FACULTAD DE INGENIERIA
Universidad de Buenos Aires

75.06 Organización de Datos

Trabajo Práctico Nº2

Machine Learning

Grupo 7

Nombre del grupo: Cobra Kai

Integrantes del grupo:

Nombre	Padrón
Eleonora Luna	96444
Germán Bobadilla	90123
Antonella Briglia	90903
Santiago Schuchinsky	96400

1er Cuatrimestre 2018

Introducción

El objetivo del presente trabajo fue determinar la probabilidad de que un usuario se postule a un determinado aviso laboral. Para lograr esto, utilizamos los datos provistos y algoritmos de machine learning para hacer las predicciones. En primer instancia, tomamos los datos, de los cuales ya teníamos una noción por el trabajo práctico realizado anteriormente, y los procesamos nuevamente para poder utilizarlos en los algoritmos de machine learning. Luego de este pre procesamiento, utilizamos los modelos generados para entrenar diversos algoritmos y predecir si un usuario se postulará a un aviso o no. A continuación, detallaremos cómo se han procesado los datos, qué algoritmos se utilizaron y cuáles de ellos han logrado predecir con mejor eficacia que los otros.

Pre-procesamiento de datos

Al revisar la información, nos encontramos con que varios datos podrían traernos problemas a la hora de predecir. Como por ejemplo, datos duplicados o datos inválidos. Si no se los procesaba, y se los utilizaba para entrenar a nuestros modelos, podrían contribuir negativamente a la hora de predecir. Para solucionar estos problemas, realizamos lo siguiente:

Duplicados en csv

Para lidiar con este problema, se comenzó por eliminar los duplicados en cada csv. Es decir, si alguna fila era idéntica a otra esa fila fue eliminada. También podía pasar, como en el caso del csv con la información sobre el género y la edad de los postulantes, que un id de postulante estuviese repetido con distinta información. En ese caso, se preservó la última entrada, ya que se supone que esa fue la última actualización de la información realizada por el postulante.

Valores NaN

Algunos csv contenían valores NaN. Por ejemplo, en el csv que contiene la información sobre el género y edad de los postulantes, la columna de ‘fechanacimiento’ contenía varios valores Nan. Por lo tanto, a la hora de crear una columna ‘Edad’, a esos valores se los rellenaron con la media de la edad de esos postulantes. En el csv de detalles de avisos, en la columna de ‘nivel laboral’, también habían varios valores Nan, por lo tanto se rellenaron con el label ‘Otro’.

En aquellos casos en los cuales la mayoría de los valores de las columnas eran NaN, decidimos descartarlas por completo ya que se consideró que no aportan suficiente información para lograr el trabajo. Unos ejemplos de este caso fueron la columna ‘mapacalle’ y la columna ‘ciudad’ del csv de los detalles de los avisos.

Valores Inválidos

En otros casos, los csv contenían valores inválidos. Por ejemplo, en el csv que contiene la información sobre el género y edad de los postulantes, en la columna ‘género’ habían algunos valores que eran ‘0.0’. Por supuesto que no es un valor representativo para dicha columna, por lo tanto, se reemplazaron estos valores por ‘NO_DECLARA’.

Datos faltantes de idavisos

Analizando el csv de postulaciones se encontraron varios id de avisos que no aparecían en el csv de detalles de los avisos. Por lo tanto, no se contaba con información alguna de estos y al hacer un merge con el dataframe de postulaciones muchos valores quedaban en NaN. Como ninguno de estos avisos formaba parte del set que se quería predecir, se decidió descartar estos avisos ya que no aportaban información suficiente.

Datos a predecir en set de entrenamiento

Se pudo observar que en el csv de postulaciones “hasta el 15 de abril” aparecían postulaciones hasta el 17 de abril inclusive, con lo cual varios de los datos que se querían predecir formaban parte del set de entrenamiento. Para evitar la mala predicción se decidió crear una función que elimina las tuplas que forman parte del set de datos a predecir.

Outliers

Al calcular las edades de los postulantes mediante la fecha de nacimiento que fue provista, se pudo observar que habían postulantes con edades de 7 a 245 años. Algunas de ellas claramente imposibles de ser reales y otras que entran en un rango menor a los 18 años. Antes de eliminarlos, se chequeó si la edad de alguno de los postulantes que se encontraban entre los datos a predecir entraba en estos rangos. Como se encontraron varias edades por debajo de los 18 años y un par arriba de los 100 años, se decidieron dejar todos los datos.

Tuplas de gente postulada y no postulada

Para poder entrenar se necesitaban tanto las tuplas de gente que se postuló a algún aviso y las tuplas de gente que no se postuló a algún aviso.

Para eso se utilizaron las tuplas (idpostulante,idaviso) del csv de postulaciones como las tuplas para la gente postulada.

Para la gente no postulada, dependiendo el modelo, se utilizaron los ‘idpostulante’ con la información más completa (los que no fueron descartados en la fase de preprocesamiento) y se samplearon junto con todos los ‘idaviso’ posibles. Al hacer esta intersección podía pasar que algunas tuplas formarán parte del csv de postulaciones, por lo tanto, esas tuplas se descartaron y de esta manera se logró obtener samples de la gente no postulada.

Feature Engineering

Una vez que nos encargamos de los datos que podrían traernos problemas para entrenar nuestros modelos, procedimos a evaluar para los features qué transformación podríamos aplicarles para lograr la mejor predicción. A continuación detallamos cómo manejamos cada uno de los casos.

Feature Género

Al ser pocos los valores que podía abarcar la columna 'Género', se decidió aplicar un One Hot Encoding para cada uno de esos valores. En este caso: FEM, MASC, o, NO_DECLARA.

Feature Edad

Para este feature se probaron varias opciones. Primero dejamos la edad calculada según la fecha de nacimiento sin alterar.

Luego, se probó dividiendo las edades en distintos grupos por rango. Se probaron varias combinaciones de rangos y luego, a cada rango se le aplicó un one hot encoding.

Se pudo observar, luego de varios intentos, que la opción que dio mejores resultados fue la primera.

Feature Educación

En este caso se decidió asignarle un peso a cada estado del nivel de estudio. Los estados que existen son : Graduado, En Curso o Abandonado. Como se considera más relevante que alguien se haya graduado de algún nivel educativo le asignamos un peso mayor a este estado, el segundo peso a 'En Curso' y por último al estado 'Abandonado'. Se generó una columna por cada nivel educativo y luego, para cada fila se preservó el valor más alto obtenido en ese nivel educativo.

Por ejemplo, si un postulante aparece dos veces con nivel Universitario y una vez con estado 'Graduado' y otra vez con estado 'En Curso', se preserva el mayor, es decir, el estado Graduado. El otro, se descarta. Si no existe información para algún postulante para un cierto nivel, se rellena con ceros.

Feature Nombre Área de Trabajo (Ventas, Adm,etc)

En este caso, las áreas de trabajo eran bastantes y muchas de ellas eran áreas más específicas de un área más general. Por lo tanto, se decidió reducir las áreas e incluir esas áreas más específicas en las áreas más generales.

Por ejemplo, todas las ingenierías fueron incluidas en una sola área llamada ‘Ingeniería’. De esta manera, logramos reducir considerablemente la cantidad de áreas y una vez hecho esto aplicamos un One Hot Encoding para cada una de ellas. Comparamos el resultado con el resultado obtenido aplicando un One Hot Encoding sin reducir las áreas y el resultado fue considerablemente mayor.

Feature Tipo de Trabajo,Nivel Laboral,Nombre Zona

Estas tres características formaban parte del csv de los detalles de los avisos. Las tres contaban con una cantidad de valores menor, por lo tanto se decidió probar con dos opciones. Una fue aplicar un One Hot Encoder y la otra aplicar un label encoder sobre los valores existentes. Se pudo observar, que la primera opción logró mejores resultados para cada característica.

Feature vistas

En este caso se decidió crear una columna llamada ‘Visto’ la cual indica si cierta tupla (idaviso, idpostulante) está incluida o no en el csv de vistas. Es decir si ese postulante visitó alguna vez ese aviso. La columna se rellena con un 1 si está incluida y con un 0 si no lo está. El 1 representa que el postulante visitó el aviso y el 0 que no lo hizo.

Feature tasa de postulación de aviso y postulante

Para este feature, primero se obtuvo la cantidad de vistas y postulaciones para cada aviso. Luego se realizó la división de la cantidad de postulaciones sobre la cantidad de visitas, obteniendo así un valor entre 0 y 1 que representa la tasa de postulación de cada aviso. Se realiza la misma operatoria para los postulantes obteniendo la tasa de postulación de cada uno.

Modelos

En base a estos features, creamos muchos modelos y combinaciones, siendo dos los principales. El primer Modelo el que utiliza la edad, el género de las personas y el feature ‘visto’; con un set de datos reducido. El modelo recién nombrado fue el que obtuvo mejor puntaje en Kaggle y mejor performance con Gradient Boosting Classifier.

El segundo Modelo se le agregan a los features que tiene el anterior la educación y el área de trabajo de los avisos. Este Modelo obtiene buen resultado con el set de entrenamiento y su mejor performance es con Decision Tree Classifier.

Algoritmos probados

Una vez que tuvimos nuestros datos listos para ser entrenados, utilizamos varios algoritmos para generar las predicciones. Algunos modelos funcionaron mejores que otros, aunque entre algunos otros las diferencias fueron mínimas. Probamos utilizando nuestro set de datos y dividiéndolo en dos partes, una para entrenar y otra para testear (reservamos para esto una porción pequeña del set completo). Luego, para los algoritmos que nos dieron un mejor resultado, utilizamos los datos provistos para hacer las predicciones, generar el csv con el formato correspondiente, y subirlo a la competencia de Kaggle. Los algoritmos que probamos fueron los siguientes:

Random Forest Classifier

Este algoritmo fue uno de los cuales obtuvo mejores resultados a la hora de predecir. Los resultados obtenidos fueron parecidos a los obtenidos con el Decision Tree Classifier. Tiene sentido ya que el Random Forest está compuesto por una colección de Decision Trees. Pero, como cada árbol es entrenado con un subset de los features y luego se combinan los resultados de cada uno, hay menos probabilidades de que ocurra overfitting. La desventaja que se presentó es que tarda un tiempo considerable para entrenar y generar las predicciones.

Gradient Boosting Classifier

Este algoritmo logró resultados similares al Random Forest. Pero, a medida que se incluyen más features los resultados tienden a ser menores. La diferencia no termina siendo tan significativa como con Perceptrón pero sigue siendo menor a los resultados obtenidos con el Random Forest o el Decision Tree Classifier. También, tarda bastante más tiempo en calcular el resultado que los algoritmos mencionados. Sin embargo, con uno de los enfoques realizados para hacer la predicción en el cual se minimiza el número de features, fue el caso en el que se obtuvo el mayor puntaje en Kaggle (apenas superior al R.F.).

Decision Tree Classifier

Este algoritmo logró en casi todas nuestras pruebas, un resultado muy parecido al Random Forests. Es decir, un resultado mayor al resto de los clasificadores. La gran ventaja que encontramos con este algoritmo fue que el tiempo que tardaba en calcular el resultado era considerablemente menor. Pero, al ser un único árbol hay muchas más probabilidades de que se produzca overfitting si el set de datos es muy grande.

Perceptron

Este algoritmo nos resultó muy rápido para nuestro set. Fue el algoritmo que menor tiempo tardó en correr pero, a medida que se iban incluyendo más features los resultados

obtenidos resultaron ser bastante más bajos que los demás. Con pocos features predecía bastante parecido a los que resultaron ser los mejores clasificadores, pero su precisión disminuye a medida que se aumentan los features.

Naive Bayes

Este algoritmo, sin bien funcionó rápidamente para nuestro set de datos, no marcó ninguna mejoría frente a otros algoritmos.

KNN

Este algoritmo, comparado con los demás, es extremadamente lento. Mientras que al resto le toma un promedio de 10 o 20 minutos dependiendo el enfoque, datos e hiperparametros, KNN tarda más de una hora con pocos features (siendo interminable con más features). Y la predicción es de las que menor score obtiene.

Ridge Classifier

Ridge Classifier utiliza un modelo lineal que obtiene un puntaje un poco menor al de los algoritmos que mejor resultados brindan. Sin embargo vale la pena destacar que es extremadamente rápido. Por lo que es una buena opción si lo que se quiere obtener es una relativa buena predicción en un muy corto tiempo.

Score en Kaggle con el modelo de mayor puntaje:

Algoritmo con mejor resultado: Gradient Boosting Classifier

Algoritmo	Hiper-parámetros	Score en Kaggle
GradientBoostingClassifier	n_estimators=100	0.92275
AdaBoostClassifier	n_estimators=50	0.92269
RandomForestClassifier	n_estimators=50	0.92034
BaggingClassifier	n_estimators=10	0.92034
DecisionTreeClassifier	-	0.92034
RidgeClassifier	alpha=1.0	0.88860
KNeighborsClassifier	n_neighbors=5	0.74366

Conclusiones

Se pudo observar que los features que más relevancia y mas información aportaron fueron aquellos que estaban relacionados con las vistas de los avisos. Lo cual tiene sentido ya que es mucho más probable que una persona se postule a un aviso si lo vio previamente o si vio algún aviso con características similares. Al tener esto en cuenta, nuestras predicciones mejoraron considerablemente.

También, el nombre del área de los avisos complementada con el género de los postulantes logró incrementar el score de las predicciones. En el Trabajo Práctico anterior se había podido notar que si bien habían areas comunes en las preferencias de hombres y mujeres, habían muchas otras que no aparecían en el ranking de uno o de otro, o incluso otras que si aparecian pero variaban las preferencias.

Por otra parte, en cuanto a la educación se había podido observar que la mayoría de los postulantes que tenían como estado de sus estudios 'abandonado' no eran muy propensos a postularse a algún trabajo. Al ponderar estos estados, teniendo en cuenta los resultados obtenidos anteriormente, se vio reflejada la suba en la precisión de las predicciones.

Por último, ciertos features consiguieron una suba de la precisión pero en cantidades mínimas. Como por ejemplo, el nombre de la zona del aviso, el tipo de trabajo y el nivel laboral. Y, al ser incluidos con los demás features no aportaban información o terminaban degradando la precisión.

Otras características que no aportaron información fueron aquellas en las cuales los datos eran pocos o los valores eran los mismos para todos los casos. Por ejemplo, el id del país era el mismo para todos los avisos y en el caso de la característica 'mapacalle' la mayoría de los datos eran NaN, con lo cual se cree que es un campo que muchas personas deciden no completar y no aporta información para este trabajo.

Link al repositorio de Github

https://github.com/czuczinsky/TP2-Machine-Learning