

Juan Esteban Salamanca

Andrés Felipe Fuentes

Laura Valentina Pinto

Entrega final de principios de M&S

En el presente documento se encuentra el análisis de la base de datos de sobre el estado financiero de varias empresas donde contamos con las siguientes variables:

**Variables Numéricas:**

- Edad: Representa la edad de la empresa en años.
- Ingresos: Los ingresos anuales en dólares.
- Gastos: Gastos anuales en dólares.
- Balance Deuda: El balance total de deuda de la empresa.
- Puntaje Crédito: Un puntaje crediticio que varía entre 300 y 850.
- Numero Productos: El número de productos financieros utilizados por la empresa.
- Historial Incidentes: Número de incidentes financieros previos que ha enfrentado la empresa.

**Variables Categóricas:**

- Región: La región geográfica donde opera la empresa (Norte, Sur, Este, Oeste).
- Tipo Empresa: Tipo de empresa según su estructura (Pequeña, Mediana, Grande).
- Sector: El sector económico en el que opera la empresa (Tecnología, Salud, Manufactura, Comercio, Servicios).

**Variable Para Predecir (Objetivo):**

- Estado Financiero: Representa si la empresa es financieramente saludable (1) o si está en riesgo financiero (0).

Por consiguiente, se hace el estudio pertinente para poder predecir si las empresas se encuentran en buen estado o no.

## Exploración de los Datos:

```
df=pd.read_csv('C:/Users/soyju/Downloads/PRINCIPIOS M85/ENTREGA FINAL/datos_clasificacion.txt',sep='\t')
df
```

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: Datos obtenidos de la base de datos.

```
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 2: Tipo de dato de cada variable.

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

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 3: Eliminación de datos.

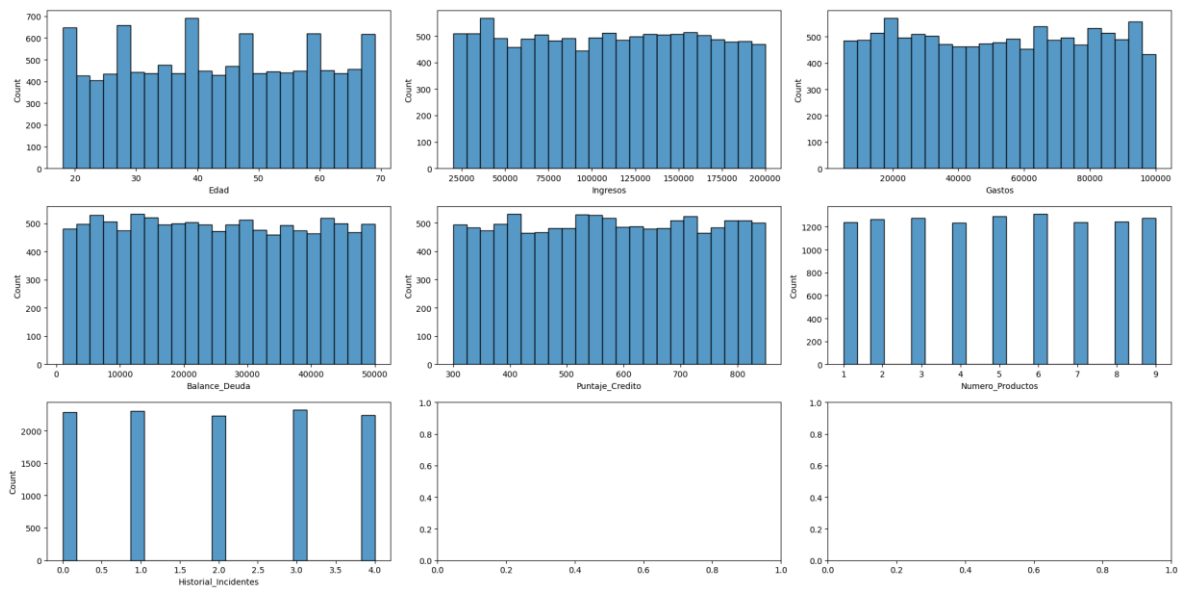


Imagen 4: Gráficas de conteo de datos de variables numéricas.

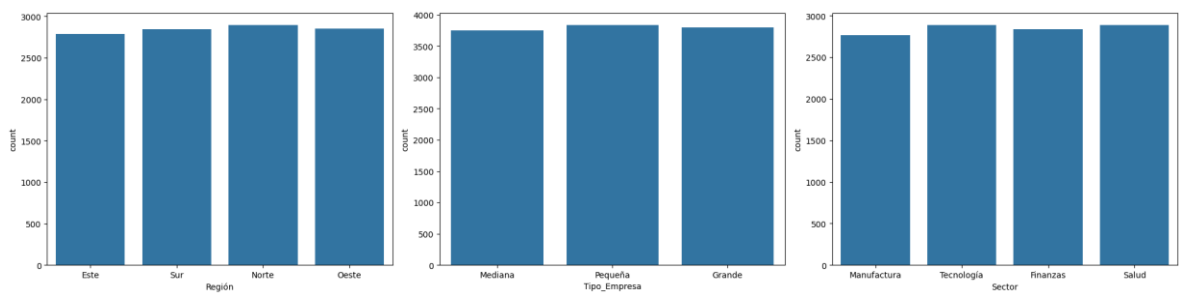


Imagen 5: Gráficas de conteo de datos de variables categóricas.

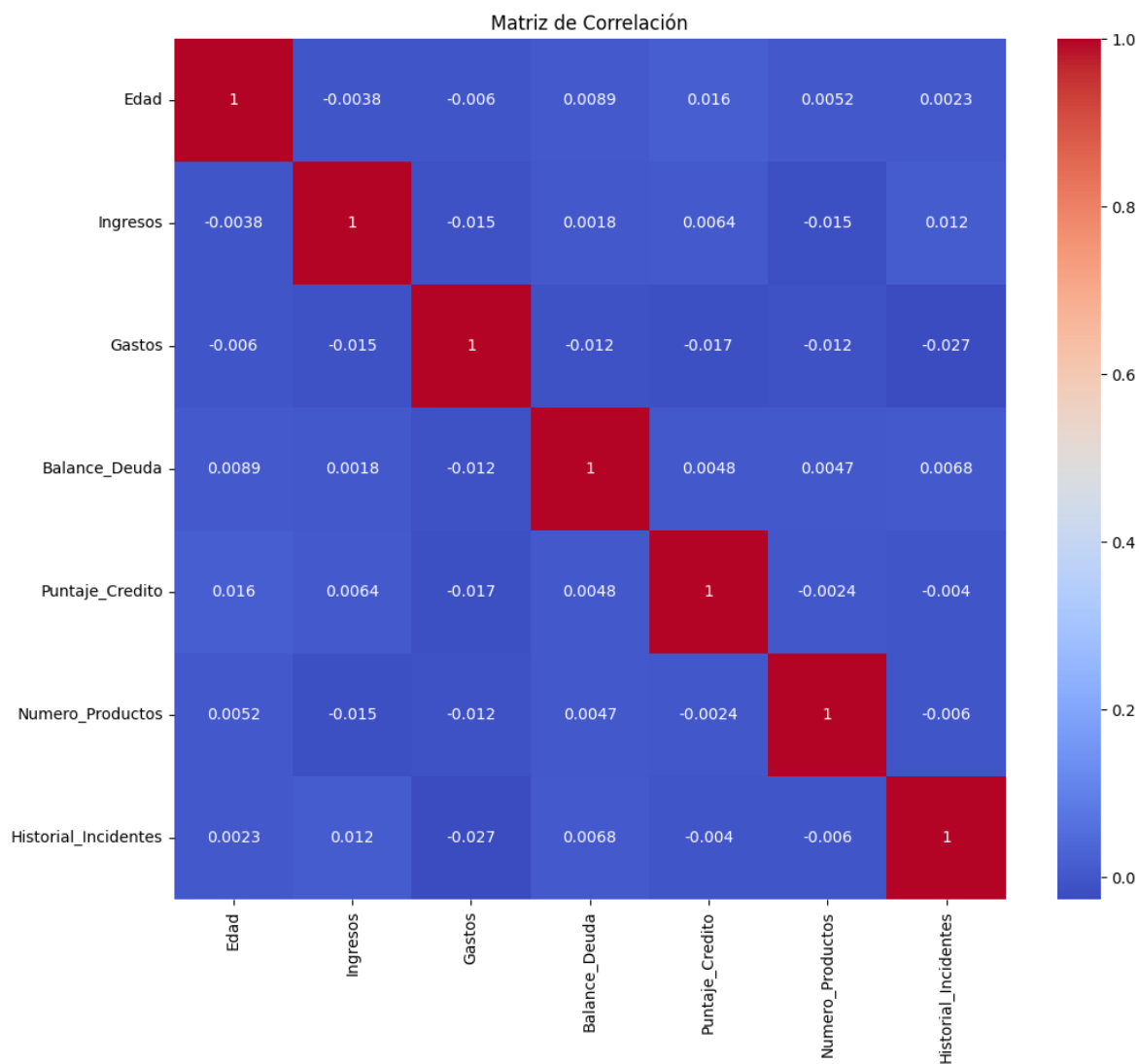


Imagen 6: Matriz de correlación de las variables numéricas.

## Preprocesamiento de los Datos:

### Estandarización:

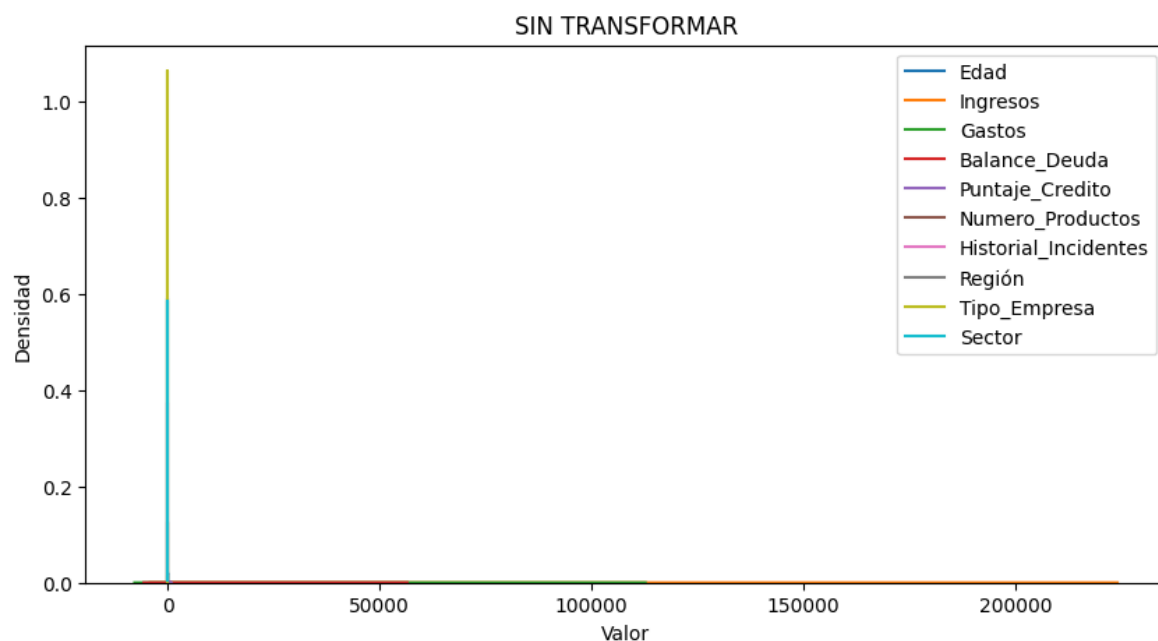


Imagen 7: Gráfica de datos sin transformar.

### Codificación de Variables Categóricas:

Al aplicar el comando dummy, las variables categóricas ( Tipo\_Empresa, Región y Sector) toman números específicos para su debido uso, presentados en la siguiente imagen:

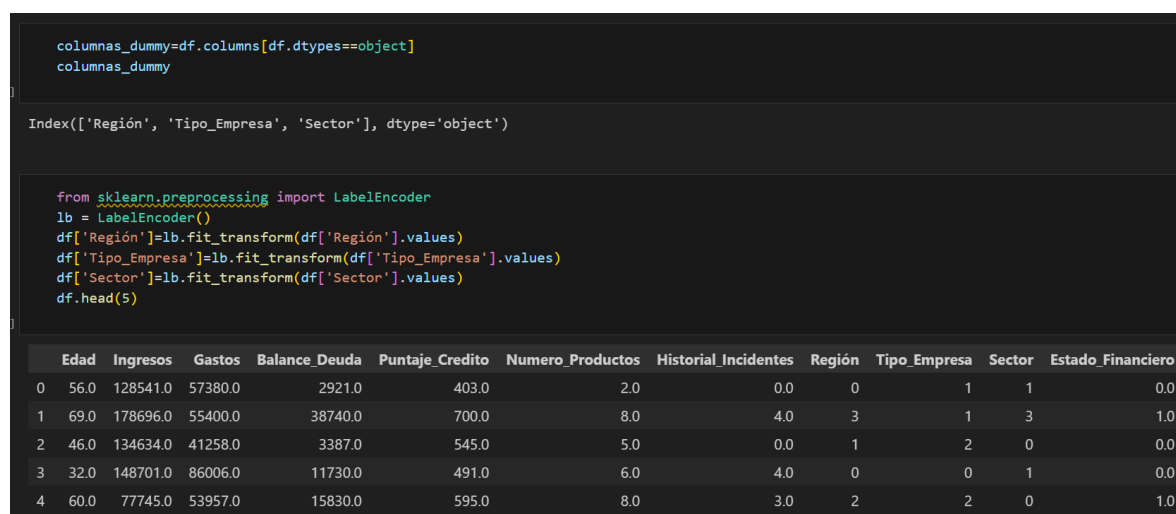


Imagen 8: Aplicación del comando dummy a variables categóricas.

**Región:**

- Este: 0
- Norte: 1
- Oeste: 2
- Sur: 3

**Tipo\_Empresa:**

- Grande:0
- Mediana:1
- Pequeña: 2

**Sector:**

- Finanzas:0
- Manufactura: 1
- Salud: 2
- Tecnología:3

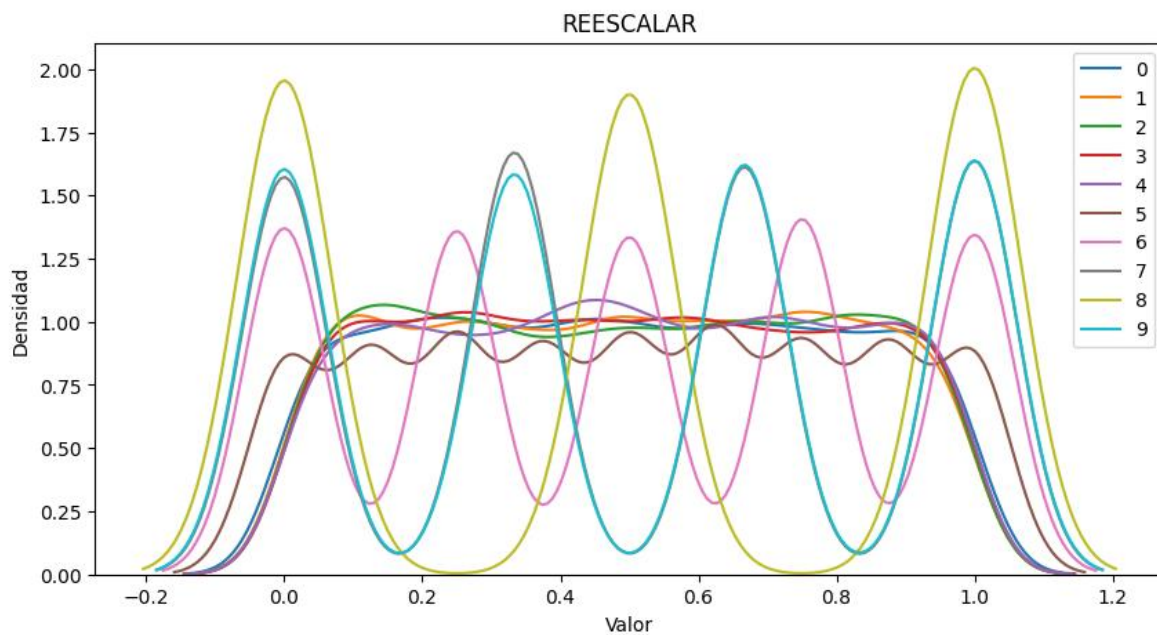
**Reducción de Dimensionalidad:**

Imagen 9: Gráfica de reescalar.

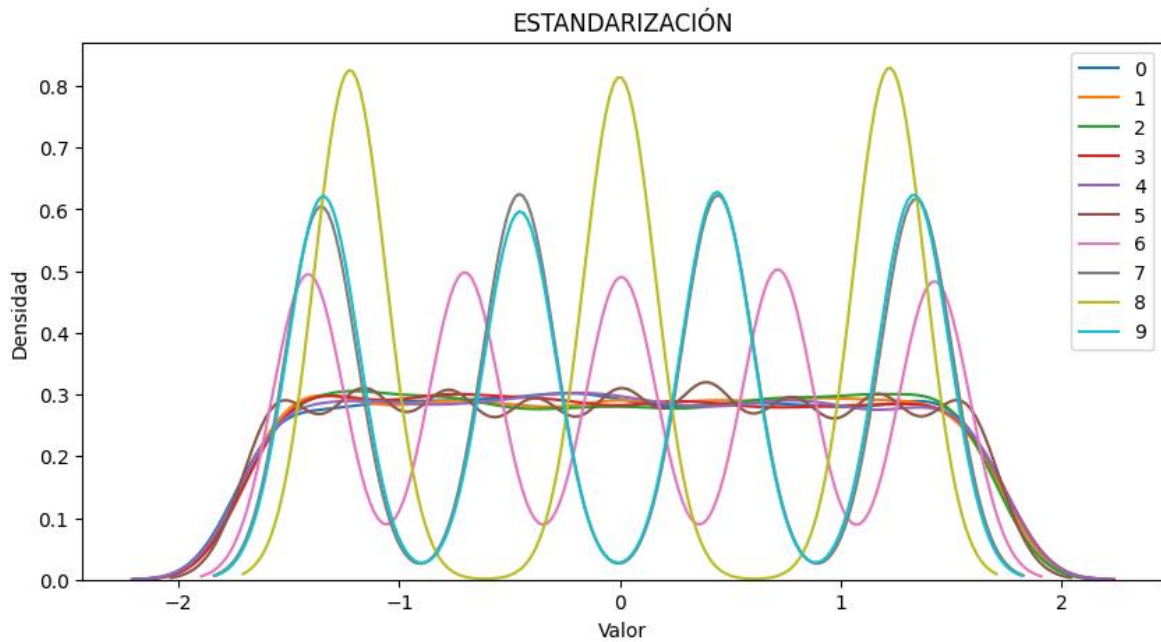
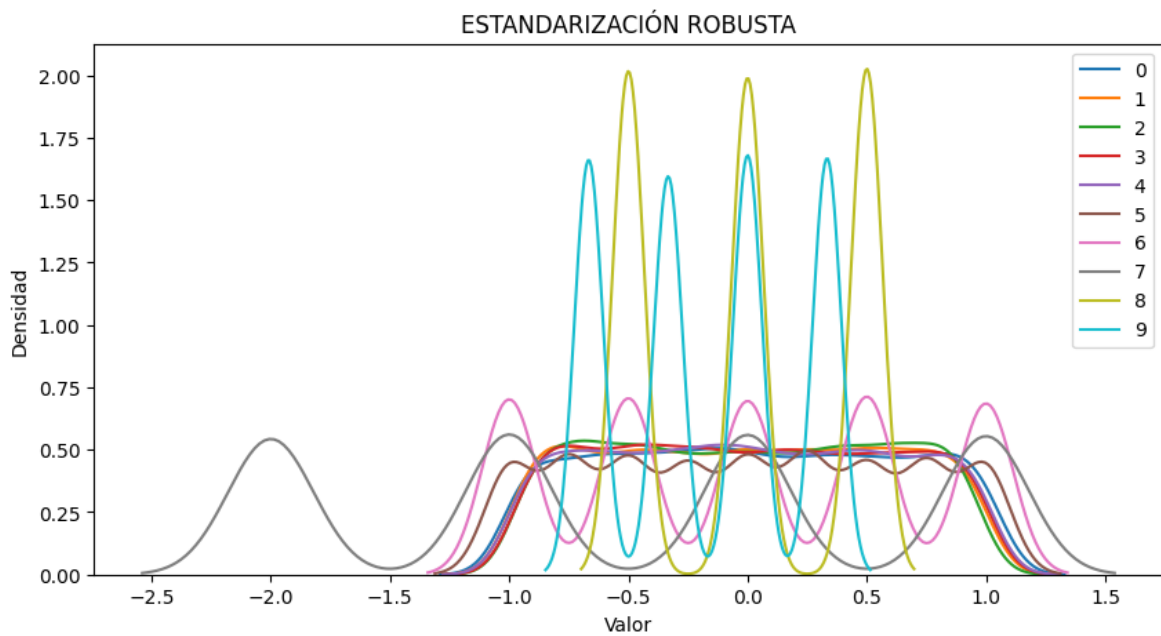


Imagen 10: Gráfica de estandarización.



Gráfica 11: Gráfica de estandarización robusta.

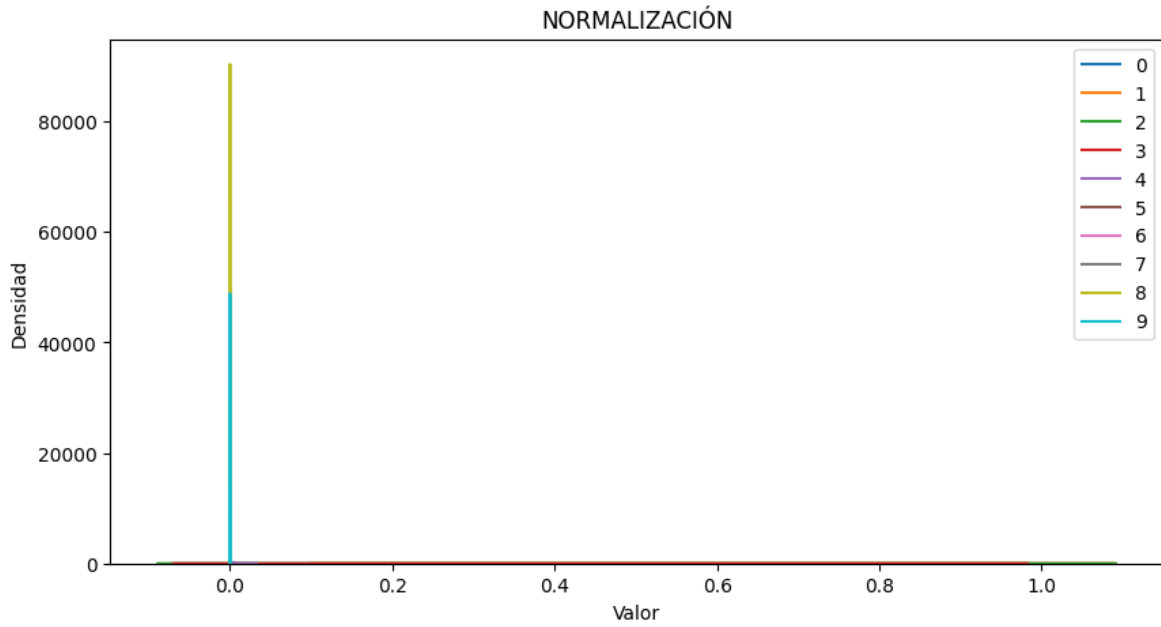


Imagen 12: Gráfica de normalización.

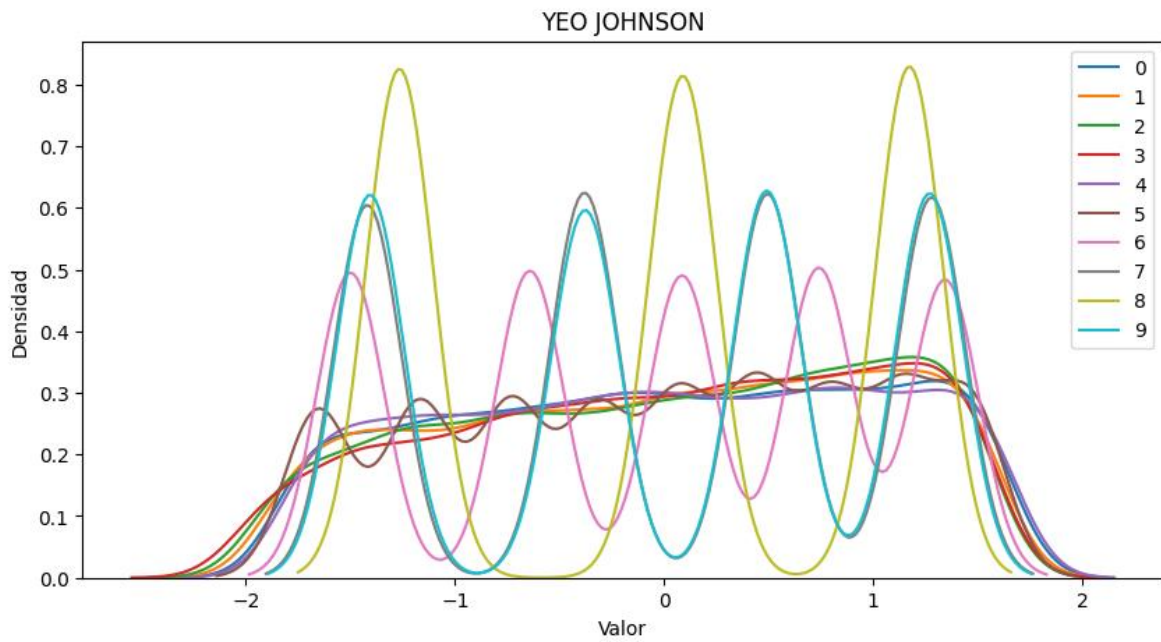


Imagen 13: Gráfica de Yeo Johnson.

Después de tener los datos con el modelo de Yeo Johnson, se procede con los modelos de clasificación:



## Construcción del Modelo de Clasificación:

### SVC

```
from sklearn import svm
svc = svm.SVC()
svc=svc.fit(X_train_std,y_train)
svc
```

SVC

SVC()

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

El valor del accuracy es de 0.49054945054945054

Imagen 14: Modelo SVC.

### Tree

```
from sklearn import tree
tr = tree.DecisionTreeClassifier()
tr = tr.fit(X_train_std,y_train)
```

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

El valor del accuracy es de 0.5010989010989011

Imagen 15: Modelo Decisión Tree.

### Random

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=10)
clf = clf.fit(X_train_std,y_train)
```

+ Code

+ Markdown

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

El valor del accuracy es de 0.5248351648351648

Imagen 16: Modelo Random Forest.

Comparando los modelos, podemos observar que el mejor modelo de los 3 escogidos es Random Forest debido a que nuestro conjunto de datos es muy amplio y este modelo arroja un rendimiento bastante alto de 0.5248 al combinar múltiples árboles de decisión ayuda a mejorar la precisión y reducir el sobreajuste que se hace. Además este modelo permite poder modelar el comportamiento de los datos de un mejor forma en comparación con SVC y Tree.

Resultados para Decision Tree:				
	precision	recall	f1-score	support
0.0	0.50	0.49	0.49	1142
1.0	0.50	0.51	0.50	1133
accuracy			0.50	2275
macro avg	0.50	0.50	0.50	2275
weighted avg	0.50	0.50	0.50	2275
Matriz de Confusión:				
[[556 586]				
[558 575]]				

Imagen 17: Métricas del modelo Decision Tree.

Resultados para Random Forest:				
	precision	recall	f1-score	support
0.0	0.50	0.49	0.49	1142
1.0	0.50	0.51	0.50	1133
accuracy			0.50	2275
macro avg	0.50	0.50	0.50	2275
weighted avg	0.50	0.50	0.50	2275
Matriz de Confusión:				
[[557 585]				
[558 575]]				

Imagen 18: Métricas del modelo Random Forest.

Resultados para SVM:					
	precision	recall	f1-score	support	
0.0	0.50	0.49	0.49	1142	
1.0	0.49	0.50	0.50	1133	
accuracy			0.49	2275	
macro avg	0.49	0.49	0.49	2275	
weighted avg	0.49	0.49	0.49	2275	

Matriz de Confusión:

```
[[554 588]
 [565 568]]
```

Imagen 19: Métricas del modelo SVC.

### Optimización del Modelo:

```
from sklearn.model_selection import GridSearchCV

param_grid={'n_estimators':[1,10,100,1000], 'criterion':['gini', 'entropy', 'log_loss'],
            'max_features':['sqrt', 'log2']}
modelo=RandomForestClassifier()
modelo_grilla=GridSearchCV(modelo,param_grid,cv=5,scoring='accuracy')

modelo_grilla.fit(X_train_std,y_train)
```

Python

GridSearchCV

- best\_estimator\_: RandomForestClassifier
  - RandomForestClassifier

```
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)}')
```

Python

El valor del accuracy es de 0.9777777777777777

Imagen 20: Modelo grilla

Al hacer la optimización del modelo Random Forest se tiene un aumento a 0,97 lo que es un comportamiento mucho al valor obtenido por el no optimizado de 0,52, haciendo este modelo optimizado algo que se aproxime más a la realidad del comportamiento de los datos.

```
y_hat_prueba=modelo_grilla.predict(X_test_std)
precision= precision_score(y_test,y_hat_prueba,average='weighted')
recall=recall_score(y_test,y_hat_prueba,average='weighted')
f1=f1_score(y_test,y_hat_prueba,average='weighted')
accuracy = accuracy_score(y_test, y_hat_prueba)

+ Code + Markdown

print("Precisión: ", round(precision*100,2))
print("Exhaustividad: ", round(recall*100,2))
print("Puntuación F1: ", round(f1*100,2))
print("Exactitud: ", round(accuracy*100,2))

Precisión:  97.89
Exhaustividad:  97.78
Puntuación F1:  97.76
Exactitud:  97.78
```

Imagen 21: Modelo grilla con variable test.

```
y_hat_entrenamiento=modelo_grilla.predict(X_train_std)
precision= precision_score(y_train,y_hat_entrenamiento,average='weighted')
recall=recall_score(y_train,y_hat_entrenamiento,average='weighted')
f1=f1_score(y_train,y_hat_entrenamiento,average='weighted')
accuracy = accuracy_score(y_train, y_hat_entrenamiento)

print("Precisión: ", round(precision*100,2))
print("Exhaustividad: ", round(recall*100,2))
print("Puntuación F1: ", round(f1*100,2))
print("Exactitud: ", round(accuracy*100,2))

Precisión:  96.3
Exhaustividad:  96.19
Puntuación F1:  96.2
Exactitud:  96.19
```

Imagen 22: Modelo grilla con variable real.

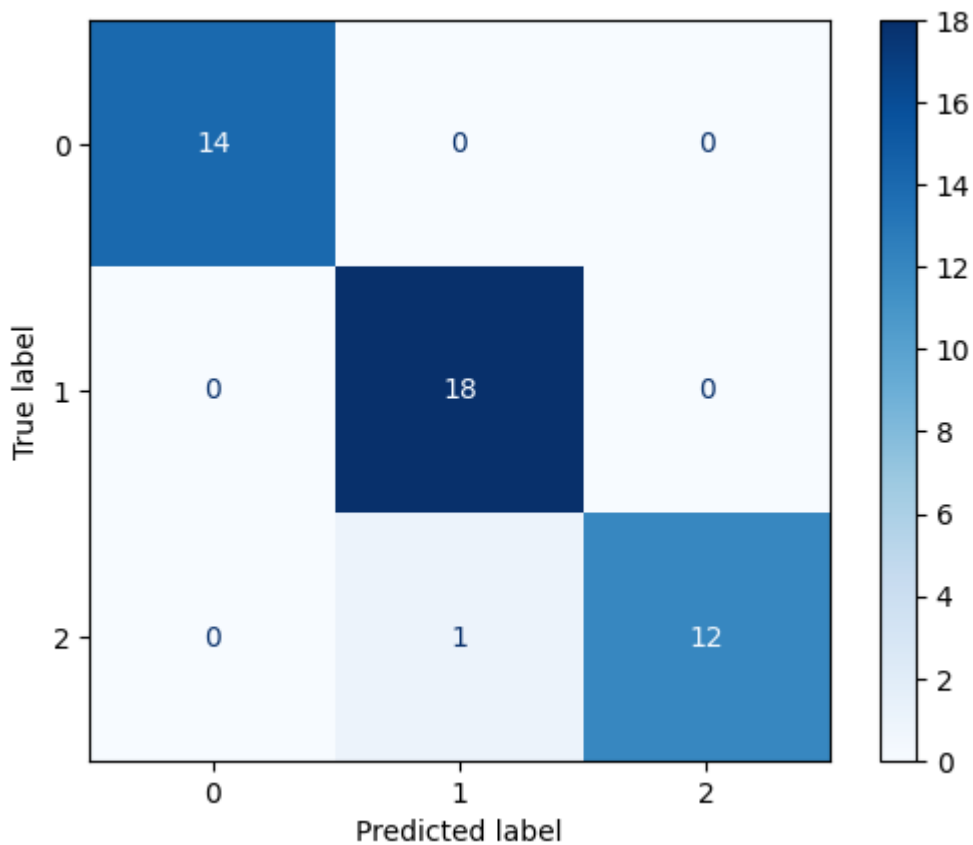


Imagen 23: Matriz de confusión de datos.

Basándonos en esta matriz, podemos decir que el modelo tiene un buen desempeño en las clases 0 y 1, debido a que la mayoría de los datos de fueron clasificados correctamente, por ejemplo, el modelo clasificó correctamente 14 datos como de la clase 0, 18 de la clase 1 y 12 de la clase 2. Sin embargo, hay cierta confusión en la clase 2, ya que un dato fue clasificado incorrectamente como de la clase 1.