Instituto Tecnológico y de Estudios Superiores de Monterrey

Materia:

Analítica de datos y herramientas de inteligencia artificial II

(Gpo 501)

Entregable:

Actividad 7

Regresión Logística

Alumnos:

Andrés Salmerón García A01731809

Julen Ugartechea Repetto A01735646

Yael Mojica Pérez A01735620

Fecha de entrega:

18 de Octubre del 2023

3. Preprocesamiento

Lo primero que se realizó, fue la aplicación de la función '.info()' sobre el data frame, para poder conocer todos los detalles sobre las variables que se tienen, de qué tipo son, etc.

```
#Empezamos con un data.info() para conocer el tipo de columnas con las que contamos
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252000 entries, 0 to 251999
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Id                252000 non-null  int64
 1   Income            252000 non-null  int64
 2   Age               252000 non-null  int64
 3   Experience        252000 non-null  int64
 4   Married/Single    252000 non-null  object
 5   House_Ownership   252000 non-null  object
 6   Car_Ownership     252000 non-null  object
 7   Profession        252000 non-null  object
 8   CITY              252000 non-null  object
 9   STATE             252000 non-null  object
 10  CURRENT_JOB_YRS   252000 non-null  int64
 11  CURRENT_HOUSE_YRS 252000 non-null  int64
 12  Risk_Flag         252000 non-null  int64
dtypes: int64(7), object(6)
memory usage: 25.0+ MB
```

Para corroborar que no existiera ningún dato nulo dentro de las columnas, aplicamos también la función '.isnull()', la cual nos mostró que no existen datos nulos dentro de la base.

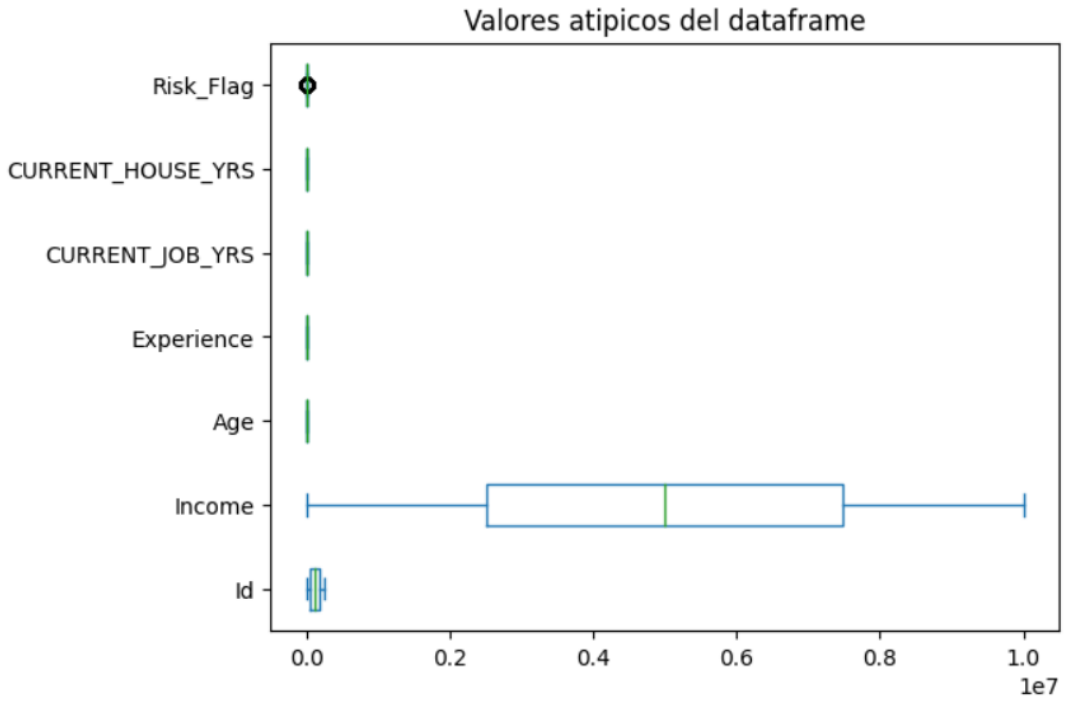
```
#Se corroboran los valores nulos de las columnas con valores cuantitativos
valores_nulos = df.isnull().sum()
valores_nulos
```

```
Id                 0
Income             0
Age                0
Experience         0
Married/Single     0
House_Ownership    0
Car_Ownership      0
Profession         0
CITY               0
STATE              0
CURRENT_JOB_YRS    0
CURRENT_HOUSE_YRS  0
Risk_Flag          0
dtype: int64
```

Una vez que se confirmó que no existían datos nulos en la base de datos, se continuó con la limpieza de la base, buscando la eliminación de los outliers, primero realizando un diagrama de caja para poder observarlos. Al correr el código del diagrama, nos dimos cuenta de que la base tampoco mostraba ningún tipo de valor atípico, por lo que consideramos la limpieza de la base y el preprocesamiento como terminados.

```
#Realizamos un diagrama de caja o bigote de cada columna del dataframe para poder visualizar de manera grafica los outliers de las columna
fig = plt.figure(figsize = (15, 8))
df.plot(kind='box', vert=False)
plt.title('Valores atipicos del dataframe')
plt.show() #Dibujamos el histograma
```

```
<Figure size 1500x800 with 0 Axes>
```




```
df
```

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	Rewa	Madhya_Pradesh
1	2	7574516	40	10	single	rented	no	Software_Developer	Parbhani	Maharashtra
2	3	3991815	66	4	married	rented	no	Technical_writer	Alappuzha	Kerala
3	4	6256451	41	2	single	rented	yes	Software_Developer	Bhubaneswar	Odisha
4	5	5768871	47	11	single	rented	no	Civil_servant	Tiruchirappalli[10]	Tamil_Nadu
...	...	...	...	...	...	...	...	...	...	...
251995	251996	8154883	43	13	single	rented	no	Surgeon	Kolkata	West_Benga
251996	251997	2843572	26	10	single	rented	no	Army_officer	Rewa	Madhya_Pradesh
251997	251998	4522448	46	7	single	rented	no	Design_Engineer	Kalyan-Dombivli	Maharashtra
251998	251999	6507128	45	0	single	rented	no	Graphic_Designer	Pondicherry	Puducherry
251999	252000	9070230	70	17	single	rented	no	Statistician	Avadi	Tamil_Nadu

252000 rows × 13 columns

4. Casos de correlación logística

Para este punto, lo primero que se realizó fue una tabla de frecuencia de la primera variable a analizar, la cual fue la variable ‘CITY’, y se imprimieron las primeras 10 columnas para poder conocer cuáles eran las ciudades que se repetían más, ya que estas tendrían un mayor impacto para la realización de la regresión lineal.

```
#Generamos una tabla de frecuencia paa conocer las ciudades que mas se repiten para aplicar la regresión lineal a estos
#Ya que estos serían los que tendrían mayor impacto
tabla = freq_tbl(df['CITY'])
#Ajusto el índice de mi dataframe
tabla = tabla.set_index('CITY')
tabla = tabla.head(10)
tabla
```

CITY	frequency	percentage	cumulative_perc
Vijayanagaram	1259	0.004996	0.004996
Bhopal	1208	0.004794	0.009790
Bulandshahr	1185	0.004702	0.014492
Saharsa[29]	1180	0.004683	0.019175
Vijayawada	1172	0.004651	0.023825
Srinagar	1136	0.004508	0.028333
Indore	1130	0.004484	0.032817
New_Delhi	1098	0.004357	0.037175
Hajipur[31]	1098	0.004357	0.041532
Satara	1096	0.004349	0.045881

Se creó un filtro llamado df1, el cual contiene solamente las filas en las que aparece la ciudad ‘Vijayawada’ y ‘Srinagar’ y se imprimieron las primeras cinco filas para corroborar que funcione el filtro:

```
df1=df[(df["CITY"]=="Vijayawada") | (df["CITY"]=="Srinagar")]
df1.head()
```

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE	CURRENT_JOB
110	111	1197375	41	0	single	rented	yes	Microbiologist	Srinagar	Jammu_and_Kashmir	
190	191	6862187	22	7	single	rented	no	Scientist	Srinagar	Jammu_and_Kashmir	
718	719	3753805	68	0	single	rented	no	Physician	Srinagar	Jammu_and_Kashmir	
736	737	960205	66	16	single	rented	no	Comedian	Vijayawada	Andhra_Pradesh	
863	864	1239373	64	4	single	rented	no	Lawyer	Srinagar	Jammu_and_Kashmir	

Utilizando este filtro, se definieron las variables independiente y dependiente, siendo ‘Income’ y ‘CITY’ respectivamente, se dividió el conjunto de datos en prueba y entrenamiento, se escalaron los datos, se definió el algoritmo ‘LogisticRegression’ de la librería ‘sklearn.linear_model’ y se realizó la predicción, todo esto, se puede observar en el siguiente código:

```
#Declaramos las variables dependientes e independientes para la regresión logística
Vars_Indep= df1[['Income']]
Var_Dep= df1['CITY']

#Redifinimos las variables
X=Vars_Indep
y=Var_Dep

#Dividimos el conjunto de datos en la parte de entrenamiento y prueba:
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.3, random_state=None)

#Se escalan los datos
escalar=StandardScaler()

#Para realizar el escalamiento de las variables 'X' tanto de entrenamiento como de prueba, utilizando
X_train=escalar.fit_transform(X_train)
X_test=escalar.fit_transform(X_test)

#Definimos el algoritmo a utilizar
from sklearn.linear_model import LogisticRegression
algoritmo=LogisticRegression()

#Entrenamos el modelo
algoritmo.fit(X_train, y_train)

#Realizamos una predicción
y_pred=algoritmo.predict(X_test)
```

Se verificó la matriz de confusión y se calcularon los cuatro valores que nos interesan de cada una de las variables, las cuales son la precisión del modelo, la exactitud, la sensibilidad y el puntaje de F1. Todo esto se realizó en el siguiente código:

```
#Verifico la matriz de Confusión
from sklearn.metrics import confusion_matrix
matriz=confusion_matrix(y_test, y_pred)
print('Matriz de confusión:')
print(matriz)
print()
print()

#Calculo la precisión del modelo
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_pred, average="binary", pos_label="Vijayawada")
print('Precisión del modelo:', precision)
print()
print()

#Calculo la exactitud del modelo
from sklearn.metrics import accuracy_score

exactitud = accuracy_score(y_test, y_pred)
print('Exactitud del modelo:', exactitud)
print()
print()

#Calculo la sensibilidad del modelo
from sklearn.metrics import recall_score

sensibilidad = recall_score(y_test, y_pred, average="binary", pos_label="Vijayawada")
print('Sensibilidad del modelo:', sensibilidad)
print()
print()
```

Se le aplicó el mismo código de la regresión logística a la variable ‘CITY’ dos veces más, pero sobre otros nuevos frames llamados df2 y df3, los cuales contienen solamente los datos de las columnas con las ciudades “Bulandshahr” y “Saharsa[29]” dentro del filtro df2 e “Indore” y “Bhopal” dentro del filtro df3. Los resultados fueron los siguientes:

	1. CITY	2. CITY	3. CITY
Precisión	0.5722	0.5131	0.5559
Exactitud	0.5988	0.5380	0.5641
Sensibilidad	0.6547	0.5799	0.4970
F1 Score	0.6106	0.5444	0.5248

Si tomamos como dato importante la sensibilidad, vemos que la regresión logística que tuvo un mejor porcentaje, fue el primero, ya que tuvo un porcentaje de 65.47%, además, cabe mencionar que en los demás datos, de precisión, exactitud y F1, igualmente fue más alto al de las otras regresiones, lo que nos daría como resultado que la primera regresión logística es la más correcta, según los estados, variables y parámetros utilizados.

Para obtener estos cuatro valores de cada una de las variables que nos interesa, utilizamos el mismo procedimiento que se realizó con esta variable, desde la tabla de frecuencia de la nueva variable, la creación del primer filtro df1 con los valores de las columnas que nos interesan de cada variable, hasta realizar la regresión logística sobre la variable en cada u no de los filtros creados, para finalmente calcular y obtener los valores de precisión, exactitud, sensibilidad y puntaje F1 de cada variable en los distintos filtros.

```
#Generamos una tabla de frecuencia paa conocer las profesiones que mas se repiten para aplicar la regresión lineal a estos
#Ya que estos serían los que tendrían mayor impacto
tabla2 = freq_tbl(df['Profession'])
#Ajusto el índice de mi dataframe
tabla2 = tabla2.set_index('Profession')
tabla2 = tabla2.head(10)
tabla2
```

Profession	frequency	percentage	cumulative_perc
Physician	5957	0.023639	0.023639
Statistician	5806	0.023040	0.046679
Web_designer	5397	0.021417	0.068095
Psychologist	5390	0.021389	0.089484
Computer_hardware_engineer	5372	0.021317	0.110802
Drafter	5359	0.021266	0.132067
Magistrate	5357	0.021258	0.153325
Fashion_Designer	5304	0.021048	0.174373
Air_traffic_controller	5281	0.020956	0.195329
Comedian	5259	0.020869	0.216198

```
df4=df[(df["Profession"]=="Physician") | (df["Profession"]=="Statistician")]
df4.head()
```

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE	CURRENT_JOB_YRS	CURRE
12	13	9120988	28	9	single	rented	no	Physician	Erode[17]	Tamil_Nadu	9	
29	30	4386333	31	16	single	rented	no	Physician	Shimoga	Karnataka	3	
89	90	6097344	27	16	single	rented	yes	Statistician	Kottayam	Kerala	10	
98	99	5083653	35	14	single	rented	yes	Statistician	Ambattur	Tamil_Nadu	12	
123	124	9867887	79	9	single	rented	no	Physician	Chapra	Bihar	4	

```
df5=df[(df["Profession"]=="Web_designer") | (df["Profession"]=="Psychologist")]
df5.head()
```

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE
54	55	1213131	67	8	single	rented	no	Psychologist	Agartala	Tripura
102	103	4417164	42	12	single	rented	no	Web_designer	Vellore	Tamil_Nadu
121	122	8312895	35	12	single	rented	no	Psychologist	Bhagalpur	Bihar
133	134	9397962	58	3	married	rented	no	Psychologist	Saharanpur	Uttar_Pradesh
134	135	3643187	77	20	single	rented	no	Web_designer	Bellary	Karnataka

```
df6=df[(df["Profession"]=="Computer_hardware_engineer") | (df["Profession"]=="Statistician")]
df6.head()
```

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE
32	33	4128828	21	10	single	rented	no	Computer_hardware_engineer	Khammam	Telangana
89	90	6097344	27	16	single	rented	yes	Statistician	Kottayam	Kerala
96	97	3449511	50	19	single	rented	yes	Computer_hardware_engineer	Rajpur_Sonarpur	West_Bengal
98	99	5083653	35	14	single	rented	yes	Statistician	Ambattur	Tamil_Nadu
115	116	5522159	22	6	single	rented	no	Computer_hardware_engineer	Anantapur	Andhra_Pradesh

Los resultados que obtuvimos en los filtros de la variable profession fueron los siguientes:

	4. Profession	5. Profession	6. Profession
Precisión	0.5	0.4863	0.5232
Exactitud	0.4950	0.4878	0.5331
Sensibilidad	0.9456	0.5280	0.3621
F1 Score	0.6541	0.5063	0.4280

En este caso se tuvo como resultado que la 4ta regresión logística es la que obtuvo mejores resultado en comparación a la regresión logística 5 y 6, esto debido a que para la 4ta regresión logística se usaron como categorías las 2 profesiones que más se repetían, esto con el objetivo de tener un mejor análisis, ya que este captaría y usaría más datos que las otras regresiones logísticas, lo que daría como resultado un mejor análisis, es por ello que este tuvo mejores porcentajes que la regresión logística 5 y 6,.

La siguiente variable que se analizó fue la variable 'STATE', para esta variable, solo se realizaron dos filtros, df7 y df8, y se les aplicó a estos el mismo procedimiento desde la tabla de frecuencia de la variable, hasta las regresiones logísticas:

```
#Generamos una tabla de frecuencia paa conocer los estados que mas se repiten para aplicar la regresión lineal a estos
#Ya que estos serían los que tendrían mayor impacto
tabla3 = freq_tbl(df['STATE'])
#Ajusto el índice de mi dataframe
tabla3 = tabla3.set_index('STATE')
tabla3 = tabla3.head(10)
tabla3
```

STATE	frequency	percentage	cumulative_perc
Uttar_Pradesh	28400	0.112698	0.112698
Maharashtra	25562	0.101437	0.214135
Andhra_Pradesh	25297	0.100385	0.314520
West_Bengal	23483	0.093187	0.407706
Bihar	19780	0.078492	0.486198
Tamil_Nadu	16537	0.065623	0.551821
Madhya_Pradesh	14122	0.056040	0.607861
Karnataka	11855	0.047044	0.654905
Gujarat	11408	0.045270	0.700175
Rajasthan	9174	0.036405	0.736579

```
df7=df[(df["STATE"]=="Madhya_Pradesh") | (df["STATE"]=="Maharashtra")]
df7.head()
```

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE	CURRENT_JOB_
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	Rewa	Madhya_Pradesh	
1	2	7574516	40	10	single	rented	no	Software_Developer	Parbhani	Maharashtra	
5	6	6915937	64	0	single	rented	no	Civil_servant	Jalgaon	Maharashtra	
17	18	3666346	56	12	single	rented	no	Politician	Bhusawal	Maharashtra	
27	28	9643150	24	13	single	rented	no	Comedian	Indore	Madhya_Pradesh	

```
df7=df[(df["STATE"]=="Madhya_Pradesh") | (df["STATE"]=="Maharashtra")]
df7.head()
```

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	Rewa	Madhya_Pradesh
1	2	7574516	40	10	single	rented	no	Software_Developer	Parbhani	Maharashtra
5	6	6915937	64	0	single	rented	no	Civil_servant	Jalgaon	Maharashtra
17	18	3666346	56	12	single	rented	no	Politician	Bhusawal	Maharashtra
27	28	9643150	24	13	single	rented	no	Comedian	Indore	Madhya_Pradesh

Los resultados de las regresiones en estos casos fueron los siguientes:

	7. STATE	8. STATE
Precisión	0.6455	0.5193
Exactitud	0.6456	0.5207
Sensibilidad	0.0045	0.9984
F1 Score	0.0089	0.6832

Para este caso solo queda mencionar que hay una gran diferencia entre los porcentajes de las regresiones 7 y 8, ya que esto es por las variables dependientes que se usaron, ya que estos resultaron en buenas o malas matrices, es por ello que la regresión logística 8 es mejor que la 7.

Por último se volvió a repetir el mismo procedimiento, con la variable ‘Experience’, con la única diferencia siendo que en lugar de crear dos filtros más, se utilizaron intervalos y categorías como se muestra en el siguiente código:

```
intervalos=[-math.inf, 4, math.inf]
categorias=['Menores', 'Mayores']
df['Experience_Intervalos']=pd.cut(x=df['Experience'], bins=intervalos, labels=categorias)
df.head()
```

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	Rewa
1	2	7574516	40	10	single	rented	no	Software_Developer	Parbhani
2	3	3991815	66	4	married	rented	no	Technical_writer	Alappuzha
3	4	6256451	41	2	single	rented	yes	Software_Developer	Bhubaneswar
4	5	5768871	47	11	single	rented	no	Civil_servant	Tiruchirappalli[10]

Los resultados que se obtuvieron en estas dos últimas regresiones logísticas fueron los siguientes:

	9. Experience	10. Experience
Precisión	0.7390	0.9426
Exactitud	0.8921	0.8910
Sensibilidad	0.8106	0.9144
F1 Score	0.7732	0.9283

Finalmente tenemos las ultimas 2 regresiones, que aunque ambas tienen resultados muy buenos, la que dio mejores resultados fue la ultima regresión, esto por la elección de categorias, ademas, cabe mencionar que se usaron como variables a

5. Conversión de variables de tipo dicotómica

En este punto, se realizó un solo código para cambiar las variables necesarias a el tipo variable dicotómica utilizando primero la función "logistic_regression_and_metrics", dividiendo el conjunto como se realizó anteriormente, al igual que escalando los datos, creando y entrenando el modelo, realizando la predicción y calculando los coeficientes de precisión, exactitud y sensibilidad, como se muestra a continuación:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_score, accuracy_score, recall_score

def logistic_regression_and_metrics(data, independent_var, dependent_var):
    # Dividir el conjunto de datos en entrenamiento y prueba
    X = data[independent_var]
    y = data[dependent_var]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=None)

    # Escalar los datos
    escalar = StandardScaler()
    X_train = escalar.fit_transform(X_train)
    X_test = escalar.transform(X_test)

    # Crear y entrenar el modelo de regresión logística
    model = LogisticRegression()
    model.fit(X_train, y_train)

    # Realizar una predicción
    y_pred = model.predict(X_test)

    # Calcular los coeficientes de precisión, exactitud y sensibilidad
    precision = precision_score(y_test, y_pred, average="binary", pos_label=data[dependent_var].unique()[0])
    accuracy = accuracy_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred, average="binary", pos_label=data[dependent_var].unique()[0])

    return precision, accuracy, recall
```

Después, se aplicó la función del código anterior para cada una de las combinaciones de variables de los filtros creados, desde df1 hasta df8 y los intervalos para las últimas dos.

```python
# Ahora aplicamos la función para cada combinación de variables

# Combinación 1: 'CITY' - 'Vijayawada' vs 'Srinagar'
precision1, accuracy1, recall1 = logistic_regression_and_metrics(df1, ['Income'], 'CITY')

# Combinación 2: 'CITY' - 'Bulandshahr' vs 'Saharsa[29]'
precision2, accuracy2, recall2 = logistic_regression_and_metrics(df2, ['Income'], 'CITY')

# Combinación 3: 'CITY' - 'Indore' vs 'Bhopal'
precision3, accuracy3, recall3 = logistic_regression_and_metrics(df3, ['Income'], 'CITY')

# Combinación 4: 'Profession' - 'Physician' vs 'Statistician'
precision4, accuracy4, recall4 = logistic_regression_and_metrics(df4, ['Experience'], 'Profession')

# Combinación 5: 'Profession' - 'Web_designer' vs 'Psychologist'
precision5, accuracy5, recall5 = logistic_regression_and_metrics(df5, ['Experience'], 'Profession')

# Combinación 6: 'Profession' - 'Computer_hardware_engineer' vs 'Statistician'
precision6, accuracy6, recall6 = logistic_regression_and_metrics(df6, ['Experience'], 'Profession')

# Combinación 7: 'STATE' - 'Maharashtra' vs 'Madhya_Pradesh'
precision7, accuracy7, recall7 = logistic_regression_and_metrics(df7, ['Income', 'Age', 'Experience', 'CURRENT_JOB_YRS', 'CURRENT_HOUSE_YRS', 'Risk_Flag'], 'STATE')

# Combinación 8: 'STATE' - 'Andhra_Pradesh' vs 'West_Bengal'
precision8, accuracy8, recall8 = logistic_regression_and_metrics(df8, ['Income', 'Age', 'CURRENT_HOUSE_YRS'], 'STATE')

# Combinación 9: 'Experience' - Menores vs Mayores
precision9, accuracy9, recall9 = logistic_regression_and_metrics(df, ['CURRENT_JOB_YRS'], 'Experience_Intervalos')

# Combinación 10: 'Experience' - Menores vs Mayores (otra vez)
precision10, accuracy10, recall10 = logistic_regression_and_metrics(df, ['CURRENT_JOB_YRS'], 'Experience_Intervalos')
```

Por último, se creó un data frame que contuviera todos los resultados y se imprimió en forma de tabla, lo cual se puede observar en el siguiente código con su respectivo resultado al ejecutarlo:

```python
# Crear un DataFrame con los resultados
results_df = pd.DataFrame({
    'Combinación': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Variable Independiente': ['Income', 'Income', 'Income', 'Experience', 'Experience', 'Experience', 'Income, Age, Experience, CURRENT_JO
    'Variable Dependiente': ['CITY', 'CITY', 'CITY', 'Profession', 'Profession', 'Profession', 'STATE', 'STATE', 'Experience_Intervalos', '
    'Precision': [precision1, precision2, precision3, precision4, precision5, precision6, precision7, precision8, precision9, precision10],
    'Accuracy': [accuracy1, accuracy2, accuracy3, accuracy4, accuracy5, accuracy6, accuracy7, accuracy8, accuracy9, accuracy10],
    'Recall': [recall1, recall2, recall3, recall4, recall5, recall6, recall7, recall8, recall9, recall10]
})

# Imprimir la tabla de resultados
print(results_df)
```

	Combinación	Variable Independiente
0	1	Income
1	2	Income
2	3	Income
3	4	Experience
4	5	Experience
5	6	Experience
6	7	Income, Age, Experience, CURRENT_JOB_YRS, CURR...
7	8	Income, Age, CURRENT_HOUSE_YRS
8	9	CURRENT_JOB_YRS
9	10	CURRENT_JOB_YRS

	Variable Dependiente	Precision	Accuracy	Recall
0	CITY	0.563467	0.556999	0.522989
1	CITY	0.600575	0.591549	0.580556
2	CITY	0.553672	0.529915	0.280802
3	Profession	0.528582	0.533012	0.827624
4	Profession	0.562108	0.520544	0.271351
5	Profession	0.525394	0.527132	0.364964
6	STATE	0.000000	0.640769	0.000000
7	STATE	0.523280	0.524054	0.963376
8	Experience_Intervalos	0.743759	0.892698	0.811483
9	Experience Intervalos	0.740795	0.891759	0.811728