[1]Universidad Sergio Arboleda
Presentado por
Guillermo De Mendoza
Nataly Jineth Roa Peña

Análisis de datos a partir de información encontrada para cada base de datos

Análisis de datos para 11 tablas con 11 países de latinoamérica, el objetivo principal de este análisis es reconocer y eliminar NaN.

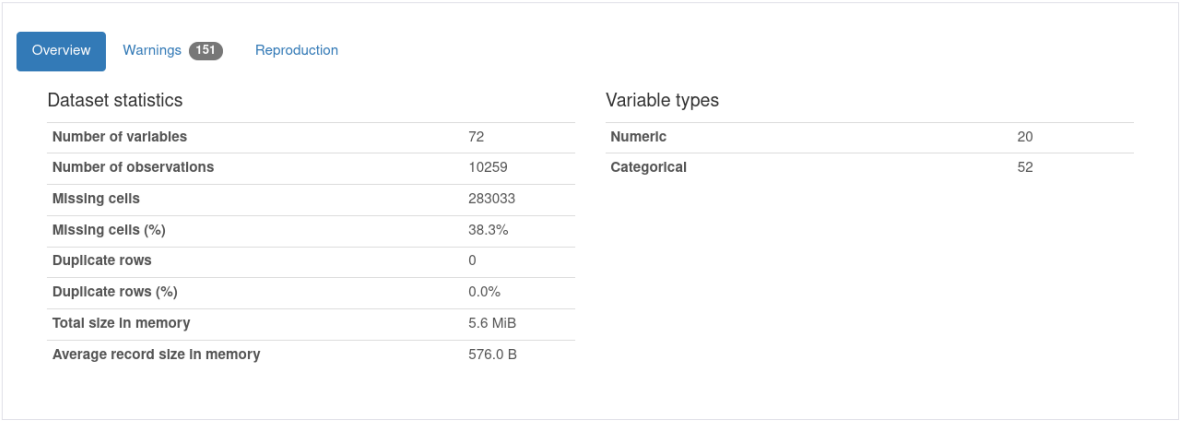
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10259 entries, 0 to 10258
Data columns (total 72 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Unnamed: 0                              10259 non-null  int64
 1   Organization Name                       10259 non-null  object
 2   Organization Name URL                   10259 non-null  object
 3   Industries                              10127 non-null  object
 4   Headquarters Location                   10259 non-null  object
 5   Description                             10259 non-null  object
 6   CB Rank (Company)                       10259 non-null  object
 7   Headquarters Regions                    8259 non-null   object
 8   Estimated Revenue Range                 6124 non-null   object
 9   Operating Status                        10259 non-null  object
 10  Founded Date                            9897 non-null   object
 11  Founded Date Precision                  9897 non-null   object
 12  Exit Date                               1300 non-null   object
 13  Exit Date Precision                     1300 non-null   object
 14  Company Type                            10033 non-null  object
 15  Website                                 10180 non-null  object
 16  Twitter                                 7386 non-null   object
 17  Facebook                                7494 non-null   object
 18  LinkedIn                                8440 non-null   object
 19  Contact Email                           8324 non-null   object
```

[1] Bitacora-Mineria de datos

```
 20  Phone Number                                       6451 non-null   object
 21  Number of Articles                                 5782 non-null   object
 22  Hub Tags                                           433 non-null    object
 23  Full Description                                   9012 non-null   object
 24  Industry Groups                                    10127 non-null  object
 25  Number of Founders                                 7528 non-null   float64
 26  Founders                                           7528 non-null   object
 27  Number of Employees                                9871 non-null   object
 28  Number of Funding Rounds                           7860 non-null   float64
 29  Funding Status                                     6663 non-null   object
 30  Last Funding Date                                  7860 non-null   object
 31  Last Funding Amount                                6647 non-null   float64
 32  Last Funding Amount Currency                       6650 non-null   object
 33  Last Funding Amount Currency (in USD)              6647 non-null   float64
 34  Last Funding Type                                  7860 non-null   object
 35  Last Equity Funding Amount                         6351 non-null   float64
 36  Last Equity Funding Amount Currency                6354 non-null   object
 37  Last Equity Funding Amount Currency (in USD)       6351 non-null   float64
 38  Last Equity Funding Type                           7451 non-null   object
 39  Total Equity Funding Amount                        6957 non-null   float64
 40  Total Equity Funding Amount Currency               6957 non-null   object
 41  Total Equity Funding Amount Currency (in USD)      6957 non-null   float64
 42  Total Funding Amount                               7400 non-null   float64
 43  Total Funding Amount Currency                      7400 non-null   object
 44  Total Funding Amount Currency (in USD)             7400 non-null   float64
 45  Top 5 Investors                                    6861 non-null   object
 46  Number of Lead Investors                           4954 non-null   float64
 47  Number of Investors                                6868 non-null   float64
 48  Number of Acquisitions                             1032 non-null   float64
 49  Acquisition Status                                 1663 non-null   object
 50  IPO Status                                         10259 non-null  object
 51  IPO Date                                           567 non-null    object
 52  Stock Symbol                                       566 non-null    object
 53  Stock Symbol URL                                   567 non-null    object
 54  Stock Exchange                                     565 non-null    object
 55  Last Leadership Hiring Date                        355 non-null    object
 56  Number of Events                                   2297 non-null   float64
 57  CB Rank (Organization)                             10259 non-null  object
 58  BuiltWith - Active Tech Count                      9919 non-null   float64
 59  Apptopia - Number of Apps                          1673 non-null   float64
 60  Apptopia - Downloads Last 30 Days                  1064 non-null   object
 61  G2 Stack - Total Products Active                   4844 non-null   float64
 62  IPqwery - Patents Granted                          3088 non-null   object
 63  IPqwery - Trademarks Registered                    3088 non-null   object
 64  IPqwery - Most Popular Patent Class                1504 non-null   object
 65  IPqwery - Most Popular Trademark Class             2692 non-null   object
 66  Aberdeen - IT Spend                                1098 non-null   float64
 67  Aberdeen - IT Spend Currency                       1145 non-null   object
 68  Aberdeen - IT Spend Currency (in USD)              1098 non-null   float64
 69  Ciudad                                             10259 non-null  object
 70  Departamento                                       10000 non-null  object
 71  Pais                                               10259 non-null  object
dtypes: float64(19), int64(1), object(52)
memory usage: 5.6+ MB
```

- Información básica de los datos



Overview | Warnings 151 | Reproduction

Dataset statistics

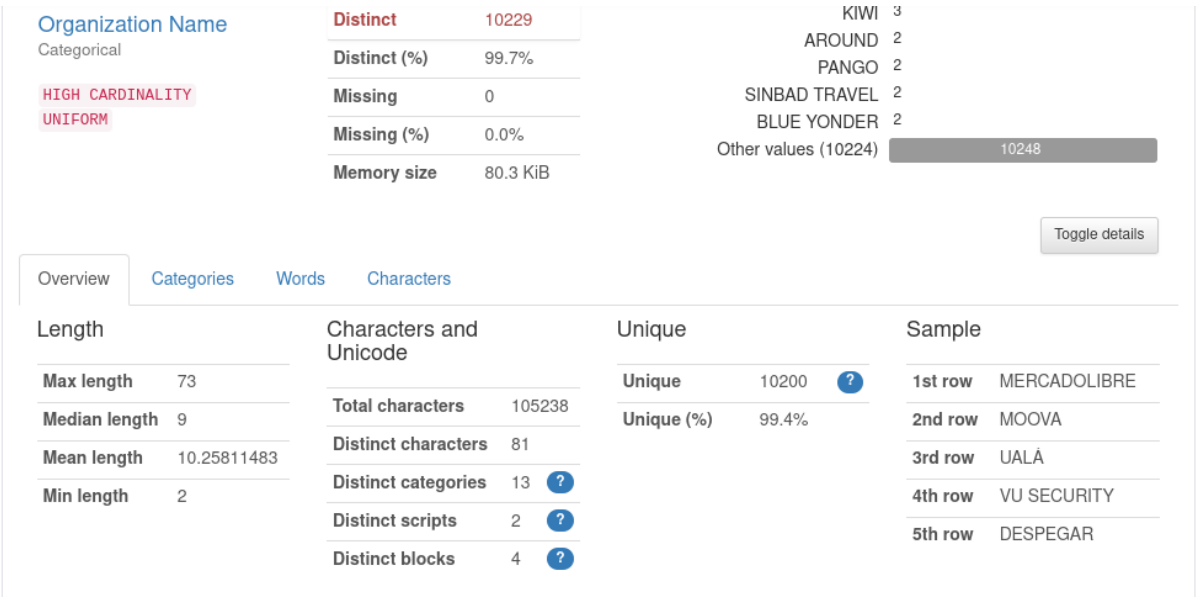
Number of variables	72
Number of observations	10259
Missing cells	283033
Missing cells (%)	38.3%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	5.6 MiB
Average record size in memory	576.0 B

Variable types

Numeric	20
Categorical	52

Aquí encontramos información descriptiva de la unión de los data frame

- Información descriptiva de las variables



Organization Name
Categorical

HIGH CARDINALITY
UNIFORM

Distinct	10229
Distinct (%)	99.7%
Missing	0
Missing (%)	0.0%
Memory size	80.3 KiB

KIWI	3
AROUND	2
PANGO	2
SINBAD TRAVEL	2
BLUE YONDER	2
Other values (10224)	10248

Toggle details

Overview | Categories | Words | Characters

Length

Max length	73
Median length	9
Mean length	10.25811483
Min length	2

Characters and Unicode

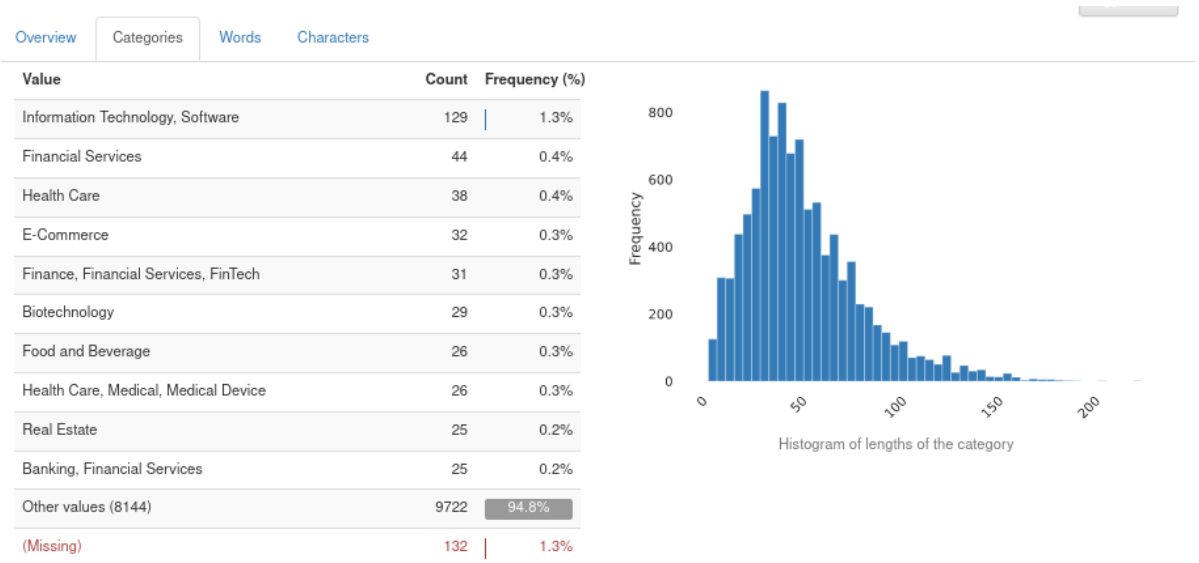
Total characters	105238
Distinct characters	81
Distinct categories	13
Distinct scripts	2
Distinct blocks	4

Unique

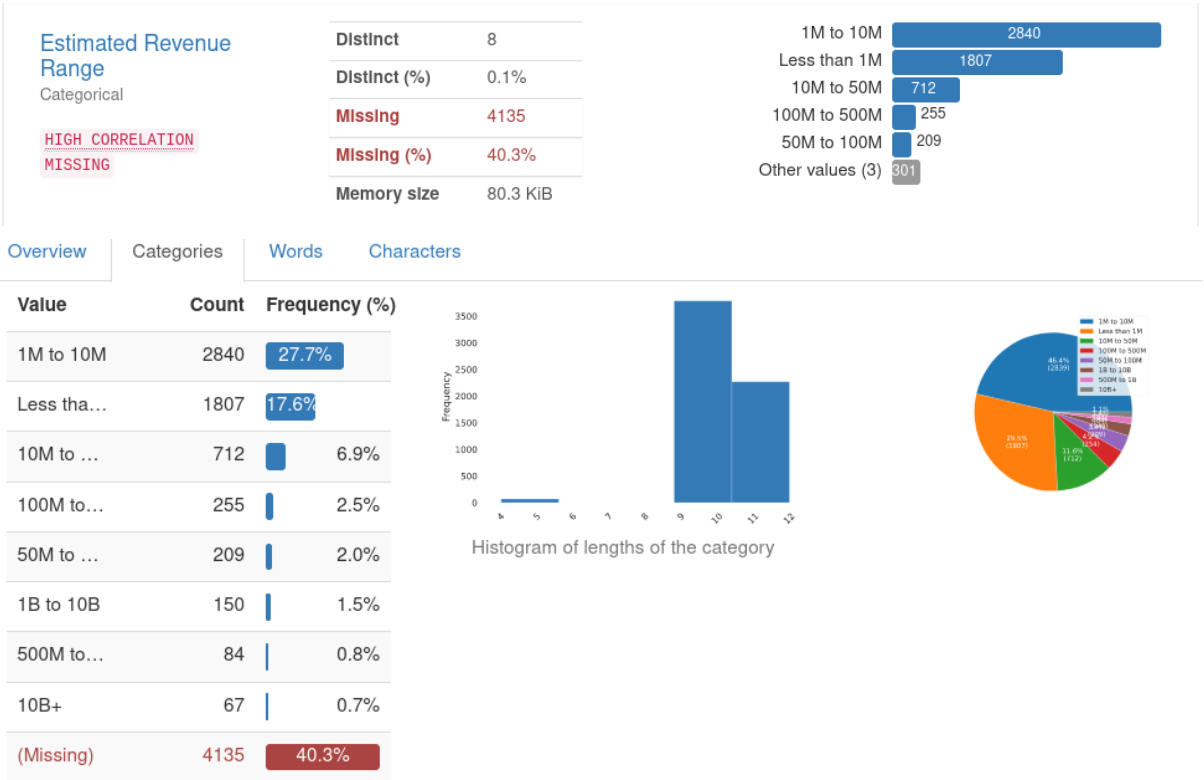
Unique	10200
Unique (%)	99.4%

Sample

1st row	MERCADOLIBRE
2nd row	MOOVA
3rd row	UALÁ
4th row	VU SECURITY
5th row	DESPEGAR



podemos denotar la distribución de [Industries](#) a demás



Función para detectar el porcentaje de NaN por cada columna para los 11 archivos.
Entradas:

- datos, los data frame que se quieren analizar
- **per nulls** cuanto es lo mínimo de porcentaje que es tolerable para el análisis.

para el análisis de estos datos notamos que lo mínimo de consideración es el 50%, pero al nivel de levantamiento de los datos el negocio dejó claro que para ellos lo mínimo es el 70% de los NaN

```
In [3]: df_ar['Number of Events'].isna().sum()/len(df_ar['Number of Events']) # porcentaje de datos nulos en la columna
Out[3]: 0.917
```

```
In [4]: def cols_90per_nulls(data):
            count = 0
            cols_to_drop = {}
            for col in data.columns:
                per_nulls = data[col].isna().sum()/len(data[col])
                if per_nulls >= 0.5:
                    cols_to_drop[col] = per_nulls
                    # print(col, per_nulls)
                    count+=1
                else:
                    None

            print('Number of cols with >50% nulls:', count)
            return cols_to_drop
```

¿Cual es la salida?

nos entrega un análisis por variable con la cantidad en valor del porcentaje del total de NaN.

Out[7]:

	Arg	Bra	Chi	Col	Ger	Isr	Mex	Spa	Swi	Uru	Usa
Estimated Revenue Range	0.543	NaN	0.621	0.603	NaN	NaN	0.604	NaN	NaN	0.559846	NaN
Exit Date	0.900	0.912	0.940	0.938	0.832	0.810	0.906	0.894	0.880	0.930502	0.706
Exit Date Precision	0.900	0.912	0.940	0.938	0.832	0.810	0.906	0.894	0.880	0.930502	0.706
Closed Date	0.985	0.989	0.984	0.989	0.997	0.992	0.992	0.994	0.993	0.992278	0.992
Closed Date Precision	0.971	0.980	0.965	0.978	0.996	0.991	0.980	0.988	0.992	0.984556	0.992
...	...	...	...	...	...	...	...	...	...	...	...
Aberdeen - IT Spend Currency (in USD)	1.000	1.000	1.000	1.000	0.727	1.000	1.000	0.836	0.847	1.000000	NaN
School Method	1.000	0.998	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000000	1.000
Number of Founders	NaN	NaN	NaN	0.564	NaN	NaN	NaN	NaN	NaN	0.621622	NaN
Founders	NaN	NaN	NaN	0.564	NaN	NaN	NaN	NaN	NaN	0.621622	NaN
Headquarters Regions	NaN	NaN	NaN	NaN	NaN	1.000	NaN	NaN	1.000	NaN	NaN

Función para corregir espacios, que recibe:

- la palabra para borrar el espacio en blanco

su salida, es una palabra sin espacios.

```
In [24]: #Funcion que corrige espacios
         def correct_word(word):

             new_word = word.split()[0]
             return new_word
```

Esta función es creada con el fin de encontrar todos los nombres relacionados con LaTaM, y así poder realizar un filtro para hacer el análisis descriptivo que lo relacione, que recibe esta función:

1. df= data frame
2. col= el nombre de la variable donde se debe buscar la palabra

1. Cuánto capital se ha invertido en LaTAM durante el último año. Desagregue gráficamente por país.

Para iniciar a hacer este análisis necesitamos hacer un primer procesamiento de los datos para encontrar la información y el análisis adecuado.

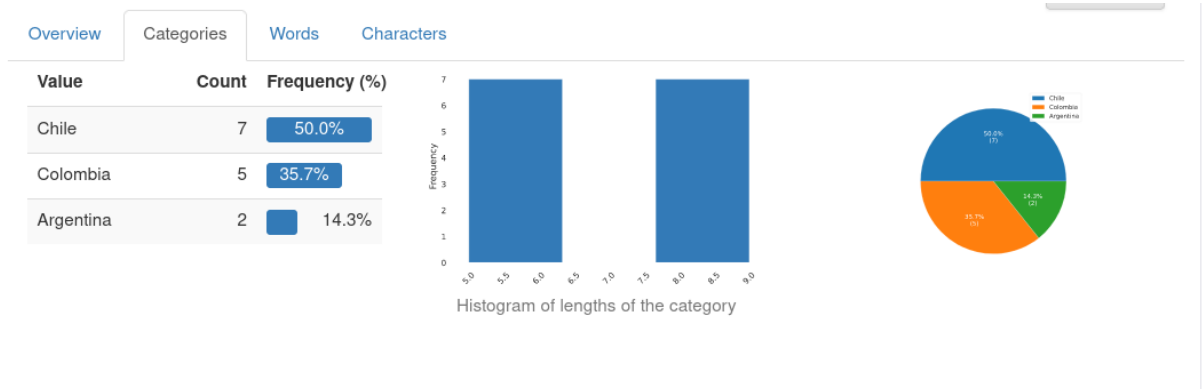
```
def obtener_palabras(df, col):
    lista_palabras = []
    for word in df[col]:
        if 'LATAM' in word:
            lista_palabras.append(word)
        else:
            continue

    return lista_palabras
```

```
list_latam = obtener_palabras(df, 'Organization Name')
#df[df['Organization Name'] in list_latam]
list_latam
```

```
['DTA LATAM',
 'OM LATAM',
 'LATAM BRASIL',
 'CUPONATIC LATAM',
 'LATAMLEAP',
 'LATAM AIRLINES GP',
 'GROUPON LATAM',
 'SCM LATAM',
 'IBR LATAM',
 'OMNI LATAM',
 'SOY STARTUP LATAM',
 'FRANQUICIAS LATAM',
 'MDALATAM UNIVERSITY',
 'INVEST LATAM']
```

De la función anterior encontramos todos los nombres que están relacionados con LaTaM.

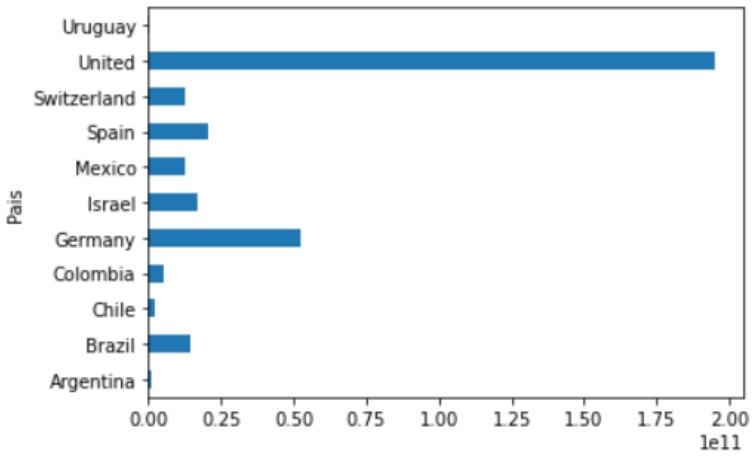
Iniciamos el análisis de cuál es el país que más invierte en LaTaM y análisis descriptivo de cada variable.



2. Haga una comparación entre Colombia con cada uno de los otros países. Analice.

```
df_ciudad=df.groupby(by='Pais')['Last Funding Amount Currency (in USD)'].sum()
df_ciudad[:20].plot(kind='barh')
#plt.xscale('log')
```

```
<AxesSubplot:ylabel='Pais'>
```




```
Pais
Argentina       9.348042e+08
Brazil          1.456192e+10
Chile           2.437126e+09
Colombia        5.049966e+09
Germany         5.265208e+10
Israel          1.715243e+10
Mexico          1.245483e+10
Spain           2.050172e+10
Switzerland     1.259551e+10
United          1.952392e+11
Uruguay         4.873965e+08
Name: Last Funding Amount Currency (in USD), dtype: float64
```

Colombia y Chile están por la misma cantidad de inversión y son de los países con menor inversión.

Análisis descriptivo para para solucionar las preguntas del 3-6

Vamos a crear un filtro

```
[53]: df_colombia=df[(df['Ciudad'] == 'Colombia')]
      df_colombia.head(3)
```

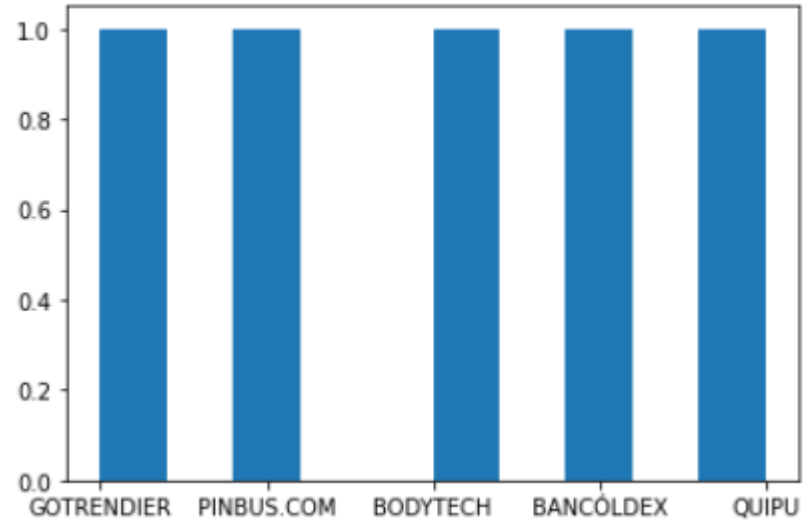
[53]:

	Unnamed: 0	Organization Name	Organization Name URL	Industries	Headquarters Location	Description	CB Rank (Company)	Headquarters Regions	Estimated Revenue Range	Operating Status	...	IPqwery - Patents Granted	IPqw Trader Regis
6055	55	GOTRENDIER	https://www.crunchbase.com /organization/gotren...	E-Commerce, Fashion, Internet	Colombia, Nuevo Leon, Mexico	GoTrendier is a fashionable community.	18,596	Latin America	NaN	Active	...	NaN	
6234	234	PINBUS.COM	https://www.crunchbase.com /organization/pinbus...	Online Portals, Service Industry, Transportation	Colombia, Nuevo Leon, Mexico	Pinbus.com provides an online service to purch...	67,261	Latin America	Less than 1M	Active	...	NaN	
6514	514	BODYTECH	https://www.crunchbase.com /organization/bodytech	Fitness, Health Care, Sports, Wellness	Colombia, Nuevo Leon, Mexico	The second-largest gym company in Latin America	148,726	Latin America	1M to 10M	Active	...	NaN	

3 rows × 72 columns

- Nombre de organizaciones que están en colombia

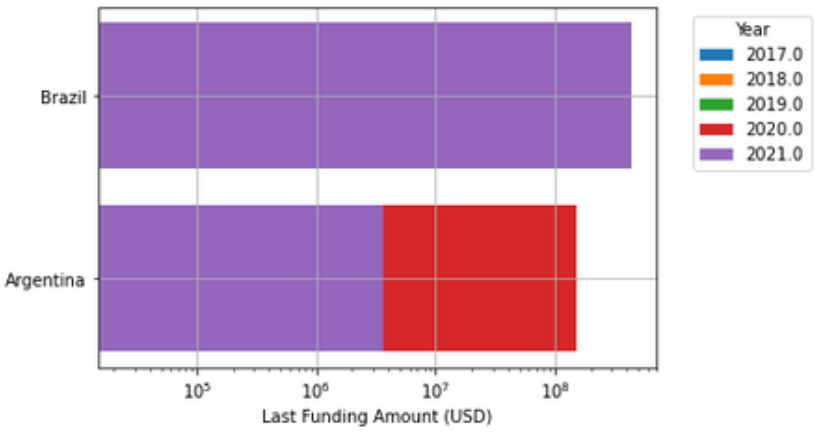
```
plt.hist(df_colombia['Organization Name']);
```



5. Muestre el crecimiento porcentual mensual de ingresos por inversión en Colombia en comparación con los demás países.

[117]:
```
list_years = np.sort(list(df['Year'][:10].unique()))
for year in list_years:

    plt.barh(df[df['Year'] == year]['Pais'][:10],
             df[df['Year'] == year]['Last Funding Amount Currency (in USD)'][:10],
             label=year
            )

plt.legend(bbox_to_anchor=(1.05, 1), title='Year')
plt.xscale('log')
plt.grid()

plt.xlabel('Last Funding Amount (USD)')
```

[117]:
```
Text(0.5, 0, 'Last Funding Amount (USD)')
```



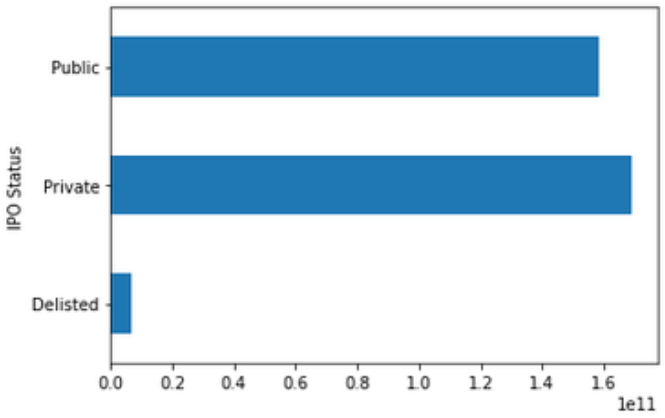

```
df['Pais'].value_counts(normalize=True)
```

```
Spain          0.097475
Brazil         0.097475
Chile          0.097475
Switzerland    0.097475
Colombia       0.097475
Argentina      0.097475
Israel         0.097475
Mexico         0.097475
Germany        0.097475
United         0.097475
Uruguay        0.025246
Name: Pais, dtype: float64
```

4. Muestre gráficamente los exits de capital privado en Colombia por deal size.

```
[125]: df2=df.groupby(by='IPO Status')['Last Funding Amount Currency (in USD)'].sum()
       df2.plot(kind='barh')
```

```
[125]: <AxesSubplot:ylabel='IPO Status'>
```

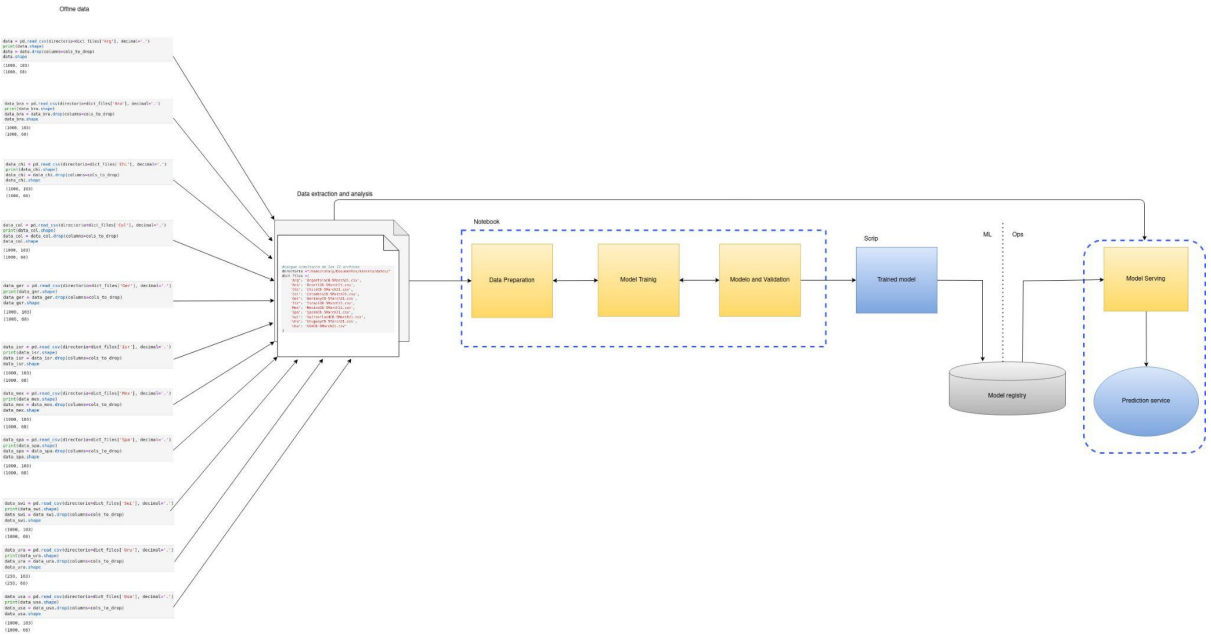


6. ¿De acuerdo con los hallazgos, que le hace falta a Colombia para lograr más inversión?

Notamos que Colombia solo tiene inversión en IPO Status al nivel privado, la inversiones en otros campos, al nivel de latinoamérica vemos como Argentina tiene una historia de inversión muy alineada a las compañías y al nivel de tecnología.

cuando inicie el análisis me concentré en conocer cada variable de forma descriptiva, y ver cual era la que me daba mas informacion al nivel de la información, una que me llamó mucho la atención fue la variable de tipos de servicio, están generando gran impacto en cada país, tanto al nivel privado como publico, asi que mi conclusión es que colombia debe invertir en la transformación digital y todo lo que la rodea.

PUNTO 2.1



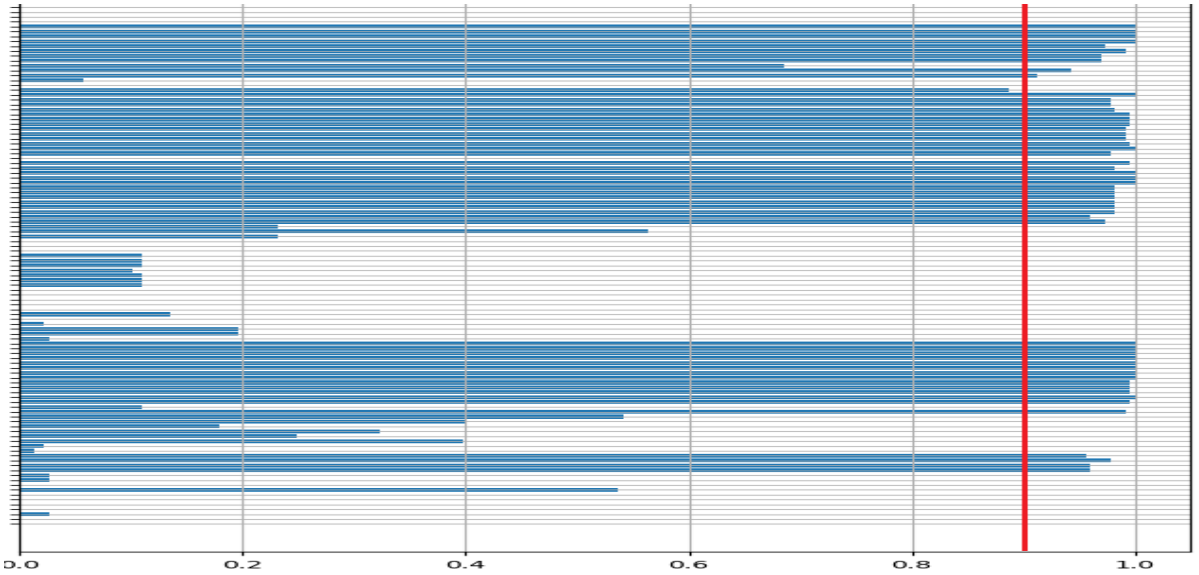
ETL: Procesamiento de extracción, transformación y carga de datos:

paso 0: lectura del set de datos

```
df = pd.read_csv('..\\resources\\ColombiaFlagTop100.csv')
```

paso 1: proceso de limpieza de las columnas de datos con un porcentaje de nulos superior al 90%

```
def percentajeOfEmptyFieldsNumericColumn(dataframe,columnName):
    return df[columnName].isnull().sum() / len(dataframe)

columnNames = []
columnEmptyPercentajes = []

for columnName in df.columns:
    emptyFields = percentajeOfEmptyFieldsNumericColumn(df,columnName)
    columnNames.append(columnName)
    columnEmptyPercentajes.append(emptyFields)

plt.barh(columnNames,columnEmptyPercentajes)
plt.grid()
plt.show()
```



Resumen de limpieza de columnas:

```
LIMIT_EMPTY_PERCENTAJE = 0.9
deletedColumns = 0
for columnName in df.columns:
    emptyFields = percentajeOfEmptyFieldsNumericColumn(df,columnName)
    if emptyFields >= LIMIT_EMPTY_PERCENTAJE :
        del df[columnName]
        deletedColumns+=1
print("Deleted columns: %d, columns remaining: %d"%(deletedColumns,len(df.columns)))
```

```
Deleted columns: 54, columns remaining: 53
```

paso 2: Los campos de rangos de valores se dividen en dos columnas independientes, donde se tendrá el máximo y mínimo como valores numéricos

Campo: “Estimated Revenue Range”

```
def convertRangeValue(inputValue):
    value = inputValue.strip()
    if(value=="$1M"):
        return 1000000
    elif(value=="$10M"):
        return 10000000
    elif(value=="$50M"):
        return 50000000
    elif(value=="$100M"):
        return 100000000
    elif(value=="$500M"):
        return 500000000
    elif(value=="$1B"):
        return 1000000000
    elif(value=="$10B"):
        return 10000000000
    else:
        print("OTRO",inputValue)
```

```
columnName = "Estimated Revenue Range"
#Estimated Revenue Range
df["minEstimatedRevenueRange"] = np.zeros(df.shape[0])
df["maxEstimatedRevenueRange"] = np.zeros(df.shape[0])

for index in range(len(df)):
    content = df.iloc[index][columnName]
    if pd.isnull(df.iloc[index][columnName]):
        df.iloc[index, df.columns.get_loc('minEstimatedRevenueRange')] = 0
        df.iloc[index, df.columns.get_loc('maxEstimatedRevenueRange')] = 0
        continue

    if(content=="Less than $1M"):
        minValue = 0
        maxValue = 1000000
    else:
        contentSplitted = content.split("to")
        minValue = convertRangeValue(contentSplitted[0])
        maxValue = convertRangeValue(contentSplitted[1])
    df.iloc[index, df.columns.get_loc('minEstimatedRevenueRange')] = minValue
    df.iloc[index, df.columns.get_loc('maxEstimatedRevenueRange')] = maxValue

del df[columnName]
```

Campo: “Number of Employees”

```
columnName = "Number of Employees"
#Estimated Revenue Range
df["minNumberEmployees"] = np.zeros(df.shape[0])
df["maxNumberEmployees"] = np.zeros(df.shape[0])

for index in range(len(df)):
    content = df.iloc[index][columnName]
    if pd.isnull(df.iloc[index][columnName]):
        df.iloc[index, df.columns.get_loc('minNumberEmployees')] = 0
        df.iloc[index, df.columns.get_loc('maxNumberEmployees')] = 0
        continue
    if(content=="01-Oct"):
        df.iloc[index, df.columns.get_loc('minNumberEmployees')] = 1000
        df.iloc[index, df.columns.get_loc('maxNumberEmployees')] = 2000
        continue
    if(content=="Nov-50"):
        df.iloc[index, df.columns.get_loc('minNumberEmployees')] = 2000
        df.iloc[index, df.columns.get_loc('maxNumberEmployees')] = 5000
        continue
    if(content=="10001+"):
        df.iloc[index, df.columns.get_loc('minNumberEmployees')] = 10000
        df.iloc[index, df.columns.get_loc('maxNumberEmployees')] = 99999
        continue
    contentSplitted = content.split("-")
    minValue = int(contentSplitted[0].strip())
    maxValue = int(contentSplitted[1].strip())
    df.iloc[index, df.columns.get_loc('minNumberEmployees')] = minValue
    df.iloc[index, df.columns.get_loc('maxNumberEmployees')] = maxValue

del df[columnName]
```

Paso 3: Se refactoriza campos de contención de atributos booleanos que felicitan su procesamiento, al indicar si son o no son de un tipo específico, y si estos tienen o no un valor.

```
df["isCompanyTypeForProfit"] = np.zeros(df.shape[0])
df["Company Type"] = df["Company Type"].fillna("")
for index in range(len(df)):
    content = df.iloc[index]["Company Type"]
    if(content=="For Profit"):
        df.iloc[index, df.columns.get_loc("isCompanyTypeForProfit")] = 1

df["isCompanyPrivate"] = np.zeros(df.shape[0])
df["IPO Status"] = df["IPO Status"].fillna("")
for index in range(len(df)):
    content = df.iloc[index]["IPO Status"]
    if(content=="Private"):
        df.iloc[index, df.columns.get_loc("isCompanyPrivate")] = 1

df["hasWebsite"] = np.zeros(df.shape[0])
df["Website"] = df["Website"].fillna("")
for index in range(len(df)):
    content = df.iloc[index]["Website"]
    if(content!=""):
        df.iloc[index, df.columns.get_loc("hasWebsite")] = 1

df["hasTwitter"] = np.zeros(df.shape[0])
df["Twitter"] = df["Twitter"].fillna("")
for index in range(len(df)):
    content = df.iloc[index]["Twitter"]
    if(content!=""):
        df.iloc[index, df.columns.get_loc("hasTwitter")] = 1
```

```
df["hasLinkedIn"] = np.zeros(df.shape[0])
df["LinkedIn"] = df["LinkedIn"].fillna("")
for index in range(len(df)):
    content = df.iloc[index]["LinkedIn"]
    if(content!=""):
        df.iloc[index, df.columns.get_loc("hasLinkedIn")] = 1

df["hasContactEmail"] = np.zeros(df.shape[0])
df["Contact Email"] = df["Contact Email"].fillna("")
for index in range(len(df)):
    content = df.iloc[index]["Contact Email"]
    if(content!=""):
        df.iloc[index, df.columns.get_loc("hasContactEmail")] = 1

df["hasPhoneNumber"] = np.zeros(df.shape[0])
df["Phone Number"] = df["Phone Number"].fillna("")
for index in range(len(df)):
    content = df.iloc[index]["Phone Number"]
    if(content!=""):
        df.iloc[index, df.columns.get_loc("hasPhoneNumber")] = 1
```

```
del df["Company Type"]
del df["Website"]
del df["Twitter"]
del df["Facebook"]
del df["LinkedIn"]
del df["Contact Email"]
del df["Phone Number"]
del df["IPO Status"]
```

Paso 4: Se reemplazan columnas de campos numericos con tontenidos nulos, y se llenan con zeros “0”

```
numericColumnsNames = [
    "CB Rank (Company)",
    "Number of Articles",
    "Number of Founders",
    "Number of Funding Rounds",
    "Last Funding Amount",
    "Last Funding Amount Currency (in USD)",
    "Last Equity Funding Amount",
    "Last Equity Funding Amount Currency (in USD)",
    "Total Equity Funding Amount",
    "Total Equity Funding Amount Currency (in USD)",
    "Total Funding Amount",
    "Total Funding Amount Currency (in USD)",
    "Number of Lead Investors",
    "Number of Investors",
    "Number of Events",
    "CB Rank (Organization)",
    "BuiltWith - Active Tech Count",
    "G2 Stack - Total Products Active"
]

for columnName in numericColumnsNames:
    df[columnName] = df[columnName].fillna(0)
```

Paso 5: Convertir valores flotantes en formato de comas “,” a punto flotante

```
for index in range(len(df)):
    content = df.iloc[index]["CB Rank (Company)"]
    contentSplitted = content.split(",")
    buildedStringNumber = contentSplitted[0] + "." + contentSplitted[1]
    df.iloc[index, df.columns.get_loc("CB Rank (Company)")] = float(buildedStringNumber)

for index in range(len(df)):
    content = df.iloc[index]["CB Rank (Organization)"]
    contentSplitted = content.split(",")
    buildedStringNumber = contentSplitted[0] + "." + contentSplitted[1]
    df.iloc[index, df.columns.get_loc("CB Rank (Organization)")] = float(buildedStringNumber)
```

Paso 6: Carga de datos en un archivo CSV, el cual se guardará en la carpeta local del repositorio GIT

```
df.to_csv('ETL-out.csv')
```

PUNTO 2.2

Se realiza un análisis de componentes principales, con el fin de identificar la viabilidad en la segmentación entre los grupos de empresas, si esta se encuentra en el top 100 o no, mediante las entradas de valores de las variables del set de datos.

Paso 1: Cargar librerías

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

Paso 2: Dividir el set de datos en dos data frames independientes, teniendo en mente los valores que intentarán predecir si un input está en el top 100 o no lo esta

```
columns_X = [
    "CB Rank (Company)",
    "Number of Articles",
    "Number of Founders",
    "Number of Funding Rounds",
    "Last Funding Amount Currency (in USD)",
    "Last Equity Funding Amount Currency (in USD)",
    "Total Equity Funding Amount Currency (in USD)",
    "Total Funding Amount Currency (in USD)",
    "Number of Lead Investors",
    "Number of Investors",
    "Number of Events",
    "BuiltWith - Active Tech Count",
    "G2 Stack - Total Products Active",
    "minEstimatedRevenueRange",
    "maxEstimatedRevenueRange",
    "minNumberEmployees",
    "maxNumberEmployees",
    "isCompanyTypeForProfit",
    "isCompanyPrivate",
    "hasWebsite",
    "hasTwitter",
    "hasFacebook",
    "hasLinkedIn",
    "hasContactEmail",
    "hasPhoneNumber",
    "isOperating"
]

X = pd.read_csv('.\\ETL-out.csv')
for column in X.columns:
    if (not column in columns_X ):
        del X[column]
```

```
columns_y = ["isInTop100"]

y = pd.read_csv('.\\ETL-out.csv')
for column in y.columns:
    if (not column in columns_y ):
        del y[column]
```

Paso 3: Normalización de los datos

```
In [7]: X_std = StandardScaler().fit_transform(X)
        print('NumPy covariance matrix: \n%s' %np.cov(X_std.T))
```

```
NumPy covariance matrix:
[[ 1.00438596e+00 -2.11156754e-01 -1.43730342e-01 -3.32805283e-01
  -8.02292062e-02 -8.22356536e-02 -9.65216552e-02 -1.03842498e-01
  -3.56678802e-01 -2.78995262e-01 -1.14998906e-01 -4.02030028e-01
  -1.62577893e-02 -7.78712116e-02 -6.93291065e-02 -1.06416093e-01
  -8.02612925e-02  7.55587646e-02  4.46427110e-03 -3.16944405e-01
   8.72646526e-02 -1.06679724e-01 -4.51783278e-01 -1.49436247e-01
  -3.64969443e-02 -2.75104043e-01]
 [-2.11156754e-01  1.00438596e+00  1.53488870e-01  4.02691551e-01
   4.45435834e-01  2.27045931e-01  6.63445049e-01  7.44253285e-01
   5.94477664e-01  5.83520626e-01  2.41758771e-01  2.00728330e-01
   3.31240551e-01 -2.06304928e-04 -2.10462872e-02  3.06203921e-01
   2.74240177e-01  2.77653056e-02 -1.84637271e-01  5.53590647e-02
   8.47168752e-02  8.11627232e-02  1.59254627e-01  7.56520474e-02
  -1.03684613e-01  7.64099972e-02]
 [-1.43730342e-01  1.53488870e-01  1.00438596e+00  3.19939333e-01
  -1.25668184e-01 -9.27451780e-02 -2.45826873e-03 -6.84368526e-02
   1.99863524e-01  2.25113257e-01  2.07895829e-02  8.75087955e-02
   1.05601788e-01 -4.14159466e-02 -7.73287015e-02 -6.27403460e-02
  -8.02990417e-02  1.51365663e-01  1.03843118e-01  5.76233436e-02
```

Paso 4: Calcular los autovectores y autovalores

```
In [8]: cov_mat = np.cov(X_std.T)

        eig_vals, eig_vecs = np.linalg.eig(cov_mat)

        print('Eigenvectors \n%s' %eig_vecs)
        print('\nEigenvalues \n%s' %eig_vals)
```

```
Eigenvectors
[[-1.75359657e-01 -2.22497369e-01 -1.55346717e-01  1.67807117e-01
   3.45530708e-01 -6.82032912e-02 -1.71510046e-03  4.31490068e-03
   4.99627238e-02  3.09313877e-02 -2.36229711e-01  3.46136650e-02
  -8.27202558e-02 -2.01113353e-01 -1.65538884e-01 -5.62846988e-02
  -9.57596533e-02 -3.82551831e-01 -5.76098513e-02  4.98794505e-02
  -5.02716300e-01 -2.04614601e-01 -1.54525256e-01  2.90765376e-01
   1.94787146e-01  5.28523501e-02]
 [ 3.73857895e-01  5.51072829e-02 -1.13041137e-01  3.51970160e-02
   7.76206125e-02  1.42462161e-01 -5.87004664e-03 -8.11026756e-02
   1.71508161e-01  5.33442839e-01 -8.41627309e-02 -3.62546293e-02
  -1.47760857e-01  3.13286666e-01  2.85110269e-02  1.12918206e-01
   3.29660458e-01  2.87943661e-01  2.12043332e-02 -2.93834468e-02
  -2.53851116e-01 -2.10330154e-01 -1.60570937e-01  9.65355371e-02
  -9.47216435e-02 -9.31152794e-02]
 [ 5.56929732e-02  2.78045063e-01  5.08915621e-02 -1.23398566e-02
   2.70984312e-01  1.14177943e-01 -4.25656523e-03 -2.82634247e-02
  -3.13912227e-02 -5.18929852e-02  2.07183602e-01 -5.78086175e-02
   1.78805641e-01  3.00924260e-02  2.32844152e-01  1.81356874e-01
  -4.98122375e-02  5.23451101e-02 -5.12921901e-01  4.39474384e-01
```

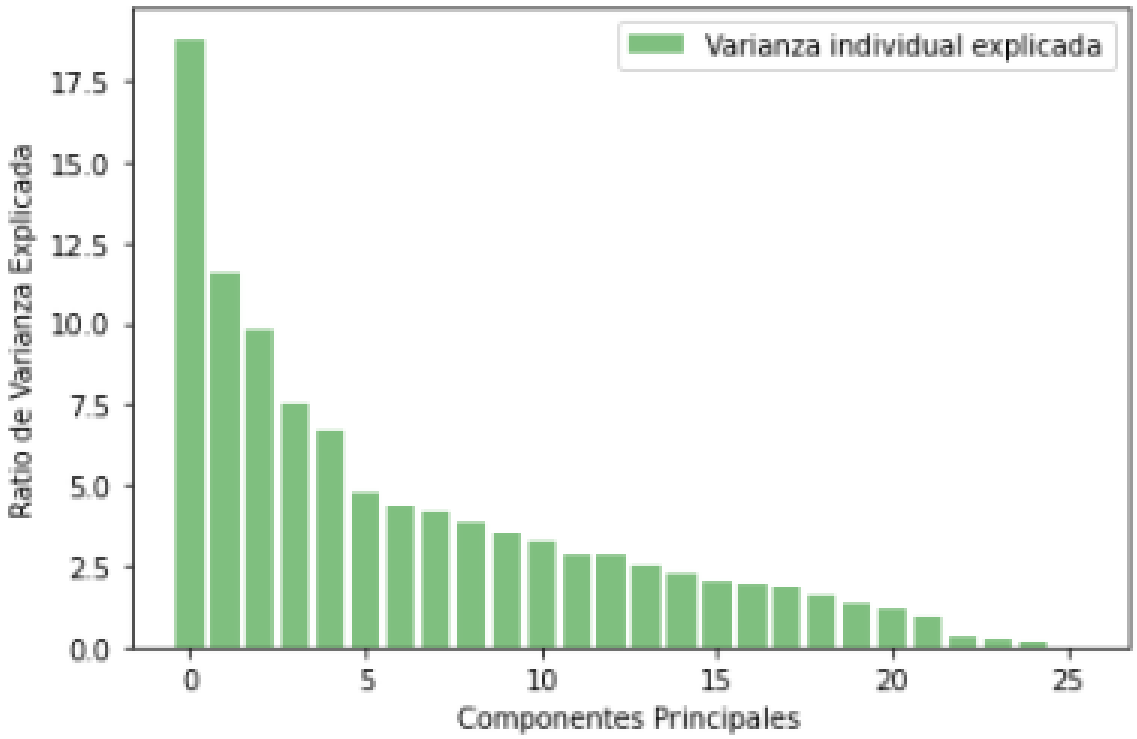
```
#  Hacemos una lista de parejas (autovector, autovalor)
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]

# Ordenamos estas parejas den orden descendiente con la función sort
eig_pairs.sort(key=lambda x: x[0], reverse=True)

# Visualizamos la lista de autovalores en orden desdenciente
print('Autovalores en orden descendiente:')
for i in eig_pairs:
    print(i[0])
```

```
Autovalores en orden descendiente:
4.912245750429123
3.0201922150701974
2.5502945378034636
1.969536169238358
1.7398580222523303
1.2546216086426745
1.1270109561812742
1.0936117108648045
0.9987229604633341
0.9108858284347017
0.8510344617635391
0.7449555053936743
0.7331995594865258
0.6455660670288373
0.5869022551758594
0.5123039373932055
0.5009543384323201
0.48570335600248954
0.4115936400921066
0.3493964105869995
0.3100625396577722
0.23473645162199175
0.07919950582824664
0.05938023436048195
0.031314844291713974
0.000752221223259712
```

Paso 5: Seleccionar los autovectores correspondientes a las componentes principales y graficar

```
# A partir de los autovalores, calculamos la varianza explicada
tot = sum(eig_vals)
var_exp = [(i / tot)*100 for i in sorted(eig_vals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)

tamanio = len(columns_X)

# Representamos en un diagrama de barras la varianza explicada por cada autovalor, y la acumulada
with plt.style.context('seaborn-pastel'):
    plt.bar(np.arange(0,tamanio), var_exp, alpha=0.5, align='center', label='Varianza individual explicada', color='g')
    plt.ylabel('Ratio de Varianza Explicada')
    plt.xlabel('Componentes Principales')
    plt.legend(loc='best')
    plt.tight_layout()
```

En la siguiente gráfica se evidencia que la mayor parte de la varianza se encuentra correspondida en la primera componente

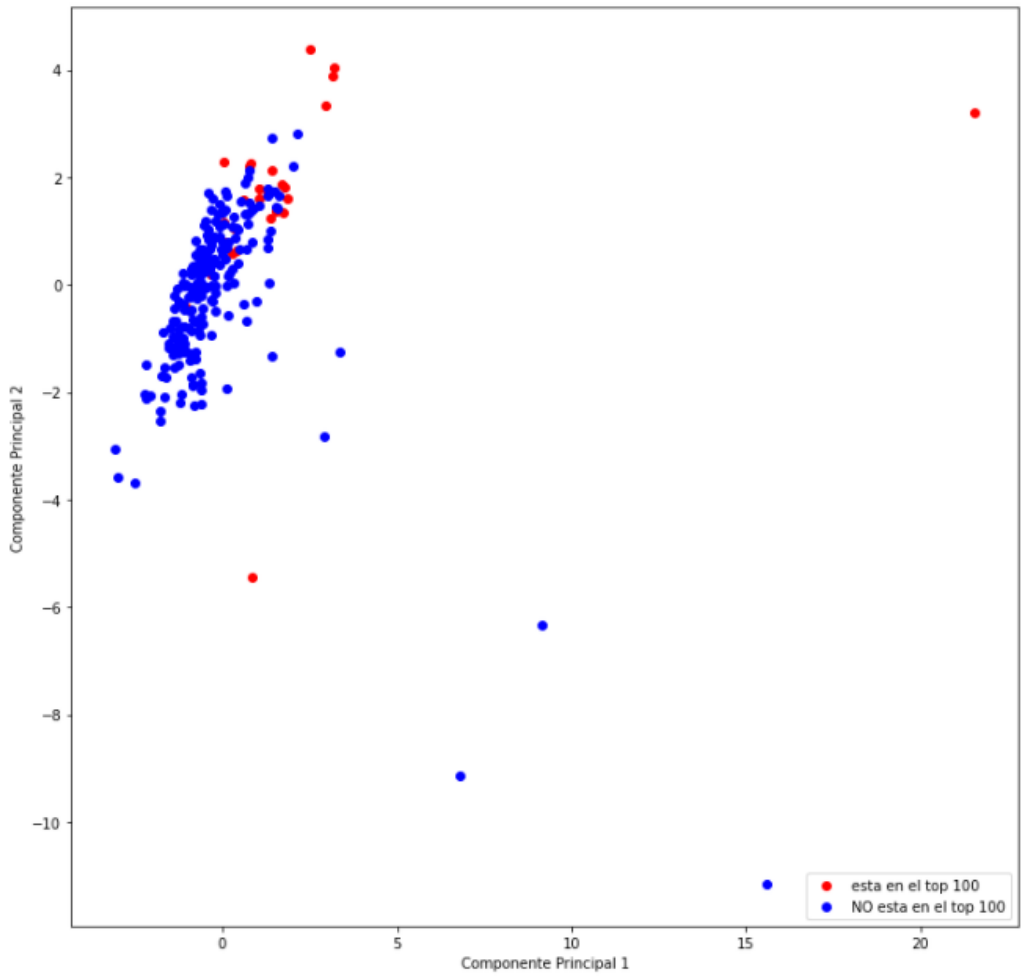


Paso 6: Proyectar en un espacio de dimensionalidad de 2D

```
#Generamos la matríz a partir de los pares autovalor-autovector
matrix_w = np.hstack((eig_pairs[0][1].reshape(tamanio,1),
                      eig_pairs[1][1].reshape(tamanio,1)))

print('Matriz W:\n', matrix_w)

Y = X_std.dot(matrix_w)
```

```
plt.rcParams["figure.figsize"]=12,12

X_values_1 = []
Y_values_1 = []

X_values_0 = []
Y_values_0 = []

for i in range(len(Y)):
    if(y.iloc[i]["isInTop100"]==1):
        X_values_1.append(Y[i][0])
        Y_values_1.append(Y[i][1])
    else:
        X_values_0.append(Y[i][0])
        Y_values_0.append(Y[i][1])

plt.scatter(X_values_1,Y_values_1,color="r",label="esta en el top 100")
plt.scatter(X_values_0,Y_values_0,color="b",label="NO esta en el top 100")
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.legend(loc="lower right")
```

Obteniendo la siguiente representación de valores:



Donde se visualiza que con los valores de entrada es viable realizar una segmentación de los mismos con un posible predictor de empresas.

PUNTO 2.3

A continuación se realiza una regresión lineal mediante el método de “Least squares”, el cual permitirá indicar según los valores de entrada de una empresa si esta puede entrar en el top 100 de empresas financiadas en colombia, en una escala de valores entre y no limitados a 0 y 1.

Para hacer la regresión lineal se utiliza R con el siguiente algoritmo:

C:\Users\admin\Desktop\R\componentes.R - R Editor

```
file = "
      ..\\MMA_cursoMetodosAnalisisDatos
      \\Guillermo De Mendoza
      \\Mineria de datos - semestre 4
      \\Tarea4\\Plantilla.txt"
datos<-read.table(file, header=T, row.names=1)
rec = glm(isInTop100~.,family="binomial", data=datos)
rec
```

Desde la consola imprimimos los coeficientes:

```
> rec

Call:  glm(formula = isInTop100 ~ ., family = "binomial", data = datos)

Coefficients:
                                  (Intercept)      minEstimatedRevenueRange
                                    -8.318e+15                    -1.065e+07
                               NumberofArticles      maxEstimatedRevenueRange
                                     1.035e+13                     1.520e+06
                               NumberofFounders           minNumberEmployees
                                     7.974e+13                     1.269e+11
                           NumberofFundingRounds           maxNumberEmployees
                                     4.432e+13                    -9.499e+10
                LastFundingAmountCurrency.inUSD.       isCompanyTypeForProfit
                                     1.814e+07                     2.235e+15
 LastEquityFundingAmountCurrency.inUSD.              isCompanyPrivate
                                    -2.272e+07                     3.121e+15
TotalEquityFundingAmountCurrency.inUSD.                    hasWebsite
                                     1.612e+07                     1.726e+15
      TotalFundingAmountCurrency.inUSD.                    hasTwitter
                                    -1.523e+07                    -2.841e+14
                         NumberofLeadInvestors                   hasFacebook
                                     1.950e+14                     1.535e+14
                            NumberofInvestors                   hasLinkedIn
                                     5.363e+13                    -8.461e+13
                               NumberofEvents                hasContactEmail
                                     5.336e+13                     2.420e+14
                         CBRank.Organization.                 hasPhoneNumber
                                    -7.560e+12                     1.643e+13
                    BuiltWith.ActiveTechCount                    isOperating
                                     6.765e+12                     3.395e+14
                  G2Stack.TotalProductsActive
                                    -2.587e+12
```

R Console

```
Degrees of Freedom: 228 Total (i.e. Null);  202 Residual
Null Deviance:      166.1
Residual Deviance: 1514         AIC: 1568
```

Detalles:

```
> summary(rec)

Call:
glm(formula = isInTop100 ~ ., family = "binomial", data = datos)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
 -8.49    0.00    0.00    0.00    8.49

Coefficients:
                                        Estimate Std. Error    z value
(Intercept)                           -8.318e+15  6.958e+07 -119534933
NumberofArticles                       1.035e+13  1.377e+06    7515182
NumberofFounders                       7.974e+13  4.004e+06   19917156
NumberofFundingRounds                  4.432e+13  3.869e+06   11453253
LastFundingAmountCurrency.inUSD.       1.814e+07  5.609e-01   32336220
LastEquityFundingAmountCurrency.inUSD. -2.272e+07  5.783e-01  -39282565
TotalEquityFundingAmountCurrency.inUSD. 1.612e+07  5.652e-01   28516804
TotalFundingAmountCurrency.inUSD.      -1.523e+07  5.562e-01  -27372977
NumberofLeadInvestors                  1.950e+14  7.681e+06   25382912
NumberofInvestors                      5.363e+13  1.829e+06   29327240
NumberofEvents                         5.336e+13  6.515e+06    8190902
CBRank.Organization.                   -7.560e+12  6.425e+04 -117662957
BuiltWith.ActiveTechCount              6.765e+12  2.693e+05   25120818
G2Stack.TotalProductsActive            -2.587e+12  6.668e+05   -3879473
minEstimatedRevenueRange               -1.065e+07  1.750e-01  -60848875
maxEstimatedRevenueRange               1.520e+06  1.946e-02   78128585
minNumberEmployees                     1.269e+11  9.235e+03   13745469
maxNumberEmployees                     -9.499e+10  1.525e+03  -62296785
isCompanyTypeForProfit                 2.235e+15  4.142e+07   53945310
isCompanyPrivate                       3.121e+15  3.742e+07   83409238
hasWebsite                             1.726e+15  3.427e+07   50349045
hasTwitter                             -2.841e+14  1.078e+07  -26357519
hasFacebook                            1.535e+14  1.230e+07   12482453
hasLinkedIn                            -8.461e+13  1.196e+07   -7071993
hasContactEmail                        2.420e+14  1.390e+07   17416297
hasPhoneNumber                         1.643e+13  1.086e+07    1513523
isOperating                            3.395e+14  2.378e+07   14279982
```

Analisis de variables en la regresión:

```
(Intercept)                                <2e-16 ***
NumberofArticles                           <2e-16 ***
NumberofFounders                           <2e-16 ***
NumberofFundingRounds                      <2e-16 ***
LastFundingAmountCurrency.inUSD.           <2e-16 ***
LastEquityFundingAmountCurrency.inUSD.     <2e-16 ***
TotalEquityFundingAmountCurrency.inUSD.    <2e-16 ***
TotalFundingAmountCurrency.inUSD.          <2e-16 ***
NumberofLeadInvestors                      <2e-16 ***
NumberofInvestors                          <2e-16 ***
NumberofEvents                             <2e-16 ***
CBRank.Organization.                       <2e-16 ***
BuiltWith.ActiveTechCount                  <2e-16 ***
G2Stack.TotalProductsActive                <2e-16 ***
minEstimatedRevenueRange                   <2e-16 ***
maxEstimatedRevenueRange                   <2e-16 ***
minNumberEmployees                         <2e-16 ***
maxNumberEmployees                         <2e-16 ***
isCompanyTypeForProfit                     <2e-16 ***
isCompanyPrivate                           <2e-16 ***
hasWebsite                                 <2e-16 ***
hasTwitter                                 <2e-16 ***
hasFacebook                                <2e-16 ***
hasLinkedIn                                <2e-16 ***
hasContactEmail                            <2e-16 ***
hasPhoneNumber                             <2e-16 ***
isOperating                                <2e-16 ***
```

Con lo cual concluimos que no se debe descartar ninguna de las variables de entrada en la regresión

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance:  166.13  on 228  degrees of freedom
Residual deviance: 1513.83  on 202  degrees of freedom
AIC: 1567.8

Number of Fisher Scoring iterations: 18
```