Organización de datos

1er cuatrimestre 2018

TP N 2

https://github.com/ruloweb/7506-organizacion-de-datos/tree/master/TP2

Equipo Excluded Outliers (Grupo 17)	
Sebastián Leonardi	sebastian.leonardi@gmail.com
Fernando Pazos	quini.coppe@gmail.com
José Sánchez	Josegabriel.st@gmail.com
Romina Zatyracz	romi_z@yahoo.com

Tabla de contenido

Introducción

El presente trabajo tiene como objetivo analizar el desarrollo y los resultados obtenidos de la participación en la competencia “Predicción de Postulaciones a Avisos Laborales” ().

La modalidad de trabajo fue compartir las mejoras entre los diferentes participantes para poder sumarlas a cada trabajo, debatiendo las ventajas y desventajas que cada uno apreciaba. No todos los algoritmos performaron según lo esperado por ende solo se publicaron los resultados con mayor performance.

Se probaron varios algoritmos como regresión logistica, random forest, perceptrón, árbol de decisión. Nos hubiese gustado aplicar un bag of words para los feautres titulo y descripción pero por una cuestión de tiempo, no nos fue posible.

En cuanto a la preparación de los datos, se tuvieron en cuenta supuesto que son explicados en el desarrollo del TP, como también decisiones para eliminar, modificar, mantener y agregar datos/features en base a determinados supuestos y/o criterios.

Se presentan tres modelos, de los cuales uno no fue posible obtener resultados, pero presenta una nueva manera de organizar los datos.

Análisis de Datasets

Los datasets contienen información de postulantes, avisos, visitas a avisos y postulaciones a avisos, en diferentes periodos en el tiempo:

- **datos_navent_fiuba/**
  - fiuba_1_postulantes_educacion.csv
  - fiuba_2_postulantes_genero_y_edad.csv
  - fiuba_3_vistas.csv
  - fiuba_4_postulaciones.csv
  - fiuba_5_avisos_online.csv
  - fiuba_6_avisos_detalle.csv
  - fiuba_6_avisos_detalle_missing_nivel_laboral.csv
- **fiuba_hasta_15_abril/**
  - fiuba_1_postulantes_educacion.csv
  - fiuba_2_postulantes_genero_y_edad.csv
  - fiuba_3_vistas.csv
  - fiuba_4_postulaciones.csv
  - fiuba_5_avisos_online.csv
  - fiuba_6_avisos_detalle.csv
- **fiuba_desde_15_abril/**
  - fiuba_1_postulantes_educacion.csv
  - fiuba_2_postulantes_genero_y_edad.csv
  - fiuba_3_vistas.csv
  - fiuba_6_avisos_detalle.csv

Ademas, se cuenta con el dataset al cual se deberá aplicar el modelo predictivo, y contiene las columnas: ID de registro, ID de aviso e ID de postulante:

- test_final_100k.csv

El modelo tiene que ser capaz de predecir la probabilidad de que la persona se postule a dicho aviso.

Procesamiento de Datasets

Para el Modelo 1 fueron utilizados los datos de **fiuba_hasta_15_abril** y **fiuba_desde_15_abril**. Ademas seria posible integrar los datos de **datos_navent_fiuba**.

En primer instancia se concatenaron los datasets:

- fiuba_1_postulantes_educacion.csv
- fiuba_2_postulantes_genero_y_edad.csv
- fiuba_6_avisos_detalle.csv

Y en el caso de duplicados se elimina el registro mas antiguo (correspondiente a los datasets en **fiuba_hasta_15_abril**). Se tomo esta decision bajo la observación de que algunos postulantes figuran en ambos periodos, pero tienen mas información registrada (edad, nivel educativo) en el periodo mas reciente.

Del los datasets fiuba_1_postulantes_educacion.csv y fiuba_2_postulantes_genero_y_edad.csv se calculo una variable ordinal del nivel educativo, ordenada en base a la “importancia” del mismo. Dicho valor termino no utilizándose en el modelo, ya que reduce el nivel predictivo, probablemente por el sesgo de importancia utilizado. También se derivo la edad del participante y la variable sexo se transformo a variable binaria, donde 1 indica masculino.

Luego se realizo un **outter merge** sobre la variable **idpostulante**, y el resultado se almaceno en el archivo **postulantes.pkl**.

Del dataset de aviso, solo se verifico que no existan duplicados y se eliminan las variables con muchos valores nulos (idpais, ciudad, mapacalle). El resultado se almacena en el archivo **avisos.pkl**.

En los datasets de visitas puede existir mas de un registro por postulante y aviso, por lo que se agrupo por idaviso e idpostulante y se agrego la variable **visita_cantidad**, la cual refleja la cantidad de veces que el postulante visito dicho aviso. Los dataframes resultantes se almacenaron en los archivos **visitas_train.pkl** (fiuba_hasta_15_abril) y **visitas_test.pkl** (fiuba_desde_15_abril).

Por ultimo, solo se cuenta con el dataset de postulaciones para el primer periodo (fiuba_hasta_15_abril), ya que el siguiente periodo (fiuba_desde_15_abril), corresponde a los registros que se quieren predecir.

Para este dataset también puede existir uno o más registros por aviso y postulante, pero en este caso solo nos interesa si hubo al menos una postulación, por lo que se eliminaron los duplicados.

Luego se realizó un merge de postulaciones y avisos, y notamos que existen postulaciones que no tienen visitas, lo cual no debería ser posible. Asumimos que en el primer periodo (fiuba_hasta_15_abril) el dataset de visitas está incompleto y suponemos que en el segundo periodo (fiuba_desde_15_abril) las visitas estarán en mejores condiciones. Por lo tanto ejecutamos un **inner merge**, el cual deja afuera a las postulaciones sin visitas.

Otra forma seria asignar una visita a las postulaciones que no tienen. El usuario tiene que generar al menos una visita antes de poder postularse. Dicha propuesta se probó obteniendo un resultado inferior, por lo que se dejó a un lado. Además, generaba un dataset de entrenamiento de mas de 10 millones de registros, lo cual entorpecía el proceso ya que requería de muchos recursos.

Hasta ahora, en el dataset de postulaciones, todos los registros corresponden a casos positivos, o sea postulaciones que si se realizaron, por lo que se agregó la variable **target** con valor 1 en todos los casos.

Por último, se generaron aproximadamente la misma cantidad de casos negativos (desarrollado en el apartado siguiente), se mezclaron con los casos positivos y el resultado se almaceno en el archivo **postulaciones_visitas_train.pkl.**

Dataset de Entrenamiento

El set de entrenamiento es el resultado de un **merge** entre los archivos postulaciones_visitas_train.pkl, postulantes.pkl y avisos.pkl, utilizando las variables **idaviso** e **idpostulante**.

Para que el modelo sea capaz de aprender, necesitamos casos positivos y casos negativos (para un aviso y un postulante dado, existe o no existe postulación).

Ideamos dos maneras de obtener casos negativos:

En vez de un inner merge entre postulaciones y visitas, hacer **left merge** entre las visitas y las postulaciones, de esta forma tendremos visitas con postulaciones y sin postulaciones, casos positivos y negativos respectivamente.

Como el dataset de visitas también tiene las variables idaviso e idpostulante, sigue siendo posible realizar merge con los datasets de avisos y postulantes.

Esta solución se buscó implementar en el Modelo 3, pero no fue posible por cuestiones de tiempo.

El segundo método para obtener casos negativos es el utilizado en el Modelo 1 y Modelo 2 y está basado en dos grandes **supuestos**.

Generamos al azar combinaciones de **idaviso** e **idpostulante**, la misma cantidad que el set de casos positivos. Luego descartamos las combinaciones que existen en el dataset de casos positivos, correspondiente al primer periodo (de existir serán muy pocas, ya que la probabilidad de un caso positivo en el conjunto de todas las combinaciones posibles es muy bajo).

Las combinaciones resultantes son el conjunto de:

1. Postulaciones que se realizaron antes del primer periodo.
2. Postulaciones que si se harán en el futuro.
3. Postulaciones que nunca se hicieron y nunca se harán.

Este modelo **SUPONE** que de existir postulaciones pasadas y futuras (casos 1 y 2) en las combinaciones generadas, estas son pocas (investigar si es posible demostrar), por lo tanto representan en su mayoría a avisos y postulantes con poca probabilidad de postulación.

Luego, realizamos un **left merge** entre el dataset de casos negativos y las visitas del primer periodo (fiuba_hasta_15_abril). Esto nos permite tener

registros de casos negativos que si poseen visitas. Para los registros donde las visitas son nulas (debido al **left merge**) les asignamos visitas 0.

En este último punto realizamos otro **SUPUESTO**: las mayoría de las postulaciones en el caso 3 no poseen visita, dicho de otro modo, la probabilidad de que un usuario se postule a un aviso que visitó es alta. Por lo tanto, los registros donde idaviso e idpostulante no corresponden a una postulación pero si tienen visita son pocos, asignarles vista 0 es un error y hará que el modelo aprenda mal, pero al ser la minoría de casos el impacto es despreciable y terminaremos con un modelo funcional.

Estos supuestos se pondrán a prueba (grosso modo) al ejecutar el modelo. De ser válidos obtendremos un score operativo.

Por último, al dataset de casos negativos se le agrego la variable **target** en 0 y se mezclo con el dataset de casos positivos.

Algunas consideraciones:

- El dataset resultante tiene una o más visitas para todos los casos positivos y 0 visitas para la mayoría de los casos negativos, por lo que existe una correlatividad extremadamente alta entre visita y **target**. Esto nos traerá problemas a la hora de entrenar, ya que obtendremos siempre scores muy elevados y será difícil identificar los cambios que aumentan el performance del modelo.
- Este modelo depende fuertemente de las visitas, por lo que no puede aplicarse en casos en donde dicha variable no exista (imaginamos que la mayoría de los casos), un ejemplo seria mostrar publicidad de avisos que el usuario aun no vio y que tienen mucha probabilidad de convertir una postulación.

Modelo 1

Para este modelo, las únicas variables numéricas son edad y visita_cantidad, las demás son categóricas y fueron procesadas con **OneHotEncoder**.

Como se predijo en los apartados anteriores, existe una fuerte correlación entre visita_cantidad y el target, por lo que el accuracy de train es muy alto (0.99). Esto nos impide comparar diferentes algoritmos y features en la etapa de training, ya que todos los modelos devuelven un resultado elevado.

Falta realizar feature engineer.

El mejor score obtenido con este modelo en el Leaderboard Publico fue de **0.924** con un **LogisticRegression**.

Como el fuerte se encuentra en la variable visita_cantidad, un modelo lineal simple puede aprender fácilmente la alta correlación entre esa variable y el target.

El score elevado nos invita a pensar que los supuestos descriptos anteriormente pueden ser verdaderos.

Modelo 2

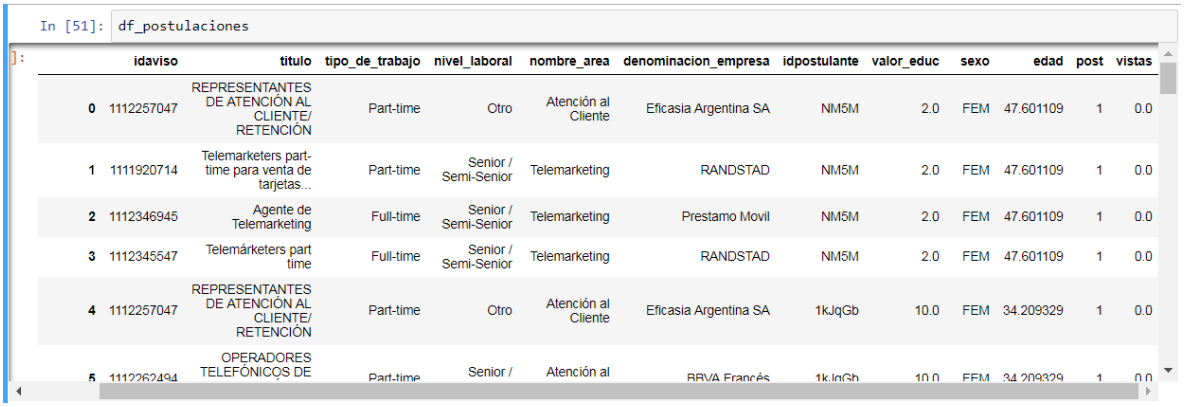
En este modelo se generó un notebook destinado a producir un archivo con los datos del set de entrenamiento. En este archivo figuran las postulaciones hasta el 15 de abril y está conformado por un frame con 12 columnas: idaviso, título del trabajo, nivel laboral, tipo de trabajo, nombre del área de trabajo, denominación de la empresa, id postulante, educación, sexo, edad y número de vistas del aviso. La educación se cuantificó entre 0 y 20 según el nombre y estado del curso; además, como muchos postulantes ingresaron más de un nivel de educación, se eligió el de más alta puntación. Este frame se generó a partir de los frames de postulaciones concatenados y haciendo operaciones de merge con todos los frames de los archivos provistos.

Hasta este punto, los resultados del frame de postulaciones generado son:

```
postulaciones válidas 6084165
avisos con detalles 25288
número de postulantes 504407
```

Además, para entrenar el modelo, es necesario que existan registros de postulaciones “no hechas”. Para tal se generaron aleatoriamente 3000000 de combinaciones de idaviso e idpostulante y se generó un frame con las mismas columnas del frame anterior. Se adicionó al frame anterior eliminando las líneas duplicadas y agregándose una décimo segunda columna **post** que indica si la postulación se realizó (es decir que estaba en el frame original de postulaciones) o no (es decir que fue una de las generadas aleatoriamente). Para estas postulaciones “no realizadas” también se agregan en la columna correspondiente el número de vistas, dato extraído de los frames de vistas como hecho con las postulaciones efectivamente realizadas.

El frame final del set de entrenamiento tiene, así, 9082431 líneas y 12 columnas. Se almacena en un archivo 'training.csv'. La siguiente pantalla muestra un detalle de este frame.



```
In [51]: df_postulaciones
```

	idaviso	titulo	tipo_de_trabajo	nivel_laboral	nombre_area	denominacion_empresa	idpostulante	valor_educ	sexo	edad	post	vistas
0	1112257047	REPRESENTANTES DE ATENCIÓN AL CLIENTE/ RETENCIÓN	Part-time	Otro	Atención al Cliente	Eficasia Argentina SA	NM5M	2.0	FEM	47.601109	1	0.0
1	1111920714	Telemarketers part-time para venta de tarjetas...	Part-time	Senior / Semi-Senior	Telemarketing	RANDSTAD	NM5M	2.0	FEM	47.601109	1	0.0
2	1112346945	Agente de Telemarketing	Full-time	Senior / Semi-Senior	Telemarketing	Prestamo Movil	NM5M	2.0	FEM	47.601109	1	0.0
3	1112345547	Telemárketers part time	Full-time	Senior / Semi-Senior	Telemarketing	RANDSTAD	NM5M	2.0	FEM	47.601109	1	0.0
4	1112257047	REPRESENTANTES DE ATENCIÓN AL CLIENTE/ RETENCIÓN	Part-time	Otro	Atención al Cliente	Eficasia Argentina SA	1kJqGb	10.0	FEM	34.209329	1	0.0
5	1112262494	OPERADORES TELEFÓNICOS DE	Part-time	Senior /	Atención al	BBVA Francés	1kJqGb	10.0	FEM	34.209329	1	0.0

Seguidamente, en este mismo notebook, se trabajó el archivo 'test_final_100k.csv' para el armado del set de prueba. También haciendo operaciones de merge con los frames anteriores con las vistas a partir del 15 de abril se completa un frame similar al anterior pero evidentemente sin la columna **post**, pues es exactamente el resultado de esta columna lo que el algoritmo de machine learning debe predecir. Este frame, con 100000 registros de 11 columnas o features, se almacena en un archivo 'testing.csv'.

El segundo notebook está destinado a ejecutar el algoritmo de machine learning con el set de entrenamiento para después probarlo con el set de prueba. Lo primero que hacemos es tomar una muestra de 4000000 de registros del set de entrenamiento con el cual se trabajará. Las columnas de título del trabajo y denominación de la empresa no serán utilizadas en este modelo. Las variables categóricas del nivel laboral son: Senior/Semi-Senior, Junior, Otro, Jefe/Supervisor/Responsable, Gerencia/Alta Gerencia/Dirección; las de tipo de trabajo son: Full-time, Part-time, Pasantia, Por Horas, Temporario, Por Contrato, Teletrabajo, Fines de Semana, Primer empleo, Voluntario; hay 188 áreas laborales diferentes, y evidentemente dos sexos: MASC y FEM.

Seguidamente, se cuantificarán las variables categóricas. Para esto primero se confeccionan matrices de VDM (value distance measure) calculando las probabilidades de cada categoría en cada clase. En nuestro caso, las clases son determinadas por la columna **post** , es decir que existen las clases post=0 (no postuló) y post=1 (postuló). El siguiente frame muestra un ejemplo de las probabilidades calculadas para cada nivel laboral en cada clase:

	post=0	post=1
Senior / Semi-Senior	0.671958	0.637698
Jefe / Supervisor / Responsable	0.060096	0.030234
Junior	0.164106	0.220783
Otro	0.077758	0.081141
Gerencia / Alta Gerencia / Dirección	0.012833	0.006330

Seguidamente se calculan las distancias entre cada categoría como la suma para cada clase del valor absoluto de la diferencia de probabilidades. Esta matriz de distancias evidentemente tiene la diagonal principal igual a cero (porque la distancia entre una categoría y ella misma es 0), y es simétrica porque la distancia entre la categoría A y la B es la misma que entre B y A. Finalmente, a partir de esta matriz de distancias entre las diferentes categorías de cada feature, se calculas las coordenadas de cada categoría através de **multidimensional scaling.** Para esto se calculan los autovalores y autovectores de la matriz de distancias al cuadrado centralizada (promedio de sus filas y columnas igual a cero). Las coordenadas de las categorías serás dadas por los primeros q autovectores multiplicados por la raíz cuadrada de los

q autovalores diferentes de cero. En nuestro caso, en todos los features categóricos (nombre_área, tipo_de_trabajo_nivel_laboral y sexo) las coordenadas siempre fueron unidimensionales (un solo autovalor diferente de cero, esto es q=1, calculado con una precisión del 95%). Se substituyen las filas categóricas por sus coordenadas y se substituyen los valores faltantes (Nan) por valores adecuados (p.ej. la edad promedio en caso de edad no declarada).

Se ejecutaron en este modelo algoritmos **perceptron** tanto el provisto en **sklearn.linear_model**
como otro programado por nosotros. Para ejecutar este algoritmo dividimos el set de
entrenamiento en una matriz y un vector de test con 3000000 de registros, para después verificar el hiperplano separador generado con el 1000000 de registros restantes. El resultado fue de
473627 vectores del lado errado del hiperplano, cuyas coordenadas son `w= [-4.15361701, -0.69242865, 105.14197102, 1.25490292, 102.72118984, -1.3740235, 234.3842892 ].`

El frame del set de prueba pasó por el mismo tratamiento que el set de entrenamiento, y
multiplicando la matriz generada por este frame por el vector del hiperplano nos da `35258`
registros del lado de las postulaciones (post=1). Sometido el archivo generado a la plataforma
Kaggle, el resultado fué un score de 0,53 aproximadamente.

Se ejecutó también con la totalidad set de entrenamiento un árbol de decisión obtenido mediante
from sklearn import tree. La predicción hecha con el set de prueba arrojó un resultado de 50220
postulaciones. El archivo obtenido con estas postulaciones se subió a la plataforma Kaggle dando
un resultado aproximadamente igual a 0,68.

Finalmente, se ejecutó con la totalidad del set de entrenamiento un algoritmo de random forest
obtenido de **sklearn.ensemble import RandomForestClassifier**.
La predicción hecha con el set de prueba nos dió una cantidad de `62506` postulaciones sobre un
total de 100000 registros. Subido el archivo con estas predicciones a la plataforma Kaggle, nos dió un score de 0,75 aproximadamente.
En todos los casos, los resultados obtenidos con este modelo fueron inferiores a los obtenidos en el modelo 1, nuestro mejor score.

Modelo 3

En la búsqueda de probar otras opciones de estructuración de los datos para usarlos como dataset inicial, se pensó en integrar la información dispersa en varios dataframes para a formar uno solo, pero tener como punto de partida las vistas.

En el otro approach, se parte de generar los casos negativos aleatoriamente (en los cuales un postulante no concreta la postulación), aquí esa información está presente en las vistas de los avisos.

Si tomamos todo el universo de avisos y todo el universo de postulantes, y hacemos el producto cartesiano de todos, obtendremos todas las posibles combinaciones de avisos-postulantes. En la realidad eso no se da, y generarlo no aporta más información de aprendizaje al algoritmo, porque hay claramente situaciones que no se van a dar.

De ese conjunto total de combinaciones, está el subconjunto de postulantes que vieron un aviso, lo analizaron y decidieron postularse (o no). Esas combinaciones las tenemos en los dataframes por separado. Por un lado las visitas al aviso y por otro lado, los postulantes.

Como en el dataframe de las visitas están listadas todas, incluso repetidas si un mismo postulante visito varias veces el aviso, se pueden agrupar por aviso/postulante y contar los timestamp.

Así se obtiene la valiosa información de la cantidad de visitas por un postulante a un mismo aviso. Y si hacemos el merge con las postulaciones, vamos a poder relacionar la cantidad de vistas, con la información posterior si efectivamente se postuló (o no lo hizo, lo cual es valioso también como dato). Esto aporta más información basada en el comportamiento de un potencial postulante, y ayudando a la predicción del mismo, ya que los casos aleatorios no lo hacen.

Dataframe de postulantes

	idpostulante	educacion	sexo	edad
4030	zvPXaPJ	2	False	23.0
71952	mzb3rQ6	2	False	26.0
140971	ak50Vr9	6	True	36.0
412205	3NL4zDJ	11	False	28.0
76930	lDXQdr3	11	False	23.0
415020	2zmj2G0	11	False	29.0
75607	lDw4a5G	10	False	23.0
44285	qevAwaN	5	True	33.0
361247	8A1p1j	10	False	28.0
5223	zv8WoBk	10	True	25.0

Dataframe de avisos

	idaviso	titulo	descripcion	nombre_zona	tipo_de_trabajo	nivel_laboral	nombre_area	denominacion_empresa
3252	1112225463	Outsourcing Manager	For international TELCO company we are looki...	Gran Buenos Aires	Full-time	Senior / Semi-Senior	Telecomunicaciones	CERUTTI & ASOCIADOS CONSULTING
4073	1112292689	Responsable Coordinación Logística	Para Importante Operadora Multinacional busca...	Gran Buenos Aires	Full-time	Jefe / Supervisor / Responsable	Logística	TALENT RECRUITERS
2031	1112345187	Jefe de Repuestos-Concesionaria	Nuestro c liente, importante concesionaria ...	Gran Buenos Aires	Full-time	Senior / Semi-Senior	Ventas	RANDSTAD
4269	1112296360	Ingeniero Eléctrico	Importante Compañía Multinacional Incorporar...	Gran Buenos Aires	Full-time	Senior / Semi-Senior	Mantenimiento	ARRIVE RRHH
9832	1112496759	05 May - C# (.NET) Chief Software Architect (...	Al aplicar a la vacante será redirigido a la ...	Gran Buenos Aires	Teletrabajo	Senior / Semi-Senior	Tecnologia / Sistemas	CrossOver
10	1001724563	Paramédico	Nos encontramos en un proceso de selección pa...	Gran Buenos Aires	Full-time	Senior / Semi-Senior	Salud	Hochschild Mining
6426	1112467397	Director Creativo	CARESTINO INCORPORA: DIRECTOR CREATIVO (ZO...	Gran Buenos Aires	Full-time	Jefe / Supervisor / Responsable	Diseño Multimedia	BIAMER SRL
13813	1112392476	CAJERO/A JR. SUC. SANTA FE	Orientamos nuestra búsqueda a estudiantes ava...	Gran Buenos Aires	Full-time	Junior	Tesorería	BBVA Francés
11398	1112203334	importante restaurant seleccionará camareros	Se requiere - Muy buena presencia - Muy bue...	Gran Buenos Aires	Full-time	Senior / Semi-Senior	Gastronomia	PIOLA
3300	1112226448	Analista de Planeamiento de Inventario (Zona ...	En KaizenRH buscamos URGENTE Analista de Pla...	Gran Buenos Aires	Full-time	Senior / Semi-Senior	Abastecimiento	Kaizen Recursos Humanos

Dataframe de vistas agrupada por cantidad idaviso/idpostulante

	idpostulante	idaviso	vistas
1329761	JBe65PJ	1112416123	2
251306	2zm1e1V	1112411350	1
1379987	JBx11VO	1112457520	2
550520	6LdPkr	1112454807	2
2355348	ZZzJKZ	1112167719	1
953548	A3X0MeG	1112334791	1
2780041	eko9VBB	1112426585	1
1845588	QNLwRoN	1112320618	1
307360	3NdRNqa	1112390398	7
2142435	YBMKZX	1112441384	1
1298837	GNdwwqN	1112036398	1
2128475	Y5851V	1112432399	2
3240291	rm2A3Ad	1112360052	2
2639718	dYJeMWb	1112439298	1
1402707	KBaM66j	1112440594	1

Así en el mismo dataframe tenemos un conjunto de registros que reflejan los ID de postulante y de aviso, la cantidad de visitas del mismo y si efectivamente se postuló.

Dataframe con vistas, postulantes y avisos unificado

	idpostulante	idaviso	vistas	postulo	educacion	sexo	edad
1939443	RzrY83E	1112390566	1	1.0	11.0	False	34.00
1173957	EzEVeGo	1112369030	1	0.0	8.0	True	29.00
2512831	akjzXv0	1112413716	1	0.0	8.0	True	24.00
3157659	pzdr54Z	1112410056	2	1.0	11.0	True	23.00
3348863	vVjbEMe	1112448287	1	0.0	10.0	True	29.37
715813	8MB34Jl	1112424076	2	1.0	2.0	True	26.00
617918	6rP3PGx	1112424876	1	0.0	10.0	False	19.00
583183	6r2Bapj	1112366936	1	0.0	10.0	True	27.00
2568894	bOVr3G9	1112334788	1	1.0	2.0	False	23.00
3225757	qew8m2x	1112366747	2	0.0	4.0	True	26.00
1538877	MVd29Qe	1112423698	2	1.0	2.0	True	32.00
3267157	rmdX68x	1112430622	2	0.0	2.0	True	28.00
2769373	ekOwNDB	1112315654	1	0.0	10.0	False	21.00
3301856	vV9BPBx	1112280891	1	1.0	8.0	True	22.00
3022791	mzdmGQ3	1111970596	1	1.0	10.0	False	24.00
3476407	xkdNaXj	1112409128	3	1.0	7.0	False	40.00
1226181	EzepKK6	1112335409	1	1.0	6.0	False	25.00
21088	0zP2zoY	1112467026	4	1.0	2.0	False	23.00

Una vez armada la base de registros, se procede a agregar la información adicional referida a los avisos y datos de los postulantes.

Los campos a tener en cuenta son:

Dataframe Postulantes

Sexo: se transforma en valores binarios.

Edad: se normaliza a valores entre 0-1

Nivel de estudios: se normaliza a valores entre 0-1

Dataframe Avisos

Nombre Zona: se cargan las zonas laborales transformándose en columnas por one hot encoding

Tipo de Trabajo: ídem Nombre Zona

Nivel laboral: ídem Nombre Zona

Nombre Área: ídem Nombre Zona

Titulo Aviso: Se eliminan las palabras poco significativas: artículos, preposiciones y otras. (Este último punto, no implementado).

Descripción Aviso: se limpian los tags de HTML (implementado). Se eliminan las palabras poco significativas: artículos, preposiciones y otras. (Este último punto, no implementado).

Dataframe final unificado y normalizado

	idpostulante	idaviso	vistas	postulo	educacion	sexo	edad	Buenos Aires (fuera de GBA)	Capital Federal	GBA Oeste	...	Por Horas	Primer empleo	Teletrabajo	Temporario	Voluntario	Gerencia / Alta Gerencia / Dirección	S F
0	0z5Dmrd	1112384041	1	1.0	0.55	True	0.401869	0	0	0	...	0	0	0	0	0	0	
1	0z5Dmrd	1112418961	1	0.0	0.55	True	0.401869	0	0	0	...	0	0	0	0	0	0	
2	0z5Dmrd	1112420060	1	1.0	0.55	True	0.401869	0	0	0	...	0	0	0	0	0	0	
3	0z5Dmrd	1112426253	2	0.0	0.55	True	0.401869	0	0	0	...	0	0	0	0	0	0	
4	0z5JW1r	1112165954	2	0.0	0.45	True	0.355140	0	0	0	...	0	0	0	0	0	0	
5	0z5JW1r	1112394797	1	1.0	0.45	True	0.355140	0	0	0	...	0	0	0	0	0	0	
6	0z5JW1r	1112417455	3	1.0	0.45	True	0.355140	0	0	0	...	0	0	0	0	0	0	
7	0z5JW1r	1112428361	2	1.0	0.45	True	0.355140	0	0	0	...	0	0	0	0	0	1	
8	0z5JW1r	1112450843	2	0.0	0.45	True	0.355140	0	0	0	...	0	0	0	0	0	0	
9	0z5VvGv	1111413600	1	0.0	0.10	True	0.196262	0	1	0	...	0	0	0	0	0	0	