UTADEO
UNIVERSIDAD DE BOGOTÁ JORGE TADEO LOZANO

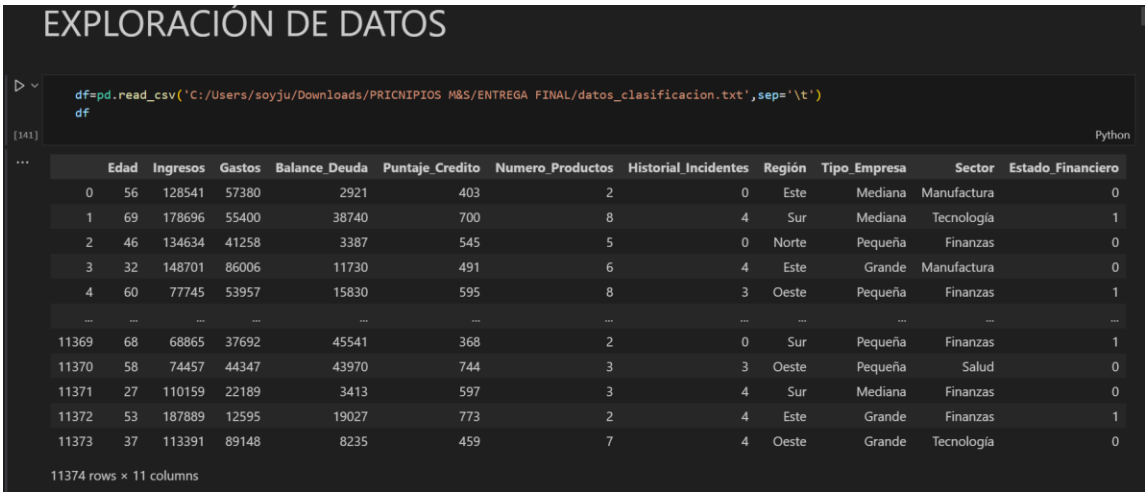
PRINCIPIOS DE MODELADO Y SIMULACIÓN

SEGUNDA ENTREGA GRUPO 5

Paula Juliana Cañón Ávila[1], Valentina Pérez Yaya[2], Laura Lucia Cuellar Morales[3].

paula.canona@utadeo.edu.co, valentina.perezy@utadeo.edu.co, laural.cuellarm@utadeo.edu.co.

1. Análisis de los datos y preprocesamiento.



EXPLORACIÓN DE DATOS

```
df=pd.read_csv('C:/Users/soyju/Downloads/PRICNIPIOS M&S/ENTREGA FINAL/datos_clasificacion.txt',sep='\t')
df
```

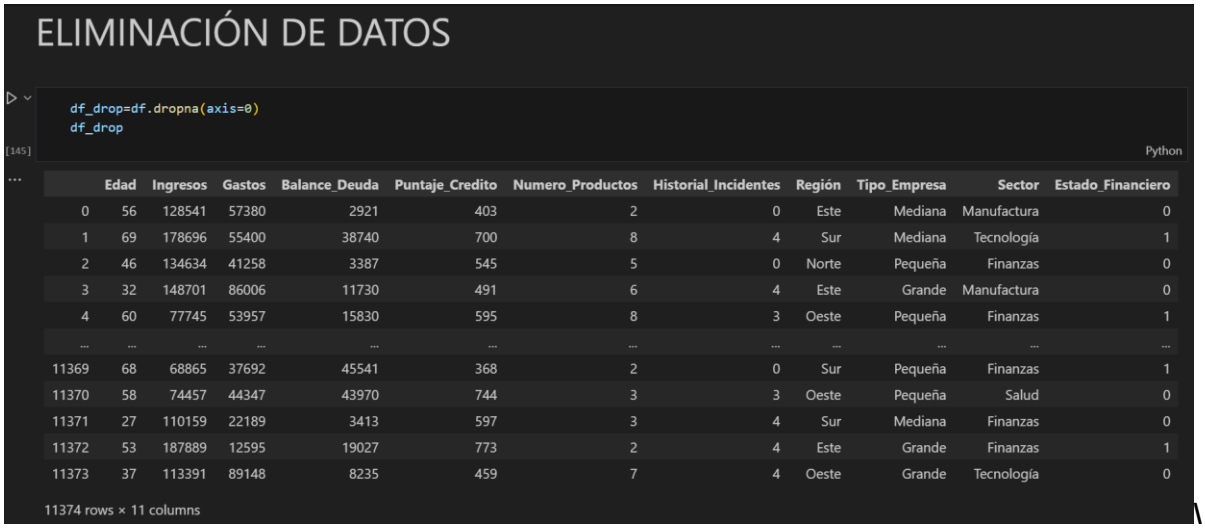
[141] Python

	Edad	Ingresos	Gastos	Balance_Deuda	Puntaje_Credito	Numero_Productos	Historial_Incidentes	Región	Tipo_Empresa	Sector	Estado_Financiero
0	56	128541	57380	2921	403	2	0	Este	Mediana	Manufactura	0
1	69	178696	55400	38740	700	8	4	Sur	Mediana	Tecnología	1
2	46	134634	41258	3387	545	5	0	Norte	Pequeña	Finanzas	0
3	32	148701	86006	11730	491	6	4	Este	Grande	Manufactura	0
4	60	77745	53957	15830	595	8	3	Oeste	Pequeña	Finanzas	1
...	...	...	...	...	...	...	...	...	...	...	...
11369	68	68865	37692	45541	368	2	0	Sur	Pequeña	Finanzas	1
11370	58	74457	44347	43970	744	3	3	Oeste	Pequeña	Salud	0
11371	27	110159	22189	3413	597	3	4	Sur	Mediana	Finanzas	0
11372	53	187889	12595	19027	773	2	4	Este	Grande	Finanzas	1
11373	37	113391	89148	8235	459	7	4	Oeste	Grande	Tecnología	0

11374 rows × 11 columns

Imagen 1. Carga de datos, csv

Primeramente, se realiza una exploración de datos y se obtiene un primer análisis de que datos se utilizarán.



ELIMINACIÓN DE DATOS

```
df_drop=df.dropna(axis=0)
df_drop
```

[145] Python

	Edad	Ingresos	Gastos	Balance_Deuda	Puntaje_Credito	Numero_Productos	Historial_Incidentes	Región	Tipo_Empresa	Sector	Estado_Financiero
0	56	128541	57380	2921	403	2	0	Este	Mediana	Manufactura	0
1	69	178696	55400	38740	700	8	4	Sur	Mediana	Tecnología	1
2	46	134634	41258	3387	545	5	0	Norte	Pequeña	Finanzas	0
3	32	148701	86006	11730	491	6	4	Este	Grande	Manufactura	0
4	60	77745	53957	15830	595	8	3	Oeste	Pequeña	Finanzas	1
...	...	...	...	...	...	...	...	...	...	...	...
11369	68	68865	37692	45541	368	2	0	Sur	Pequeña	Finanzas	1
11370	58	74457	44347	43970	744	3	3	Oeste	Pequeña	Salud	0
11371	27	110159	22189	3413	597	3	4	Sur	Mediana	Finanzas	0
11372	53	187889	12595	19027	773	2	4	Este	Grande	Finanzas	1
11373	37	113391	89148	8235	459	7	4	Oeste	Grande	Tecnología	0

11374 rows × 11 columns

Imagen 2. Eliminación de datos.

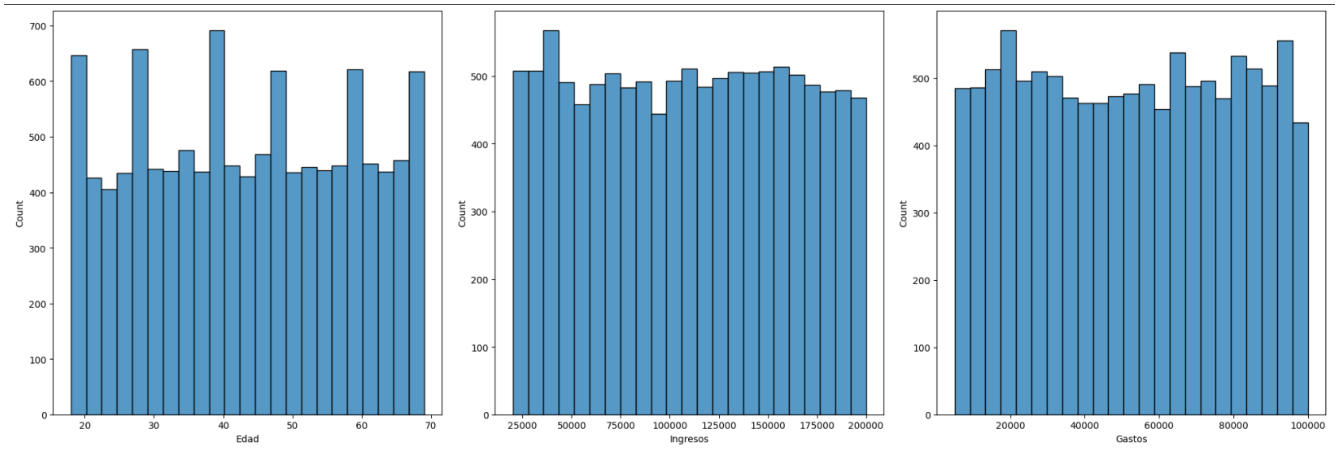
Para esta parte se utiliza **df.dropna(axis=0):** Esta función elimina todas las filas (axis=0) que contienen valores faltantes (NaN) en el DataFrame df. Esto es útil para limpiar los datos antes de realizar análisis o modelado, asegurando que no haya valores faltantes que puedan afectar los resultados.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11374 entries, 0 to 11373
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Edad                  11374 non-null  int64
 1   Ingresos              11374 non-null  int64
 2   Gastos                11374 non-null  int64
 3   Balance_Deuda         11374 non-null  int64
 4   Puntaje_Credito       11374 non-null  int64
 5   Numero_Productos      11374 non-null  int64
 6   Historial_Incidentes  11374 non-null  int64
 7   Región                11374 non-null  object
 8   Tipo_Empresa          11374 non-null  object
 9   Sector                11374 non-null  object
 10  Estado_Financiero     11374 non-null  int64
dtypes: int64(8), object(3)
memory usage: 977.6+ KB
```

Imagen 3. Total de columnas.

Visualización de Datos: Una vez cargados los datos, se muestra el contenido del DataFrame df. La tabla resultante tiene 11 columnas con los siguientes encabezados: ‘Edad’, ‘Ingresos’, ‘Gastos’, ‘Deuda’, ‘Puntaje Crédito’, ‘Numero Productos’, ‘Historial Crediticio’, ‘Tipo Empleo’, ‘Sector Industria’, ‘Estado Financiero’ y ‘Manufactura’. La tabla contiene datos numéricos y categóricos para diferentes individuos o entidades



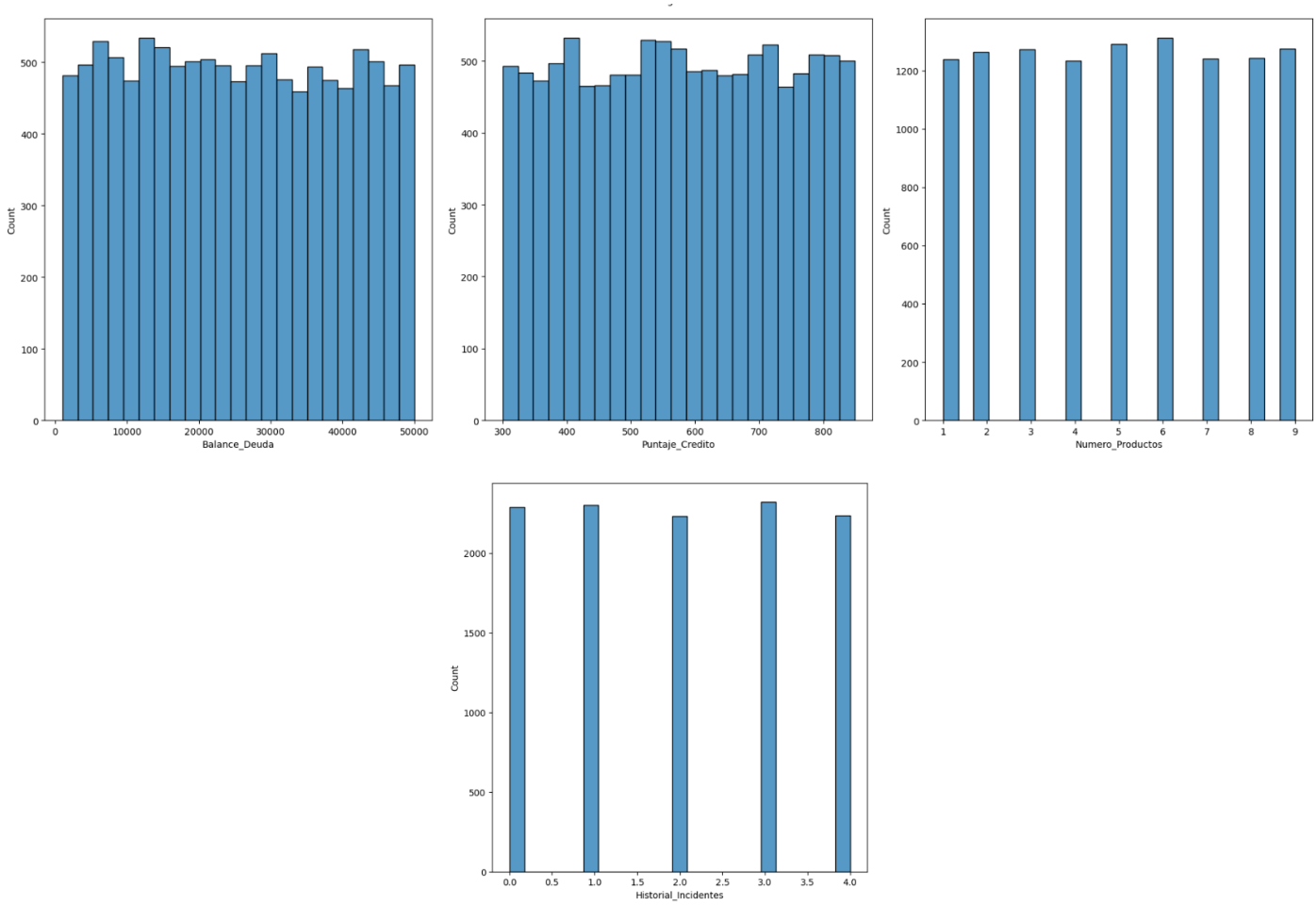


Imagen 4. Graficas de análisis de datos

Con estos histogramas analizamos la distribución de nuestro conjunto de datos. Luego de esto se realizó el preprocesamiento de los datos.

PREPROSESAMIENTO DE DATOS

ESTANDARIZACIÓN

```
X = df.loc[:, ~df.columns.str.contains('Estado_Financiero')]
Y = df.loc[:, df.columns.str.contains('Estado_Financiero')]
print("Separación de datos usando Pandas")
print(X.shape, Y.shape)
```

152] Python

```
Separación de datos usando Pandas
(11374, 10) (11374, 1)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(X,Y, test_size=0.3, random_state=1,stratify=Y)
```

153] Python

```
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
X_train_ree = mms.fit_transform(X_train)
X_test_ree = mms.transform(X_test)
```

154] Python

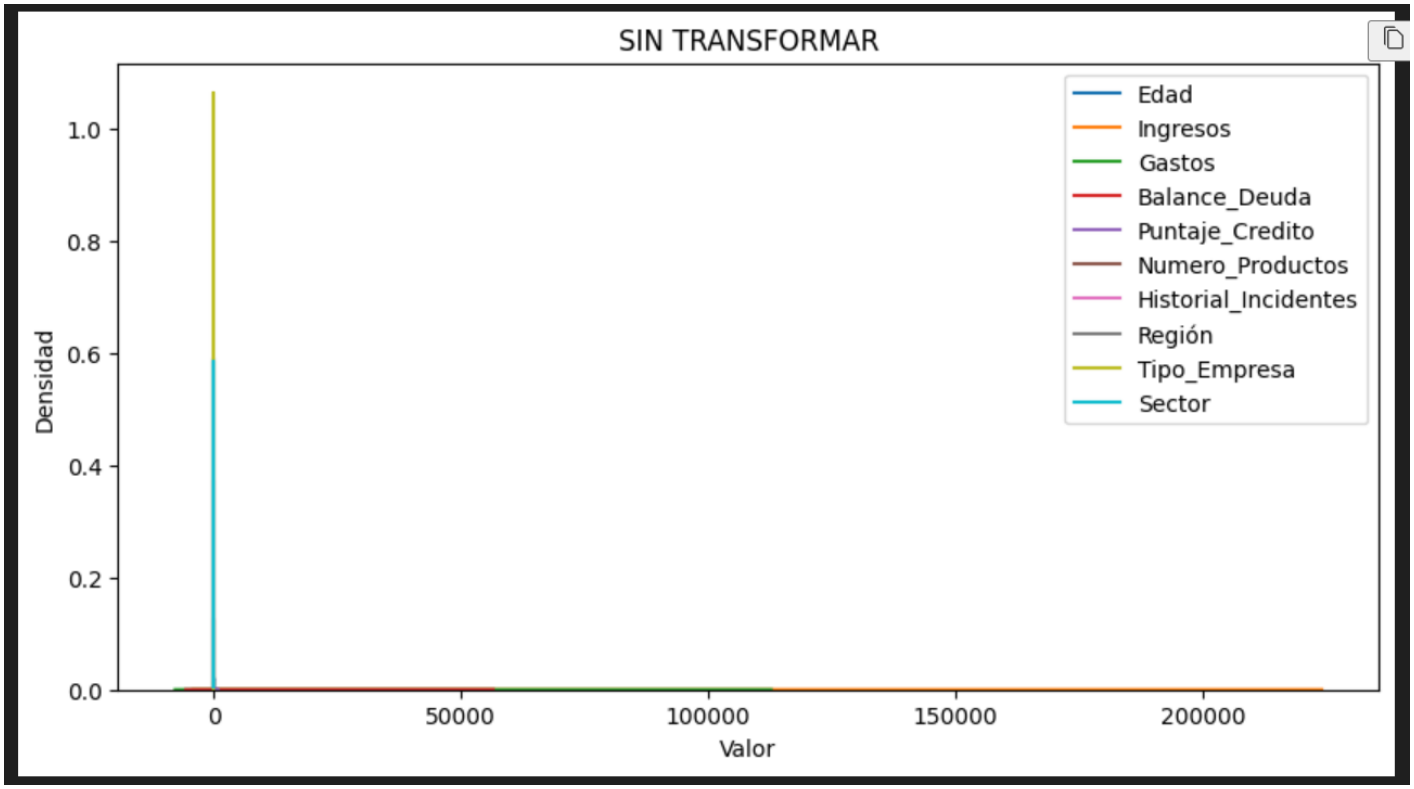


Imagen 5. Datos sin transformar

REDUCCIÓN DE DIMENSIONALIDAD (PCA)

```
[161]
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
X_train_ree = mms.fit_transform(X_train)
X_test_ree = mms.transform(X_test)
```

```
[162]
X_train_ree_df=pd.DataFrame(X_train_ree)
X_train_ree_df
```

	0	1	2	3	4	5	6	7	8	9
0	0.254902	0.177629	0.951398	0.383026	0.287796	0.875	1.00	0.666667	1.0	0.666667
1	0.333333	0.186236	0.305698	0.698045	0.785064	1.000	0.00	0.333333	0.0	0.333333
2	0.705882	0.236195	0.889067	0.047843	0.085610	0.000	0.00	0.333333	1.0	0.333333
3	0.784314	0.168229	0.917369	0.053088	0.577413	0.000	0.75	1.000000	0.5	1.000000
4	0.392157	0.662403	0.192996	0.459260	0.120219	0.500	0.00	0.000000	0.0	0.666667
...	...	...	...	...	...	...	...	...	...	...
9094	0.352941	0.626084	0.267194	0.367637	0.453552	0.625	0.75	0.333333	0.0	1.000000

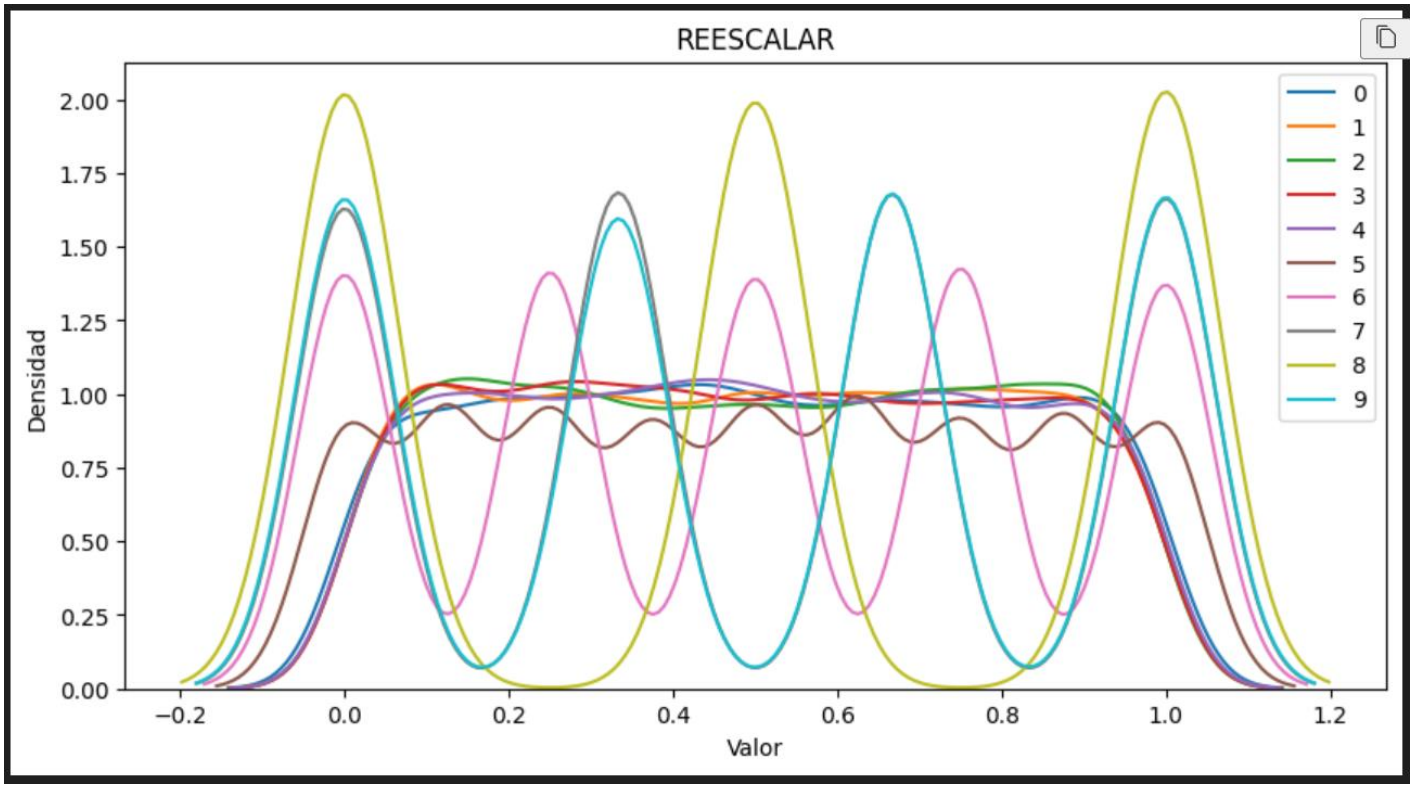


Imagen 6. Datos rescaldados.

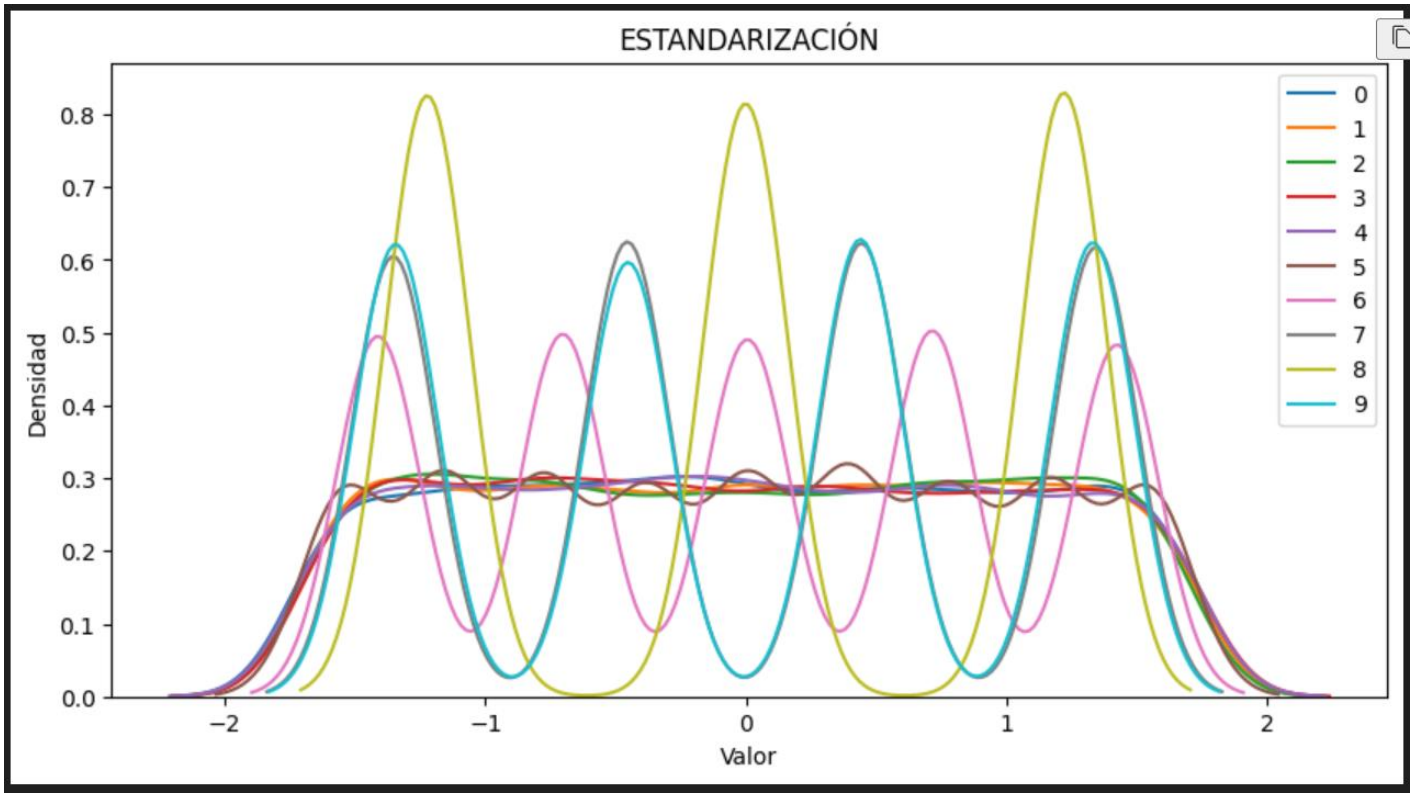


Imagen 7. Datos estandarizados.

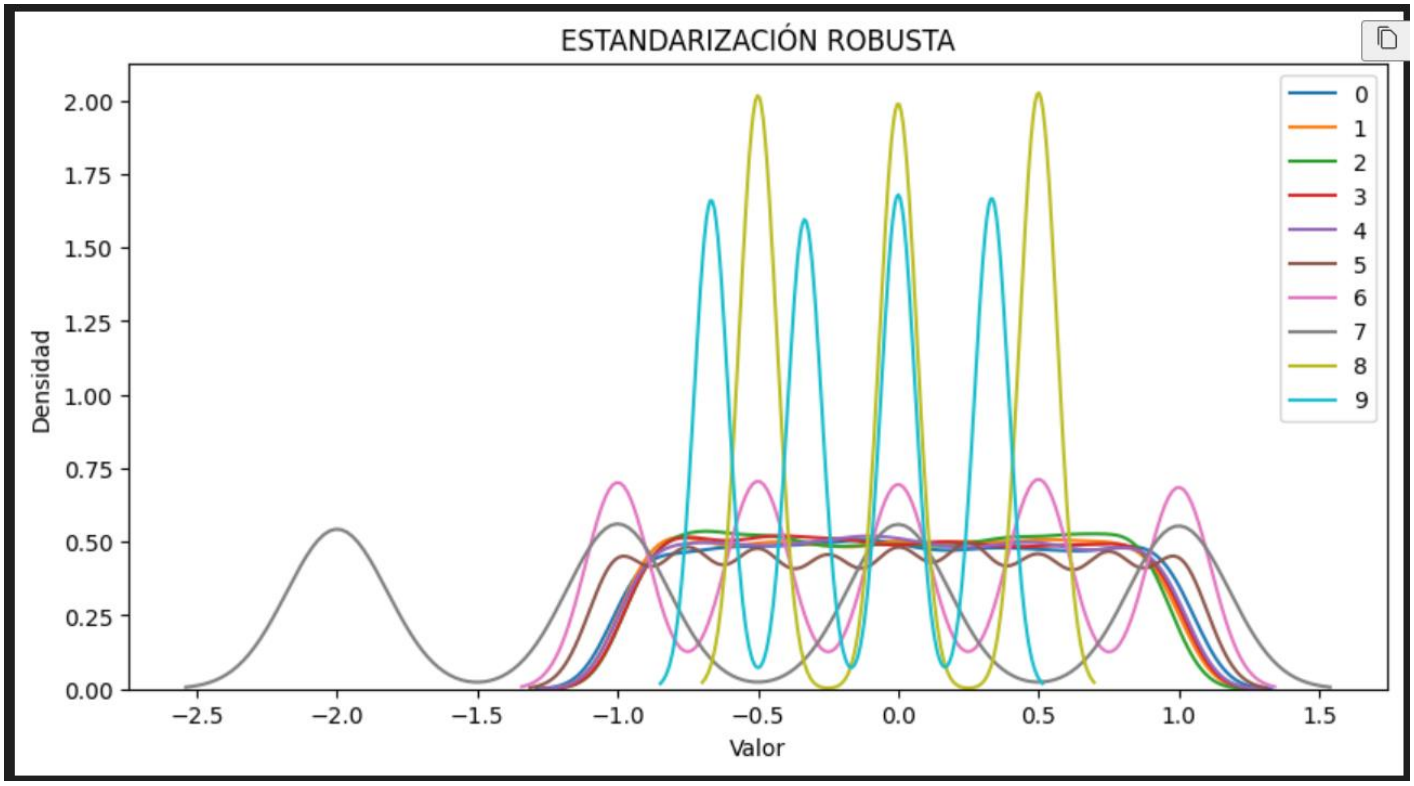


Imagen 8. Estandarización robusta.

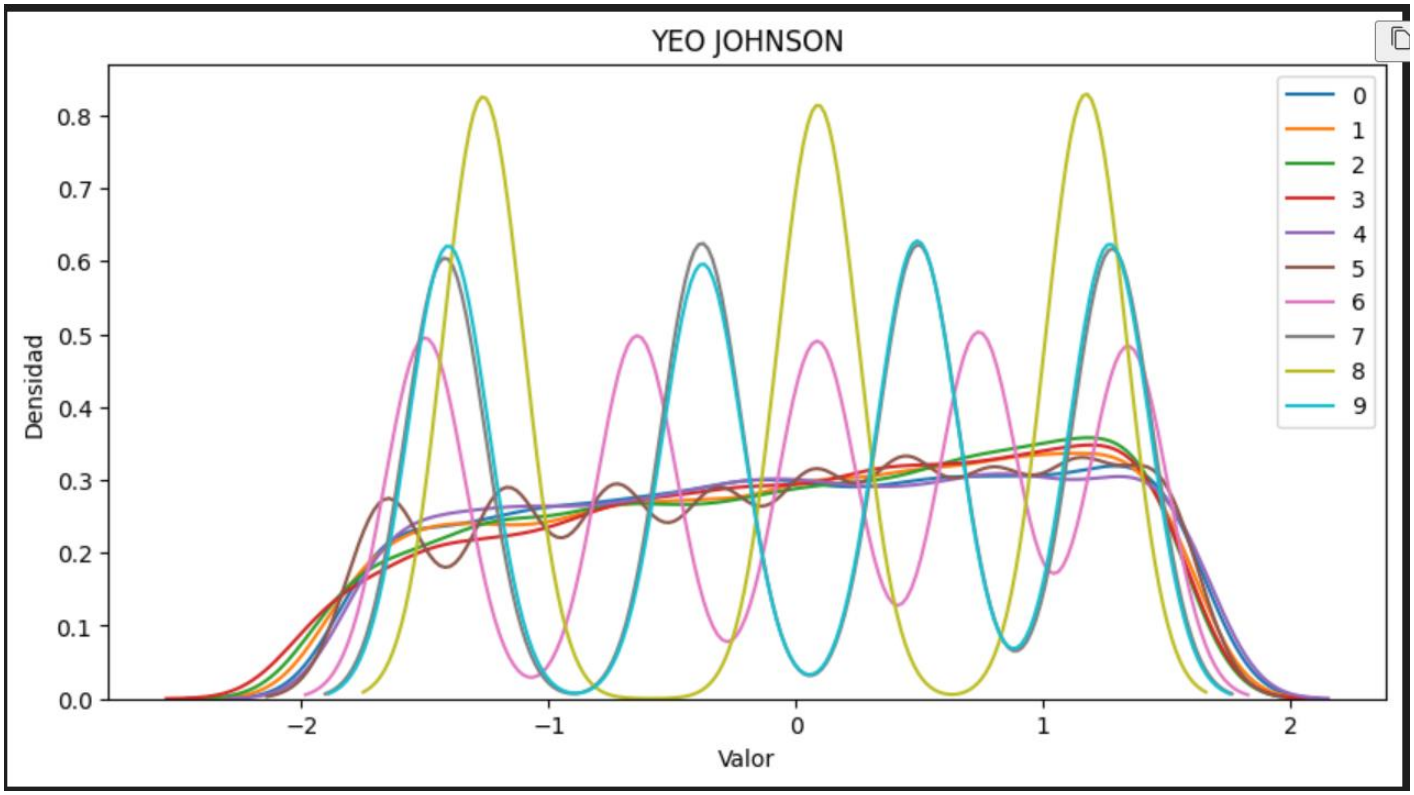


Imagen 9. Yeo Johnson.

Con este modelo de Yeo Johnson buscamos estabilizar la varianza y mejorar la normalidad de los datos. A continuación, te explico sus principales características y aplicaciones.

Transformación de Yeo-Johnson:

- La transformación de Yeo-Johnson se utiliza para hacer que los datos se asemejen más a una distribución normal, lo que puede mejorar la precisión de los modelos estadísticos y de machine learning.
- La gráfica muestra cómo diferentes categorías responden a esta transformación, con variaciones en la densidad a lo largo del rango de valores

2. Comparación de los modelos.

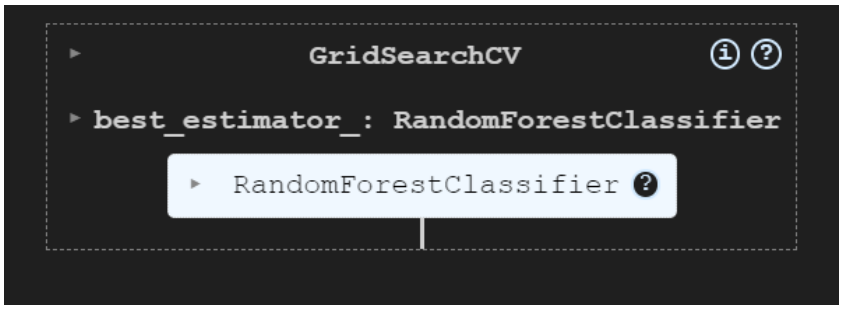
Después de normalizar y estandarizar los datos bajo el método de Yeo Johnson, utilizamos diferentes modelos para predecir los siguientes comportamientos de los datos problema, se implementaron un total de 5 modelos que arrojaron diferentes valores de rendimiento para la predicción de datos y tienen diferentes características como se muestra en la siguiente tabla comparativa:

Modelo	Precisión sin optimización	Tiempo de Entrenamiento	Aplicaciones Comunes	Características
Perceptron	0.4967	Rápido	Clasificación binaria, problemas de datos linealmente separables, reconocimiento de patrones.	Modelo simple de red neuronal, adecuado para datos linealmente separables.
Regression	0.4927	Rápido	Predicción de resultados binarios, análisis de riesgo, marketing, diagnóstico médico.	Bueno para problemas de clasificación binaria, proporciona probabilidades de clase.
SVC	0.4905	Lento	Clasificación de texto y imágenes, detección de fraudes, bioinformática.	Eficaz en espacios de alta dimensión, utiliza diferentes núcleos para la separación de datos.
Tree	0.5011	Moderado	Análisis de crédito, diagnóstico médico, segmentación de clientes, predicción de ventas.	Fácil de interpretar, puede manejar datos categóricos y continuos.

Modelo	Precisión sin optimización	Tiempo de Entrenamiento	Aplicaciones Comunes	Características
Random Forest	0.5248	Moderado	Detección de fraudes, clasificación de imágenes, predicción de enfermedades, análisis de mercado.	Combina múltiples árboles de decisión para mejorar la precisión y reducir el sobreajuste.

3. **Explicación del proceso de optimización y el modelo final seleccionado.**

Para la optimización del modelo, se escogió el método Random Forest ya que en el resultado precisión este fue el de mayor valor (0.5248). Se usó el método de optimización Grid Search el cual es una técnica de validación cruzada. Lo que se realiza es que este modelo ejecuta a través de los diferentes parámetros que se introducen en la cuadrícula de parámetros y extraer los mejores valores y combinaciones de parámetros

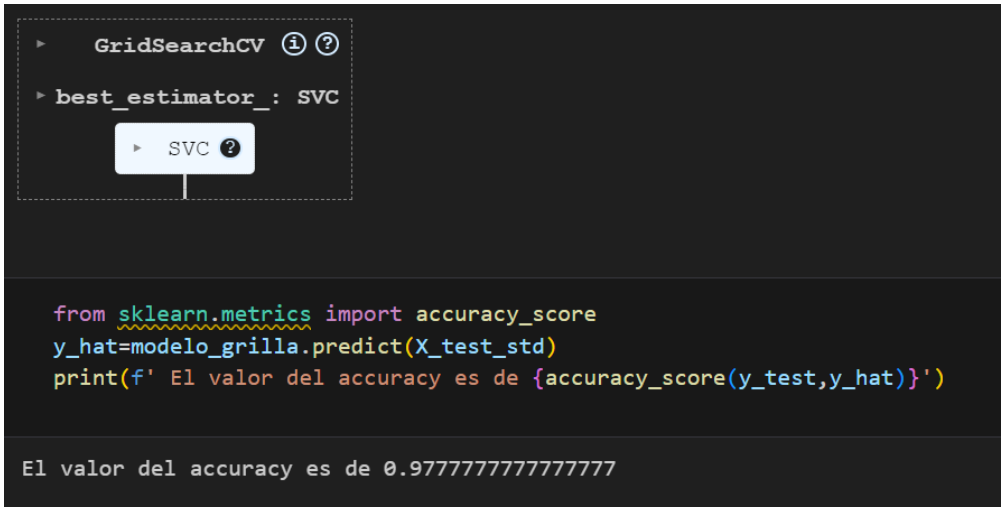


Para obtener el valor con la optimización, se obtuvo un valor accuracy el cual hace referencia a la fracción de predicciones que el modelo realizó correctamente el cual va de un rango de 0 a 1 y este se obtuvo un valor total de 0,97

```
from sklearn.metrics import accuracy_score
y_hat=modelo_grilla.predict(X_test_std)
print(f' El valor del accuracy es de {accuracy_score(y_test,y_hat)}')
```

```
El valor del accuracy es de 0.9777777777777777
```

Por otro lado, también se quiso optimizar el peor modelo que nos arrojó un valor de 0.4905, para el modelo de SCV también se realizó la optimización por el método de grilla.




```
from sklearn.metrics import accuracy_score
y_hat=modelo_grilla.predict(X_test_std)
print(f' El valor del accuracy es de {accuracy_score(y_test,y_hat)}')
```

```
El valor del accuracy es de 0.9777777777777777
```

Después de esta optimización, se obtuvo un resultado también de 0,97. Es decir que nuestra optimización de grilla sirve para optimizar cualquier método que su resultado halla sido bajo sin optimizarlo.