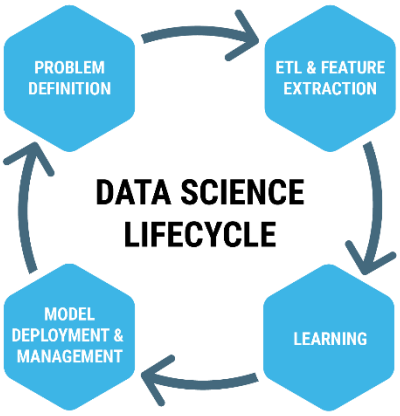
Trabajo Práctico N°2: Organización de Datos (75.06)

Machine Learning



Integrantes	Padrón
Gonzalo Coda	97740
Github repository: https://github.com/Goncod/datos-TP/tree/master/final	

INDICE

1. Análisis y limpieza de csv's

Lo primero que se vio y se realizó en esta segunda parte del trabajo práctico fue un merge de los archivos correspondientes entre sí, es decir, todos los fiuba_1_postulantes_educación por un lado, luego fiuba_2_postulantes_genero_y_edad, y así siguiendo, generando 6 csv finales que se utilizarán en el resto del TP. Aprovechando el análisis y limpieza hecho para la primera parte, estos 6 archivos finales se limpiaron de la misma manera.

El archivo final para el entrenamiento de los algoritmos de ML es un merge de: fiuba_1_postulantes_educacion.csv, fiuba_2_postulantes_genero_y_edad.csv y fiuba_4_postulaciones. Dada la dimensión del csv, no se pudo agregar también los trabajos vistos por los usuarios.

2. Transformaciones realizadas a los datos y feature engineerig

La transformación más simple realizada fue convertir la fecha de nacimiento de cada persona en la edad correspondiente. Luego de eso se analizaron varios métodos para transformar los datos de strings a enteros, de los cuales usamos tres:

- **Find and Replace**: se trata de sustituir las palabras por algún valor numérico manualmente. Esto se utilizó en el csv de la educación de los postulantes dado que no debe tener el mismo peso una persona con Doctorado - graduado que otro con Postgrado - en curso.
- **Label Encoding**: simplemente convierte cada valor de una columna a un número categórico. Se aplicó este método al sexo, nombre de la zona, nombre de área y al tipo de trabajo.
- **One Hot Encoding**: la estrategia de este método es convertir el valor de cada categoría en una nueva columna y asignar un valor 1 o 0 a dicha columna. Esto fue utilizado en el nivel laboral.

One hot Encoding nos resuelve el problema de que los algoritmos malinterpreten los valores que asigna (i.e Label Encoding) pero la desventaja que presenta es que aumenta considerablemente el número de columnas del dataset.

Los valores de las columnas fueron transformados con uno u otro método en base a lo que creemos que es más relevante que el algoritmo reconozca (y que no sea muy pesado el dataset, dado que se necesita mucha memoria).

3. Investigación y entrenamiento de algoritmos de ML

Se investigaron varios algoritmos de la librería sklearn de python y se analizaron sus distintos hiperparámetros. Se vieron tanto algoritmos supervisados lineales como no lineales, también se analizó KNN (no lineal y no supervisado) y dos ensambles. Se pasan a listar a continuación:

❖ Lineales

- **Linear Discriminant Analysis**: Lamentablemente este algoritmo lanzaba error de Memoria y no se pudo probar bien su eficiencia.
- **Logistic Regression:** No sabemos la causa de por qué este algoritmo demoró tanto en entrenar y su predicción en un principio no fue la mejor. Luego se descartó

❖ No lineales:

- **Decision Tree**: se probó con dos criterios: Gini y Entropía. Su predicción (según como están organizados nuestros datos) fue muy baja en ambos casos y prácticamente igual.
- **GaussianNB (Naive Bayes):** Fue el algoritmo más rápido probado. Su predicción fue un poco mayor al 50% y se probó con dos csv de training distintos: uno como el que uso para el resto de algoritmos y el otro con más columnas, utilizando One Hot Encoding en nombre_zona, tipo_de_trabajo y nivel_laboral (se generó un csv con 37 columnas). Lamentablemente esto no mejoró mucho el resultado.
- **K-Neighbords**: se probó con distintos k-hiperparámetros: 5, 9 y 80. Ambos dieron resultados similares. El más preciso fue k = 80.
- **SVC:** Al igual que Logistic Regression demoró mucho en entrenar y no se pudo probar su nivel de predicción, se descartó.
- **Random Forest:** Es un poco rápido. Se probó con un n_estimators 30 y se obtuvieron resultados similares al de GaussianNB. Con un número de estimadores de 50 apróximadamente, requiere mucha memoria y no se pudo probar.

❖ Ensambles:

- **AdaBoost**
- **Stochastic Gradient Boosting**

Ambos ensambles lanzaron error de memoria, por ende tampoco se pudo analizar bien cómo funcionan, su nivel de precisión en la predicción.

Todos los algoritmos fueron probados en una computadora de 12GB de RAM y un procesador i7, no estamos seguros de por qué algunos no funcionaron como se esperaba.

4. Algoritmo final utilizado

El algoritmo final utilizado es GaussianNB, logrando una baja tasa de precisión (53%, un 1% menor que KNN) pero aun así siendo el más veloz y casi el más alto entre los algoritmos probados.

5. Conclusiones

Lo que observamos es que GaussianNB fue un algoritmo que demoró 10 minutos en predecir y tuvo una precisión de 53%, frente a KNN que en todos los casos de K demoraba aproximadamente 3 horas para lograr al final un 54%. Debido a la baja tasa de predicción en todos los algoritmos podemos concluir que nuestros datos no están bien transformados o que no hemos elegido bien los hiperparámetros. Por no ser tanta la diferencia entre GaussianNB y KNN, habría que volver a transformar los datos para ver si se logra una mejor precisión con GNB.

Los otros algoritmos se han probado y han obtenido bajo porcentaje. Por ejemplo, Decision Tree en ambos casos obtuvo un 18% de precisión en su predicción.

Creemos que se deberían obtener puntajes un poco más altos, por lo que nos vuelve a remitir a que nuestros datos están mal formateados o los hiperparámetros no fueron correctamente elegidos.

6. Comentarios

Antes de probar los algoritmos con el csv de test, se intentó entrenar todos los modelos utilizando K-fold Cross Validation para evitar overfitting (k = 5, k = 10, k = 25) pero en la mayoría fueron operaciones que no terminaron nunca (se han dejado más de 10 horas sin obtener un resultado).

Nos enfrentamos ante un mundo nuevo de data-science que lleva mucho tiempo entrenar un buen algoritmo para predecir con la mayor exactitud posible las cosas, hay que dedicarle y hay que dejar la computadora varias horas trabajando. Además se necesita de un estudio riguroso de los tipos de datos con los que uno va a trabajar, cómo convertir esos datos, los features engineering, etc., cosas que llevan mucho tiempo determinar.

Un mundo interesante para seguir aprendiendo.