UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

75.06 Organización de Datos

Trabajo Práctico 2
Primer Cuatrimestre de 2018

Grupo 11: Los Maggios

Bravo Arroyo, Víctor Manuel	98882
Calvani, Sergio Alejandro	98588
Pérez Ondarts, Flavio	96786

Link de GitHub: https://github.com/SergioCalvani/TP2-Datos

Índice

1. Introducción

Este informe tiene el objetivo de explicar el criterio que se utilizó para aplicar distintos algoritmos de Machine Learning para predecir si un usuario se postuló en cierto aviso o no. El informe en cuestión se dividirá en tres grandes partes:

- **Procesamiento de Datos:** Primero mostraremos los criterios que hemos utilizado para tomar los datos que vimos como necesarios para el entrenamiento del algoritmo, así mismo como fue el proceso para obtener los datos de la forma que vimos que era necesaria.
- **Algoritmos:** Luego se describirán los algoritmos utilizados, el criterio de porque se utilizó dicho algoritmo y los resultados que nos dieron.
- **Conclusión Final:** Por último, una conclusión final de cuál fue el algoritmo utilizado finalmente, el resultado que nos arrojó y por qué creemos que funcionó de mejor manera.

Así mismo fue de vital importancia utilizar los resultados del análisis realizado en el Trabajo Práctico Nro. 1 ya que podíamos saber qué datos tienen mayor peso que otros, por esto mismo, a lo largo de este informe se recurrirá a los datos obtenidos en dicho trabajo.

2. Procesamiento de Datos

Para comenzar hablaremos del procesamiento que hicimos de los datos, estos datos son acerca de los avisos de búsquedas laborales en la página https://www.zonajobs.com.ar/, estos datos fueron provistos amablemente por la empresa Navent, en archivos de tipo .CSV, los cuales contenían la información de tantos sus usuarios como los de los avisos. Los datos en cuestión tenían la información que utilizamos en el Trabajo Práctico Nro. 1 y se le agregaron los mismos archivos pero después de la fecha del 15 de Abril, por lo que había información un poco más reciente en relación al trabajo anterior. Para explicar detalladamente los criterios que se utilizaron, es explicará cada archivo por separado.

2.1. Género y Edad

Primero hablaremos sobre los datos que utilizamos en relación al género y la edad, para ello lo dividiremos en dos secciones para profundizar su explicación. En este archivo se encuentra la fecha de nacimiento de cada usuario y su género.

2.1.1. Género

Transformamos la columnas del género de los usuarios al tipo booleano, indicando si son o no usuarios masculinos. A los que no declararon sexo se les otorga un sexo random, aunque como se pudo observar en el primer trabajo, es una cantidad ínfima.

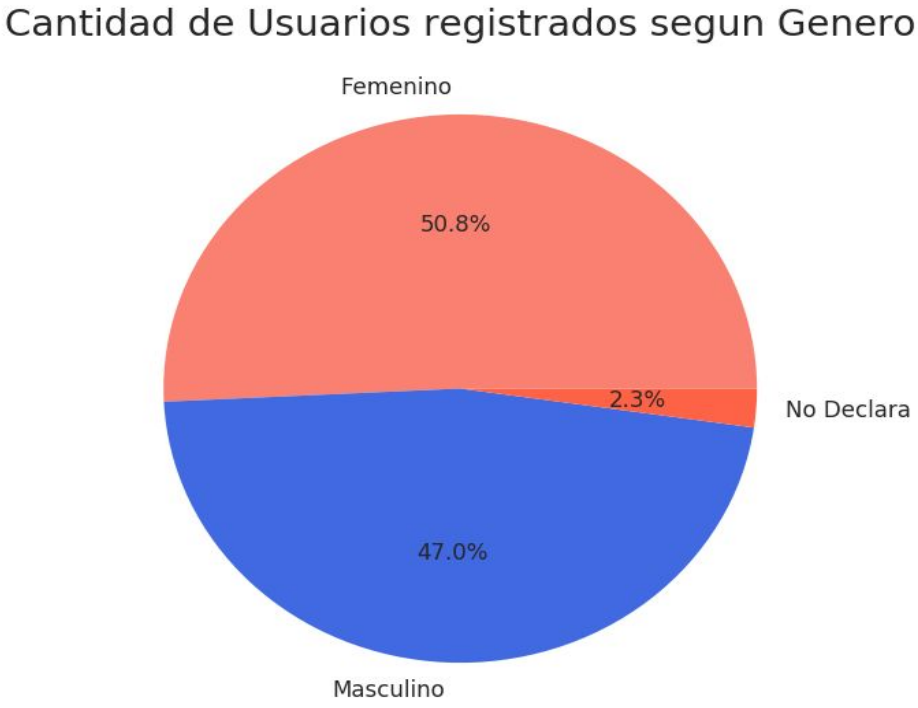

Figura 2.1: Género de los Usuarios Registrados

	idpostulante	sexo_masculino
0	eo2p	True
1	1d2B	True
2	EBO0	False
3	a6MKW	True
4	6MWd4	True

Figura 2.2: Vista del procesamiento por Género

2.1.2. Edad

Con respecto a la Edad, lo que primero se realizó fue filtrar las edades que no tienen coherencia alguna, para ello solamente utilizamos las edades mayores a 18 años y menores a 75. Para los usuarios que no declararon su edad, lo que se hizo fue rellenar con números cuya probabilidad corresponden a una distribución exponencial, manteniendo la media original

	idpostulante	edad	sexo_masculino
0	eo2p	37.0	True
1	1d2B	42.0	True
2	EBO0	44.0	False
3	a6MKW	44.0	True
4	6MWd4	43.0	True

Figura 2.3: Género y Edad ya procesados

2.2. Educación

A continuación vamos a detallar qué datos utilizamos con respecto a la educación, cabe recordar que se encontraban dos atributos, uno era el estado (el cual podía ser Abandonado, En Curso o Graduado) y el otro el nivel (secundario, universitario, terciario, posgrado, master y doctorado). Primero lo que hicimos fue tomar los usuarios cuyo estado sea Graduado, ya que debido al análisis realizado en el trabajo anterior pudimos observar que era este el que mejor caracterizaba su educación, como se puede observar en la siguiente imagen.

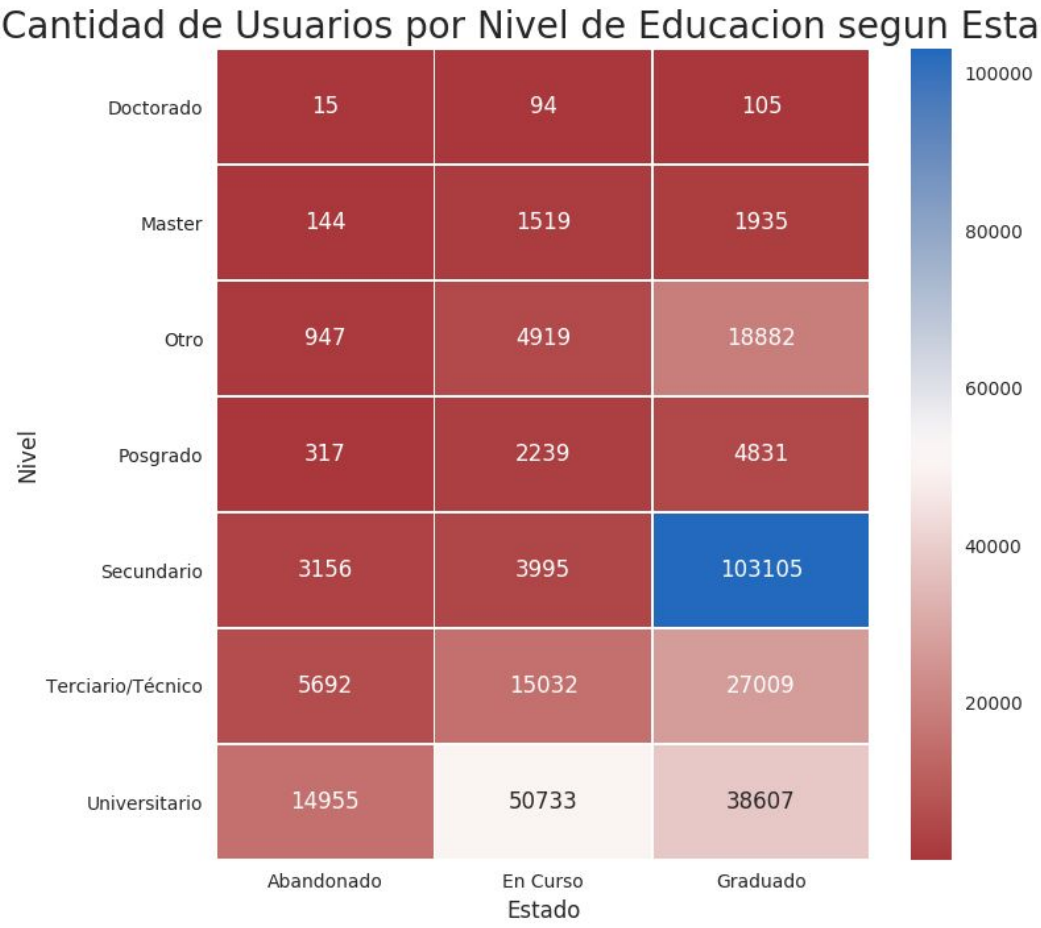

Figura 2.4: Nivel de Educación General

Una vez que nos quedamos con los usuarios graduados, lo que hicimos fue quedarnos solamente con los usuarios cuyo nivel sea Secundario, Universitario o Superior, en este caso se engloba a Master, Posgrado y Doctorado. Una vez realizado esto se generaron distintas columnas en base a los niveles tomados, cuyo valor es un 1 si efectivamente pertenece a ese nivel o un 0 de no serlo.

	idpostulante	titulo_universitario	titulo_secundario	titulo_superior
0	0z5Dmrd	1	0	0
1	0z5JW1r	0	1	0
2	0z5VvGv	0	1	0
3	0zB01pE	0	0	0
4	0zB026d	0	1	0

Figura 2.5: Usuarios con su nivel en base al estado “Graduado”

Una vez realizado esto, lo que se realizó fue crear un nuevo un nuevo DF llamado *'usuarios'*, donde cada fila corresponde a un usuario y posee las columnas que corresponden a los datos sobre su educación, edad y género.

	idpostulante	edad	sexo_masculino	titulo_universitario	titulo_secundario	titulo_superior
0	eo2p	37.0	True	1.0	1.0	1.0
1	1d2B	42.0	True	0.0	0.0	0.0
2	EBO0	44.0	False	1.0	0.0	1.0
3	a6MKW	44.0	True	1.0	0.0	1.0
4	6MWd4	43.0	True	1.0	0.0	0.0

Figura 2.6: Nuevo Dataframe 'Usuarios'

2.3. Avisos

Ahora bien, ya tenemos los datos de los usuarios casi listos pero falta tomar los datos sobre los avisos, para ello se tomó los CSV en los que se encuentra la información detallada de cada aviso en cuestión. Para comenzar, lo que se hizo fue eliminar los avisos que estuvieran duplicados, quedándonos con el último disponible, luego eliminamos las columnas donde casi todos sus datos eran nulos, caso como 'mapacalle' , 'ciudad' e 'idpais'. Ahora nuevamente vamos a dividir esta sección en varias partes para explicar cada parte con mayor nivel de detalle.

2.3.1. Tipos de Trabajo

Como se pudo analizar en el trabajo anterior, pudimos observar que hay una tendencia favorable para los tipos de trabajo Full-Time y Part-Time, mientras que en los otros hay muy poca cantidad de avisos como se puede observar en el gráfico a continuación

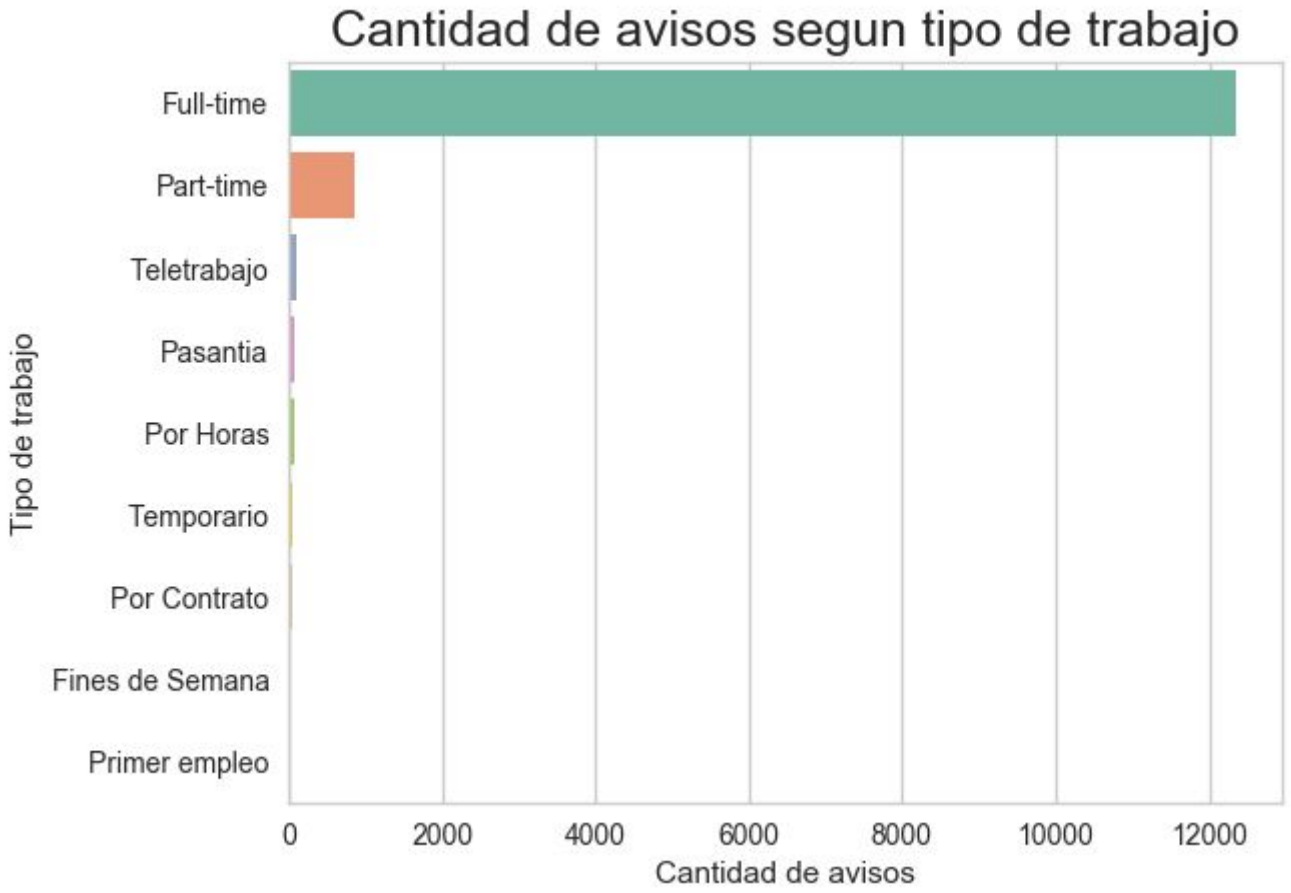

Figura 2.7: Cantidad de Avisos según Tipo de Trabajo

Por consiguiente, solamente se utilizaron los avisos que presentan el tipo de trabajo Full-Time y Part-Time, para ello, hicimos el mismo proceso que en los casos anteriores, creamos dos columnas que nos indiquen si el trabajo es Full-time o si es Part-time.

	idaviso	Full-time	Part-time
4	1000872556	True	False
8	9240880	True	False
19	1110185164	True	False
21	1110513885	True	False
32	1111034024	True	False

Figura 2.8: Columnas creadas según si es Part-Time o Full-Time

2.3.2. Nivel Laboral

Con respecto al nivel laboral, en el análisis posterior se pudo ver que hay una amplia tendencia favorable para el nivel Senior / Semi-Senior, pero como hay solamente 5 niveles se tomaron todas los niveles. Para ello, transformamos la columna de nivel laborar en 5 features que indiquen si el puesto de trabajo corresponde o no a determinado nivel.

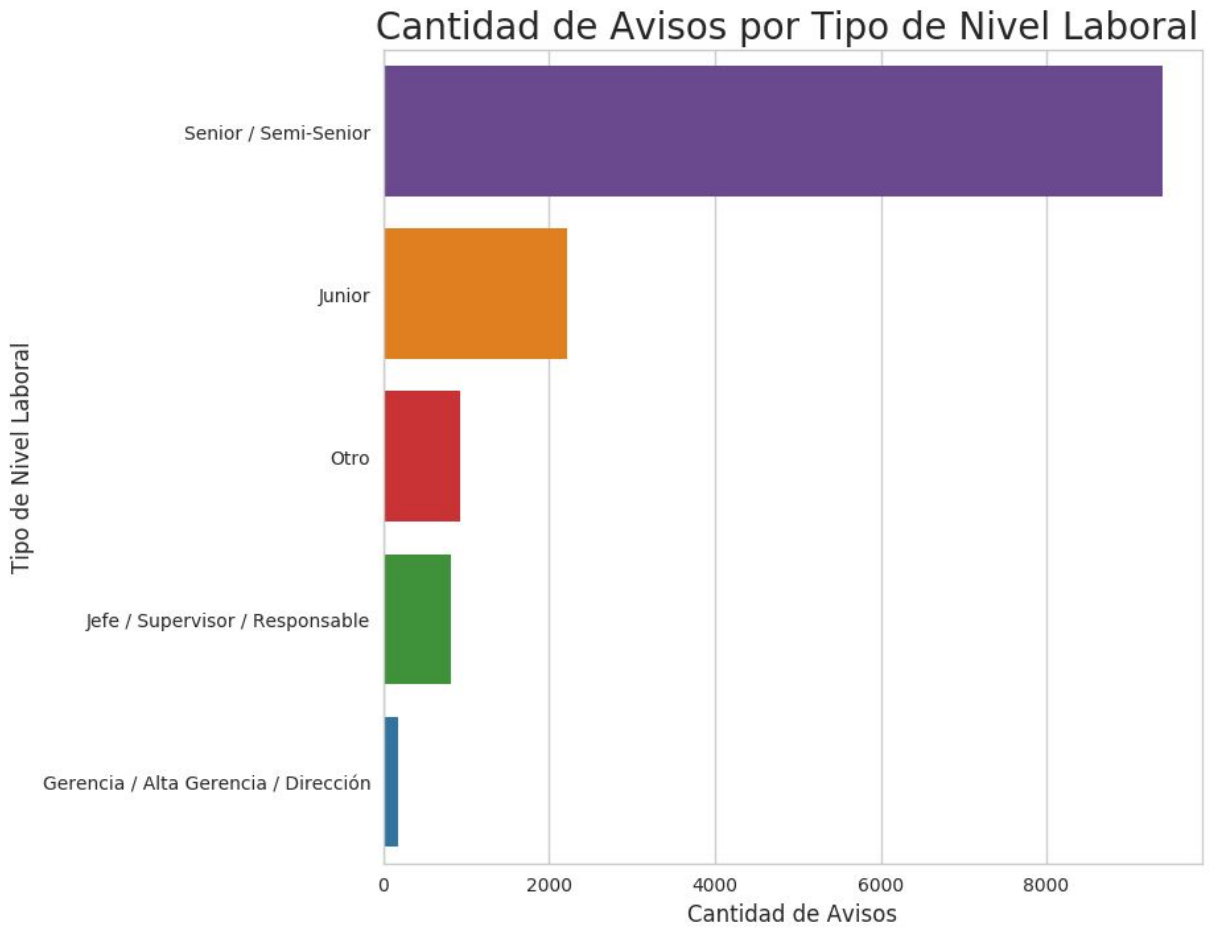

Figura 2.9: Cantidad de Avisos por Nivel Laboral

	idaviso	senior	junior	otro	Jefe	Gerencia
4	1000872556	True	False	False	False	False
8	9240880	False	False	False	True	False
19	1110185164	False	True	False	False	False
21	1110513885	False	True	False	False	False
32	1111034024	True	False	False	False	False

Figura 2.10: Columnas generadas en relación al Nivel Laboral

2.3.3. Zona Laboral

En cuanto a las zonas de trabajo, como se observó en el trabajo anterior, la mayor cantidad son de Gran Buenos Aires y Capital Federal, por lo que generamos una columna booleana que indique si pertenece al Gran Buenos Aires o no.

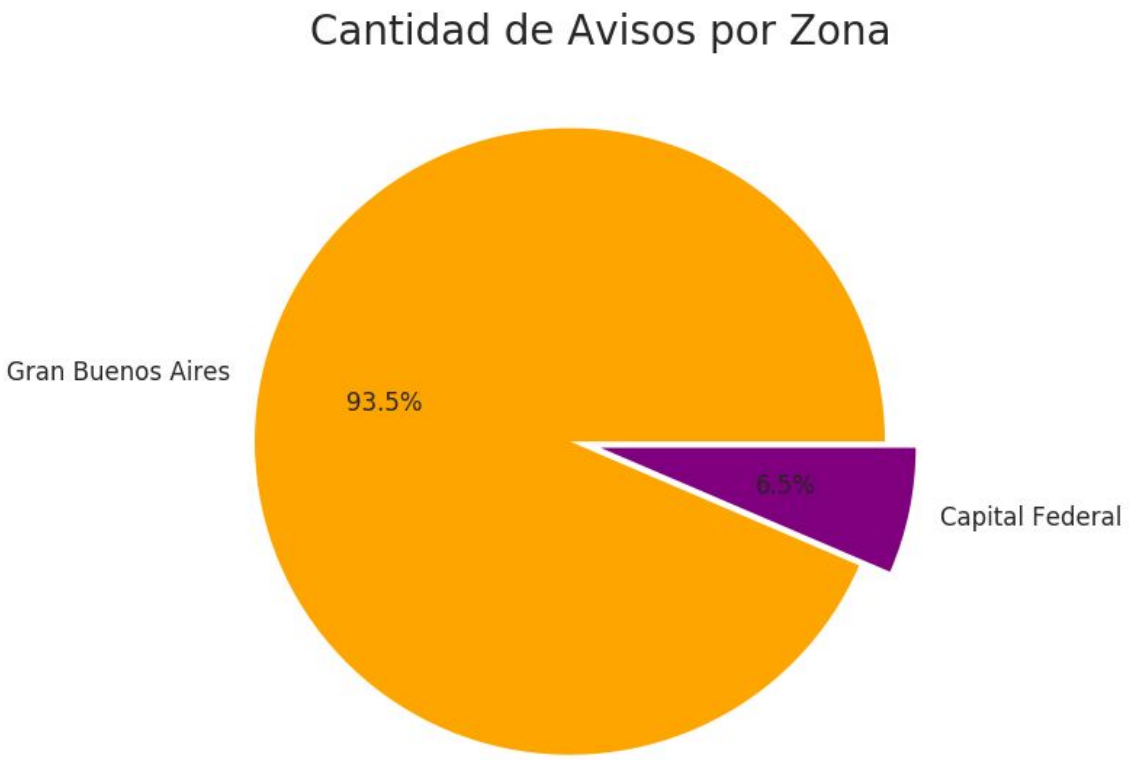

Figura 2.11: Cantidad de Avisos por Zona

	idaviso	GBA
4	1000872556	True
8	9240880	False
19	1110185164	True
21	1110513885	True
32	1111034024	True

Figura 2.12: Columna creada a raíz de la Zona Laboral

2.3.4. Áreas de Trabajo

En lo que respecta en las áreas de trabajo de cada aviso, lo que se hizo fue tomar en cuenta los principales que se encuentren en la mayor cantidad de avisos, caso que como se enunció en el trabajo anterior, también se repiten en cuanto a las áreas con mayor cantidad de postulaciones y vistas.

Figura 2.13: Áreas con mayor cantidad de Avisos

Por lo que se, agregamos features que indiquen si el aviso corresponde a una determinada Área Laboral, en este caso tomamos las primeras 17 áreas.

	idaviso	Ventas	Comercial	Administración	Producción	Programación	Contabilidad	Tecnologia / Sistemas	Atención al Cliente	Mantenimiento
4	1000872556	False	False	False	True	False	False	False	False	False
8	9240880	False	True	False	False	False	False	False	False	False
19	1110185164	False	False	True	False	False	False	False	False	False
21	1110513885	False	False	False	False	False	False	False	False	False
32	1111034024	False	True	False	False	False	False	False	False	False

Figura 2.14: Vista previa de las Columnas generadas a partir de las Áreas

Por último se eliminaron las columnas que ya fueron procesadas o que nos dimos cuenta que no generaban ningún tipo de información relevante, por lo que se procedió a dejar solamente los features que generamos nosotros

2.4. Vistas

Para la parte de las vistas mucho procesamiento no fue necesario, lo único que pudimos obtener fue la cantidad de vistas que tuvo un postulante a determinado aviso, quedando de la siguiente manera

	idpostulante	idaviso	visitas
0	0002q	1789742	4
1	0002q	1807692	6
2	0002q	1808090	5
3	0005E	1744005	3
4	0005E	1785430	2

Figura 2.15: Cantidad de visitas de cada postulante a determinado aviso

2.5. Postulaciones

Con postulaciones obtuvimos información que nos permitió caracterizar tanto a los postulantes como a los avisos.

2.5.1. Postulantes

Como vimos en secciones anteriores, los postulantes tienen una clara inclinación por ciertas áreas sobre otras y la zona se disputa principalmente en el Gran Buenos Aires, por lo tanto para caracterizarlos mejor calculamos la cantidad de postulaciones que hicieron dentro de las áreas principales y la cantidad de postulaciones que hicieron en el Gran Buenos Aires.

	idpostulante	edad	sexo_masculino	postulaciones_gba	postulaciones_Ventas	postulaciones_Comercial	...	postulaciones_Atención al Cliente
0	eo2p	37.0	True	2	0.0	1.0	...	0.0
1	1d2B	42.0	True	3	1.0	0.0	...	0.0
2	EBO0	44.0	False	7	0.0	3.0	...	0.0
3	a6MKW	44.0	True	3	0.0	1.0	...	0.0
4	6MWd4	43.0	True	3	0.0	2.0	...	0.0

Figura 2.16: Vista Previa de la Cantidad de Postulaciones a Áreas y en el Gran Buenos Aires por cada postulante

2.5.2 Avisos

Para los avisos nos interesó la relación inversa, es decir, que caracteristicas tenian los postulantes para cada aviso, y entonces calculamos la proporción de hombres que postulaban, la proporción de estudiantes universitarios y finalmente la cantidad total de postulantes.

	idaviso	postulaciones_aviso	postulaciones_masculinas	postulaciones_universitarias
0	8725750	137	0.583942	0.262774
1	11740890	13	0.461538	0.538462
2	12543760	1080	0.426852	0.476852
3	12812680	159	0.119497	0.132075
4	17903700	175	0.108571	0.377143

Figura 2.17: Vista previa de la Caracterización de Avisos según cantidad de postulantes y proporción de Postulantes Masculinos y Universitarios.

Cabe recordar, que para los datos desde el 15 de abril contamos con usuarios para los cuales no tenemos información sobre a que se postularon con lo cual nos deja una cierta cantidad de filas con valores nulos y la decisión que tomó para poder seguir ocupando sus datos fue la de rellenar aquellos valores faltantes con cero, puesto no tenemos información de que se hayan postulado a ningún aviso.

Luego, ya tenemos un set de datos que nos sirve para entrenar a un algoritmo de Machine Learning. Este algoritmo debe predecir si un postulante se postuló o no a un determinado aviso basándose en los features que definimos. Se podría decir que se trata de un problema de clasificación binaria, donde las categorías posibles de clasificación son *'postulación'* o *'no_postulación'*. Bajo esta caracterización del problema, ahora nuestro set de entrenamiento resulta incompleto, ya que solo tenemos datos de una categoría (*'postulación'* o columna *'postulación'* = True). Por lo tanto debemos generar datos que correspondan a pares de usuarios y avisos cuya categoría corresponda a *'no_postulación*'.

	idaviso	idpostulante	GBA	nombre_area	Full-time	Part-time	senior	junior	otro	Jefe	...	I
0	1112257047	NM5M	True	Atención al Cliente	False	True	False	False	True	False	...	(
1	1111920714	NM5M	True	Telemarketing	False	True	True	False	False	False	...	(
2	1112346945	NM5M	True	Telemarketing	True	False	True	False	False	False	...	(
3	1112345547	NM5M	True	Telemarketing	True	False	True	False	False	False	...	(
4	1112237522	5awk	True	Contabilidad	True	False	True	False	False	False	...	(

5 rows × 57 columns

Figura 2.18: Vista Previa de las Postulaciones.

2.6. No Postulaciones

Implícitamente ya tenemos la información faltante, pues sabiendo cómo se relaciona cada postulante y aviso sabemos a cuales no se postuló, pero si observamos los datos nos damos cuenta que cada usuario solo se postuló a una cantidad muy pequeña del total de avisos disponibles con lo cual la cantidad total filas que se tendrían con cada postulante y aviso al cual no se postuló sería enorme y por lo tanto agregar esta información en su totalidad provocaría que no entraran los datos en memoria. Consecuentemente, para solucionar esta problemática, se tomaron postulantes y avisos al azar de tal manera de que la cantidad de filas de los *'no_postulados'* fuera similar a la cantidad de los 'postulados', y por supuesto, eliminando aquellos avisos que pudieran producir contradicciones en las que un postulante se postuló y no se postuló a un aviso, quedándonos siempre con el primero.

	idpostulante	idaviso	GBA	nombre_area	Full-time	Part-time	senior	junior	otro	Jefe	...	postulaciones_Atención al Cliente
0	YjQ9511	1111438862	True	Atención al Cliente	True	False	False	True	False	False	...	0.0
1	vVeGPY5	1112346693	True	Construcción	True	False	False	False	True	False	...	0.0
2	EWv5r4	1112230559	True	Logística	True	False	False	False	True	False	...	0.0
3	5mrwNPq	1112377508	True	Trabajo social	True	False	False	False	False	True	...	0.0
4	2zPLrpw	1112513501	True	Contabilidad	True	False	False	True	False	False	...	0.0

Figura 2.19: Vista Previa de las *'No Postulaciones'*

2.7. Set de datos final

El csv test_final tiene formato (id, idaviso, idpostulante) y por lo tanto para que tenga el mismo formato que los datos anteriores buscamos sus features correspondientes y se los agregamos. Claramente, el data frame formado tendrá una columna menos, la de postulaciones, ya que es esta la que el algoritmo de Machine Learning deberá predecir.

	id	idaviso	idpostulante	edad	sexo_masculino	titulo_universitario	titulo_secundario	titulo_superior	cantidad_p
0	0	739260	6M9ZQR	42.0	False	True	False	True	2.0
1	1	739260	6v1xdL	30.0	True	False	False	False	68.0
2	2	739260	ezRKm9	36.0	False	True	False	False	1.0
3	3	758580	1Q35ej	68.0	True	True	False	True	0.0
4	4	758580	EAN4J6	32.0	False	False	False	False	1.0

5 rows × 27 columns

Figura 2.19: Vista previa de los datos a los cuales les queremos predecir la categoría.

3. Algoritmos de Machine Learning

Los algoritmos utilizados pertenecen a las bibliotecas de scikit learn de los cuales utilizamos Gaussian Naïve Bayes, Árboles de Decisión, Random Forest, Multilayer Perceptron,Perceptrón y SVM. El objetivo de este apartado es dar una breve descripción del algoritmo, el criterio por el cual se utilizó y los distintos resultados que fuimos obteniendo.

3.1. Perceptrón

Primero comenzaremos con el primer algoritmo que utilizamos, este es el caso de perceptrón. Perceptrón es un algoritmo de clasificación lineal que se basa en el campo de las redes neuronales. A nuestro criterio, debido al procesamiento que hemos realizado con los datos (es decir, la de features booleanos), creemos que este tipo de algoritmo podría funcionar de buena manera.

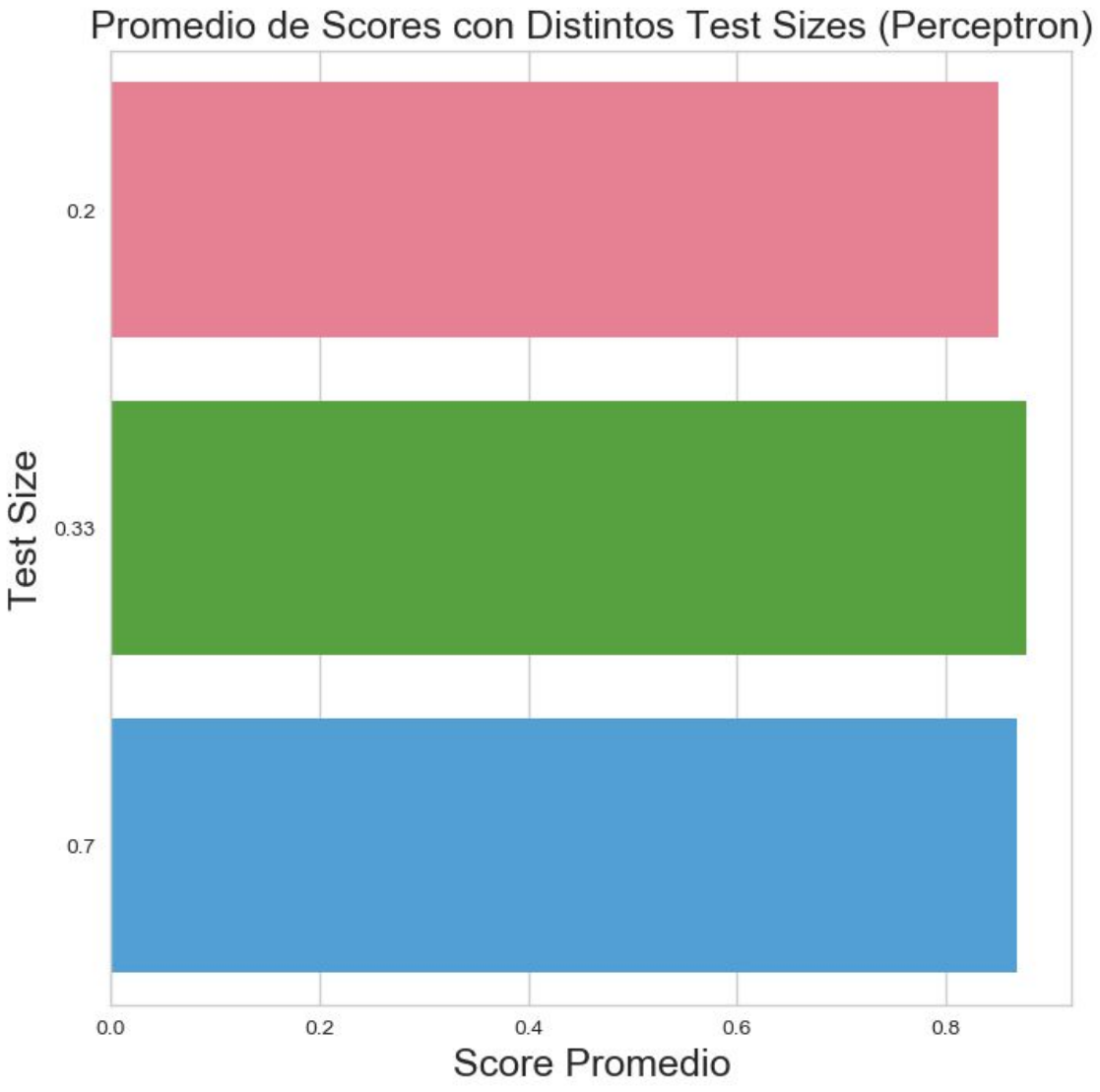

Figura 3.1: Promedios de Scores con Distintos Test Sizes (Perceptrón)

Como se puede ver en la figura anterior, los resultados que obtuvimos fue lo que teníamos en mente, con un promedio realmente importante con cualquier tamaño del set de datos, aunque con una leve superioridad con un set de entrenamiento del 67%. Al ser el primer algoritmo que utilizamos los features fueron cambiando, aunque se llegó al pico (0.90171) con los descritos en el apartado anterior.

3.2. SVM

Luego probamos otro algoritmo de clasificación lineal al igual que Perceptrón, en este caso SVM. Es muy utilizado actualmente y tiene una gran cantidad de versiones y variaciones, pero se pudo ver que los Scores que nos dieron fueron muy inferiores a los obtenidos en el algoritmos anterio. A continuación, se mostraran los Scores promedios con los diferentes tamaños de set de entrenamientos

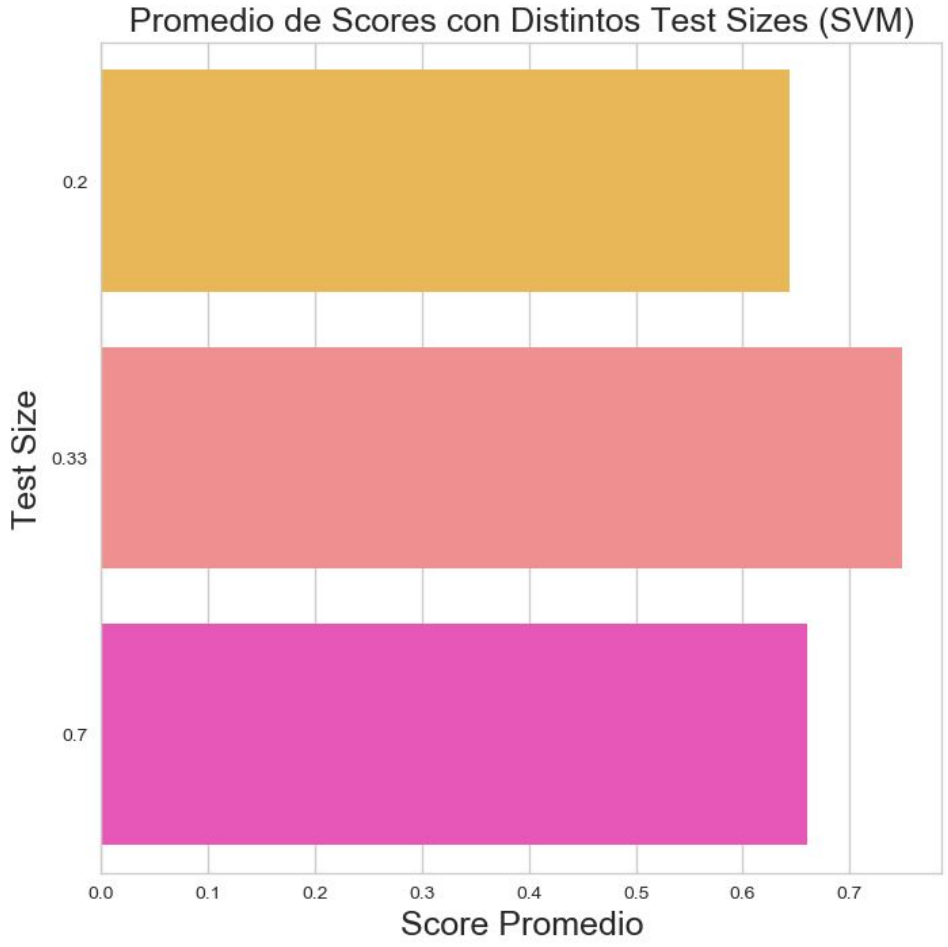

Figura 3.2: Promedios de Scores con Distintos Test Sizes (SVM)

Sorprendentemente los Scores bajaron considerablemente en relación a lo visto en Perceptrón, también se puede observar que hay bastante variación en los Scores con diferentes tamaños en el set de entrenamiento, con el mayor puntaje obtenido con un set de entrenamiento del 67%. Los features utilizados son, nuevamente, los mismos que se enunciaron en el apartado anterior y también se probó con un mayor rango de edades, otra distribución en relación a los géneros y variando las Áreas de Trabajo.

3.3. Gaussian Naïve Bayes

Se quiso utilizar Gaussian Naïve Bayes por el hecho de que es un algoritmo que si bien no es de *'el mejor'* de todos los algoritmos pero tampoco es el peor, por eso nos resultó interesante que resultados generaban. Como se esperaba, los resultados que se generaron no fueron mejores a los obtenidos por Perceptrón. Ahora observemos qué Scores promedios nos dieron con diferentes tamaños de sets de entrenamientos

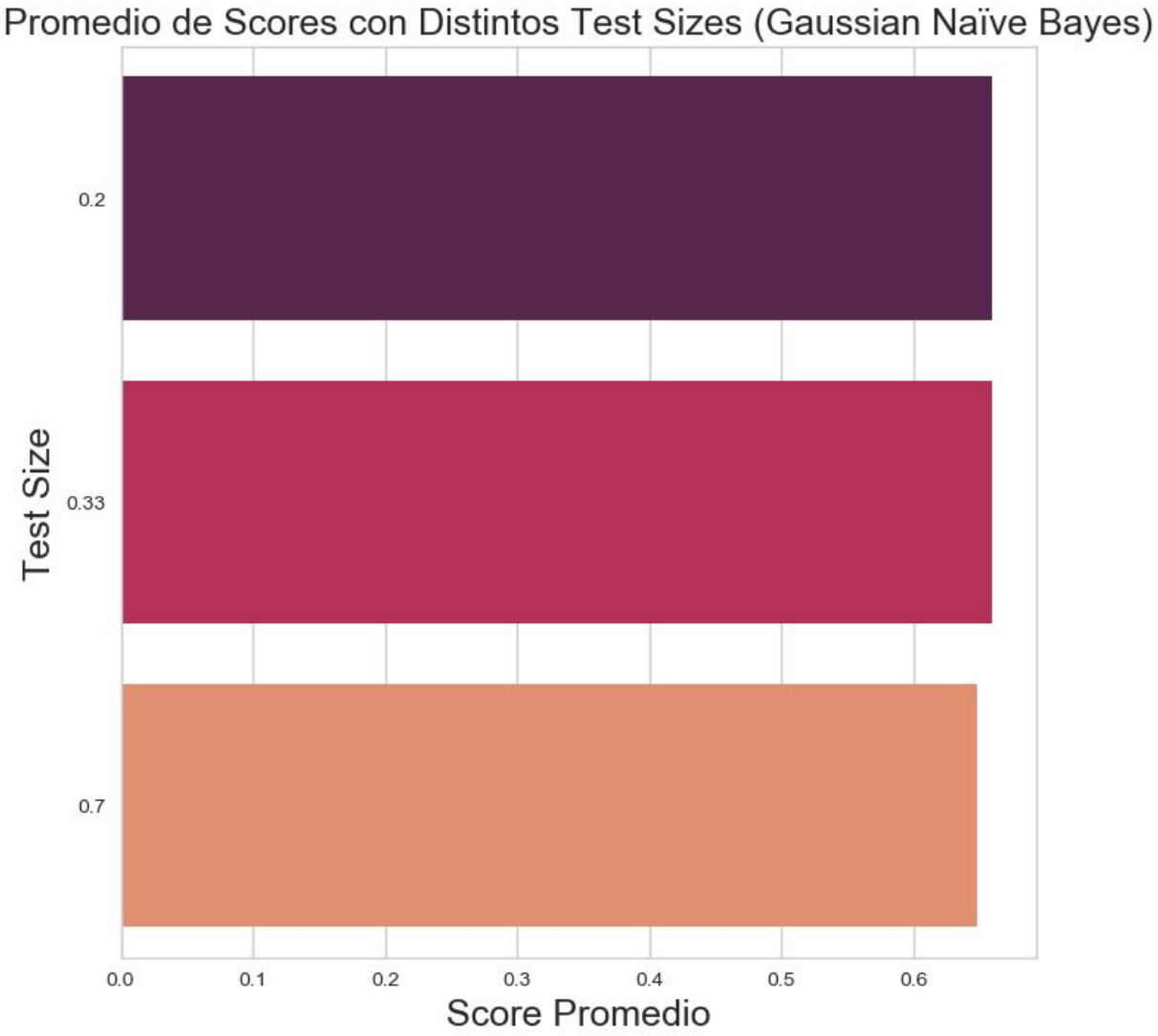

Figura 3.3: Promedios de Scores con Distintos Test Sizes (Naïve Bayes)

Como se dijo anteriormente, los resultados son significativamente menores en relación a los dos primeros algoritmos que se han visto pero podemos ver que aun modificando los tamaños de los sets de entrenamiento, la variación de los resultados es significativamente nula generando nuevamente el mejor resultado con un set de entrenamiento del 67%. Los features utilizados son los descritos en el apartado anterior y luego se tomaron más Áreas de Trabajo y Tipos de Trabajo.

3.4. Árboles de Decisión

Ahora bien, también nos generó interés probar un tipo de algoritmo que sea un poco distinto en relación a los que se vió anteriormente, por eso se inclinó por los Árboles de Decisión. Es un tipo de árbol binario y según cómo se procesaron los datos, nuestra hipótesis es que este algoritmo puede llegar a arrojar buenos resultados, para ello, veamos los Scores que nos generaron

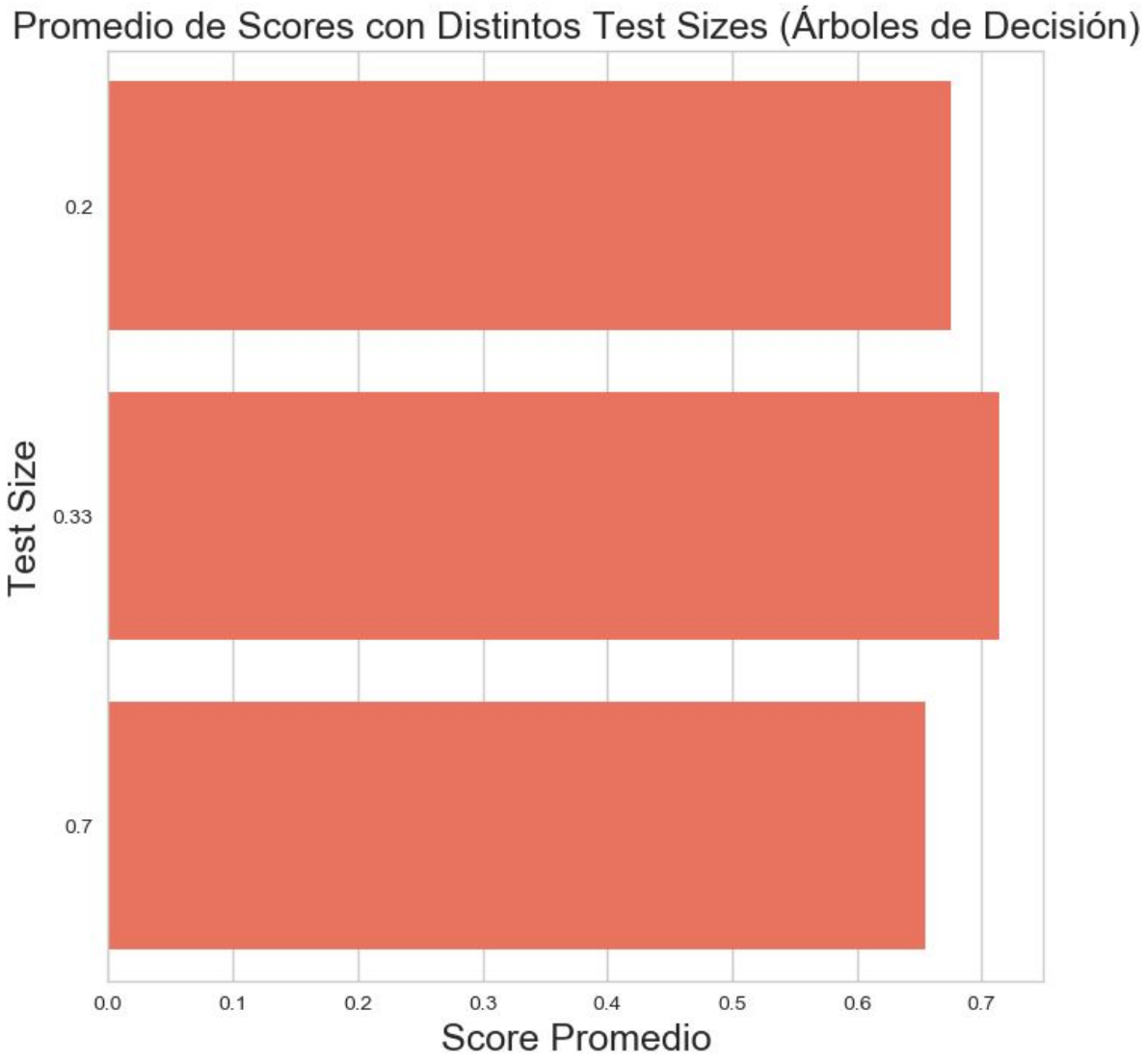

Figura 3.4: Promedios de Scores con Distintos Test Sizes (Árboles de Decisión)

Para nuestra sorpresa, los resultados que obtuvimos no son los que esperábamos, si bien se esperaba que los Scores sean menos a Perceptrón, no nos imaginábamos que iba a haber tanta diferencia. Se sigue la tendencia que el el que mejor resultado arrojó es el que tiene un set de entrenamiento del 67% del set total. Los features utilizados fueron los que se describieron en el apartado anterior, luego también se probó utilizar variables no binarias pero sus resultados no generaron ningún impacto positivo.

3.5. Random Forest

Como se vio en el ítem anterior Árboles de Decisión, queríamos ver qué sucedía con un algoritmo específico de este, como es Random Forest. Como se dijo, este algoritmo es del tipo de árboles de decisión, son de los más populares ya que suelen generar buenos resultados con cualquier tipo de set de datos. En este caso, pudimos observar que el Score fue un poco inferior al obtenido con Perceptrón, con poca variación con los diferentes tamaños del set de entrenamiento. A continuación veremos un gráfico que nos permitirá observar mejor este caso

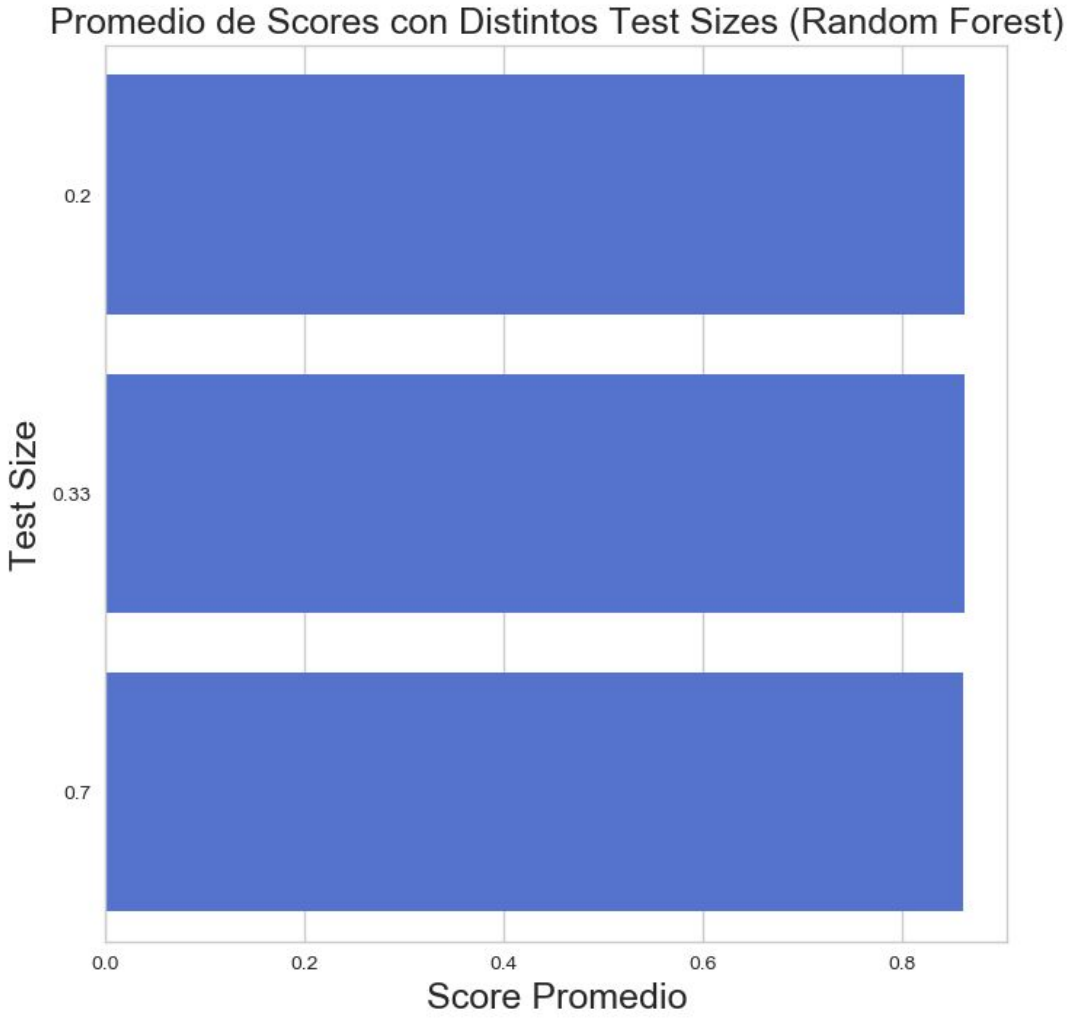

Figura 3.5: Promedios de Scores con Distintos Test Sizes (Random Forest)

Como se puede observar en la figura anterior, prácticamente no hay variación en relación a los distintos tamaños del set de entrenamiento, a su vez, se puede ver que el Score se mantiene realmente alto con un Score de más o menos un 0.86. Los features que se utilizaron fueron los mismos que se describen anteriormente y luego, se tomaron en cuenta otra distribución de género en relación con los que No Declaran,las Top Áreas de Trabajo en relación a las Postulaciones Y los Niveles de Educación según el nivel 'En Curso', ya que como se vió en el análisis Exploratorio, lo tienen una gran mayoría, pero no hubo ningún tipo de variación.

3.6. Multilayer Perceptron

Como se pudo ver durante todo este apartado, Perceptrón fue el algoritmo que hasta ahora mejor funcionó, ahora bien, nos propusimos ver que sucede con un algoritmo el cual es una derivación de este, y ahí encontramos Multilayer Perceptron. Multilayer Perceptrón es una red neuronal artificial formada por múltiples capas, por lo que tiene capacidad para resolver problemas que no son linealmente separables, lo cual es la principal limitación del perceptrón. Nuevamente, veremos a continuación el promedio de Scores que obtuvimos

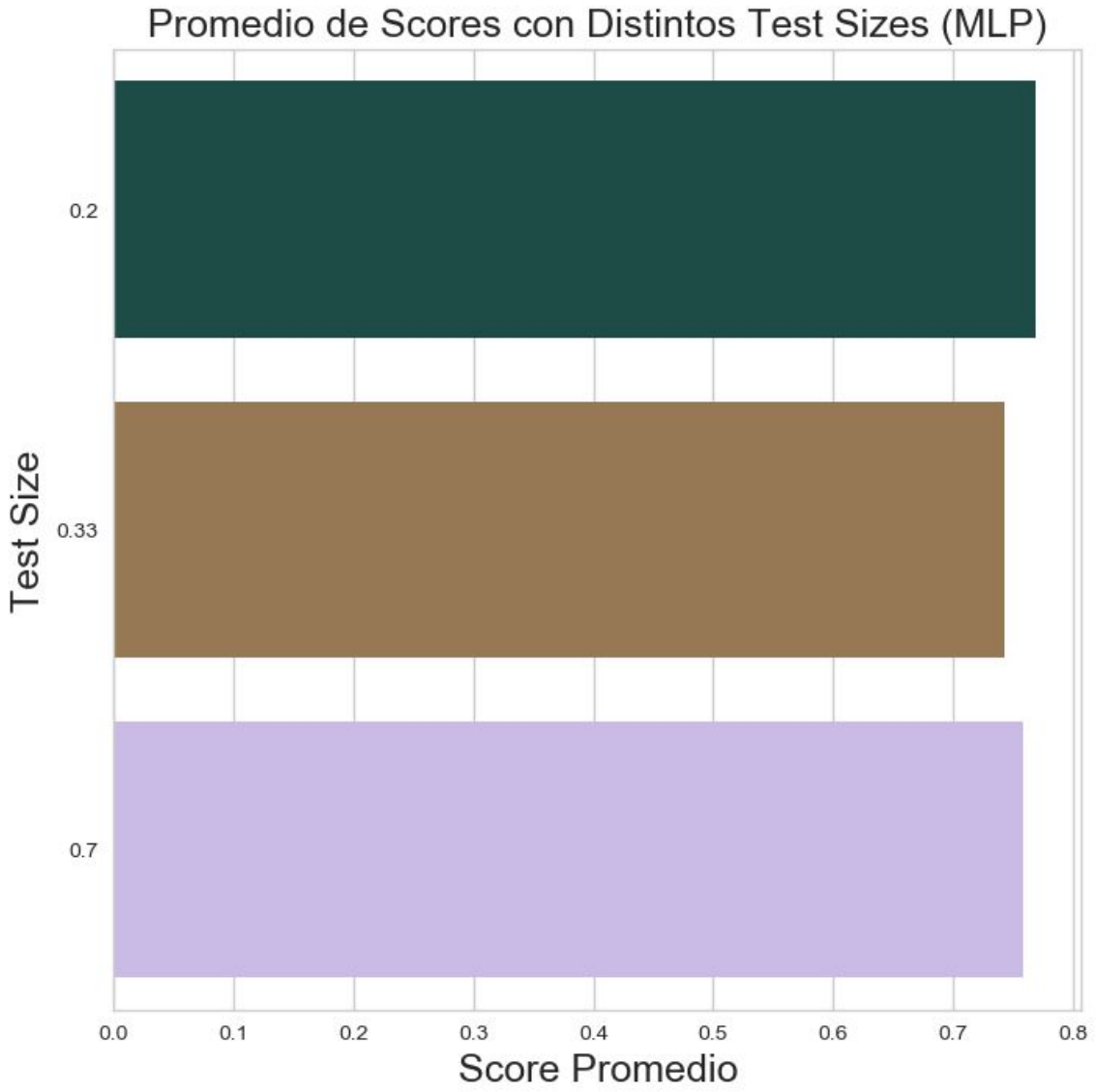

Figura 3.6: Promedios de Scores con Distintos Test Sizes (Multilayer Perceptron)

Nuevamente nos encontramos con sorpresas, a pesar de que Perceptrón fue el algoritmo que mejor funcionó, Multilayer Perceptron no nos dio de los mejores resultados. Se puede observar que el promedio es realmente menor en comparación a Perceptrón y también, esto ya es comparándolo con los otros algoritmos, es el primer algoritmo en el que con un mayor set de entrenamiento (80% en

este caso) funciona mejor en relación con los otros tamaños. En relación con los features, nuevamente se utilizaron los que se enunciaron en el apartado anterior, también se probó con un set más reducido del anterior y probando otra distribución de géneros y de Áreas de Trabajo.

3.7. Tamaño de los Sets de Entrenamiento

Para finalizar, nos resultó interesante tocar el tema del tamaño del set de entrenamiento, en el sentido de cómo son los resultados en función de cuánto entrenamiento tiene, para ello observemos la figura que se encuentra a continuación

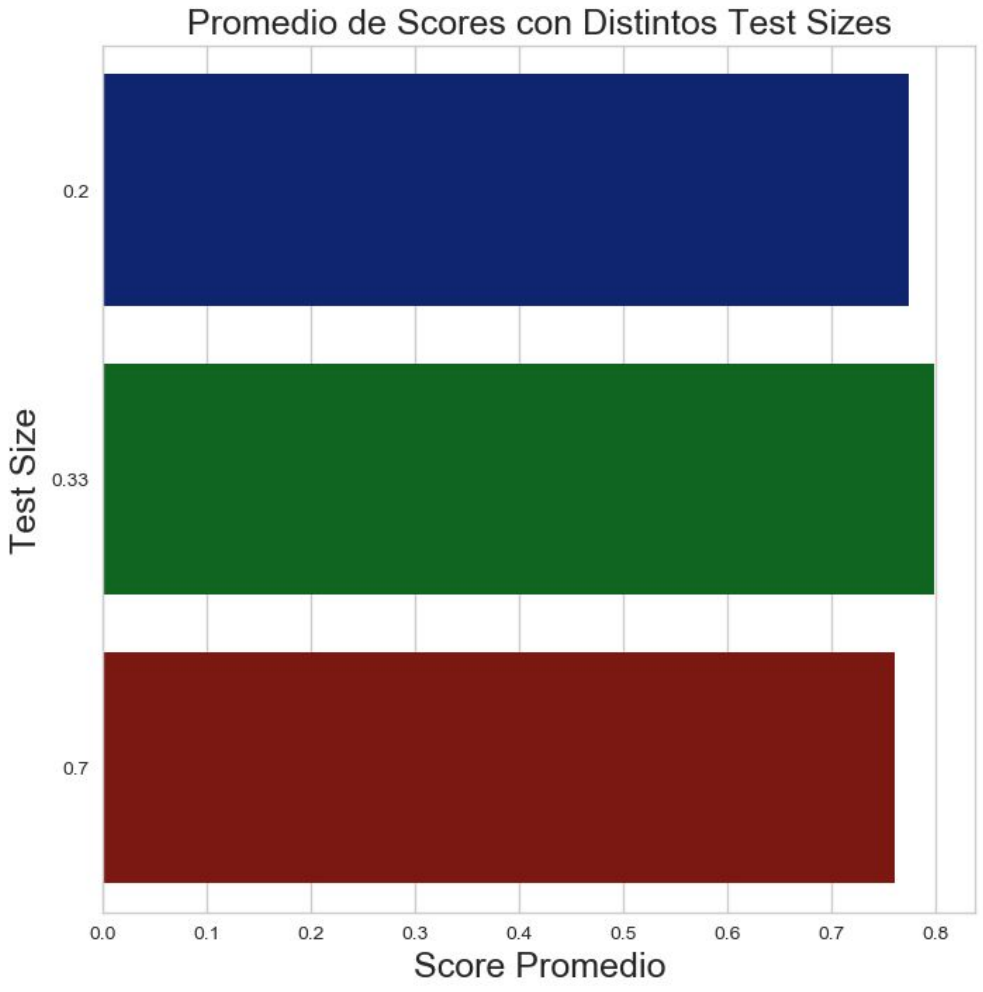

Figura 3.7: Promedio de Scores Total con Distintos Test Sizes

Los resultados finales concuerdan con lo visto a lo largo de todo este apartado, ya que en la mayoría de los algoritmos utilizados (salvo MLP) cuando el test size era de 0.33 (67% del set de entrenamiento), mientras que en el caso de cuánto más o menos tamaño (también nos referimos a tiempo de entrenamiento) se puede ver que no hay diferencias tan marcadas, lo cual sigue respetando lo que se vió a lo largo de este informe. Nuestra opinión acerca de esto es que nos parece coherente ya que no tiene ni mucho entrenamiento (lo cual puede generar OverFitting) ni poco entrenamiento, creemos que es una distribución equitativa por lo cual el algoritmo puede aprender de buena manera y generar una predicción acorde a lo esperado.

4. Conclusión Final

Para finalizar vamos a decir cuál fue el algoritmo que utilizamos para la tener nuestro mejor Score en la competencia y nuestra opinión por la cual se obtuvo. Claramente el que utilizamos fue el algoritmo de Perceptrón pero utilizamos la técnica de Ensamble, debido a que como se vió en clase, es poco usual que un solo algoritmo genere muy buenos resultados, entonces lo que hicimos fue ensamblar los mejores scores que obtuvimos (todos de Perceptrón) y logramos alcanzar el score que tenemos actualmente, con 0.90171.

La idea de este informe era la de mostrar el procesamiento de los datos que se hizo para alcanzar el score que obtuvimos finalmente y los diversos algoritmos que se utilizaron a nuestro criterio, ya sea porque pensabamos que ibamos a tener un buen resultado o porque queríamos ver cómo se comportaba nuestro set de datos con otro tipo de algoritmos.