Análisis deserción de empleados

Guillermo Ortega Arzate

Una empresa de productos médicos está interesada en atraer y mantener al mejor talento porque sabe que es la clave del éxito para cualquier organización. Si un empleado abandona la empresa, se está provocando una pérdida de tiempo y dinero debido, entre otras cosas, a la inversión en capacitación y a la experiencia acumulada del empleado. Desde luego que hay algunas formas de deserción que son inevitables, como cuando un empleado se retira o cambia de ciudad de residencia. Sin embargo, existen algunos factores que se pueden controlar por parte de la empresa con el objetivo de minimizar la deserción al mejorar las condiciones de trabajo. A la empresa le interesa saber cuáles son los factores que hacen que un empleado siga con ellos y cuáles son los que se deben cambiar debido a que provocan que los empleados se vayan.

Se espera que el archivo empleadosRETO.csv esté cargado en el ambiente de colaboratory. Existe una **fecha incorrecta** en el archivo original, el registro dice 30 de febrero de 2012, se modificó a **29 de febrero de 2012**.

```
import os
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Verificar que exista el archivo
if os.path.exists('/content/empleadosRETO.csv') == False:
  raise SystemExit("No existe el archivo de 'empleadosRETO.csv'. Cargue primero el archivo al ambiente de Colab.")

EmpleadosAttrition = pd.read_csv('/content/empleadosRETO.csv')

# Eliminar columnas EmployeeCount, EmployeeNumber, Over18, StandardHours
### REVISAR LA VARIABLE OVERTIME   ###
### Se decide no eliminar la variable Overtime en este momento, podría ser indicativa si existe preferencia o no en un empleado por el trabajo extra ###
EmpleadosAttrition = EmpleadosAttrition.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis=1)

# Crear la columna Year a partir de la columna HiringDate, la fecha está en formato m/d/Y
EmpleadosAttrition['HiringDate'] = pd.to_datetime(EmpleadosAttrition['HiringDate'].str.strip(), format='%m/%d/%Y')
EmpleadosAttrition['Year'] = pd.DatetimeIndex(EmpleadosAttrition['HiringDate']).year

# Crear columna YearAtCompany considerando antigüedad hasta 2018
EmpleadosAttrition['YearsAtCompany'] = 2018 - EmpleadosAttrition['Year']

# Renombar columna DistanceFromHome a DistanceFromHome_km
EmpleadosAttrition = EmpleadosAttrition.rename(columns={'DistanceFromHome': 'DistanceFromHome_km'})

# Eliminar la palabra km de la columna DistanceFromHome_km
EmpleadosAttrition['DistanceFromHome_km'] = EmpleadosAttrition['DistanceFromHome_km'].str.replace('km', '')

# Crear columna DistanceFromHome como entero
EmpleadosAttrition['DistanceFromHome'] = EmpleadosAttrition['DistanceFromHome_km'].astype(int)

# Eliminar columnas Year, HiringDate y DistanceFromHome_km
EmpleadosAttrition = EmpleadosAttrition.drop(['Year', 'HiringDate', 'DistanceFromHome_km'], axis=1)
```

```
# Nuevo dateframe con el sueldo promedio por departamento
SueldoPromedioDepto = EmpleadosAttrition.groupby(['Department'])['MonthlyIncome'].mean()
# Renombar columna de la serie
SueldoPromedioDepto = SueldoPromedioDepto.rename('SueldoPromedio')

# Escalar SueldoPromedio para que tenga valor entre 0 y 1
SueldoPromedioDepto = (SueldoPromedioDepto - SueldoPromedioDepto.min()) / (SueldoPromedioDepto.max() - SueldoPromedioDepto.min())

# Convertir la columna BusinessTravel, Department, EductionField, Gender, JobRole, MaritalStatus, Attrition y Overtime a one-hot numéricas
EmpleadosAttrition = pd.get_dummies(EmpleadosAttrition, columns=['BusinessTravel', 'Department', 'EducationField', 'Gender', 'JobRole', 'MaritalStatus', 'Attrit

# Calcular correlación con respecto a Attrition
Correlacion = EmpleadosAttrition.corrwith(EmpleadosAttrition['Attrition_Yes'])

# Selecionar sólo las columnas que tengan una correlación mayor o igual 0.1
Correlacion = Correlacion[abs(Correlacion) >= 0.1]

# Crear dataframe a partir de EmpleadosAttrition con las columnas que tengan un correlación mayor a 0.1
EmpleadosAttritionFinal = EmpleadosAttrition[Correlacion.index].copy()

# Agregar la columna Attrition utilizando los valores de Attrition_Yes
EmpleadosAttritionFinal.loc[:, 'Attrition'] = EmpleadosAttritionFinal['Attrition_Yes']

# Crear variable EmpleadosAttritionPCA con los componentes principales del EmpleadosAttrionFinal

# Separar las características (X) de la variable objetivo (y)
# Excluimos las columnas de Attrition para el PCA, ya que son la variable objetivo
EmpleadosAttritionFinal = EmpleadosAttritionFinal.drop(columns=['Attrition_Yes', 'Attrition_No'])
X = EmpleadosAttritionFinal.drop(columns=['Attrition'])
y = EmpleadosAttritionFinal['Attrition']

# Estandarizar las características
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Convertir las características escaladas de nuevo a un DataFrame para mejor manejo
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

# PCA
# Se elige n_components=0.80 para dejar los componentes que explican el 80% de la varianza total
pca = PCA(n_components=0.80)
X_pca = pca.fit_transform(X_scaled_df)

# Crear un DataFrame con los componentes principales
pca_columns = [f'PC{i+1}' for i in range(X_pca.shape[1])]
EmpleadosAttritionPCA = pd.DataFrame(data=X_pca, columns=pca_columns)

# Añadir la columna de la variable objetivo 'Attrition' de nuevo al DataFrame PCA
EmpleadosAttritionPCA['Attrition'] = y.reset_index(drop=True)

print("Ratio de varianza explicada por cada componente principal:")
print(pca.explained_variance_ratio_)
print(f"Varianza total explicada: {pca.explained_variance_ratio_.sum():.2f}")

print("\nPrimeras 5 filas de EmpleadosAttritionPCA:")
display(EmpleadosAttritionPCA.head())
```

```
# Unir las columnas de los componentes principales al DataFrame EmpleadosAttritionFinal
df_soloPCA = EmpleadosAttritionPCA.drop(columns=['Attrition'])
EmpleadosAttritionFinal = pd.merge(EmpleadosAttritionFinal, df_soloPCA, left_index=True, right_index=True)

print("Primeras 5 filas de EmpleadosAttritionFinal con los componentes PCA:")
display(EmpleadosAttritionFinal.head())

# Guardar como csv
EmpleadosAttritionFinal.to_csv('EmpleadosAttritionFinal.csv', index=False)
print("Se generó archivo EmpleadosAttritionFinal.csv")
```

```
Ratio de varianza explicada por cada componente principal:
[0.21942395 0.10717548 0.08103248 0.07016553 0.06381979 0.06010521
 0.05655883 0.05161279 0.05033804 0.04737347]
Varianza total explicada: 0.81

Primeras 5 filas de EmpleadosAttritionPCA:
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	Attrition
0	4.033729	-0.710986	-0.059165	-1.624672	-0.486734	-0.169310	1.702556	1.668108	1.815519	-0.832728	0
1	-0.903752	-0.992742	-0.721055	-0.713655	0.385828	0.363012	1.538820	0.222375	-0.025040	0.358982	0
2	-3.431326	-0.808339	2.895546	1.688174	-0.135030	0.909283	1.086065	0.875658	-0.174607	1.158442	1
3	1.687870	-0.719987	1.614177	-0.978045	2.050525	-0.459754	-1.390107	-0.868729	-1.238779	0.922202	0
4	1.021265	2.337563	0.012527	0.084812	0.087954	0.093123	0.261178	-0.360835	0.635105	0.507537	1

```
Primeras 5 filas de EmpleadosAttritionFinal con los componentes PCA:
```

	Age	EnvironmentSatisfaction	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIncome	TotalWorkingYears	YearsInCurrentRole	YearsAtCompany	BusinessTravel_N Tra
0	50	4	3	4	4	17399	32	4	5	
1	36	2	3	2	2	4941	7	2	3	
2	21	2	3	1	2	2679	1	0	1	
3	52	2	3	3	2	10445	18	6	8	
4	33	2	3	3	3	13610	15	6	7	

5 rows × 30 columns

```
Se generó archivo EmpleadosAttritionFinal.csv
```

No fue posible conectarse al servicio de reCAPTCHA. Comprueba tu conexión a Internet y vuelve a cargar la página para obtener un desafío de reCAPTCHA.