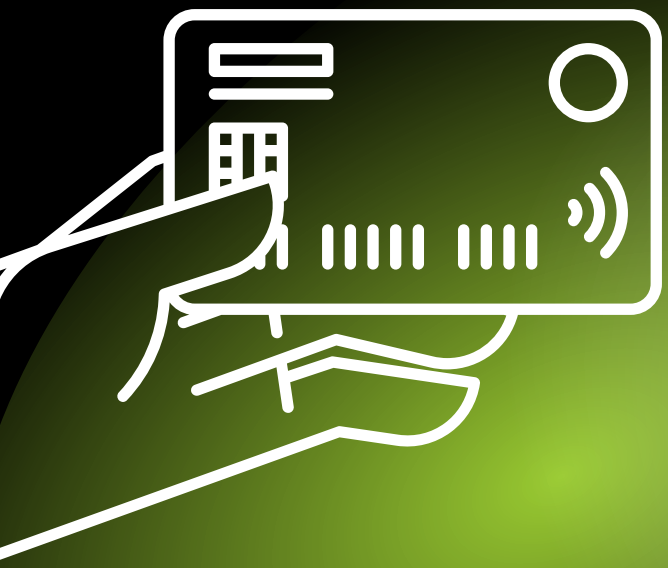
ESTUDIO DEL DATASET

ANALISIS DE RIESGOS CREDITICIOS

ESTUDIANTES:
KAREN TATIANA PIMIENTO MARTINEZ
DONOVAN DAVID TORRES VAHOS



ESTUDIO DEL DATASET

Planteamiento del problema

El análisis de riesgos crediticios se refiere a la evaluación de la probabilidad de que un individuo incumpla en el pago de un préstamo. Se utiliza información como la edad, ingresos, historial de vivienda, duración del empleo y calificación crediticia para determinar el riesgo de otorgar un préstamo. Los factores como la tasa de interés, el monto del préstamo y el porcentaje de ingresos dedicado al préstamo también se consideran.

Objetivo General:

Evaluar y mejorar el análisis de riesgos crediticios mediante técnicas de aprendizaje automático (Decision Tree Regressor, Regresión de Bosque Aleatorio y Regresión de Vector de Soporte) para optimizar la toma de decisiones crediticias.

Objetivos Específicos:

- Construir un modelo de Decision Tree Regressor, Implementar un modelo de Regresión, desarrollar un modelo de Regresión de Vector de Soporte (SVM) para manejar datos complejos y evaluar su rendimiento.
- Comparar los modelos y determinar cuál ofrece la mejor precisión en la predicción de incumplimientos.

DATASET UTILIZADO

```
[ ]  1 df = pd.read_csv('/content/drive/MyDrive/credit_risk_dataset.csv')
     2 df
```

	person_age	person_income	person_home_ownership	person_emp_length	loan_intent	loan_grade	loan_amnt	loan_int_rate	loan_status	loan_percent_income	cb_person_default_on_file	cb_perso
0	22	59000	RENT	123.0	PERSONAL	D	35000	16.02	1	0.59	Y	
1	21	9600	OWN	5.0	EDUCATION	B	1000	11.14	0	0.10	N	
2	25	9600	MORTGAGE	1.0	MEDICAL	C	5500	12.87	1	0.57	N	
3	23	65500	RENT	4.0	MEDICAL	C	35000	15.23	1	0.53	N	
4	24	54400	RENT	8.0	MEDICAL	C	35000	14.27	1	0.55	Y	
...	...	...	...	...	...	...	...	...	...	...	...	
32576	57	53000	MORTGAGE	1.0	PERSONAL	C	5800	13.16	0	0.11	N	
32577	54	120000	MORTGAGE	4.0	PERSONAL	A	17625	7.49	0	0.15	N	
32578	65	76000	RENT	3.0	HOMEIMPROVEMENT	B	35000	10.99	1	0.46	N	
32579	56	150000	MORTGAGE	5.0	PERSONAL	B	15000	11.48	0	0.10	N	
32580	66	42000	RENT	2.0	MEDICAL	B	6475	9.99	0	0.15	N	

32581 rows × 12 columns

LOAN_STATUS

- 0: Sin incumplimiento: el prestatario pagó exitosamente el préstamo según lo acordado y no hubo incumplimiento.
- 1: Incumplimiento: el prestatario no pagó el préstamo de acuerdo con los términos acordados y no cumplió con el préstamo.

1. DECISION TREE REGRESSOR

UN DECISION TREE REGRESSOR ES UN ALGORITMO DE APRENDIZAJE AUTOMÁTICO UTILIZADO PARA PREDECIR VALORES NUMÉRICOS CONTINUOS. FUNCIONA CONSTRUYENDO UN ÁRBOL DE DECISIONES QUE DIVIDE LOS DATOS EN SUBCONJUNTOS Y ASIGNA VALORES DE PREDICCIÓN A CADA SUBCONJUNTO. ESTE MÉTODO ES ÚTIL PARA TAREAS DE REGRESIÓN, PERO PUEDE SER PROPENSO AL SOBREAJUSTE SI NO SE MANEJA ADECUADAMENTE.

- Se cargo un conjunto de datos para explorar la clasificacion de (entrenamiento=80, prueba=20) GaussianNV

```
32
33 # Dividir los datos en conjuntos de entrenamiento y prueba
34 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=21)
35
36 # Crear y entrenar el modelo Naive Bayes Gaussiano
37 est = GaussianNB()
38 est.fit(X_train, y_train)
39 print(accuracy_score(est.predict(X_test), y_test))
```

```
0.7822617768912076
```

- Precisión del modelo Decision Tree

```
28
29 # Transformar la variable objetivo en los datos de prueba
30 label_encoder = LabelEncoder()
31 label_encoder.fit(y_train)
32 y_test_encoded = label_encoder.transform(y_test)
33
34 # Evaluación del modelo
35 accuracy = model.score(X_test, y_test_encoded)
36 print("Precisión del modelo (Decision Tree):", accuracy)
```

```
Precisión del modelo (Decision Tree): 0.79975448826147
```

- Evaluación del Modelo de Árbol de Decisión con Métricas de Clasificación

```
9
10 est = DecisionTreeClassifier(max_depth=2)
11
12 s = cross_val_score(est, X, y, cv=KFold(10, shuffle=True), scoring=make_scorer(accuracy_score))
13 print("accuracy %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))
14 s = cross_val_score(est, X, y, cv=KFold(10, shuffle=True), scoring=tpr)
15 print("tpr      %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))
16 s = cross_val_score(est, X, y, cv=KFold(10, shuffle=True), scoring=tnr)
17 print("tnr      %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))
```

```
accuracy 0.802 (+/- 0.00956)
tpr      0.119 (+/- 0.00996)
tnr      0.992 (+/- 0.00089)
```

- Analisis de correlación

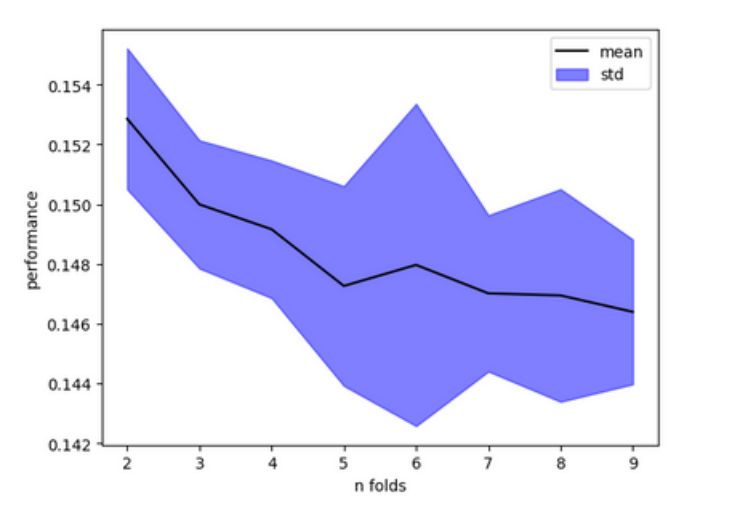
```
8 dfc.corr()
```

```
<ipython-input-33-93771e924ce6>:8: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select
  dfc.corr()
```

	person_age	person_income	person_emp_length	loan_amnt	loan_int_rate	loan_status	loan_percent_income	cb_person_cred_hist_length
person_age	1.000000	0.173202	0.163106	0.050787	0.012580	-0.021629	-0.042411	0.859133
person_income	0.173202	1.000000	0.134268	0.266820	0.000792	-0.144449	-0.254471	0.117987
person_emp_length	0.163106	0.134268	1.000000	0.113082	-0.056405	-0.082489	-0.054111	0.144699
loan_amnt	0.050787	0.266820	0.113082	1.000000	0.146813	0.105376	0.572612	0.041967
loan_int_rate	0.012580	0.000792	-0.056405	0.146813	1.000000	0.335133	0.120314	0.016696
loan_status	-0.021629	-0.144449	-0.082489	0.105376	0.335133	1.000000	0.379366	-0.015529
loan_percent_income	-0.042411	-0.254471	-0.054111	0.572612	0.120314	0.379366	1.000000	-0.031690
cb_person_cred_hist_length	0.859133	0.117987	0.144699	0.041967	0.016696	-0.015529	-0.031690	1.000000

- Graficas de dispersión

- Análisis de Rendimiento de DecisionTreeRegressor en Validación Cruzada por Número de Folds



- Informe de clasificacion

Informe de clasificación:

	precision	recall	f1-score	support
Incumplimiento	0.84	0.12	0.21	1445
Sin incumplimiento	0.80	0.99	0.89	5072
accuracy			0.80	6517
macro avg	0.82	0.56	0.55	6517
weighted avg	0.81	0.80	0.74	6517

Precisión del modelo (Decision Tree): 0.79975448826147

2. RANDOM FOREST REGRESSION

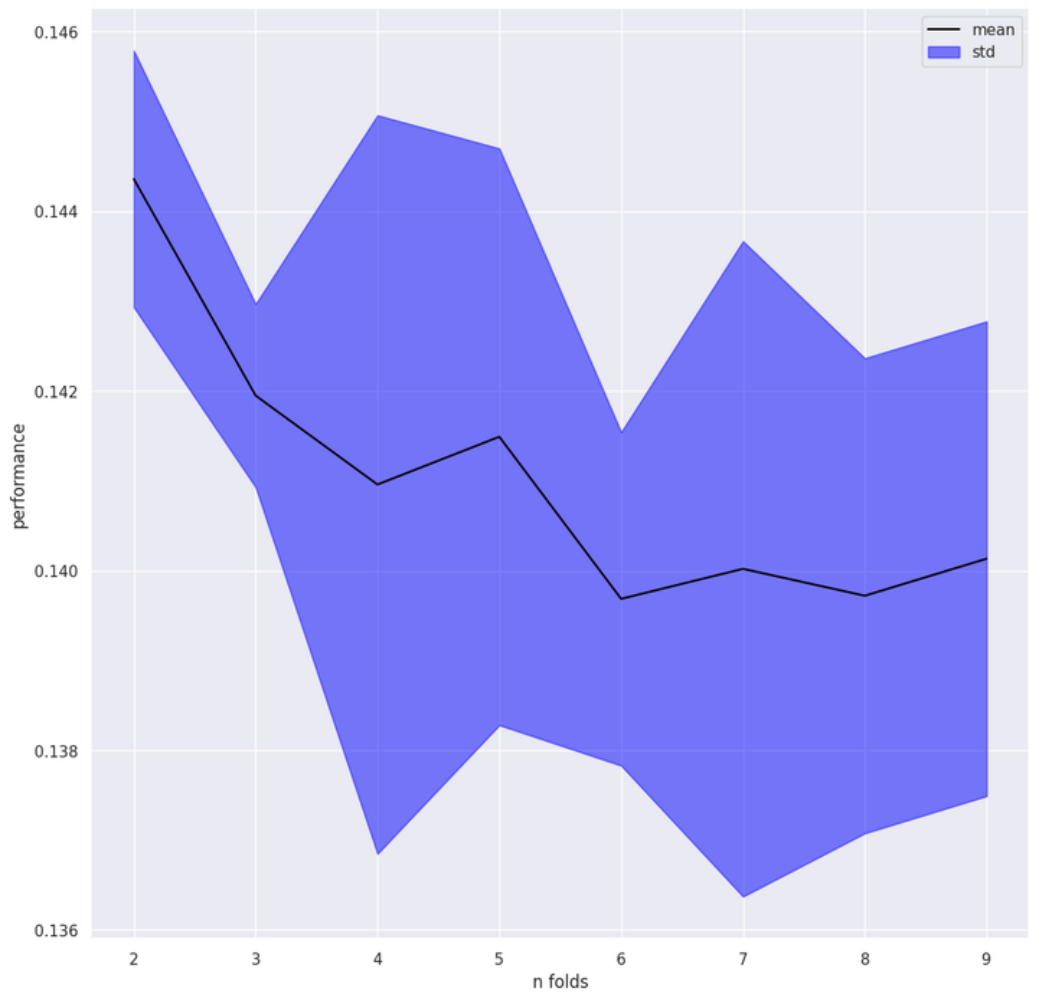
Un Random Forest Regresión emplea múltiples árboles de decisión, donde cada árbol se entrena con una muestra aleatoria del conjunto de datos y con un subconjunto aleatorio de características. Luego, combina las predicciones individuales de cada árbol para obtener una predicción final. Esto ayuda a reducir el sobreajuste (overfitting) y a mejorar la precisión predictiva del modelo.

- Se evaluaron los rendimientos de los arboles por medio de medidas de regresión

```
33
34 # Evaluar el modelo
35 mse_rf = mean_squared_error(y_test, est.predict(X_test))
36 print("MSE depth split data  %.3f"% mean_squared_error(y_test, est.predict(X_test)))
37 print("RMSE depth split data  %.3f"% np.sqrt(mean_squared_error(y_test, est.predict(X_test))))
38 print("MAE depth split data  %.3f"% mean_absolute_error(y_test, est.predict(X_test)))
```

```
MSE depth split data  0.136
RMSE depth split data  0.369
MAE depth split data  0.255
```

- Análisis de Rendimiento de RandomForestRegression en Validación Cruzada por Número de Folds



3. SUPPORT VECTOR REGRESION

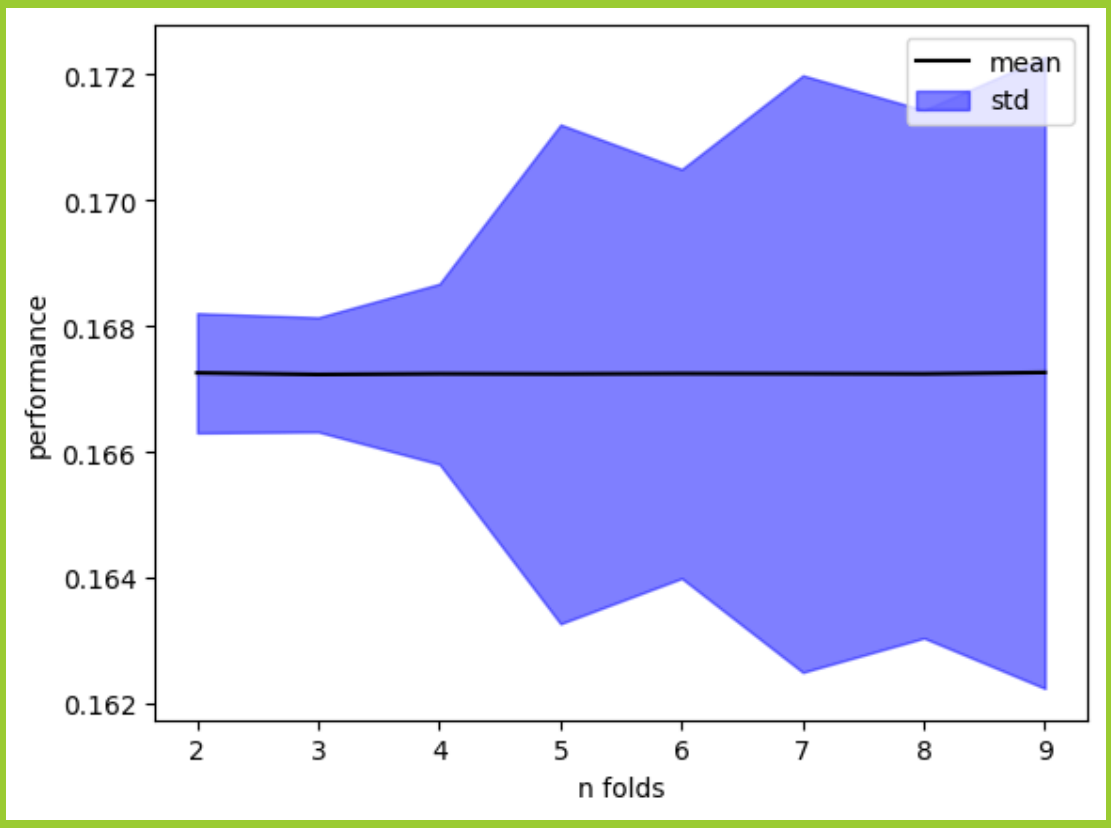
El objetivo de SVR es encontrar una función (hiperplano en el espacio de características) que se ajuste lo más cerca posible a un conjunto de puntos de datos objetivo. En lugar de buscar la mayor separación entre clases como en la clasificación, en regresión, SVR busca minimizar la diferencia entre la función predicha y los puntos de datos reales. Se enfoca en encontrar una función que se ajuste a los datos dentro de un rango de tolerancia permitido.

- Se evalua el rendimiento de la clasificacion por medio de un accuracy

```
# Evaluar el modelo
mse_rf = mean_squared_error(y_test, svm_model.predict(X_test))
print("Precisión del modelo5:", predictions)
print("MSE depth split data  %.3f"% mean_squared_error(y_test, svm_model.predict(X_test)))
print("RMSE depth split data  %.3f"% np.sqrt(mean_squared_error(y_test, svm_model.predict(X_test))))
print("MAE depth split data  %.3f"% mean_absolute_error(y_test, svm_model.predict(X_test)))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but SVR was fitted without feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but SVR was fitted without feature names
  warnings.warn(
Precisión del modelo5: [1.09918105 1.09918105 1.09918105 ... 1.09918105 1.09918105 1.09918105]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but SVR was fitted without feature names
  warnings.warn(
MSE depth split data  0.947
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but SVR was fitted without feature names
  warnings.warn(
RMSE depth split data  0.973
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but SVR was fitted without feature names
  warnings.warn(
MAE depth split data  0.882
```

- Análisis de Rendimiento de SVR en Validación Cruzada por Número de Folds



4. SUPPORT VECTOR CLASIFICATION

La Support Vector Classification (SVC), en español, clasificación de vectores de soporte, es un algoritmo de aprendizaje supervisado utilizado para tareas de clasificación. Su objetivo es encontrar un hiperplano en un espacio de alta dimensión que mejor separe las clases de los datos. La idea central detrás del algoritmo de Support Vector Machine (SVM) es encontrar el "margen máximo" que mejor separa las clases.

- Se evalua el rendimiento de la clasificacion por medio de un accuracy

```
28 # Separar datos en conjunto de entrenamiento y prueba
29 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=21)
30
31 est = SVC()
32 est.fit(X_train,y_train)
33 print(accuracy_score(est.predict(X_test), y_test))
```

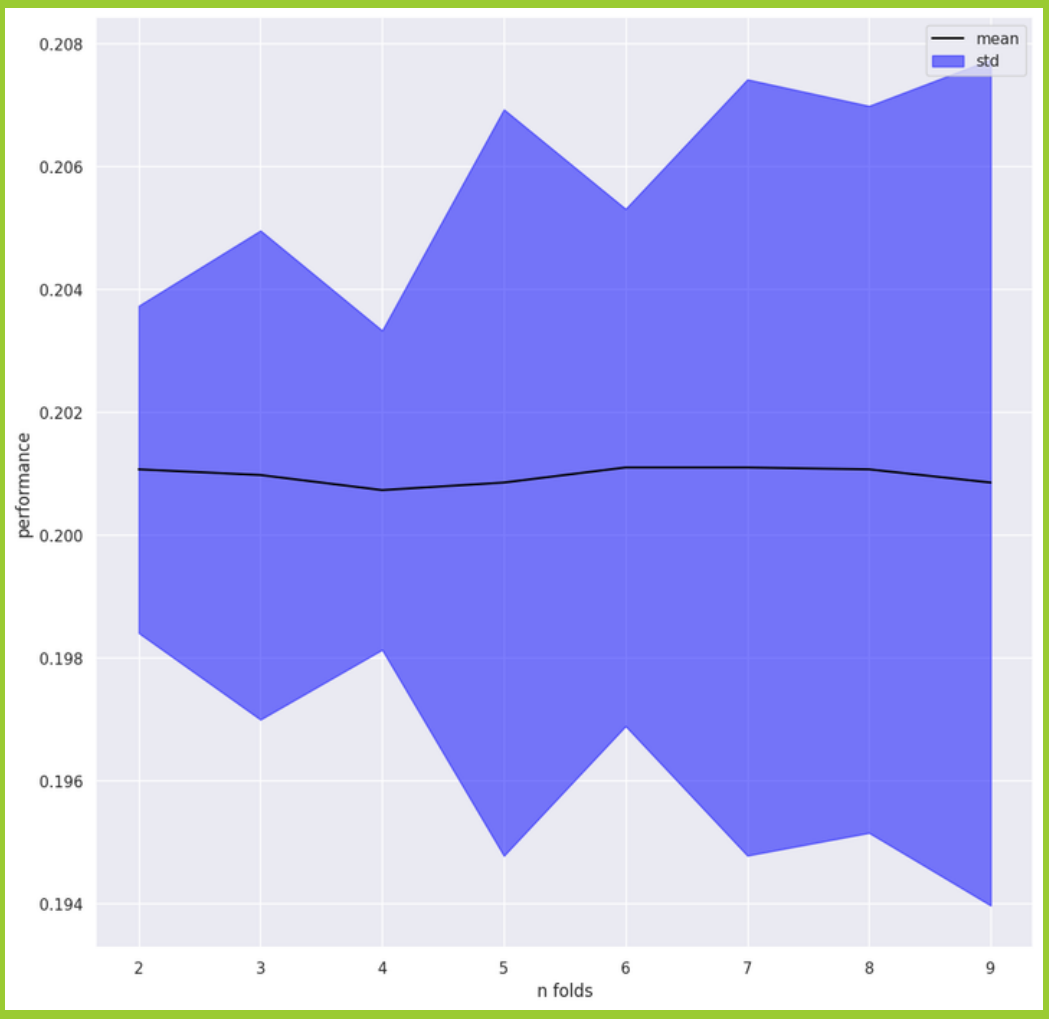
```
0.7968390363664263
```

- Evaluación del Modelo de SVC con Métricas de Clasificación

```
5
6 s = cross_val_score(est, X, y, cv=KFold(10, shuffle=True), scoring=make_scorer(accuracy_score))
7 print("accuracy %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))
```

```
accuracy 0.799 (+/- 0.00750)
```

- Análisis de Rendimiento de SVC en Validación Cruzada por Número de Folds



5. RED NEURONAL PARA CLASIFICACIÓN

La red neuronal implementada es un modelo de aprendizaje supervisado diseñado para clasificar el estado de los préstamos basado en diversas características financieras y personales de los prestatarios. Esta red utiliza una arquitectura secuencial de varias capas densamente conectadas. Incluye capas de entrada con activación ReLU, capas ocultas y una capa de salida con activación softmax para la clasificación multiclase. La red se ha entrenado y evaluado utilizando características como la edad del solicitante, ingreso, duración del empleo, historial crediticio, monto del préstamo, entre otros, para predecir si un préstamo tendrá o no incumplimiento. Tras el preprocesamiento de los datos, el modelo ha alcanzado una precisión de alrededor del 83% en la clasificación del estado de los préstamos, lo que indica su capacidad para discernir entre incumplimientos y pagos exitosos. En un escenario de la vida real, esta red neuronal puede ser utilizada por instituciones financieras para evaluar el riesgo crediticio de los solicitantes de préstamos, ayudando a tomar decisiones más informadas y precisas en la aprobación o rechazo de préstamos, minimizando así el riesgo de incumplimiento y mejorando la gestión de carteras de préstamos. A continuación se muestra el resultado final de nuestro modelo:

- Variables utilizadas para el estudio:

```
# Dividir los datos en características (X) y etiquetas (y)
X = df[['loan_grade_encoded', 'person_age', 'person_income', 'person_emp_length', 'loan_amnt', 'cb_person_cred_hist_length']]
y = df['loan_status']
```

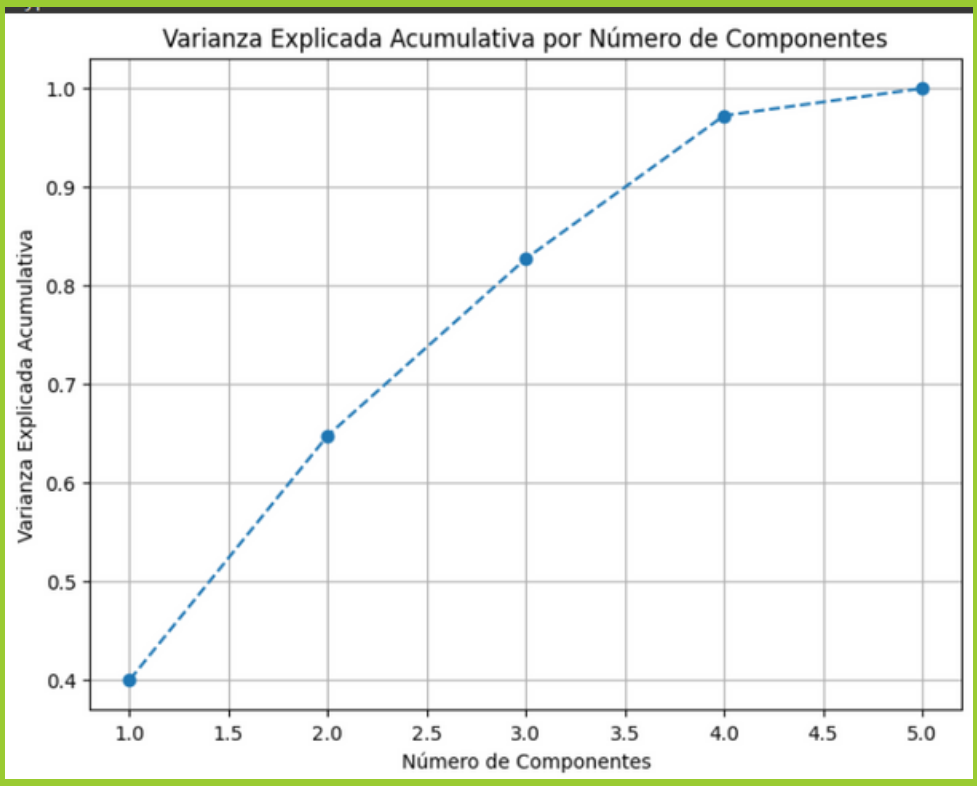
- Resultados de la implementacion de la red neuronal:

```
Epoch 1/15
358/358 [==============================] - 3s 6ms/step - loss: 0.4786 - accuracy: 0.8029 - val_loss: 0.4537 - val_accuracy: 0.8153
Epoch 2/15
358/358 [==============================] - 3s 8ms/step - loss: 0.4380 - accuracy: 0.8167 - val_loss: 0.4477 - val_accuracy: 0.8094
Epoch 3/15
358/358 [==============================] - 3s 9ms/step - loss: 0.4318 - accuracy: 0.8201 - val_loss: 0.4425 - val_accuracy: 0.8155
Epoch 4/15
358/358 [==============================] - 1s 3ms/step - loss: 0.4250 - accuracy: 0.8230 - val_loss: 0.4317 - val_accuracy: 0.8197
Epoch 5/15
358/358 [==============================] - 1s 3ms/step - loss: 0.4175 - accuracy: 0.8234 - val_loss: 0.4233 - val_accuracy: 0.8162
Epoch 6/15
358/358 [==============================] - 1s 2ms/step - loss: 0.4098 - accuracy: 0.8250 - val_loss: 0.4157 - val_accuracy: 0.8170
Epoch 7/15
358/358 [==============================] - 1s 2ms/step - loss: 0.4020 - accuracy: 0.8247 - val_loss: 0.4137 - val_accuracy: 0.8204
Epoch 8/15
358/358 [==============================] - 1s 2ms/step - loss: 0.3951 - accuracy: 0.8253 - val_loss: 0.4328 - val_accuracy: 0.8088
Epoch 9/15
358/358 [==============================] - 1s 2ms/step - loss: 0.3908 - accuracy: 0.8283 - val_loss: 0.3999 - val_accuracy: 0.8214
Epoch 10/15
358/358 [==============================] - 1s 2ms/step - loss: 0.3881 - accuracy: 0.8283 - val_loss: 0.3928 - val_accuracy: 0.8263
Epoch 11/15
358/358 [==============================] - 1s 3ms/step - loss: 0.3845 - accuracy: 0.8287 - val_loss: 0.3920 - val_accuracy: 0.8303
Epoch 12/15
358/358 [==============================] - 1s 2ms/step - loss: 0.3843 - accuracy: 0.8313 - val_loss: 0.3886 - val_accuracy: 0.8204
Epoch 13/15
358/358 [==============================] - 1s 3ms/step - loss: 0.3798 - accuracy: 0.8331 - val_loss: 0.3858 - val_accuracy: 0.8254
Epoch 14/15
358/358 [==============================] - 1s 3ms/step - loss: 0.3766 - accuracy: 0.8348 - val_loss: 0.3865 - val_accuracy: 0.8307
Epoch 15/15
358/358 [==============================] - 1s 2ms/step - loss: 0.3731 - accuracy: 0.8336 - val_loss: 0.3772 - val_accuracy: 0.8300
179/179 [==============================] - 0s 2ms/step - loss: 0.3772 - accuracy: 0.8300
Test accuracy: 0.8299580812454224
```

6. IMPLEMENTACION DE PCA

La aplicación de PCA redujo las múltiples características del conjunto de datos crediticios a tres componentes principales. Este enfoque permitió conservar alrededor del 80% al 90% de la información original. Se eligieron tres componentes tras analizar la varianza explicada por cada uno, buscando un equilibrio entre la reducción de dimensionalidad y la retención de datos esenciales. Esta selección optimizó la simplificación del conjunto de datos para un procesamiento más eficiente, facilitando la interpretación de modelos sin perder gran parte de la información crucial para el análisis de riesgos crediticios. A continuación se muestra el resultado final de nuestro modelo:

- Eleccion del numero de componentes:



- Visualizacion de los datos despues del PCA: