Proyecto del Primer y Segundo Parcial

El tema del proyecto es sobre:

Análisis Predictivo mediante el modelo de regresión lineal - regresión logística para evaluar el ROE/ROA en las empresas del Ecuador durante los años 2017-2020

1. Los datos de los indicadores se bajan aquí.
2. Crear el dataset con Pandas.
3. Tome como referencia los artículos adjuntos
4. Revisar el estado del arte y trabajos similares en: Google Scholar/Académico, DOAJ.org, Scopus, Science Direct, Web of Science.
5. Haga el análisis exploratorio correlacional en Python del año 2019
6. Crear el Modelo predictivo de regresión lineal ROE y regresión logística del ROA en pytorch.
7. Evaluar los dos modelos lineal y logístico con los años 2020, 2018 y 2017. Note que ya no debe entrenar, solo predecir con los modelos creados en el paso 7
8. Contestar las siguientes preguntas

    - ¿Qué año predice mejor y peor? Crear una tabla para comparar los resultados
    - ¿Qué categoría de empresas predice mejor y peor? Crear una tabla para comparar los resultados.

Subir TODOS los datos, scripts y documento a la carpeta proyecto-1P . Subir los script e informes a esta tarea.

Grupo # 2

> **Tema:** - Análisis Predictivo mediante el modelo de regresión lineal - regresión logística para evaluar el ROE/ROA en las empresas del Ecuador durante los años 2017-2020
>
> **Integrantes:**
>
> - Apolo Baldeon Erick Tomas
> - Mora Ferruzola Violeta Nicolle
> - Ronquillo Lamilla Cristian José
> - Tigreros Peña Dennys Ariel
> - Tola Molina Pedro Salvador

Información General

Análisis Predictivo mediante el modelo de regresión lineal - regresión logística para evaluar el ROE/ROA en las empresas del Ecuador durante los años 2017-2020

Sector: Todos

Descripción:

Pregunta 1: ¿Qué año predice mejor y peor los modelos creado?

Pregunta 2: ¿Qué categoría de empresas predice mejor y peor los modelos creados?

> **Fuente:** Datos extraidos de la Superintendencia de Bancos (SuperCIAS)
>
> **Datos Extraidos:**
>
> 1. Indicadores Financieros (2017 - 2020)
>
> - **Link:** https://reporteria.supercias.gob.ec/portal/cgi-bin/
> - **Ultima Actualización:** 07/01/2021 12:59:23
>
> 2. Directorio de empresas
>
> - **Link:** https://mercadodevalores.supercias.gob.ec/reportes/directorioCompanias.jsf

- **Ultima Actualización:** 11/07/2022 04:45:31

Carga de Datos

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import torch
6 import glob
7 import os
8
9 from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, accuracy_score
```

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
1 df = pd.read_excel('/content/drive/MyDrive/Inteligencia Artificial/Segundo Parcial/Proyecto/indicadores2019_cia.xlsx')
2 df.sample(5)
```

	AÑO	EXPEDIENTE	NOMBRE	RAMA	DESCRIPCIÓN RAMA	RAMA 6 DÍGITOS	SUBRAMA 2 DÍGITOS	LIQUIDEZ CORRIENTE	PRUEBA ÁCIDA	ENDEUDAMIENTO DEL ACTIVO	...	IMPACTO DE LA CARGA FINANCIERA	RENTABILIDA NETA DI ACTIV
49627	2019	177014	PROYJACONST S. A.	F	CONSTRUCCIÃ“N.	F4290.91	F42	0.000000	0.000000	1.000000	...	0.0	0.10357
37052	2019	149895	SUASCENCORP S.A.	G	COMERCIO AL POR MAYOR Y AL POR MENOR REPARACIÃ...	G4641.21	G46	NaN	NaN	0.000000	...	0.0	0.00195
32414	2019	140751	DISTRIBUIDORA RAMOS PADILLA S.A.	G	COMERCIO AL POR MAYOR Y AL POR MENOR REPARACIÃ...	G4659.11	G46	4.942937	4.942937	0.202309	...	0.0	0.02353
78543	2019	718220	SERVICIOS FUNERALES ASMIN SERVICIOSNOAS S.A.	S	OTRAS ACTIVIDADES DE SERVICIOS.	S9603.01	S96	0.340891	0.340891	2.170825	...	0.0	-1.10870
22602	2019	110569	IMPULSO PROYECCIONES CIA. LTDA. C. IMPYPRO	G	COMERCIO AL POR MAYOR Y AL POR MENOR REPARACIÃ...	G4711.01	G47	NaN	NaN	NaN	...	NaN	Na

5 rows × 37 columns

Limpieza de Datos

```
1 print("Las dimensiones del DataFrame son: {}".format(df.shape))
```

```
Las dimensiones del DataFrame son: (85793, 37)
```

```
1 df = df.dropna(thresh=37)
2 df.sample(5)
```

	AÑO	EXPEDIENTE	NOMBRE	RAMA	DESCRIPCIÓN RAMA	RAMA 6 DÍGITOS	SUBRAMA 2 DÍGITOS	LIQUIDEZ CORRIENTE	PRUEBA ÁCIDA	ENDEUDAMIENTO DEL ACTIVO	...	IMPACTO DE LA CARGA FINANCIERA	RENTABILI NETA ACT
25839	2019	122965	IMPORTADORA Y EXPORTADORA	G	COMERCIO AL POR MAYOR Y AL POR MENOR	G4641.11	G46	1.149023	0.653302	0.720448	...	0.014355	0.121

```
df.isnull().any()
```

```
AÑO                                   False
EXPEDIENTE                            False
NOMBRE                                False
RAMA                                  False
DESCRIPCIÓN RAMA                      False
RAMA 6 DÍGITOS                        False
SUBRAMA 2 DÍGITOS                     False
LIQUIDEZ CORRIENTE                    False
PRUEBA ÁCIDA                          False
ENDEUDAMIENTO DEL ACTIVO              False
ENDEUDAMIENTO PATRIMONIAL             False
ENDEUDAMIENTO A CORTO PLAZO           False
ENDEUDAMIENTO A LARGO PLAZO           False
COBERTURA DE INTERESES                False
ENDEUDAMIENTO DEL ACTIVO FIJO         False
APALANCAMIENTO                        False
APALANCAMIENTO FINANCIERO             False
FORTALEZA PATRIMONIAL                 False
ENDEUDAMIENTO PATRIMONIAL CORRIENTE   False
ENDEUDAMIENTO PATRIMONIAL NO CORRIENTE False
APALANCAMIENTO A CORTO Y LARGO PLAZO  False
ROTACIÓN DE CARTERA                   False
ROTACIÓN DE ACTIVO FIJO               False
ROTACIÓN DE VENTAS                    False
PERIODO MEDIO DE COBRANZA CORTO PLAZO False
PERIODO MEDIO DE PAGO CORTO PLAZO     False
IMPACTO GASTOS ADMINISTRACIÓN Y VENTAS False
IMPACTO DE LA CARGA FINANCIERA        False
RENTABILIDAD NETA DEL ACTIVO          False
MARGEN BRUTO                          False
MARGEN OPERACIONAL                    False
RENTABILIDAD NETA DE VENTAS           False
RENTABILIDAD OPERACIONAL DEL PATRIMONIO False
RENTABLIDAD FINANCIERA                False
UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS False
ROE                                   False
ROA                                   False
dtype: bool
```

Pre-Procesamiento

Remover Outliers

```
# Eliminar las filas con valores atípicos
df = df[(df['ROE']>-1) & (df['ROE']<1) &(df['ROA']>-1) & (df['ROA']<1) & (df['ENDEUDAMIENTO DEL ACTIVO']<6) & (df['RENTABILIDAD NETA DE VENTAS']>-50) &
        (df['UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS ']>-5) & (df['UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS ']<5) & (df['RENTABLIDAD FINANCIERA']<500)]
```

Categorizar Variable ROA

```
df['ROA_DIS'] = pd.qcut(df['ROA'], 3, labels=False)
df['ROA_DIS'].sample(10)
```

```
41808    1
4143     0
37811    1
3071     1
72432    1
79580    2
71052    1
30287    1
83       2
3775     1
Name: ROA_DIS, dtype: int64
```

Reenombrar etiquetas de RAMA

```
desc = ['COMERCIO AL POR MAYOR Y AL POR MENOR REPARACIÃ“N DE VEHÃ CULOS AUTOMOTORES Y MOTOCICLETAS.', 'INDUSTRIAS MANUFACTURERAS.', 'AGRICULTURA, GANADERÃ A,  SILVICULTURA Y P
       'ACTIVIDADES PROFESIONALES, CIENTÃ FICAS Y TÃ‰CNICAS.', 'ACTIVIDADES DE ALOJAMIENTO Y DE SERVICIO DE COMIDAS.','INFORMACIÃ“N Y COMUNICACIÃ“N.','ACTIVIDADES DE SERVICIOS
       'ACTIVIDADES DE ATENCIÃ“N DE LA SALUD HUMANA Y DE ASISTENCIA SOCIAL.','TRANSPORTE Y ALMACENAMIENTO.', 'ACTIVIDADES INMOBILIARIAS.', 'EXPLOTACIÃ“N DE MINAS Y CANTERAS.',
       'DISTRIBUCIÃ“N DE AGUA ALCANTARILLADO, GESTIÃ“N DE DESECHOS Y ACTIVIDADES DE SANEAMIENTO.','SUMINISTRO DE ELECTRICIDAD, GAS, VAPOR Y AIRE ACONDICIONADO.',
       'ARTES, ENTRETENIMIENTO Y RECREACIÃ“N.', 'ENSEÃ‘ANZA.', 'ACTIVIDADES FINANCIERAS Y DE SEGUROS.','ACTIVIDADES DE ORGANIZACIONES Y Ã“RGANOS EXTRATERRITORIALES.']
```

```
desc_good = ["COMERCIO AL POR MAYOR Y AL POR MENOR REPARACIÓN DE VEHÍCULOS AUTOMOTORES Y MOTOCICLETAS.", "INDUSTRIAS MANUFACTURERAS.","AGRICULTURA, GANADERÍA, SILVICULTURA Y P
 "ACTIVIDADES PROFESIONALES, CIENTÍFICAS Y TÉCNICAS.","ACTIVIDADES DE ALOJAMIENTO Y DE SERVICIO DE COMIDAS.", "INFORMACIÓN Y COMUNICACIÓN.", "ACTIVIDADES DE SERVICIOS ADMINIST
"ACTIVIDADES DE ATENCIÓN DE LA SALUD HUMANA Y DE ASISTENCIA SOCIAL.", "TRANSPORTE Y ALMACENAMIENTO.", "ACTIVIDADES INMOBILIARIAS.", "EXPLOTACIÓN DE MINAS Y CANTERAS.",
"OTRAS ACTIVIDADES DE SERVICIOS.", "DISTRIBUCIÓN DE AGUA ALCANTARILLADO, GESTIÓN DE DESECHOS Y ACTIVIDADES DE SANEAMIENTO.","SUMINISTRO DE ELECTRICIDAD, GAS, VAPOR Y AIRE ACO
 "ARTES, ENTRETENIMIENTO Y RECREACIÓN.", "ENSEÑANZA.", "ACTIVIDADES FINANCIERAS Y DE SEGUROS.","ACTIVIDADES DE ORGANIZACIONES Y ÓRGANOS EXTRATERRITORIALES."]
```

```
df['DESCRIPCIÓN RAMA'].replace(desc, desc_good, inplace=True)
```

Análisis Exploratorio de Datos

```
print("Skewness: {}".format(df['ROE'].skew()))
print("Kurtosis: {}".format(df['ROE'].kurt()))
print("---------------------------------------")
print(df['ROE'].describe())
print("---------------------------------------")
```

```
Skewness: 0.2009908413692949
Kurtosis: 2.83679171801059
---------------------------------------
count    13306.000000
mean         0.118738
std          0.276325
min         -0.999867
25%          0.006310
50%          0.069845
75%          0.211157
max          0.999334
Name: ROE, dtype: float64
---------------------------------------
```

```
print("Skewness: {}".format(df['ROA'].skew()))
print("Kurtosis: {}".format(df['ROA'].kurt()))
print("---------------------------------------")
print(df['ROA'].describe())
print("---------------------------------------")
```

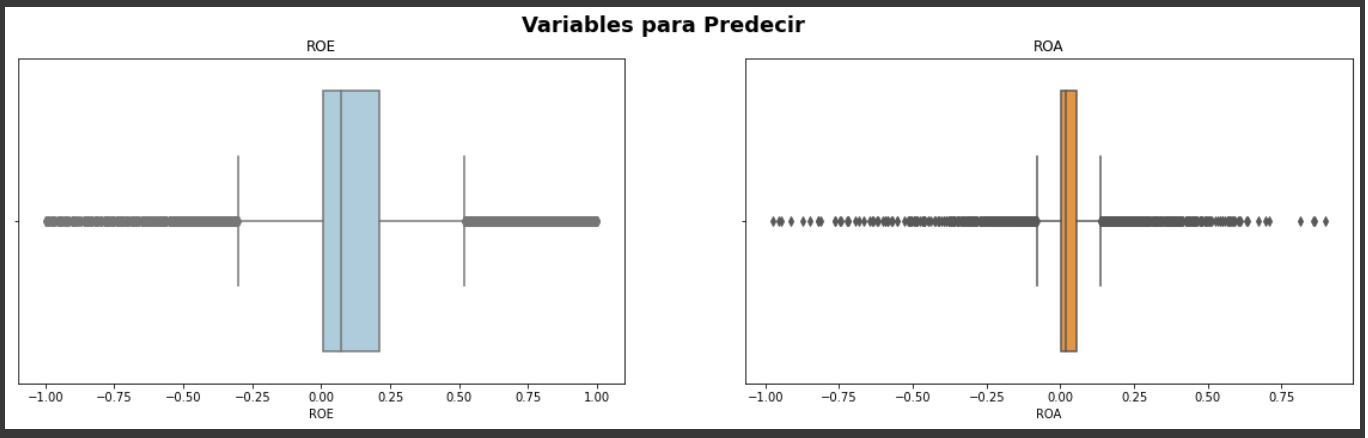
```
Skewness: -0.5395060112246742
Kurtosis: 18.04272956899672
---------------------------------------
count    13306.000000
mean         0.032126
std          0.096310
min         -0.972857
25%          0.001622
50%          0.018347
75%          0.056155
max          0.897318
Name: ROA, dtype: float64
---------------------------------------
```

Analisis de los targets o "y" de entrenamiento: ROE - ROA

- **ROA:** Skewness tiene un valor de -0.5395060112246742, lo que indica que la distribución de los datos tiene una cola ligeramente hacia la izquierda, es decir, tiene una mayor probabilidad de tener valores más bajos que la media. Kurtosis tiene un valor de 18.04272956899672, lo que indica que la distribución de los datos tiene colas más pronunciadas que una distribución normal.
- **ROE:** Skewness tiene un valor de 0.2009908413692949, lo que indica que la distribución de los datos tiene una cola ligeramente hacia la derecha, es decir, tiene una mayor probabilidad de tener valores más altos que la media. Kurtosis tiene un valor de 2.83679171801059, lo que indica que la distribución de los datos tiene colas menos pronunciadas que una distribución normal.

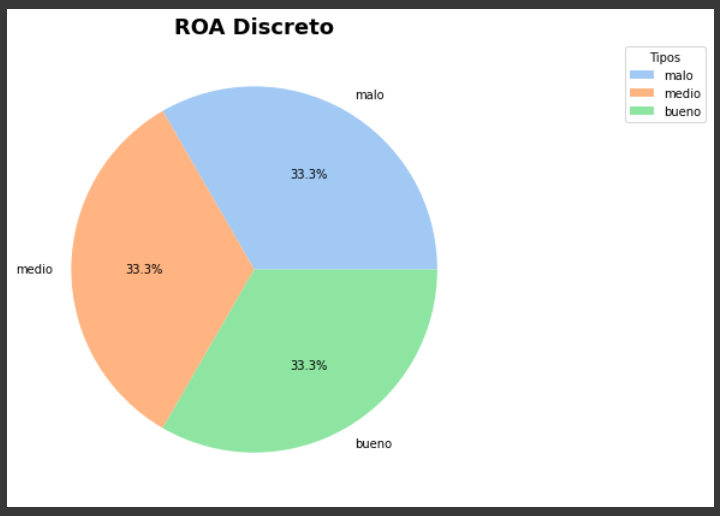
Análisis Univariado

```
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(20,5))

sns.boxplot(x = df['ROE'], ax=ax1, palette="Paired")
sns.boxplot(x =  df['ROA'],ax=ax2, palette="YlOrBr")

fig.suptitle("Variables para Predecir", fontsize=18, fontweight='bold')
ax1.set_title("ROE")
ax2.set_title("ROA")
plt.show()
```

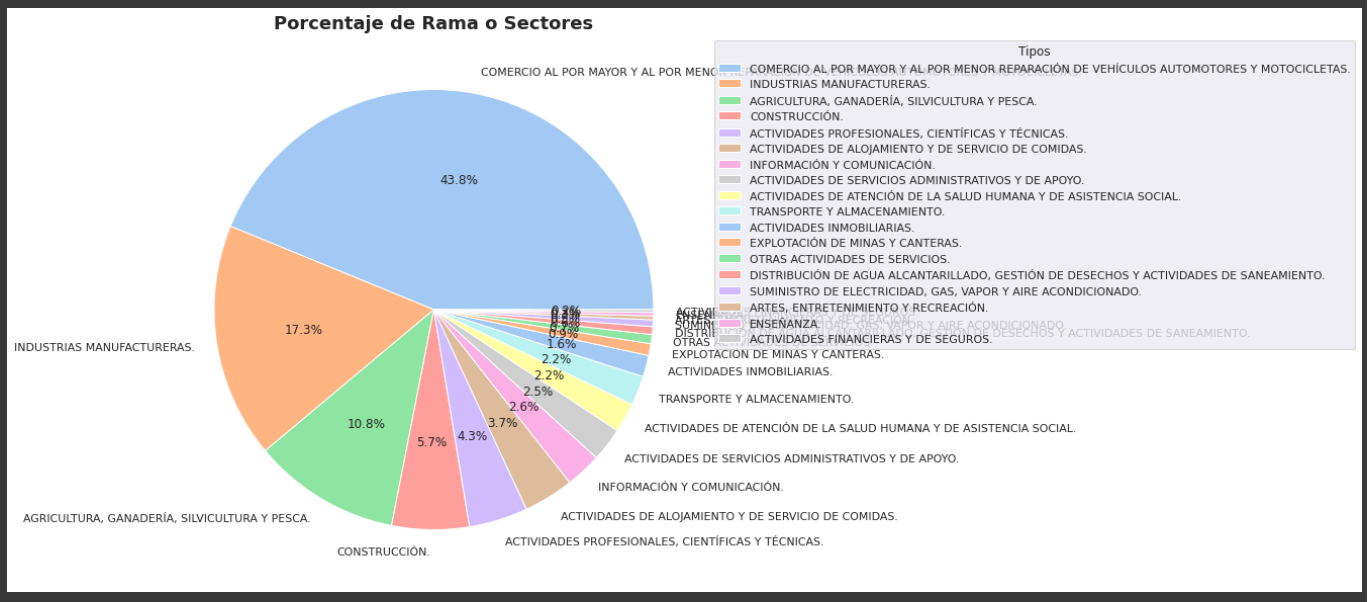



```
plt.figure(figsize=(7,7))
plt.pie(df['ROA_DIS'].value_counts(), autopct='%1.1f%%', colors=sns.color_palette('pastel'),
        labels=['malo', 'medio','bueno'])
```

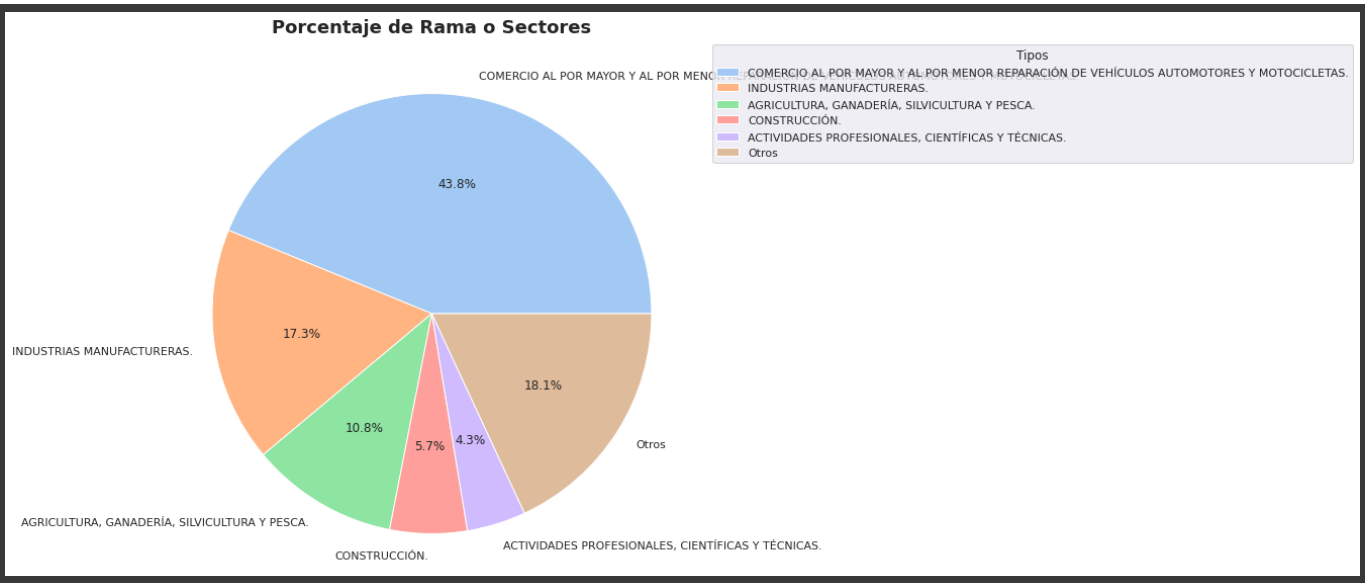
```
plt.title('ROA Discreto', fontsize=18, fontweight='bold')
plt.legend(title ="Tipos",
           bbox_to_anchor =(1, 0, 0.5, 1))
plt.show()
```




```
sns.set(rc={'figure.figsize':(10,10)})
plt.pie(df['RAMA'].value_counts(), autopct='%1.1f%%', colors=sns.color_palette('pastel'), labels=df['DESCRIPCIÓN RAMA'].value_counts().index)
plt.title('Porcentaje de Rama o Sectores', fontsize=18, fontweight='bold')
plt.legend(title ="Tipos",
           bbox_to_anchor =(1, 0, 0.5, 1))
plt.show()
```




```
ramas_count = df['DESCRIPCIÓN RAMA'].value_counts().sort_values(ascending=False)

# Obtener las 5 primeras categorías con mayor registro
top_ramas = ramas_count[:5].index

# Crear una nueva serie de datos donde las categorías no incluidas en las 5 primeras se agrupen bajo la categoría 'Otros'
ramas_new = df['DESCRIPCIÓN RAMA'].apply(lambda x: x if x in top_ramas else 'Otros')
ramas_new_count = ramas_new.value_counts()

# Construir una serie de valores a mano que incluya la categoría "Otros" al final
ramas_dict = dict(ramas_new_count[top_ramas])
ramas_dict['Otros'] = ramas_new_count['Otros']
ramas_new_count = pd.Series(ramas_dict)

# Graficar la nueva serie de datos
plt.pie(ramas_new_count, autopct='%1.1f%%', colors=sns.color_palette('pastel'), labels=ramas_new_count.index)
plt.title('Porcentaje de Rama o Sectores', fontsize=18, fontweight='bold')
plt.legend(title ="Tipos",
           bbox_to_anchor =(1, 0, 0.5, 1))
plt.show()
```



Analisis de variables "y" - ROE y ROA

- **ROE:** Comprende desde -1 al 1, como punto minimo se tiene -0.30 y punto máximo de 0.55. Con un promedio de datos de 0.11
- **ROA:** De la misma manera se tiene como punto minimo y máximo, -1 y 1 respectivamente, dando como promedio 0.03

ROA Categorizado

- Para la regresión lógisticas se toma el ROA, como es una variable continua se lo categoriza en 3 categorias y se distribuye de la siguiente manera:
  - **ROA BUENO:** 33.33% del total
  - **ROA MEDIO:** 33.33% del total
  - **ROA MALO:** 33.33% del total

ROE Continuo

- Para la regresión lineal se toma el ROE es por eso que no se lo categoriza como el ROA, en la regresión lineal se da con variables continuas

Porcentaje de Categorias por RAMA

- Existen 18 categorias que se cuentan para el siguiente analisis de regresión de los cuales los 3 primeros y los 3 ultimos son:

  3 Con mayor Registros

  - Comercio al por mayor y al por menor, reparación de Vehiculos y automotores y vehiculos
  - Industrias Manufactureras
  - Agricultura, ganaderia, silvicultura y pesca

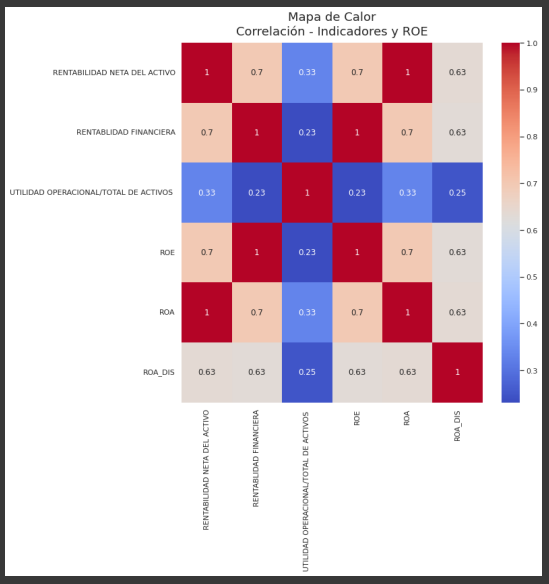
  3 Con menor Registros

  - Artes, entretenimiento y recreación
  - Enseñanza
  - Actividades financieras y de seguros

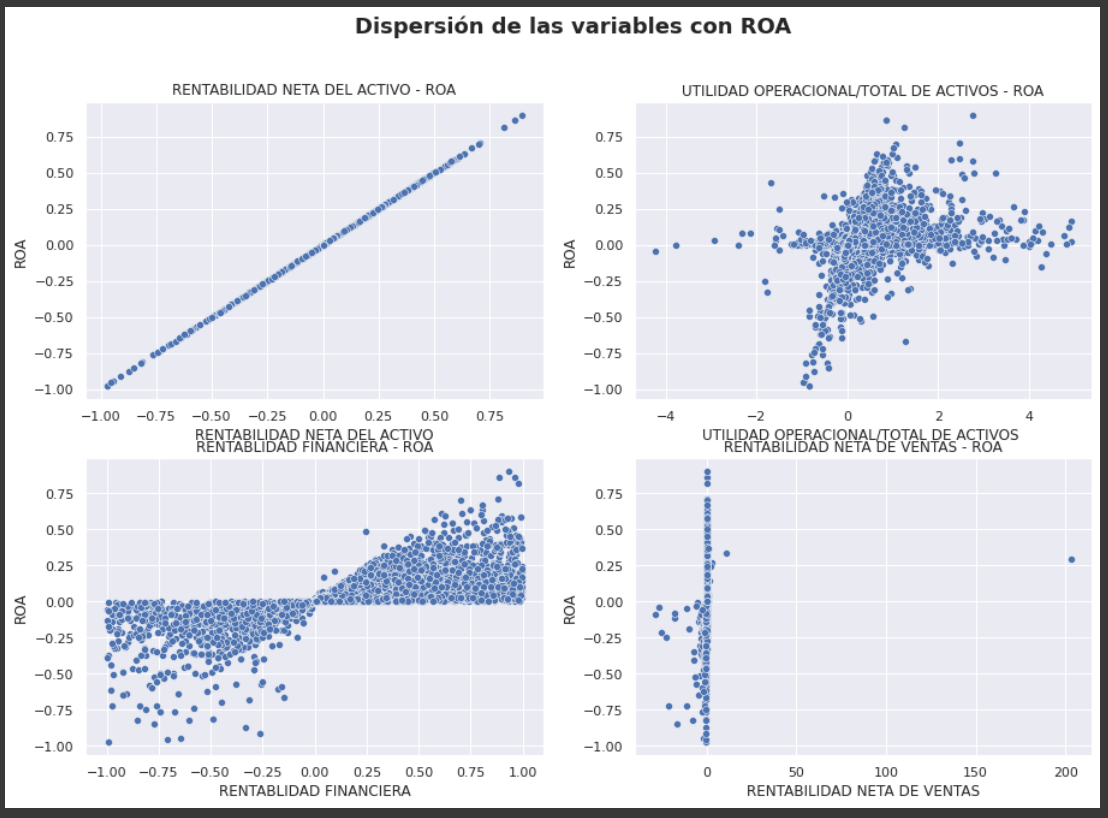
Análisis Bivariado

```
corr_matrix = df.corr()
top_corr_features = corr_matrix.index[abs(corr_matrix["ROE"])>0.2]
plt.figure(figsize=(10,10))
hm = sns.heatmap(df[top_corr_features].corr(),
                 annot=True,
                 cmap="coolwarm")
hm.set_title('Mapa de Calor\nCorrelación - Indicadores y ROE', fontdict={'fontsize':18}, pad=12)
```

```
Text(0.5, 1.0, 'Mapa de Calor\nCorrelación - Indicadores y ROE')
```




```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2, ncols=2, figsize=(15,10))

sns.scatterplot(x = df['RENTABILIDAD NETA DEL ACTIVO'],y = df['ROA'], ax=ax1)
sns.scatterplot(x = df['UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS '],y = df['ROA'], ax=ax2)
sns.scatterplot(x = df['RENTABLIDAD FINANCIERA'],y = df['ROA'], ax=ax3)
sns.scatterplot(x = df['RENTABILIDAD NETA DE VENTAS'],y = df['ROA'], ax=ax4)

fig.suptitle("Dispersión de las variables con ROA", fontsize=18, fontweight='bold')
ax1.set_title("RENTABILIDAD NETA DEL ACTIVO - ROA")
ax2.set_title("UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS - ROA")
ax3.set_title("RENTABLIDAD FINANCIERA - ROA")
ax4.set_title("RENTABILIDAD NETA DE VENTAS - ROA")
plt.show()
```



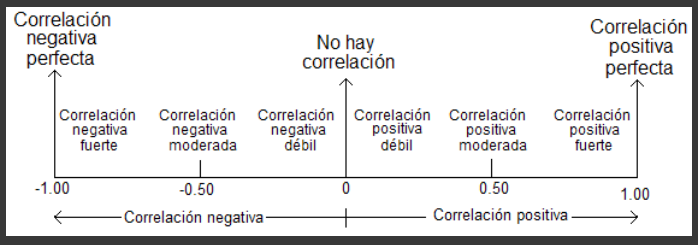
Analisis Bivariado:

- **Analisis Correlacional**

  Para el ROA se pretende determinar las variables que más se ajusten en la relación con el ROA es por eso que se tiene en cuenta el analisis Correlacional para dicha determinación de Variables, entre ellas tenemos:

  - RENTABILIDAD NETA DEL ACTIVO: 0.58 Con correlación positiva moderada
  - RENTABILIDAD NETA DE VENTAS: 0.11 Con correlación positiva debil
  - UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS: 0.47 se tiene una correlación positiva moderada
  - ROTACION DE VENTAS: 0.14, siguiendo en la tendencia de correlación positiva pero este caso es debil

- Otras variables estan:
  - ENDEUDAMIENTO
  - ENDEUDAMIENTO A CORTO PLAZO
  - ENDEUDAMIENTO A LARGO PLAZO
  - RENTABILIDAD FINANCIERA
  - ROE

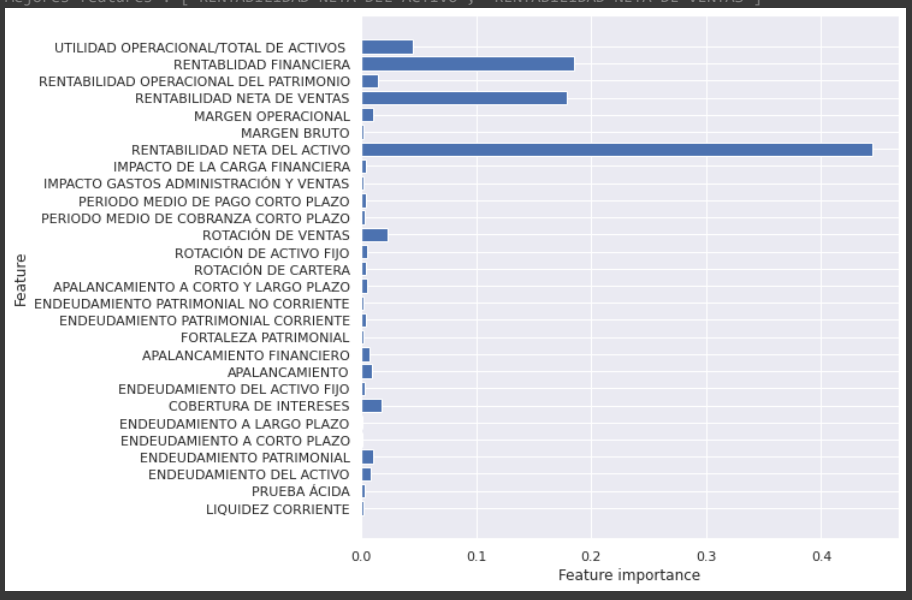


Selección de Variables

```
X_fs = df.iloc[:, 7:35]
y_fs = df['ROA_DIS']
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFECV
from sklearn.ensemble import RandomForestClassifier
cv_estimator = RandomForestClassifier(random_state =42)
X_train,X_test,Y_train,Y_test = train_test_split(X_fs, y_fs, test_size=0.3, random_state=42)
cv_estimator.fit(X_train, Y_train)
cv_selector = RFECV(cv_estimator,cv= 5, step=1,scoring='accuracy')
cv_selector = cv_selector.fit(X_train, Y_train)
rfecv_mask = cv_selector.get_support()
rfecv_features = []
for bool, feature in zip(rfecv_mask, X_train.columns):
 if bool:
   rfecv_features.append(feature)
print('Numero Optimo de features :', cv_selector.n_features_)
print('Mejores features :', rfecv_features)
n_features = X_train.shape[1]
plt.figure(figsize=(8,8))
plt.barh(range(n_features), cv_estimator.feature_importances_, align='center')
plt.yticks(np.arange(n_features), X_train.columns.values)
plt.xlabel('Feature importance')
plt.ylabel('Feature')
plt.show()
```

```
Numero Optimo de features : 2
Mejores features : ['RENTABILIDAD NETA DEL ACTIVO', 'RENTABILIDAD NETA DE VENTAS']
```



Feature Selection

- Para las selección de variables se toma en solo 4 variables para cada regresión tanto lineal como lógistica para responder lo que se quiere responder
  - **Regresión Lineal**

    Se establece mediante el mapa de correlacion las siguientes variables:
    - 'RENTABILIDAD OPERACIONAL DEL PATRIMONIO'
    - 'ENDEUDAMIENTO PATRIMONIAL'
    - 'APALANCAMIENTO'

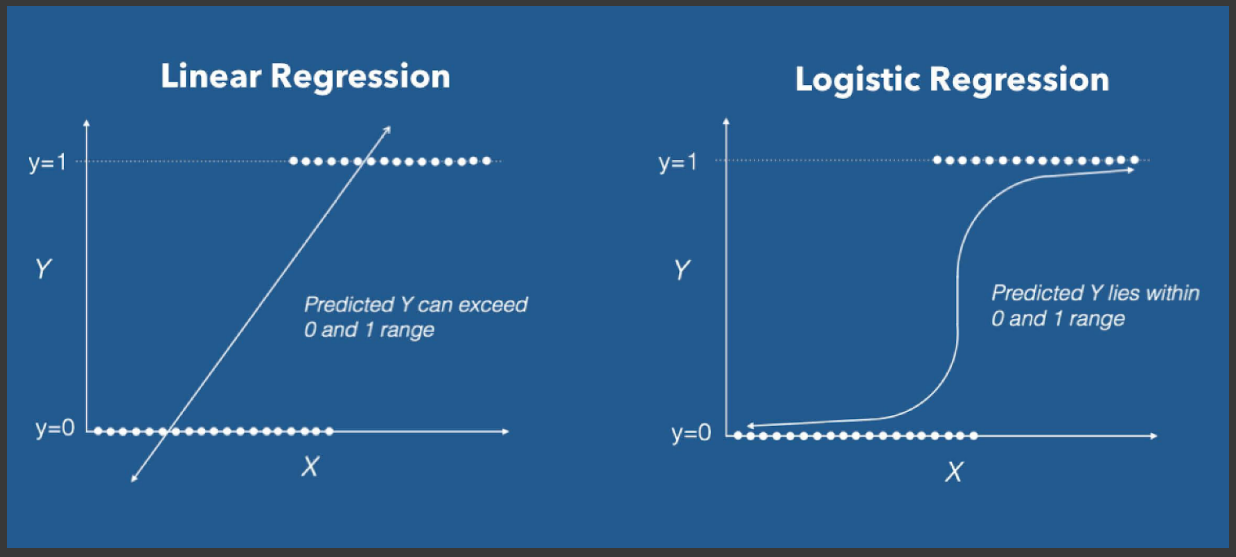
- 'FORTALEZA PATRIMONIAL'

- **Regresión Logistica**

  Se establece mediante un Featuring Selections las siguientes variables:

  - 'RENTABILIDAD NETA DEL ACTIVO'
  - 'UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS '
  - 'RENTABLIDAD FINANCIERA'
  - 'RENTABILIDAD NETA DE VENTAS'

Modelo Predictivo



Modelo predictivo de regresión lineal ROE

```
# Crear los arreglos con los inputs escogidos y el target del ROE
inputs_rl = df[['RENTABILIDAD OPERACIONAL DEL PATRIMONIO', 'ENDEUDAMIENTO PATRIMONIAL', 'APALANCAMIENTO', 'FORTALEZA PATRIMONIAL']].values
targets_rl = df[['ROE']].values
print('Input #1: ', inputs_rl[1], ' - Tamaño: ', inputs_rl.shape,
      '\nTarget #1: ', targets_rl[1], ' - Tamaño: ', targets_rl.shape)
```

```
Input #1:  [-0.09170358  1.7872771   2.787277    1.0029293 ]  - Tamaño:  (13306, 4)
Target #1:  [-0.04617106]  - Tamaño:  (13306, 1)
```

```
# Normalizar los inputs
from sklearn.preprocessing import StandardScaler
from torch.utils.data import TensorDataset, DataLoader
scaler = StandardScaler()
inputs = scaler.fit_transform(inputs_rl)
```

```
# Cambiar el tipo de dato y crear tensores
X = torch.from_numpy(inputs_rl.astype(np.float32))
Y = torch.from_numpy(targets_rl.astype(np.float32))
n_samples, n_features = X.shape
print(n_samples, n_features)
```

```
13306 4
```

```
# Crear conjuntos de datos de PyTorch a partir de los tensores
dataset_train = TensorDataset(X, Y)
dataset_train[1:2]
```

```
(tensor([[-0.0917,  1.7873,  2.7873,  1.0029]]), tensor([[-0.0462]]))
```

```
# Crear dataloaders para cargar los datos en lotes durante el entrenamiento
bs=32
train_loader = DataLoader(dataset_train,batch_size=bs,shuffle=True)
```

```
# Define la clase del modelo de regresión lineal
class ModeloRegresionLineal(torch.nn.Module):
  def __init__(self):
    super(ModeloRegresionLineal, self).__init__()
    self.linear = torch.nn.Linear(n_features, 1)

  def forward(self, x):
    y_pred = self.linear(x)
    return y_pred
```

```
# Define los epochs, learning rate, función de costo y el optimizador
epochs = 30
ta = 1e-5  #learning rate
model_rl = ModeloRegresionLineal()
```

```
funcion_costo = torch.nn.L1Loss()
optimizer = torch.optim.SGD(model_rl.parameters(), lr = ta) #gradiente descendente, actualiza los pesos w y el bias b del modelo
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'min')

for i in range(epochs):
  for x,y in train_loader:
    preds = model_rl(x)
    loss = funcion_costo(preds, y)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

  print(f"Epoch {i}/{epochs}: Loss {loss}")
```

```
Epoch 0/30: Loss 1.582852840423584
Epoch 1/30: Loss 0.6363125443458557
Epoch 2/30: Loss 0.691182553768158
Epoch 3/30: Loss 0.6884936690330505
Epoch 4/30: Loss 0.8965075612068176
Epoch 5/30: Loss 0.4686119854450226
Epoch 6/30: Loss 0.3817178010940552
Epoch 7/30: Loss 0.5758546590805054
Epoch 8/30: Loss 0.8825715780258179
Epoch 9/30: Loss 1.2418544292449951
Epoch 10/30: Loss 0.3235556185245514
Epoch 11/30: Loss 1.6967099905014038
Epoch 12/30: Loss 1.0973376035690308
Epoch 13/30: Loss 1.2854818105697632
Epoch 14/30: Loss 0.37666189670562744
Epoch 15/30: Loss 0.49281856417655945
Epoch 16/30: Loss 0.7013010382652283
Epoch 17/30: Loss 0.38611990213394165
Epoch 18/30: Loss 0.7784183025360107
Epoch 19/30: Loss 0.32213014364242554
Epoch 20/30: Loss 0.9864485263824463
Epoch 21/30: Loss 0.2880042791366577
Epoch 22/30: Loss 0.4155954122543335
Epoch 23/30: Loss 0.26024287939071655
Epoch 24/30: Loss 0.4180039167404175
Epoch 25/30: Loss 0.6704758405685425
Epoch 26/30: Loss 0.6688013672828674
Epoch 27/30: Loss 0.8429556488990784
Epoch 28/30: Loss 0.322173535823822
Epoch 29/30: Loss 0.28651705384254456
```

```
print('Weight:{} '.format(model_rl.linear.weight.detach().numpy()),
      '\nBias:{} '.format(model_rl.linear.bias.detach().numpy()))
```

```
Weight:[[ 0.24913777 -0.07784622  0.05654165 -0.03151945]]
Bias:[0.14749205]
```

Modelo predictivo de regresión lineal ROE

- El modelo se define en la clase ModeloRegresionLineal, que hereda de la clase torch.nn.Module de PyTorch. El modelo tiene una sola capa, llamada "linear", que es una capa lineal que se utiliza para realizar una operación de multiplicación matricial (X @ w.t() + b) entre los datos de entrada (X) y los pesos (w) del modelo, más un sesgo (b).
- El modelo se entrena utilizando el método de optimización de gradiente descendente estocástico (SGD), con un learning rate de 1e-5, y se utiliza la función de costo de pérdida cuadrática media (MSELoss) para evaluar el rendimiento del modelo en cada iteración del entrenamiento. El modelo se entrena durante 30 épocas, y en cada época se actualizan los pesos y el sesgo del modelo mediante el optimizador SGD.
- También se utiliza un scheduler para reducir el learning rate del optimizador cuando la función de costo no se ha reducido durante varias iteraciones.

Función Lineal Final

$$y = (0.00375612 * x1) + (-0.18859579 * x2) + (0.18879108 * x3) + (-0.2053995 * x4) - 0.07093453$$

Modelo predictivo de regresión logística del ROA

```
# Crear los arreglos con los inputs escogidos y el target del ROA
inputs_rlg = df[['RENTABILIDAD NETA DEL ACTIVO',  'UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS ', 'RENTABLIDAD FINANCIERA', 'RENTABILIDAD NETA DE VENTAS']].values
targets_rlg = df['ROA_DIS'].values
print('Input #1: ', inputs_rlg[1], ' - Tamaño: ', inputs_rlg.shape,
      '\nTarget #1: ', targets_rlg[1], ' - Tamaño: ', targets_rlg.shape)
```

```
Input #1:  [-0.01656493 -0.03290078 -0.04617107 -0.01552408]  - Tamaño:  (13306, 4)
Target #1:  0  - Tamaño:  (13306,)
```

```
scaler = StandardScaler()
x_train = scaler.fit_transform(inputs_rlg)
```

```
x_train = torch.from_numpy(x_train.astype(np.float32))
y_train = torch.from_numpy(targets_rlg.astype(np.int64))
```

```
class MulticlassLogisticRegression(torch.nn.Module):
  def __init__(self, n_features, n_classes):
    super(MulticlassLogisticRegression,self).__init__()
    self.linear = torch.nn.Linear(n_features, n_classes)
    self.softmax = torch.nn.Softmax(dim=1)
```

```
6
7   def forward(self, x):
8     y_hat = self.softmax(self.linear(x))
9     return y_hat
10
11 #función que visualiza la evolución de la perdida y la precisión en cada epoch
12 def plot_loss(epochs, loss, acc):
13   plt.figure(figsize=(10, 5))
14   xlim = len(loss)
15   plt.plot(epochs,loss)
16   plt.plot(epochs,acc)
17   plt.xlabel('Epochs')
18   plt.ylabel('Value')
19   plt.legend(('Train loss','Accuracy'),loc='upper right',shadow=True)
20   plt.title('Train Loss vs Accuracy')
21
22 #función que realiza el entrenamiento
23 def train(num_epochs, optimizer, cost, model):
24   #listas usadas para guardar los valores de pérdida, precisión, para cada epoch
25   #esta información sirve para graficar el proceso de entrenamiento
26   loss_vals = []
27   acc_vals = []
28   epoch_vals = []
29
30   #entrenamiento
31   for epoch in range(num_epochs):
32     y_hat = model(x_train)
33     loss = cost(y_hat,y_train)
34     loss.backward()
35     optimizer.step()
36     optimizer.zero_grad()
37
38     #se evalua cada 5 epochs
39     if (epoch+1)%5 == 0:
40       with torch.no_grad():
41         loss_vals.append(loss.item())
42         y_hat_class = y_hat.argmax(dim=1)
43         accuracy = (y_hat_class.eq(y_train).sum())/float(y_hat.shape[0])
44         acc_vals.append(accuracy.item())
45         epoch_vals.append(epoch+1)
46       print(f'epoch:{epoch+1} loss={loss.item()} accuracy={accuracy.item()}')
47
48   #se grafica el proceso de entrenamiento
49   plot_loss(epoch_vals, loss_vals, acc_vals)
```

```
1 model_rlg = MulticlassLogisticRegression(inputs_rlg.shape[1], len(np.unique(targets_rlg)))
2 optimizer = torch.optim.Adam(model_rlg.parameters(), lr=0.09)
3 cost = torch.nn.CrossEntropyLoss()
4 # se entrena el modelo
5 train(num_epochs=10000, optimizer=optimizer, cost=cost, model=model_rlg)
```

```
epoch:5 loss=0.9688050150871277 accuracy=0.5923643708229065
epoch:10 loss=0.9128708243370056 accuracy=0.6578235626220703
epoch:15 loss=0.8774487376213074 accuracy=0.7414700388908386
epoch:20 loss=0.8507058024406433 accuracy=0.7873139977455139
epoch:25 loss=0.8315939903259277 accuracy=0.8312039971351624
epoch:30 loss=0.8159731030464172 accuracy=0.850142776966095
epoch:35 loss=0.8039976358413696 accuracy=0.8485645651817322
epoch:40 loss=0.794280469417572 accuracy=0.8646475076675415
epoch:45 loss=0.7861695885658264 accuracy=0.8835863471031189
epoch:50 loss=0.7789802551269531 accuracy=0.8827596306800842
epoch:55 loss=0.772702693939209 accuracy=0.8884713649749756
epoch:60 loss=0.7671425938606262 accuracy=0.8897489905357361
epoch:65 loss=0.7621577978134155 accuracy=0.893130898475647
epoch:70 loss=0.7575573921203613 accuracy=0.8983917236328125
epoch:75 loss=0.75336092710495 accuracy=0.8990681171417236
epoch:80 loss=0.7494702339172363 accuracy=0.9022245407104492
epoch:85 loss=0.745836079120636 accuracy=0.9031264185905457
epoch:90 loss=0.7424418330192566 accuracy=0.9030512571334839
epoch:95 loss=0.7392492890357971 accuracy=0.9047797918319702
epoch:100 loss=0.7362363934516907 accuracy=0.9057568311691284
epoch:105 loss=0.733387291431427 accuracy=0.9072598814964294
epoch:110 loss=0.730684757232666 accuracy=0.9089884161949158
epoch:115 loss=0.728115439414978 accuracy=0.909965455532074
epoch:120 loss=0.7256685495376587 accuracy=0.9104915261268616
epoch:125 loss=0.7233341336250305 accuracy=0.9113181829452515
epoch:130 loss=0.7211030125617981 accuracy=0.9118442535400391
epoch:135 loss=0.7189679741859436 accuracy=0.9127461314201355
epoch:140 loss=0.716921865940094 accuracy=0.9133473634719849
epoch:145 loss=0.714958667755127 accuracy=0.9142491817474365
epoch:150 loss=0.7130730748176575 accuracy=0.9153013825416565
epoch:155 loss=0.7112597823143005 accuracy=0.9158274531364441
epoch:160 loss=0.709514319896698 accuracy=0.916579008102417
epoch:165 loss=0.707832932472229 accuracy=0.9171802401542664
epoch:170 loss=0.7062113881111145 accuracy=0.9176311492919922
epoch:175 loss=0.7046463489532471 accuracy=0.9183075428009033
epoch:180 loss=0.7031347751617432 accuracy=0.9191341996192932
epoch:185 loss=0.7016738057136536 accuracy=0.9192845225334167
epoch:190 loss=0.7002606391906738 accuracy=0.9199609160423279
epoch:195 loss=0.6988927125930786 accuracy=0.920261561870575
epoch:200 loss=0.697567880153656 accuracy=0.9208627939224243
epoch:205 loss=0.696283757686615 accuracy=0.9215391278266907
epoch:210 loss=0.6950386762619019 accuracy=0.921764612197876
epoch:215 loss=0.6938305497169495 accuracy=0.9224410057067871
epoch:220 loss=0.6926578283309937 accuracy=0.92319256067276
epoch:225 loss=0.6915189623832703 accuracy=0.9238689541816711
epoch:230 loss=0.6904125213623047 accuracy=0.9240943789482117
epoch:235 loss=0.689336895942688 accuracy=0.924319863319397
epoch:240 loss=0.688290536403656 accuracy=0.9247707724571228
epoch:245 loss=0.6872721314430237 accuracy=0.9252216815948486
epoch:250 loss=0.6862807273864746 accuracy=0.9255223274230957
epoch:255 loss=0.685314953327179 accuracy=0.9259732365608215
epoch:260 loss=0.6843739151954651 accuracy=0.9261987209320068
epoch:265 loss=0.6834565997123718 accuracy=0.9265744686126709
epoch:270 loss=0.682561993598938 accuracy=0.9270253777503967
epoch:275 loss=0.6816893815994263 accuracy=0.9274011850357056
epoch:280 loss=0.6808376908302307 accuracy=0.9277769327163696
epoch:285 loss=0.6800063252449036 accuracy=0.9279272556304932
epoch:290 loss=0.6791943907737732 accuracy=0.9281527400016785
epoch:295 loss=0.678400993347168 accuracy=0.9286788105964661
epoch:300 loss=0.6776258945465088 accuracy=0.9288290739059448
epoch:305 loss=0.6768680214881897 accuracy=0.9291297197341919
epoch:310 loss=0.6761270761489868 accuracy=0.9291297197341919
epoch:315 loss=0.6754024028778076 accuracy=0.9293552041053772
epoch:320 loss=0.6746931076049805 accuracy=0.929505467414856
epoch:325 loss=0.6739991903305054 accuracy=0.9296557903289795
epoch:330 loss=0.6733195781707764 accuracy=0.9298812747001648
epoch:335 loss=0.6726543307304382 accuracy=0.9303321838378906
epoch:340 loss=0.6720025539398193 accuracy=0.9304073452949524
epoch:345 loss=0.6713640689849854 accuracy=0.9307079315185547
epoch:350 loss=0.6707382798194885 accuracy=0.9308582544326782
epoch:355 loss=0.6701248288154602 accuracy=0.9313843250274658
epoch:360 loss=0.6695234179496765 accuracy=0.9316849708557129
epoch:365 loss=0.668933629989624 accuracy=0.9317601323127747
epoch:370 loss=0.6683549880981445 accuracy=0.9321358799934387
epoch:375 loss=0.6677873134613037 accuracy=0.9322110414505005
epoch:380 loss=0.6672301888465881 accuracy=0.9322110414505005
epoch:385 loss=0.6666832566261292 accuracy=0.932436466217041
epoch:390 loss=0.6661463379859924 accuracy=0.932436466217041
epoch:395 loss=0.6656190156936646 accuracy=0.9325867891311646
epoch:400 loss=0.665101170539856 accuracy=0.9327371120452881
epoch:405 loss=0.6645923256874084 accuracy=0.9331128597259521
epoch:410 loss=0.6640923619270325 accuracy=0.9333383440971375
epoch:415 loss=0.6636009812355042 accuracy=0.9334135055541992
epoch:420 loss=0.6631180047988892 accuracy=0.933864414691925
epoch:425 loss=0.6626431345939636 accuracy=0.9342402219772339
epoch:430 loss=0.662176251411438 accuracy=0.9343153238296509
epoch:435 loss=0.661716878414154 accuracy=0.9343153238296509
epoch:440 loss=0.6612651944160461 accuracy=0.9345408082008362
epoch:445 loss=0.6608206629753113 accuracy=0.9347662925720215
epoch:450 loss=0.6603833436965942 accuracy=0.9349165558815002
epoch:455 loss=0.6599528193473816 accuracy=0.9350668787956238
epoch:460 loss=0.6595290303230286 accuracy=0.9350668787956238
epoch:465 loss=0.6591118574142456 accuracy=0.9352172017097473
epoch:470 loss=0.6587011218070984 accuracy=0.9352172017097473
epoch:475 loss=0.658296525478363 accuracy=0.9354426860809326
epoch:480 loss=0.6578980088233948 accuracy=0.9358184337615967
epoch:485 loss=0.6575055122375488 accuracy=0.9358935952186584
epoch:490 loss=0.6571187376976013 accuracy=0.9359687566757202
epoch:495 loss=0.6567376255989075 accuracy=0.9361941814422607
epoch:500 loss=0.6563620567321777 accuracy=0.9362693428993225
```

```
epoch:505 loss=0.6559918522834778 accuracy=0.9367954134941101
```

Modelo predictivo de regresión logística del ROA

- El modelo se define en la clase MulticlassLogisticRegression, que hereda de la clase torch.nn.Module de PyTorch.
- El modelo tiene dos capas: una capa lineal y una capa de softmax:
  - La capa lineal se utiliza para realizar una operación de multiplicación matricial entre los datos de entrada (X) y los pesos (w) del modelo, más un sesgo (b).
  - La capa de softmax se utiliza para calcular las probabilidades para cada clase en función de los valores de la capa lineal.

Parametros

- El modelo se entrena utilizando el optimizador Adam
- Con un learning rate de 0.09
- Se utiliza la función de costo de entropía cruzada (CrossEntropyLoss) para evaluar el rendimiento del modelo en cada iteración del entrenamiento.
- El modelo se entrena durante 10000 épocas, y en cada época se actualizan los pesos y el sesgo del modelo mediante el optimizador Adam.
- El método train es el encargado de entrenar al modelo, se le pasan los parametros necesarios para el entrenamiento.

```
epoch:625 loss=0.6484388113021851 accuracy=0.9405531287193298
```

Predicciones

Predicciones con el modelo de regresión lineal ROE

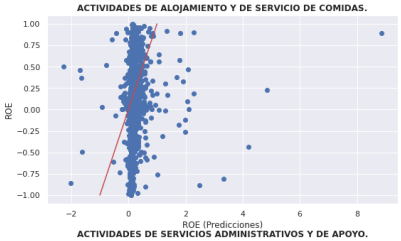
Predicciones con el modelo de regresión lineal ROE - Categoria

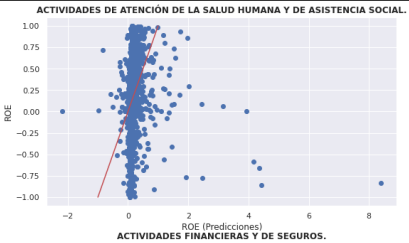
```
epoch:685 loss=0.6454028487205505 accuracy=0.9419059157371521
```

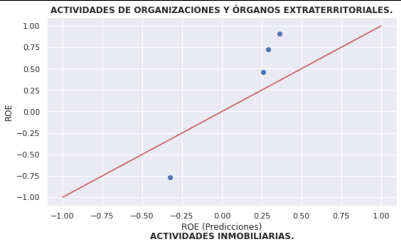
```
def categoria():
  files_joined = os.path.join('/content/drive/MyDrive/Inteligencia Artificial/Segundo Parcial/Proyecto', "indicadores*xlsx")
  list_files = glob.glob(files_joined)
  dfp = pd.concat(map(pd.read_excel, list_files), ignore_index=True)
  # Remover Outliers
  dfp = dfp[(dfp['ROE'] > -1) & (dfp['ROE'] < 1) & (dfp['ROA'] > -1) & (dfp['ROA'] < 1) & (dfp['ENDEUDAMIENTO DEL ACTIVO'] < 6) & (dfp['RENTABILIDAD NETA DE VENTAS'] > -50) &
            (dfp['UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS '] > -5) & (dfp['UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS '] < 5) & (dfp['RENTABLIDAD FINANCIERA'] < 500)]
  # Creando Metricas
  metrics = {"Categoria": [], "MAE": [], "MSE": [], "RMSE": []}
  # Agrupar por RAMA
  dfp['DESCRIPCIÓN RAMA'].replace(desc, desc_good, inplace=True)
  grouped = dfp.groupby("DESCRIPCIÓN RAMA")
  # Declarar la figura y los axs
  fig, axs = plt.subplots(nrows=(len(grouped)//3)+1, ncols=3, figsize=(30,40))
  axs = axs.ravel()
  i = 0
  for name, group in grouped:
      inputs_rl = group[['RENTABILIDAD OPERACIONAL DEL PATRIMONIO', 'ENDEUDAMIENTO PATRIMONIAL', 'APALANCAMIENTO', 'FORTALEZA PATRIMONIAL']].values
      targets_rl = group[['ROE']].values
      # Escalando los datos
      scaler = StandardScaler()
      inputs = scaler.fit_transform(inputs_rl)
      # Transformando los datos a tensores
      inputs_rl = torch.from_numpy(inputs.astype(np.float32))
      targets_rl = torch.from_numpy(targets_rl.astype(np.float32))
      # Creando el conjunto de datos de test
      dataset_test = TensorDataset(inputs_rl, targets_rl)
      test_loader = DataLoader(dataset_test, batch_size=bs, shuffle=True)
      # Evaluando el modelo
      y_pred = []
      y_true = []
      model_rl.train(False)
      for inputs, targets in test_loader:
          y_pred.extend(model_rl(inputs).data.numpy())
          y_true.extend(targets.numpy())
      axs[i].scatter(y_pred, y_true)
      axs[i].set_title(f"{name}", fontsize=12, fontweight='bold')
      axs[i].set_ylabel('ROE')
      axs[i].set_xlabel('ROE (Predicciones)')
      axs[i].plot([-1,1], [-1, 1], 'k', c='r')
      i += 1
      # Calculando metricas
      mae = mean_absolute_error(y_true=y_true, y_pred=y_pred)
      mse = mean_squared_error(y_true=y_true, y_pred=y_pred, squared=True)
      rmse = mean_squared_error(y_true=y_true, y_pred=y_pred, squared=False)
      # Imprimiendo los resultados
      metrics["Categoria"].append(name)
      metrics["MAE"].append(mae)
      metrics["MSE"].append(mse)
      metrics["RMSE"].append(rmse)
  plt.show()
  metrics_df = pd.DataFrame(metrics)
  return metrics_df
```

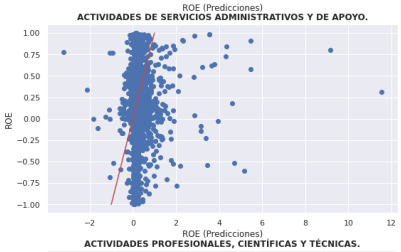
```
epoch:955 loss=0.6353154182434082 accuracy=0.9453629851341248
```

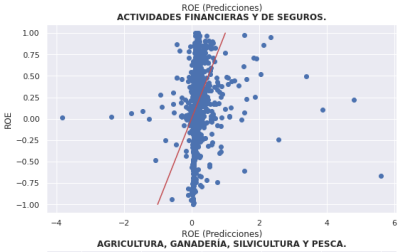
```
metrics_rl = categoria()
```

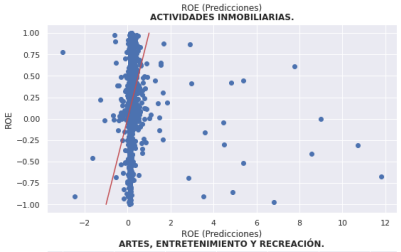

ACTIVIDADES DE ALOJAMIENTO Y DE SERVICIO DE COMIDAS.
ROE
ROE (Predicciones)

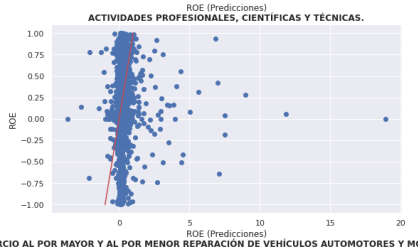

ACTIVIDADES DE ATENCIÓN DE LA SALUD HUMANA Y DE ASISTENCIA SOCIAL.
ROE
ROE (Predicciones)

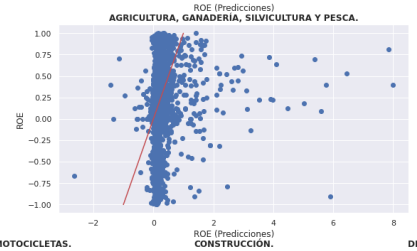

ACTIVIDADES DE ORGANIZACIONES Y ÓRGANOS EXTRATERRITORIALES.
ROE
ROE (Predicciones)

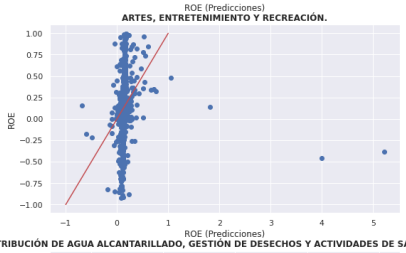

ACTIVIDADES DE SERVICIOS ADMINISTRATIVOS Y DE APOYO.
ROE
ROE (Predicciones)

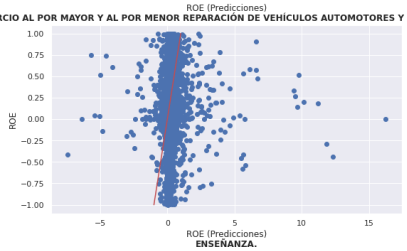

ACTIVIDADES FINANCIERAS Y DE SEGUROS.
ROE
ROE (Predicciones)

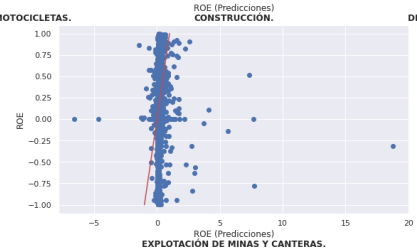

ACTIVIDADES INMOBILIARIAS.
ROE
ROE (Predicciones)


ACTIVIDADES PROFESIONALES, CIENTÍFICAS Y TÉCNICAS.
ROE
ROE (Predicciones)

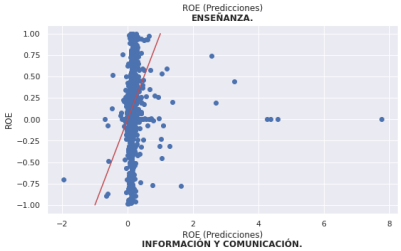

AGRICULTURA, GANADERÍA, SILVICULTURA Y PESCA.
ROE
ROE (Predicciones)

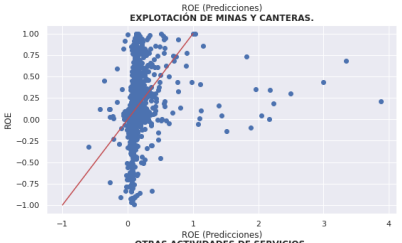

ARTES, ENTRETENIMIENTO Y RECREACIÓN.
ROE
ROE (Predicciones)

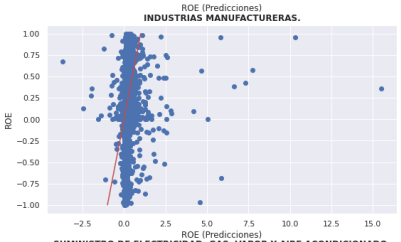

COMERCIO AL POR MAYOR Y AL POR MENOR REPARACIÓN DE VEHÍCULOS AUTOMOTORES Y MOTOCICLETAS.
ROE
ROE (Predicciones)

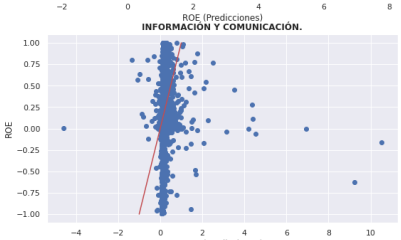

CONSTRUCCIÓN.
ROE
ROE (Predicciones)

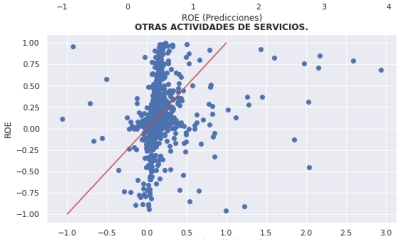

DISTRIBUCIÓN DE AGUA ALCANTARILLADO, GESTIÓN DE DESECHOS Y ACTIVIDADES DE SANEAMIENTO.
ROE
ROE (Predicciones)

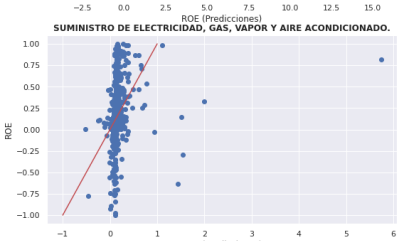

ENSEÑANZA.
ROE
ROE (Predicciones)

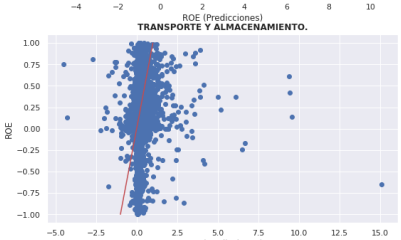

EXPLOTACIÓN DE MINAS Y CANTERAS.
ROE
ROE (Predicciones)


INDUSTRIAS MANUFACTURERAS.
ROE
ROE (Predicciones)


INFORMACIÓN Y COMUNICACIÓN.
ROE
ROE (Predicciones)


OTRAS ACTIVIDADES DE SERVICIOS.
ROE
ROE (Predicciones)


SUMINISTRO DE ELECTRICIDAD, GAS, VAPOR Y AIRE ACONDICIONADO.
ROE
ROE (Predicciones)


TRANSPORTE Y ALMACENAMIENTO.
ROE
ROE (Predicciones)

```
1 metrics_rl.sort_values("MAE", ascending=True)
```

	Categoria	MAE	MSE	RMSE
5	ACTIVIDADES INMOBILIARIAS.	0.174064	0.120180	0.346670
14	INDUSTRIAS MANUFACTURERAS.	0.200422	0.129485	0.359840
18	TRANSPORTE Y ALMACENAMIENTO.	0.206743	0.128273	0.358152
7	AGRICULTURA, GANADERÍA, SILVICULTURA Y PESCA.	0.212082	0.131772	0.363004
9	COMERCIO AL POR MAYOR Y AL POR MENOR REPARACIÓ...	0.212142	0.141425	0.376065
10	CONSTRUCCIÓN.	0.219447	0.149078	0.386106
3	ACTIVIDADES DE SERVICIOS ADMINISTRATIVOS Y DE ...	0.222980	0.141119	0.375659
13	EXPLOTACIÓN DE MINAS Y CANTERAS.	0.225074	0.132817	0.364440
4	ACTIVIDADES FINANCIERAS Y DE SEGUROS.	0.225267	0.136828	0.369903
17	SUMINISTRO DE ELECTRICIDAD, GAS, VAPOR Y AIRE ...	0.226206	0.134762	0.367099
11	DISTRIBUCIÓN DE AGUA ALCANTARILLADO, GESTIÓN D...	0.228318	0.138817	0.372582
1	ACTIVIDADES DE ATENCIÓN DE LA SALUD HUMANA Y D...	0.240345	0.164571	0.405674
16	OTRAS ACTIVIDADES DE SERVICIOS.	0.240603	0.139134	0.373007
6	ACTIVIDADES PROFESIONALES, CIENTÍFICAS Y TÉCNI...	0.240688	0.159593	0.399491
8	ARTES, ENTRETENIMIENTO Y RECREACIÓN.	0.242049	0.165277	0.406543
12	ENSEÑANZA.	0.243492	0.160503	0.400628
15	INFORMACIÓN Y COMUNICACIÓN.	0.248792	0.164463	0.405541
0	ACTIVIDADES DE ALOJAMIENTO Y DE SERVICIO DE CO...	0.250336	0.162405	0.402995
2	ACTIVIDADES DE ORGANIZACIONES Y ÓRGANOS EXTRAT...	0.407773	0.182430	0.427118

Predicciones - Regresión Lineal

- **Predicciones con el modelo de regresión lineal ROE - Categoria**

De las categorias que tienen menor error por lo tanto tiene un mejor rendimiento a la hora de predecir los ROE son:

Num	Categoria	MAE	MSE	RMSE
5	ACTIVIDADES INMOBILIARIAS.	0.297	0.21	0.458
3	ACTIVIDADES DE SERVICIOS ADMINISTRATIVOS Y DE APOYO	0.353	0.264	0.514
0	ACTIVIDADES DE ALOJAMIENTO Y DE SERVICIO DE CO...	0.361	0.286	0.535

De las categorias que tienen menor error por lo tanto tiene un mayor rendimiento a la hora de predecir los ROE son:

Num	Categoria	MAE	MSE	RMSE
17	SUMINISTRO DE ELECTRICIDAD, GAS, VAPOR Y AIRE ACONDICIONADO	0.476	0.351	0.592
16	OTRAS ACTIVIDADES DE SERVICIOS.	0.485	0.342	0.584
2	ACTIVIDADES DE ORGANIZACIONES Y Ã"RGANOS EXTRA...	1.013	1.126	1.061

> **Nota:** Estos datos son de los años: 2017, 2018, 2019 y 2020 juntos

+ Código + Texto

Predicciones con el modelo de regresión lineal ROE - Año

```
epoch:1845 loss=0.6176549196243286 accuracy=0.9541560411453247
```

```
1 def procesar_rl(año):
2     df = pd.read_excel(f'/content/drive/MyDrive/Inteligencia Artificial/Segundo Parcial/Proyecto/indicadores{año}_cia.xlsx')
3     print('Antes: ',len(df), ' ', año)
4     # Remover Outliers
5     df = df[(df['ROE'] > -1) & (df['ROE'] < 1) & (df['ROA'] > -1) & (df['ROA'] < 1) & (df['ENDEUDAMIENTO DEL ACTIVO'] < 6) & (df['RENTABILIDAD NETA DE VENTAS'] > -50) &
6             (df['UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS '] > -5) & (df['UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS '] < 5) & (df['RENTABLIDAD FINANCIERA'] < 500)]
7     # Input y Targets
8     inputs_rl = df[['RENTABILIDAD OPERACIONAL DEL PATRIMONIO', 'ENDEUDAMIENTO PATRIMONIAL', 'APALANCAMIENTO', 'FORTALEZA PATRIMONIAL']].values
9     targets_rl = df[['ROE']].values
10    # Descargando Dataset
11    df.to_csv(f'Indicadores_{año}.csv', index=False)
12    # Escalando los datos
13    scaler = StandardScaler()
14    inputs = scaler.fit_transform(inputs_rl)
15    # Transformando los datos a tensores
16    inputs_rl = torch.from_numpy(inputs.astype(np.float32))
17    targets_rl = torch.from_numpy(targets_rl.astype(np.float32))
18    # Creando el conjunto de datos de test
19    dataset_test = TensorDataset(inputs_rl, targets_rl)
20    test_loader = DataLoader(dataset_test, batch_size=bs, shuffle=True)
21    # Evaluando el modelo
22    y_pred = []
23    y_true = []
24    model_rl.train(False)
25    for inputs, targets in test_loader:
26        y_pred.extend(model_rl(inputs).data.numpy())
27        y_true.extend(targets.numpy())
28    plt.figure(figsize=(10, 5))
29    plt.scatter(y_pred, y_true)
30    plt.ylabel('ROE')
31    plt.xlabel('ROE (Predicciones)')
32    plt.plot([-1,1], [-1, 1], 'k', c='r')
```

```
    plt.show()
    # Calculando metricas
    mae = mean_absolute_error(y_true=y_true, y_pred=y_pred)
    mse = mean_squared_error(y_true=y_true, y_pred=y_pred, squared=True)
    rmse = mean_squared_error(y_true=y_true, y_pred=y_pred, squared=False)
    # Imprimiendo los resultados
    print(f"\nResultados para el año {año} - con {len(targets_rl)} datos:")
    print(f"MAE: {mae}")
    print(f"MSE: {mse}")
    print(f"RMSE: {rmse}")
```

epoch:2060 loss=0.6144905686378479 accuracy=0.956560969352722

```
#Arreglo años
years = [2017, 2018, 2019, 2020]

#Bucle para ejecutar proceso para cada año
for year in years:
    procesar_rl(year)
```

```
Antes:  81485   2017
<ipython-input-41-05e996d465d9>:32: UserWarning: color is redundantly defined by the 'color' keyword argument and the fmt string "k" (-> color=(0.1
  plt.plot([-1,1], [-1, 1], 'k', c='r')
```


```
Resultados para el año 2017 - con 46592 datos:
MAE: 0.21347905695438385
MSE: 0.13898451626300812
epoch:2640 loss=0.6049321293830872 accuracy=0.9662558436393738
```

Predicciones - Regresión Lineal

- **Predicciones con el modelo de regresión lineal ROE - Año**

En los Años se evaluaron 3: Año 2017, 2018 y 2020

Resultados para los años 2017, 2018 y 2020:

Año	MAE	MSE	RMSE
2020	0.26601243019104004	0.1771623194217682	0.42090654373168945
2018	0.2776103615760803	0.1740521788597107	0.4171956181526184
2017	0.2798231244087219	0.17468659579753876	0.4179552495479584

> Dando con el menor error al año 2020 con 26% de error

Predicciones con el modelo de regresión lógistica ROA

Predicciones con el modelo de regresión lógistica ROA - Categoria

```
epoch:2640 loss=0.6049321293830872 accuracy=0.9662558436393738
```

```python
def categoria_rlg():
  files_joined = os.path.join('/content/drive/MyDrive/Inteligencia Artificial/Segundo Parcial/Proyecto', "indicadores*xlsx")
  list_files = glob.glob(files_joined)
  dfp = pd.concat(map(pd.read_excel, list_files), ignore_index=True)
  # Remover Outliers
  dfp = dfp[(dfp['ROE'] > -1) & (dfp['ROE'] < 1) & (dfp['ROA'] > -1) & (dfp['ROA'] < 1) & (dfp['ENDEUDAMIENTO DEL ACTIVO'] < 6) & (dfp['RENTABILIDAD NETA DE VENTAS'] > -50) &
            (dfp['UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS '] > -5) & (dfp['UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS '] < 5) & (dfp['RENTABLIDAD FINANCIERA'] < 500)]

  dfp['ROA_DIS'] = pd.qcut(dfp['ROA'], 3, labels=False)
  #dfp["DESCRIPCIÓN RAMA"] = dfp[["DESCRIPCIÓN RAMA"]].values
  metrics = {"Categoria": [], "MAE": [], "MSE": [], "RMSE": [], "ACC": []}
  dfp['DESCRIPCIÓN RAMA'].replace(desc, desc_good, inplace=True)
  grouped = dfp.groupby("DESCRIPCIÓN RAMA")
  for name, group in grouped:
      inputs_rlg = group[['RENTABILIDAD NETA DEL ACTIVO',  'UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS ', 'RENTABLIDAD FINANCIERA', 'RENTABILIDAD NETA DE VENTAS']].values
      targets_rlg = group[['ROA_DIS']].values
      # Escalando los datos
      scaler = StandardScaler()
      inputs = scaler.fit_transform(inputs_rlg)
      # Transformando los datos a tensores
      inputs_rlg = torch.from_numpy(inputs.astype(np.float32))
      targets_rlg = torch.from_numpy(targets_rlg.astype(np.int64))
      # Creando el conjunto de datos de test
      dataset_test = TensorDataset(inputs_rlg, targets_rlg)
      test_loader = DataLoader(dataset_test, batch_size=bs, shuffle=True)
      # Evaluando el modelo
      y_pred = []
      y_true = []
      model_rlg.train(False)
      for inputs, targets in test_loader:
        y_hat_test = model_rlg(inputs).data.numpy()
        y_hat_class = np.argmax(y_hat_test, axis=1)
        y_pred.extend(y_hat_class)
        y_true.extend(targets.numpy())
      # Calculando metricas
      mae = mean_absolute_error(y_true=y_true, y_pred=y_pred)
      mse = mean_squared_error(y_true=y_true, y_pred=y_pred, squared=True)
      rmse = mean_squared_error(y_true=y_true, y_pred=y_pred, squared=False)
      acc = accuracy_score(y_true, y_pred)
      # Imprimiendo los resultados
      metrics["Categoria"].append(name)
      metrics["MAE"].append(mae)
      metrics["MSE"].append(mse)
      metrics["RMSE"].append(rmse)
      metrics["ACC"].append(acc)

  metrics_df_ = pd.DataFrame(metrics)
  return metrics_df_
```

```
1 metrics_rlg = categoria_rlg()
```

```
1 metrics_rlg.sort_values("ACC", ascending=False)
```

	Categoria	MAE	MSE	RMSE	ACC
9	COMERCIO AL POR MAYOR Y AL POR MENOR REPARACIÓ...	0.018918	0.018918	0.137544	0.981082
7	AGRICULTURA, GANADERÍA, SILVICULTURA Y PESCA.	0.027671	0.027671	0.166345	0.972329
12	ENSEÑANZA.	0.086716	0.086716	0.294476	0.913284
5	ACTIVIDADES INMOBILIARIAS.	0.113358	0.113358	0.336687	0.886642
14	INDUSTRIAS MANUFACTURERAS.	0.122916	0.122916	0.350594	0.877084
3	ACTIVIDADES DE SERVICIOS ADMINISTRATIVOS Y DE ...	0.127391	0.127391	0.356919	0.872609
13	EXPLOTACIÓN DE MINAS Y CANTERAS.	0.128238	0.128238	0.358103	0.871762
16	OTRAS ACTIVIDADES DE SERVICIOS.	0.211382	0.211382	0.459763	0.788618
15	INFORMACIÓN Y COMUNICACIÓN.	0.212832	0.212832	0.461337	0.787168
11	DISTRIBUCIÓN DE AGUA ALCANTARILLADO, GESTIÓN D...	0.215844	0.215844	0.464590	0.784156
1	ACTIVIDADES DE ATENCIÓN DE LA SALUD HUMANA Y D...	0.226415	0.226415	0.475831	0.773585
10	CONSTRUCCIÓN.	0.229065	0.229065	0.478608	0.770935
8	ARTES, ENTRETENIMIENTO Y RECREACIÓN.	0.262275	0.262275	0.512128	0.737725
18	TRANSPORTE Y ALMACENAMIENTO.	0.283174	0.283174	0.532141	0.716826
4	ACTIVIDADES FINANCIERAS Y DE SEGUROS.	0.297963	0.297963	0.545860	0.702037
17	SUMINISTRO DE ELECTRICIDAD, GAS, VAPOR Y AIRE ...	0.310962	0.310962	0.557640	0.689038
6	ACTIVIDADES PROFESIONALES, CIENTÍFICAS Y TÉCNI...	0.314942	0.314942	0.561197	0.685058
0	ACTIVIDADES DE ALOJAMIENTO Y DE SERVICIO DE CO...	0.343967	0.344482	0.586926	0.656290
2	ACTIVIDADES DE ORGANIZACIONES Y ÓRGANOS EXTRAT...	0.500000	0.500000	0.707107	0.500000

```
epoch:3225 loss=0.5888996124267578 accuracy=0.9906057715415955
```

Predicciones - Regresión Logistica

- **Predicciones con el modelo de regresión Logistica ROA - Categoria**

Los 3 grupos con mayor Acc son:

Num	Categoria	MAE	MSE	RMSE	ACC
9	COMERCIO AL POR MAYOR Y AL POR MENOR REPARACIÓN DE VEHÍCULOS DE MOTOR Y MOTOCICLETAS.	0.0189	0.0189	0.1375	0.981
7	AGRICULTURA, GANADERÍA, SILVICULTURA Y PESCA.	0.0277	0.0277	0.1663	0.972
12	ENSEÑANZA.	0.0867	0.0867	0.2945	0.913

Los 3 grupos con menor Acc son:

Num	Categoria	MAE	MSE	RMSE	ACC
6	ACTIVIDADES PROFESIONALES, CIENTÍFICAS Y TÉCNICAS.	0.3149	0.3149	0.5612	0.685
0	ACTIVIDADES DE ALOJAMIENTO Y DE SERVICIO DE COMIDAS.	0.344	0.344	0.5869	0.656
2	ACTIVIDADES DE ORGANIZACIONES Y ÓRGANOS EXTRATERRITORIALES.	0.500	0.500	0.7071	0.500

- El modelo tiene un mejor rendimiento en la predicción de las categorías, ya que tienen una mayor precisión (ACC más alta):
  - COMERCIO AL POR MAYOR Y AL POR MENOR REPARACIÓN DE VEHÍCULOS DE MOTOR Y MOTOCICLETAS.
  - AGRICULTURA, GANADERÍA, SILVICULTURA Y PESCA.
  - ENSEÑANZA.
- El modelo tiene un rendimiento peor en la predicción de las categorías, ya que tienen una precisión más baja (ACC más baja):
  - ACTIVIDADES PROFESIONALES, CIENTÍFICAS Y TÉCNICAS.
  - ACTIVIDADES DE ALOJAMIENTO Y DE SERVICIO DE COMIDAS.
  - ACTIVIDADES DE ORGANIZACIONES Y ÓRGANOS EXTRATERRITORIALES.

Predicciones con el modelo de regresión lógistica ROA - Año

```
epoch:3415 loss=0.5866679549217224 accuracy=0.9911318421363831
```

```
1 def procesar_rlg(año):
2     df = pd.read_excel(f'/content/drive/MyDrive/Inteligencia Artificial/Segundo Parcial/Proyecto/indicadores{año}_cia.xlsx')
3     # Remover Outliers
4     df = df[(df['ROE'] > -1) & (df['ROE'] < 1) & (df['ROA'] > -1) & (df['ROA'] < 1) & (df['ENDEUDAMIENTO DEL ACTIVO'] < 6) & (df['RENTABILIDAD NETA DE VENTAS'] > -50) &
5             (df['UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS '] > -5) & (df['UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS '] < 5) & (df['RENTABLIDAD FINANCIERA'] < 500)]
6     df['ROA_DIS'] = pd.qcut(df['ROA'], 3, labels=False)
7     # Input y Targets
8     inputs_rlg = df[['RENTABILIDAD NETA DEL ACTIVO',  'UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS ', 'RENTABLIDAD FINANCIERA', 'RENTABILIDAD NETA DE VENTAS']].values
9     targets_rlg = df[['ROA_DIS']].values
10    # Escalando los datos
11    scaler = StandardScaler()
12    inputs = scaler.fit_transform(inputs_rlg)
13    # Transformando los datos a tensores
14    inputs_rlg = torch.from_numpy(inputs.astype(np.float32))
15    targets_rlg = torch.from_numpy(targets_rlg.astype(np.int64))
16    # Creando el conjunto de datos de test
17    dataset_test = TensorDataset(inputs_rlg, targets_rlg)
18    test_loader = DataLoader(dataset_test, batch_size=bs, shuffle=True)
```

```
    # Evaluando el modelo
    y_pred = []
    y_true = []
    model_rlg.train(False)
    for inputs, targets in test_loader:
        y_hat_test = model_rlg(inputs).data.numpy()
        y_hat_class = np.argmax(y_hat_test, axis=1)
        y_pred.extend(y_hat_class)
        y_true.extend(targets.numpy())
    # Calculando metricas
    mae = mean_absolute_error(y_true=y_true, y_pred=y_pred)
    mse = mean_squared_error(y_true=y_true, y_pred=y_pred, squared=True)
    rmse = mean_squared_error(y_true=y_true, y_pred=y_pred, squared=False)
    acc = accuracy_score(y_true, y_pred)

    # Imprimiendo los resultados
    print(f"\nResultados para el año {año} - con {len(targets_rlg)} datos:")
    print(f"MAE: {mae}")
    print(f"MSE: {mse}")
    print(f"RMSE: {rmse}")
    print('Accuracy: {:.2f}%'.format(acc*100))
```

```
epoch:3615 loss=0.5846129059791565 accuracy=0.9918833374977112
```

```
#Arreglo años
years = [2017, 2018, 2019, 2020]

#Bucle para ejecutar proceso para cada año
for year in years:
    procesar_rlg(year)
```

```
Resultados para el año 2017 - con 46592 datos:
MAE: 0.14144059065934067
MSE: 0.14144059065934067
RMSE: 0.3760858820260881
Accuracy: 85.86%

Resultados para el año 2018 - con 48087 datos:
MAE: 0.1453407365816125
MSE: 0.1453407365816125
RMSE: 0.38123580180986744
Accuracy: 85.47%

Resultados para el año 2019 - con 48657 datos:
MAE: 0.12692932157757364
MSE: 0.12692932157757364
RMSE: 0.3562714156055375
Accuracy: 87.31%

Resultados para el año 2020 - con 45551 datos:
MAE: 0.07703453272156484
MSE: 0.07703453272156484
RMSE: 0.2775509551804224
Accuracy: 92.30%
```

```
epoch:3780 loss=0.5830987691879272 accuracy=0.9924845695495605
```

Predicciones - Regresión Logistica

- **Predicciones con el modelo de regresión Logistica ROA - Año**
- A partir de los datos presentados en la tabla, se puede concluir: El modelo tuvo un mejor rendimiento en el año 2020 en comparación con los años 2017 y 2018. Esto se refleja en un MAE, MSE y RMSE más bajos y en un mayor porcentaje de accuracy.
- Esto sugiere que el modelo tuvo un mejor desempeño en la predicción de los datos del año 2020 en comparación con los datos de los años 2017 y 2018.

Año	Nº de Datos	MAE	MSE	RMSE	Accuracy (%)
2020	45551	0.0769686724770038	0.0769686724770038	0.27743228448939355	92.30
2018	48087	0.1453407365816125	0.1453407365816125	0.38123580180986744	85.47
2017	46592	0.14144059065934067	0.14144059065934067	0.3760858820260881	85.86

Conclusiones preeliminares - Técnicas

```
epoch:3890 loss=0.5821661353111267 accuracy=0.992559731006622
```

Análisis Predictivo mediante el modelo de regresión lineal - regresión logística para evaluar el ROE/ROA en las empresas del Ecuador durante los años 2017-2020

Pregunta 1: ¿Qué año predice mejor y peor los modelos creado?

- Se podría concluir con certeza que el modelo de regresión logística tiene un mejor desempeño en la predicción de los datos del año 2020 en comparación con los años 2017 y 2018.
- Esto podría deberse a que el modelo de regresión logística es mejor para manejar problemas de clasificación y los datos del año 2020 presentaban una distribución de clases diferente o más compleja que los datos de los años 2017 y 2018.
- Por otro lado, el modelo de regresión lineal tendría un desempeño similar en las predicciones para los años 2017, 2018 y 2020, ya que es un modelo diseñado para manejar problemas de regresión y los datos no presentan una gran variación en su distribución.

Pregunta 2: ¿Qué categoría de empresas predice mejor y peor los modelos creados?

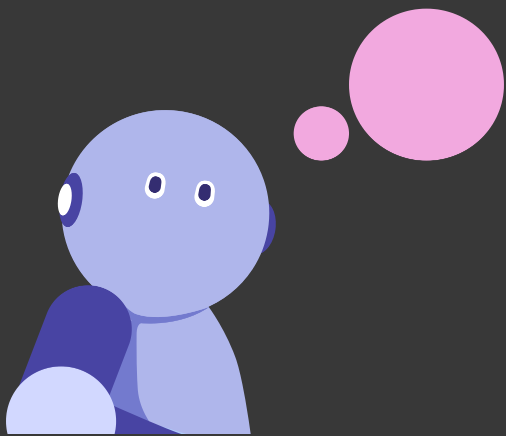
- Con base en los datos presentados, se puede concluir que el modelo de regresión lineal tiene un mejor rendimiento en la predicción de las categorías "ACTIVIDADES INMOBILIARIAS.", "ACTIVIDADES DE SERVICIOS ADMINISTRATIVOS Y DE APOYO" y "ACTIVIDADES DE ALOJAMIENTO Y DE SERVICIO DE COMIDAS.", ya que tienen un error menor en las medidas MAE, MSE y RMSE. Por otro lado, el modelo tiene un rendimiento peor en la predicción de las categorías "SUMINISTRO DE ELECTRICIDAD, GAS, VAPOR Y AIRE ACONDICIONADO",

"OTRAS ACTIVIDADES DE SERVICIOS." y "ACTIVIDADES DE ORGANIZACIONES Y ÓRGANOS EXTRATERRITORIALES.", ya que tienen un error mayor en las medidas MAE, MSE y RMSE.

- En cuanto al modelo de regresión logística, se observa que tiene un mejor rendimiento en la predicción de las categorías "COMERCIO AL POR MAYOR Y AL POR MENOR REPARACIÓN DE VEHÍCULOS DE MOTOR Y MOTOCICLETAS.", "AGRICULTURA, GANADERÍA, SILVICULTURA Y PESCA." y "ENSEÑANZA.", ya que tienen una precisión más alta (ACC más alta). Por otro lado, el modelo tiene un rendimiento peor en la predicción de las categorías "ACTIVIDADES PROFESIONALES, CIENTÍFICAS Y TÉCNICAS.", "ACTIVIDADES DE ALOJAMIENTO Y DE SERVICIO DE COMIDAS." y "ACTIVIDADES DE ORGANIZACIONES Y ÓRGANOS EXTRATERRITORIALES.", ya que tienen una precisión más baja (ACC más baja).

Resumen:

- Con respecto a los Años que se predice mejor el modelo de regresión lógistica todos los años predecidos
- La categoria que predicen mejor ambos modelos es: L - ACTIVIDADES INMOBILIARIAS.
- La peor en ambos fue la M y U: "ACTIVIDADES PROFESIONALES, CIENTÍFICAS Y TÉCNICAS" y "ACTIVIDADES DE ORGANIZACIONES Y ÓRGANOS EXTRATERRITORIALES.", puede deberse a la poca información que se prevee



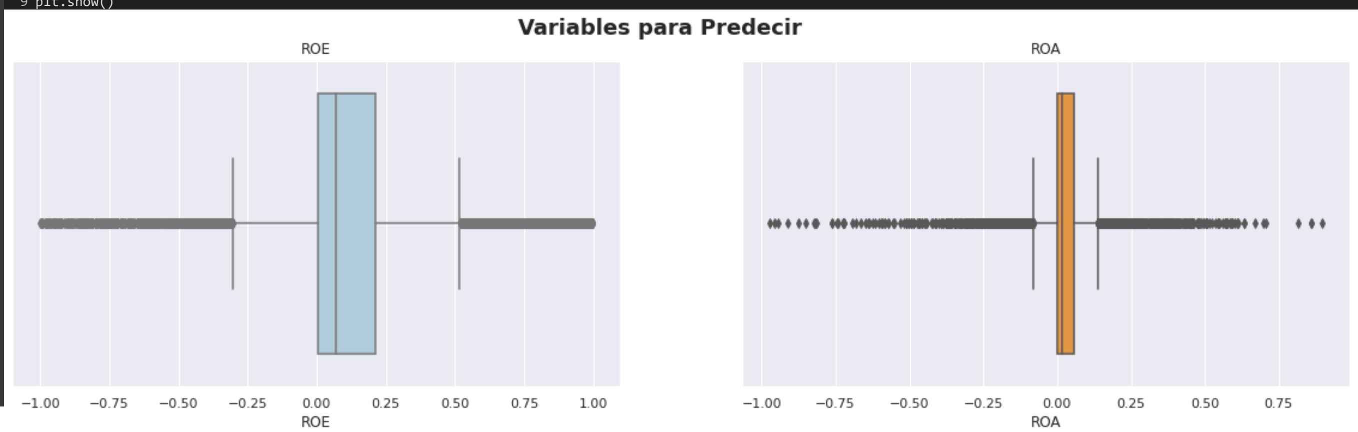
¿Y ahora que?

- Para este punto se pretende dar una explicación en base a los datos obtenidos del porque las categorias tienen menor rendimiento, para esto se realiza un analisis Profundo de solo esas categoria con las variables de entradas y salidas
- Las categorias a Analizar son:
  - ACTIVIDADES PROFESIONALES, CIENTÍFICAS Y TÉCNICAS.
  - ACTIVIDADES DE ORGANIZACIONES Y ÓRGANOS EXTRATERRITORIALES.

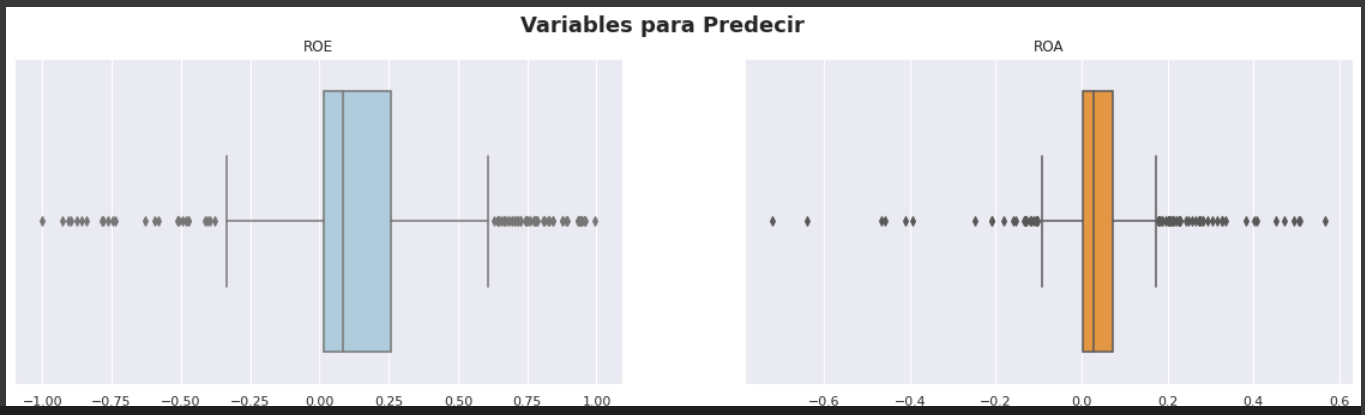
Analisis Profundo

ACTIVIDADES PROFESIONALES, CIENTÍFICAS Y TÉCNICAS.

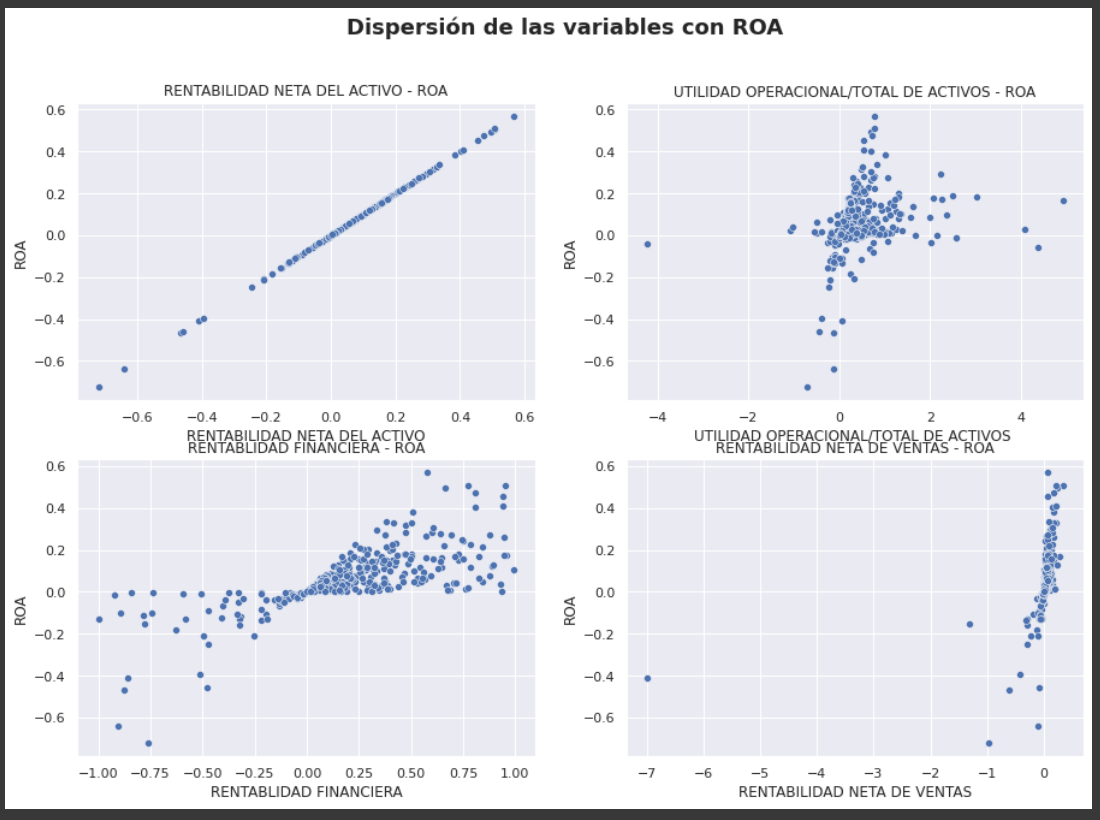
```
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(20,5))

sns.boxplot(x = df['ROE'], ax=ax1, palette="Paired")
sns.boxplot(x =  df['ROA'],ax=ax2, palette="YlOrBr")

fig.suptitle("Variables para Predecir", fontsize=18, fontweight='bold')
ax1.set_title("ROE")
ax2.set_title("ROA")
plt.show()
```



```
df_pro = df[df['DESCRIPCIÓN RAMA']=='ACTIVIDADES PROFESIONALES, CIENTÍFICAS Y TÉCNICAS.']

fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(20,5))

sns.boxplot(x = df_pro['ROE'], ax=ax1, palette="Paired")
sns.boxplot(x =  df_pro['ROA'],ax=ax2, palette="YlOrBr")

fig.suptitle("Variables para Predecir", fontsize=18, fontweight='bold')
ax1.set_title("ROE")
ax2.set_title("ROA")
plt.show()
```




```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2, ncols=2, figsize=(15,10))

sns.scatterplot(x = df_pro['RENTABILIDAD NETA DEL ACTIVO'],y = df['ROA'], ax=ax1)
sns.scatterplot(x = df_pro['UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS '],y = df['ROA'], ax=ax2)
sns.scatterplot(x = df_pro['RENTABLIDAD FINANCIERA'],y = df['ROA'], ax=ax3)
sns.scatterplot(x = df_pro['RENTABILIDAD NETA DE VENTAS'],y = df['ROA'], ax=ax4)

fig.suptitle("Dispersión de las variables con ROA", fontsize=18, fontweight='bold')
ax1.set_title("RENTABILIDAD NETA DEL ACTIVO - ROA")
ax2.set_title("UTILIDAD OPERACIONAL/TOTAL DE ACTIVOS - ROA")
ax3.set_title("RENTABLIDAD FINANCIERA - ROA")
ax4.set_title("RENTABILIDAD NETA DE VENTAS - ROA")
plt.show()
```



ACTIVIDADES DE ORGANIZACIONES Y ÓRGANOS EXTRATERRITORIALES.

- Para esta categoria se determina que los registros o datos de estas categorias son insuficientes para que la predicción sea correcta

Conclusiones Finales

Pregunta 1: ¿Qué año predice mejor y peor los modelos creado?

- Los modelos de regresión lineal y logística fueron utilizados para predecir diferentes años y los resultados muestran que ambos modelos tuvieron un mejor desempeño en la predicción de los datos del año 2020.
- Además de tener el mismo rendimiento se en ambos modelos se da como definido que los datos estan en la misma página, es decir, no cambian en su totalidad

Pregunta 2: ¿Qué categoría de empresas predice mejor y peor los modelos creados?

Categorias que Predicen Mejor: "COMERCIO AL POR MAYOR Y AL POR MENOR REPARACIÓN DE VEHÍCULOS DE MOTOR Y MOTOCICLETAS.", "ACTIVIDADES INMOBILIARIAS." y "ENSEÑANZA."

Categorias que Predicen Peor:

- ACTIVIDADES PROFESIONALES, CIENTÍFICAS Y TÉCNICAS. - Por tener datos de entradas en con tendencia positiva, es decir, que en las variables escogidas, hay datos de entradas positivos con ROA y ROE negativos, por lo que se recomienda evaluar un modelo por separado, puede deberse a la recoleccion de datos, una mal limpieza de datos o incluso diferentes forma de obtener el ROA y ROE
- ACTIVIDADES DE ORGANIZACIONES Y ÓRGANOS EXTRATERRITORIALES. - Por falta de datos