |
| 27/12/21 | 17:15:00 | 322 | 0.8609 | 0.8554 | Imputación de valores simple | Random forest | n_estimators 300, min_samples_split 35, min_samples_leaf 3 | Pruebo ajuste parámetros de RF |
| 27/12/21 | 18:39:00 | 320 | 0.8656 | 0.8581 | Imputación de valores simple | CatboostClassifier | Iterations 45, depth 6 learning_rate 0.31 | Añado otro paremtro a catboost |
| 28/12/21 | 00:50:00 | 320 | 0.8583 | 0.8515 | Imputación de valores simple | CatboostClassifier | Iterations 70, depth6, learning rate 0.31 weights todo a 1 salvo las reccomendations a 2 | No quería subir esa :') |
| 28/12/21 | 17:59:00 | 213 | 0.8669 | 0.8609 | Imputación de valores simple | LGBM | n_estimators=100, learning_rate=0.072, num_leaves=29, min_child_samples=110 | Probando LGBM con parametros |
| 28/12/21 | 20:25:00 | 213 | 0.8708 | 0.8601 | Imputaciónd de valores simple y oversampling de 2000 instancias | LGBM | n_estimators=100, learning_rate=0.072, num_leaves=29, min_child_samples=111 | Pruebo a hacer oversampling, ha ido mejor de lo esperado |
| 29/12/21 | 00:00:00 | 214 | 0.8675 | 0.8609 | Imputación de valores simple y undersampling de 200 instancias | LGBM | n_estimators=100, learning_rate=0.072, num_leaves=29, min_child_samples=111 | Pruebo el undersampling (la hora esta bien, ha sido a las 00 justas) |
| 29/12/21 | 20:06:00 | 68 | 0.8683 | 0.8630 | Imputacion de valores simple | LGBM | boosting_type="goss", top_rate=0.42, max_depth=0, subsample_for_bin=3000, n_estimators=2500, learning_rate=0.005, num_leaves=29, min_child_samples=100 | Ajusto parametros |
| 29/12/21 | 23:44:00 | 69 | 0.8683 | 0.8630 | Imputacion de valores simple | LGBM | boosting_type="goss", top_rate=0.42, max_depth=0, subsample_for_bin=3000, n_estimators=2500, learning_rate=0.005, num_leaves=29, min_child_samples=100, path_smooth=10 | Intento mejorar parametros |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 30/12/21 | 00:47:00 | 69 | 0.8682 | 0.8628 | Imputacion de valores inteligente | LGBM | boosting_type="goss", top_rate=0.42, max_depth=0, subsample_for_bin=3000, n_estimators=2500, learning_rate=0.005, num_leaves=29, min_child_samples=100, path_smooth=10 | Intento otro preprocesado sin éxito |
| 31/12/21 | 00:33:00 | 69 | 0.8684 | 0.8629 | Imputación de valores inteligente mejorada | LGBM | boosting_type="goss", top_rate=0.42, max_depth=0, subsample_for_bin=3000, n_estimators=2500, learning_rate=0.005, num_leaves=29, min_child_samples=100, path_smooth=10 | Intento otro preprocesado sin éxito |
| 31/12/21 | 21:20:00 | 69 | 0.8675 | 0.8620 | Imputacion de valores simple y normalizacion | LGBM | boosting_type="goss", top_rate=0.42, max_depth=0, subsample_for_bin=3000, n_estimators=2500, learning_rate=0.005, num_leaves=29, min_child_samples=100, path_smooth=11 | Intento preprocesado sin xito |
| 31/12/21 | 21:25:00 | 69 | 0.8683 | 0.8630 | Imputacion de valores simple y normalizacion | LGBM | boosting_type="goss", top_rate=0.42, max_depth=0, subsample_for_bin=3000, n_estimators=2500, learning_rate=0.005, num_leaves=29, min_child_samples=100, path_smooth=12 | Intento normalizar de nuev o sin éxito |

Cuadro 1: Submissions realizadas. Puede consultarse en el excel adjunto. Mejor marca 0,8630 en posición 69

### 1.4.2 *Avances*

La primera entrega fue el fichero proporcionado por los profesores que usa random forest, se usó sin ninguna modificación para probar el sistema de subidas y marcar la medida a superar con las siguientes subidas. En este fichero el único preprocesado consiste en imputar todos los valores perdidos por el más frecuente, y es a lo que nos referiremos de ahora en adelante como imputación de valores simple.

A partir de este fichero, todas las entregas están nombradas como submission<numero>.py y su correspondiente .csv para su fácil identificación.

En segundo lugar se probó un preprocesado en el que se eliminaban todas las instancias con más de tres valores nulos en la misma fila, para comprobar si mejoraba el rendimiento del algoritmo, pero no fue así. Igualmente tampoco mejoró al eliminar algunas columnas que tenían gran cantidad de valores perdidos, como industry u occupation.

Después, dado que en la primera práctica el algoritmo XGBoost dio buen resultado, se ha probado la implementación de CatBoostClassifier, ya que trabaja bien con variables

categóricas. Esta subida fue solamente el algoritmo, y en la siguiente se cambiaron los parámetros del mismo para mejorar la marca.

Dado que el resultado mejoró notablemente al cambiar los parámetros, se volvió al algoritmo random forest y se probó a modificar los parámetros allí para la siguiente subida.

Como no se mejoraron los resultados obtenidos por CatBoostClassifier, se volvió a este clasificador y se probaron más combinaciones de parámetros en las dos siguientes subidas, sin lograr mejorar la marca.

Con este algoritmo, aunque no se realizaron subidas ya que bajaba el score incluso en training, se han probado los métodos de smote, oversampling y undersampling, así como normalización minmax y standard usando las herramientas de sklearn.

Como ninguno de los preprocesados probados ha funcionado, probé el LGBMClassifier habiendo modificado los parámetros.

También, pese a que bajaba ligeramente el training probé a subir a drivendata los resultados de aplicar oversampling y undersampling por si reducían overfitting, pero no fue así.

Se probó entonces a cambiar el método de imputación de valores a la media en las variables ordinales, pero no mejoró, así como se probaron también de sklearn el KNNImputer y el IterativeImputer con distintos métodos predictivos y parámetros cada uno, pero de nuevo bajaron el score en training. Asimismo se volvió a incluir la normalización minmax y standard, pero tampoco mejoraban el resultado. Así, ninguna submission de drivendata cuenta con estos métodos, a pesar de que se ha trabajado en ellos.

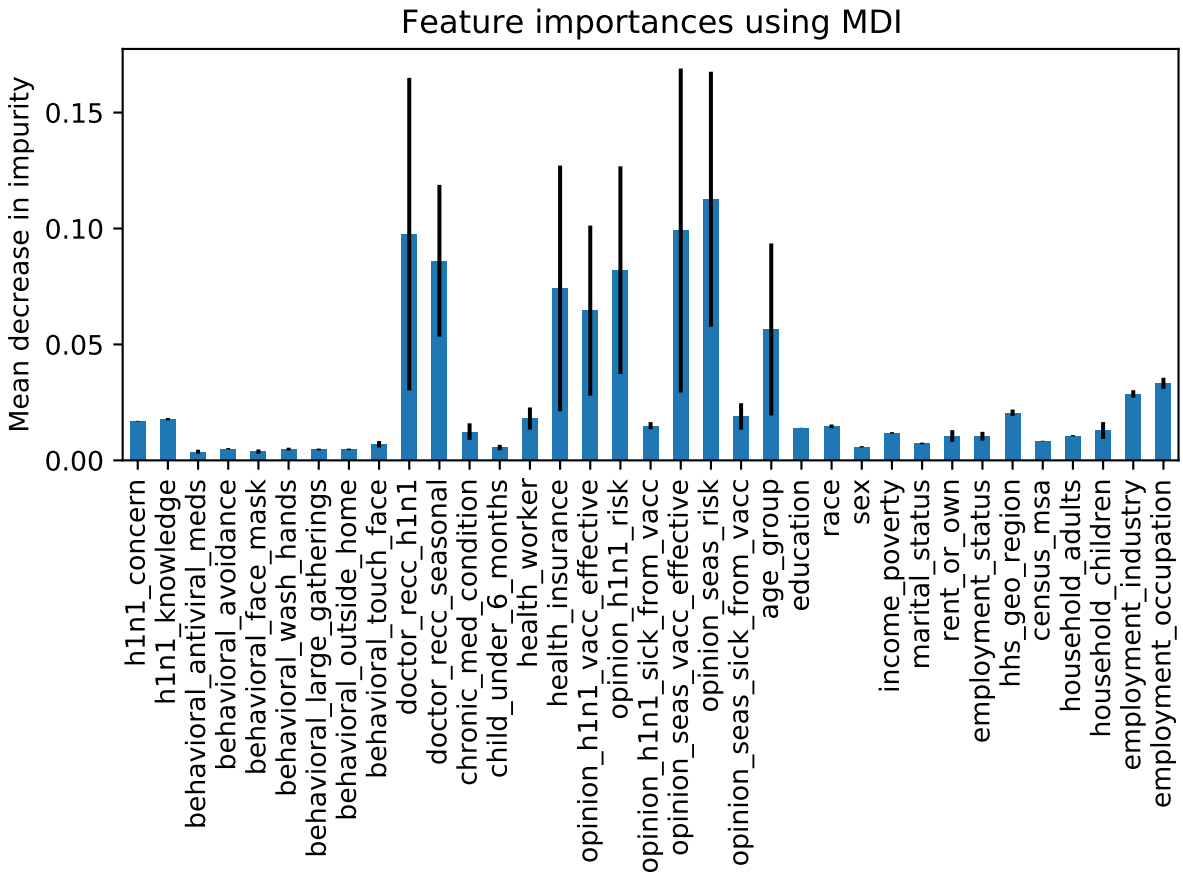


Figura 10: Importancia de los atributos según random forest

Asimismo, se probó a modificar la importancia de las variables basándonos en la relevancia de cada una de ellas, obtenidas a partir del clasificador RandomForest, como se muestra en (10).

Por tanto, las dos siguientes entregas de nuevo fueron modificar los parámetros de LGBMClassifier, obteniendo así 0.8630, que es la puntuación más alta que se ha alcanzado.

Finalmente se volvió a intentar con distintos preprocesados, donde destaca la imputación de valores a la que en la tabla nos referimos como "**imputación de valores inteligente**", pues se basa en la exploración de datos realizada antes.

En este preprocesado las personas que sean sanitarios aparecerán como empleados (employment_status) y los códigos de industry y occupation son los que se dan con más frecuencia en los sanitarios, vistos antes.

Además, si alguien está desempleado, como valor en su industry y occupation aparecerá "no trabaja"si los campos no están rellenos, variable global de sustitución.

De igual manera, si su estado es "not in labor force"se rellena con "no procede".

Los valores perdidos restantes se rellenan con el valor más frecuente distinto de estás variables globales.

El código es el siguiente:

```
no_empleado = "no empleado"
no_procede = "no procede"
empleado_healthcare_industry = "fcxhlnwr"
empleado_healthcare_occupation = "cmhcxjea"

mask_no_empleado_health = (data_x['health_worker'] == 1) &
(data_x['employment_status'] == "Unemployed") & (data_x['employment_industry'].isna())
data_x_tmp = data_x

mask = (data_x["health_worker"] == 1) &  data_x["employment_status"].isna()
data_x_tmp.loc[mask, "employment_status"] = "Employed"

mask = (data_x_tst["health_worker"] == 1) &  data_x_tst["employment_status"].isna()
data_x_tst.loc[mask, "employment_status"] = "Employed"

mask =  ~data_x_tmp["employment_status"].isna()

data_y = data_y.mask(mask)
data_x_tmp = data_x_tmp.mask(mask)
data_x_tmp.loc[mask, "employment_status"] = "unknown"

mask = data_x_tst["employment_status"].isna()
data_x_tst.drop(data_x_tst[mask].index, inplace=True)
data_x_tst.loc[mask, "employment_status"] = "unknown"

for hwdata, col in zip([empleado_healthcare_industry, empleado_healthcare_occupation],
["employment_industry", "employment_occupation"]):

    mask = (data_x["health_worker"] == 1) &
    (data_x["employment_status"] == "Employed") & (data_x[col].isna())
    data_x_tmp.loc[mask, col] = hwdata

    mask = (data_x_tst["health_worker"] == 1) &
    (data_x_tst["employment_status"] == "Employed") & (data_x_tst[col].isna())
    data_x_tst.loc[mask, col] = hwdata

    mask = (data_x['employment_status'] == "Unemployed") &
    (data_x[col].isna())
    data_x_tmp.loc[mask, col] = no_empleado
```

```
mask = (data_x_tst['employment_status'] == "Unemployed") &
(data_x_tst[col].isna())
data_x_tst.loc[mask, col] = no_empleado

mask = (data_x['employment_status'] == "Not in Labor Force") &
(data_x[col].isna())
data_x_tmp.loc[mask, col] = no_procede

mask = (data_x_tst['employment_status'] == "Not in Labor Force") &
(data_x_tst[col].isna())
data_x_tst.loc[mask, col] = no_procede

mask = data_x_tmp[col].isna()
data_x_tmp.loc[mask, col] = "unknown"
mask = data_x_tst[col].isna()
data_x_tst.loc[mask, col] = "unknown"'''

most_frequent = data_x_tmp[col].value_counts().index
selected = 0

while most_frequent[selected] == no_procede or
 most_frequent[selected] == no_empleado or most_frequent[selected] == hwdata:
    selected += 1

data_x_tmp[col].fillna(most_frequent[selected], inplace=True)
data_x_tst[col].fillna(most_frequent[selected], inplace=True)
```

## 1.5 HERRAMIENTAS CON LAS QUE SE HA TRABAJADO

- RandomForest classifier
- CatBoostClassifier
- LGBMClassifier
- sklearn.preprocessing SimpleImputer
- sklearn.preprocessing KNNImputer
- sklearn.preprocessing IterativeImputer
- sklearn.preprocessing MinMaxScaler
- sklearn.preprocessing StandardScaler
- sklearn.preprocessing OneHotEncoder
- sklearn.utils Resample
- imblearn smote
- sklearn.preprocessing LabelEncoder
- sklearn.model_selection Kfold
- sklearn.multioutput MultiOutputClassifier

# 2

## BIBLIOGRAFÍA


- Diapositivas de la asignatura.
- https://pandas.pydata.org/docs/
- https://matplotlib.org/
- https://scikit-learn.org/stable/
- https://catboost.ai/en/docs/concepts/python-reference_catboostclassifier
- https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html