Organización de Datos (75.06)

Departamento de Computación

28 de junio de 2018

1º cuatrimestre 2018

Grupo 28

Integrantes:

99689	Agostina Vasquez Isla	agostinavasquezisla@gmail.com
100118	Andres Visciglio	atvisciglio@gmail.com
100323	Luciano Ortiz	lucianortiz97@gmail.com
97161	Federico Rossini	federicorossini09@gmail.com

Link a repositorio: https://github.com/Andivisciglio/TP2-OrganizacionDeDatos-Grupo28

Índice

1. Introducción

Este trabajo consiste en una competencia de Machine Learning en donde se debe intentar determinar, para cada usuario presentado, cuál es la probabilidad de que se postule a un cierto aviso laboral. A continuación vamos a explicar la secuencia de pasos utilizados para la construcción del set de entrenamiento

2. Limpieza de datos

En la primera parte del trabajo se realizó una limpieza y reacondicionamiento de los datos provistos por Navent. Con el fin de aumentar la calidad de la informacion, los DataFrames de postulaciones, avisos y postulantes, fueron revisados y corregidos.

Comenzando por el archivo que contiene informacion sobre los usuarios, encontramos que varios de ellos no habían completado su educación. Esto pudo deberse a un error al ingresar lo datos o bien a una actitud del usuario de no querer informar su situacion actual de educación. Decidimos optar por esta segunda hipótesis y completamos los datos de éstos usuarios con la educacion minima posible de los demas (Secundario-Abandonado).

En segundo lugar, con respecto a la información sobre el género y la edad de los usuarios, observamos que había varias fechas de nacimiento incorrectas e informacion faltante sobre el sexo. Para preservar los datos de esos usuarios con fecha de nacimiento incorrecta, la solución fue asignarles la edad promedio de los demás usuarios (31 años). De esta forma no fue necesario descartar usuarios y consecuentemente perder información. Por el lado del género, se unificaron los datos incompletos o faltantes con el atributo "No Declara".

Con respecto a los avisos y las postulaciones, se decidió remover avisos cuyo ïdaviso.^{es}taba repetido y, por su parte, limitar las postulaciones por usuario. Analizando las postulaciones en la primer entrega del presente trabajo, descubrimos la existencia de usuarios que superaban de manera descomunal el promedio de postulaciones por postulante. Fue así que decidimos limitar la cantidad de postulaciones de éstos para que no adquieran un peso significante a la hora de entrenar los algoritmos. La solucion fue establecer una cantidad máxima de postulaciones por usuario, ligada al promedio de postulaciones mencionado.

2.1. Set de Entrenamiento

Consideramos importante mencionar la forma en la que fué generado el set de no-postulaciones, perteneciente al set de entrenamiento final. De manera que el algoritmo aprenda sobre las preferencias y no-preferencias de los usuarios con respecto a los avisos, fue necesario crear un set de no-postulaciones. Pero para perfeccionar la calidad del aprendizaje, nos aseguramos de que estas no-postulaciones, que fueron generadas al azar, cumplan con estas condiciones; No incluir duplas que aparezcan en el archivo de vistas o de postulaciones y no utilizar usuarios inactivos. Determinamos que un usuario es inactivo, a aquel que no figura en el archivo de vistas o bien de postulaciones.

Con respecto a la cantidad de datos utilizados para este trabajo, se concluye que la cantidad óptima es de cien mil datos. Todo comenzó con motivo de perfomance de la computadora. Entrenar con cien mil datos era rápido y los resultados eran buenos. Una vez obtenidos los parámetros de mayor eficacia, decidimos aumentar la cantidad de datos. Sin embargo, los resultados empeoraron. Suponemos que se debe a una sobrecarga innecesaria de información, o una posible caída en Overfitting.

Finalmente, cabe descatar que fueron filtrados del set de entrenamiento, las duplas idpostulante-idaviso que figuraban en el archivo test final. Esto se debe a que es incorrecto utilizar como set de validacion lo que queremos predecir. Ésto influye directamente en la deteccion óptima de los hiperparametros a utilizar.

3. Feature Engineering

Para poder realizar una predicción fue necesario procesar los datos de una forma consistente, para poder llevarlos a un formato que nos sirva para nuestro problema y así acercarse lo más posible a la estimación correcta. Tomamos como referencias las conclusiones más fuertes obtenidas en el TP1.

Una de las cosas que mas trabajo nos costó fue encontrar features que nos dieran informacion útil. Fuimos probando diferentes enfoques al problema y seleccionando los mejores resultados sobre la marcha. Finalmente nos quedamos con los siguientes:

3.1. Postulaciones a ese área en esa zona

En un comienzo, utilizamos como features independientes el área mencionada en el aviso, y la zona. Debido a los pobres resultados que eso nos dió, optamos por vincular ambos features. El resultado fue un numero categórico de rango entero 0 al 4, cuyo peso está dado por la cantidad total de postulaciones que tuvo el usuario

a ese area. Tuvimos en consideracion que el usuario puede ser de otra provincia, por lo que modificamos la funcion que genera el feature para que sólo tenga en cuenta las postulaciones que fueron en la misma zona de ese aviso. De ésta forma, el usuario no se interesará en avisos en zonas lejas a su ubicación.

3.2. Postulaciones a esa empresa

En el análisis del TP1 pudimos observar una correlación entre el renombre de una empresa con la cantidad de postulaciones que tenía. Fue así que Farmacity fue una de las empresas con mayor cantidad de postulaciones. Dicho ésto, para la segunda parte del trabajo se nos ocurrió vincular el peso que tiene el nombre de la empresa con la probabilidad de postulacion. De ésta forma, si un usuario tiene preferencia por una empresa en particular, es probable que se postule a otro aviso de esa empresa. Los resultados obtenido con éste feature demuestran su gran efectividad. El formato es similar al mencionado previamente, consistiendo en una variable númerica categórica del 0 al 4.

3.3. Interes en titulo

Uno de los features mas fuertes, que hizo diferencia significativa en las predicciones fue analizar la informacion de los titulos. La idea consiste en vincular las palabras de postulaciones anteriores de un usuario, con el nuevo aviso a predecir. De ésta forma, si muchas de las palabras coinciden, es probable que el aviso sea similar a sus postulaciones previas. Para seleccionar las palabras del titulo, fueron filtradas las más comunes y que menos informacion aportaban. Nuevamente se categoriza del 0 al 4, segun el peso del aviso.

3.4. Interes edad-genero a ese area

Encontramos una fuerte correlacion entre el género y el área a la hora de postularse.Es por esto que decidimos armar un feature que nos estime un numero (del 0 al 6) en base a la probabilidad de que el usuario esté interesado a esa área dependiendo de su genero. Los resultados fueron positivos. Probando una variante que incluía la edad en la formula, los resultados mejoraron y decidimo dejar el feature vinculando tres caracteristicas; area, edad y género.

3.5. Nivel Laboral y Tipo de Trabajo

Por último, decidimos añadir los features basicos de nivel laboral y tipo de trabajo, categorizados a variables numericas. Esto ayudó a obtener nuestro máximo puntaje, por unos pocos milésimos.

4. Algoritmos de predicción

Todos los algoritmos fueron analizados de acuerdo a sus resultados mediante cross-validation, cambiando las diferentes proporciones entre set de entrenamiento y test. A partir de ello, se eligieron los mas prometedores para probar en nuestro set de entrenamiento.

4.1. Knn

Este algoritmo fue utilizado cuando realizamos nuestras primeras predicciones, lo elegimos ya que fue el mas sencillo de entender, además de ser uno de los más conocidos y dado por la cátedra. Los resultados fueron muy distintos de acuerdo a que features elegíamos para realizar la predicción y a su vez fue uno de los algoritmos que mas tiempo tardaba en arrojar resultados. Concluimos que no era el algoritmo más adecuado para nuestro problema, de este no obtuvimos buenos resultados.

4.2. Naive Bayes

Informandonos leímos que este algoritmo no busca dependencia entre los features sino que mas bien toma a cada uno como independiente del resto, y hace que juntos logren estimar correctamente las predicciones. Esto nos parecio un buen enfoque para el armado de features y fue el primero que nos dio un buen puntaje de alrededor de 0.70. Pero luego lo dejamos de lado cuando vimos que, con los mismos features, otros algoritmos nos daban mucho mejores resultados.

4.3. Perceptron

Perceptrón es otro algoritmo dado por la cátedra, el cual nos pareció apropiado a simple vista dado que puede usarse para clasificación binaria adecuándose perfectamente al problema planteado. Nos proporciono muy buenos resultados utilizando columnas como 'postulaciones a ese área' , 'postulaciones a esa empresa', con altos scores realizando nuestras pruebas.Sin embargo al subirlo a Kaggle los resultados no eran tan prometedores.

4.4. Perceptron Multinivel

Este algoritmo lo utilizamos después de haber probado perceptrón, para abarcar todas las aristas del algoritmo y ver si eso hacia que se incremente su capacidad de predicción, pero no obtuvimos buenos resultados.

Ensambles

Bagging: Decidimos probar este ensamble ya que nos pareció muy útil la propuesta de tener n-estimadores ya que esto permite reducir el overfitting y el ruido.Por lo cual podía potencialmente incrementar el porcentaje de exactitud en el submit de algoritmos como perceptrón que ya habián arrojado buenos resultados en nuestros test. Este ensamble utilizando con el estimador base de perceptron combinado con el reemplazo y agregado de features(ej. cambio de 'postulaciones a ese area' por 'postulaciones a ese área y a esa zona', agregando 'interes en título de aviso','relación genero edad y aviso' y 'postulaciones a esa empresa'). Los resultados obtenidos fueron muy buenos. Utilizamos 50 estimadores.

Boosting: Adaboost y XGBoost: Al igual que para el caso de Bagging nos pareció muy util tener n-estimadores y además que adquieran un peso mayor los resultados que fueron mal clasificados para el algoritmo anterior. Nos proporciono muy buenos resultados utilizando 100 estimadores. Probamos tanto los algoritmos de Adaboost como los de XGBoost, obteniendo por poco margen un mayor resultado Adaboost.

5. Conclusiones

Como conclusiones para éste trabajo final, destacamos la creacion de nuevos features como un factor indispensable para obtener buenas predicciones. A su vez, la limpieza de datos y el critero de armado para el set de entrenamiento aportó a incrementar notablemente los resultados de predicción. Resaltamos la necesidad de probar una variedad de algoritmos diferentes, ya que algunos funcionan considerablemente mejor que otros.

Mencionando los datos aportados por la empresa, queríamos destacar que para un futuro analisis de datos sería ideal implementar algunas mejoras. En primer lugar, enlcarecer la informacion sobre la vistas y las postulaciones. En particular, que las postulaciones estén incluidas en las vistas, ya que ésto generó una confusión a la hora de analizar la informacion brindada. A su vez, recomendamos llevar un registro con las fechas de actividad e inactividad de los avisos que publican, para un mayor poder de analisis. Finalmente, hacer hincapié en los usuarios inactivos, eliminando o separando aquellos sin actividad en un lapzo de tiempo considerable.