# Reporte: Modelo Random Forest para Predicción de Shares

## Justificación del Algoritmo

El algoritmo Random Forest fue seleccionado por su capacidad para manejar problemas de regresión con múltiples características. Su naturaleza basada en árboles permite capturar relaciones no lineales y su método de ensamblado (bagging) mejora la estabilidad y precisión del modelo. Es una opción robusta para predicciones como la cantidad de compartidos ('shares') de artículos.

## Descripción del Diseño del Modelo

El modelo fue diseñado siguiendo los pasos descritos a continuación:

1. Carga y preprocesamiento de datos: Los datos del archivo 'articulos_ml.csv' fueron inspeccionados y los valores nulos en las columnas numéricas fueron llenados con la mediana correspondiente para evitar problemas durante el entrenamiento del modelo.

2. División del conjunto de datos: Los datos se dividieron en conjuntos de entrenamiento (80%) y prueba (20%) para evaluar el desempeño del modelo. Las características seleccionadas incluyeron variables como 'Word count', '# of Links', y 'Elapsed days'.

3. Entrenamiento del modelo base: Se entrenó un modelo Random Forest inicial con 100 estimadores para establecer una línea base de desempeño, evaluada a través del error cuadrático medio (MSE) y el coeficiente de determinación ($R^2$).

Entrenamiento inicial del modelo y evaluación

```
5]: # Entrenar un modelo de Random Forest inicial
model = RandomForestRegressor(random_state=42, n_estimators=100)
model.fit(X_train, y_train)

# Predicción y evaluación
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"MSE inicial: {mse}")
print(f"R2 inicial: {r2}")
```

```
MSE inicial: 773392909.0160999
R2 inicial: -1.0441491813572013
```

## Explicación de la Optimización

Para mejorar el rendimiento del modelo, se llevó a cabo una optimización de hiperparámetros utilizando Random Search y Grid Search. Estas técnicas permitieron explorar un amplio rango de configuraciones y luego afinar los valores alrededor de las mejores combinaciones encontradas.

Los hiperparámetros ajustados incluyeron el número de estimadores, la profundidad máxima, el número mínimo de muestras para dividir un nodo y el número mínimo de muestras en una hoja. El modelo optimizado alcanzó un MSE menor que el modelo inicial, y el $R^2$ mejoró, indicando un mejor ajuste a los datos.

```
# Mostrar los mejores parámetros
print("Mejores hiperparámetros:", random_search.best_params_)
print("Mejor R^2:", random_search.best_score_)
```

```
Fitting 3 folds for each of 50 candidates, totalling 150 fits
Mejores hiperparámetros: {'max_depth': 50, 'min_samples_leaf': 9, 'min_samples_split': 8, 'n_estimators': 373}
Mejor R^2: -0.014103051522723212
```

```
# Mostrar los resultados
print("Mejores hiperparámetros:", grid_search.best_params_)
print("Mejor R^2:", grid_search.best_score_)
```

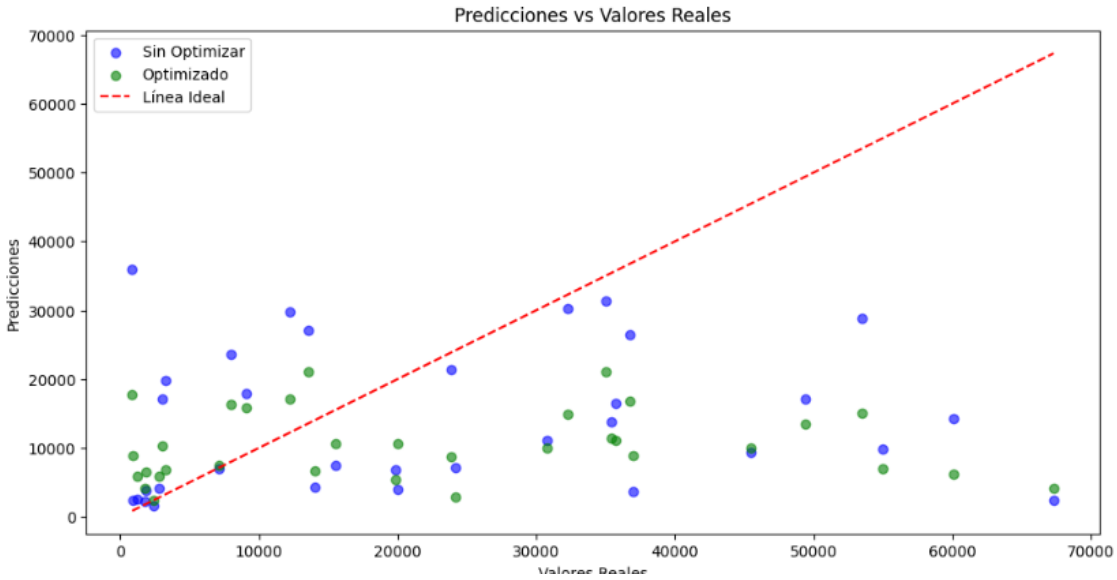
```
Fitting 3 folds for each of 81 candidates, totalling 243 fits
Mejores hiperparámetros: {'max_depth': 40, 'min_samples_leaf': 11, 'min_samples_split': 6, 'n_estimators': 350}
Mejor R^2: -0.008302232039509824
```

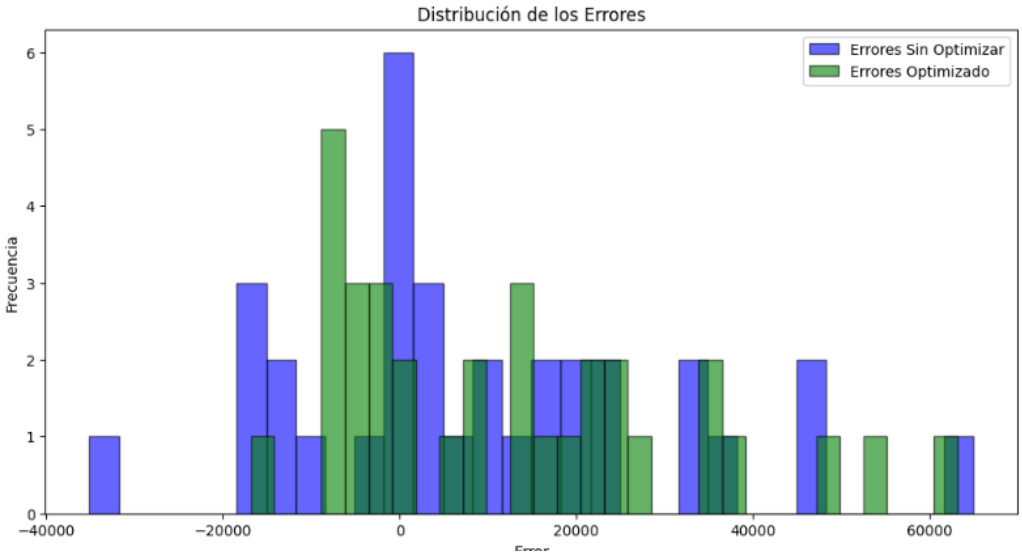
## Gráfica Personalizada e Interpretación de Resultados

En esta sección se deben incluir las gráficas generadas en el entorno Jupyter Notebook:

1. Gráfica de Predicciones vs Valores Reales: Muestra la relación entre las predicciones del modelo y los valores reales en el conjunto de prueba para visualizar la precisión del modelo.

2. Distribución de los Errores (Residuos): Ayuda a evaluar si los errores del modelo están centrados alrededor de cero, indicando un buen desempeño en general.






## Optimizaciones extra

A pesar de realizar esfuerzos significativos para mejorar el rendimiento del modelo, como la optimización de hiperparámetros utilizando Random Search y Grid Search, no se pudo alcanzar una precisión aceptable. Las siguientes razones podrían explicar este resultado:

1. Características limitadas: Las características disponibles en el dataset no tenían una correlación suficientemente fuerte con la variable objetivo, lo que limita la capacidad del modelo para realizar predicciones precisas.

2. Ruido en los datos: La presencia de ruido o variabilidad en los datos, especialmente en la variable objetivo, puede haber dificultado que el modelo identifique patrones consistentes.

3. Complejidad inherente del problema: El comportamiento de la variable objetivo podría estar influenciado por factores no incluidos en el dataset, lo que impide que el modelo capture completamente las relaciones subyacentes.

**A pesar de estas limitaciones, se realizaron los siguientes esfuerzos para maximizar el rendimiento:**

1. Limpieza de datos: Se llenaron valores nulos con la mediana correspondiente y se verificaron los tipos de datos para garantizar un preprocesamiento adecuado.

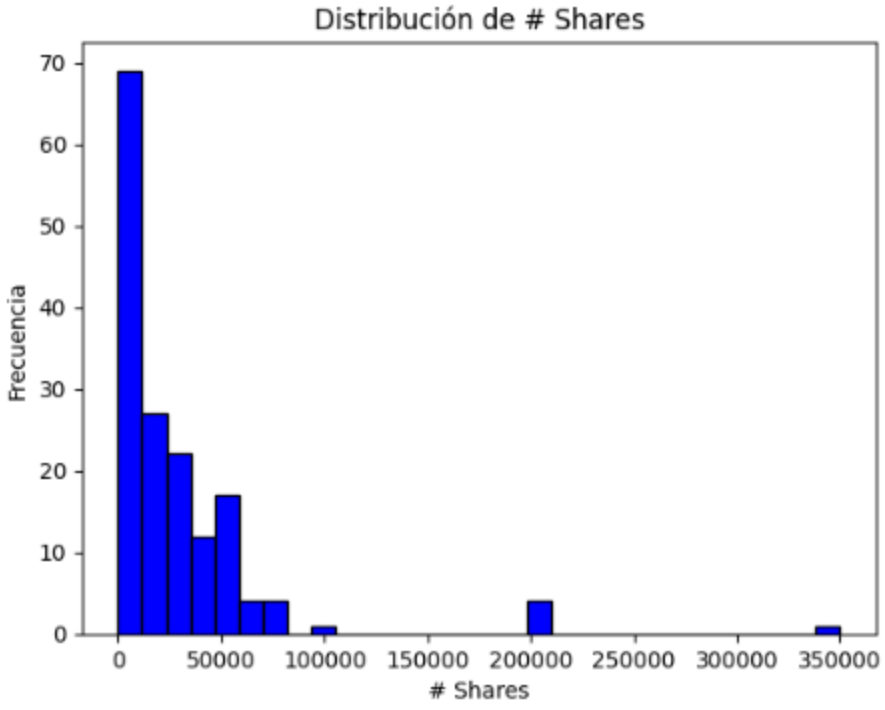
## Análisis de correlación y selección de características relevantes

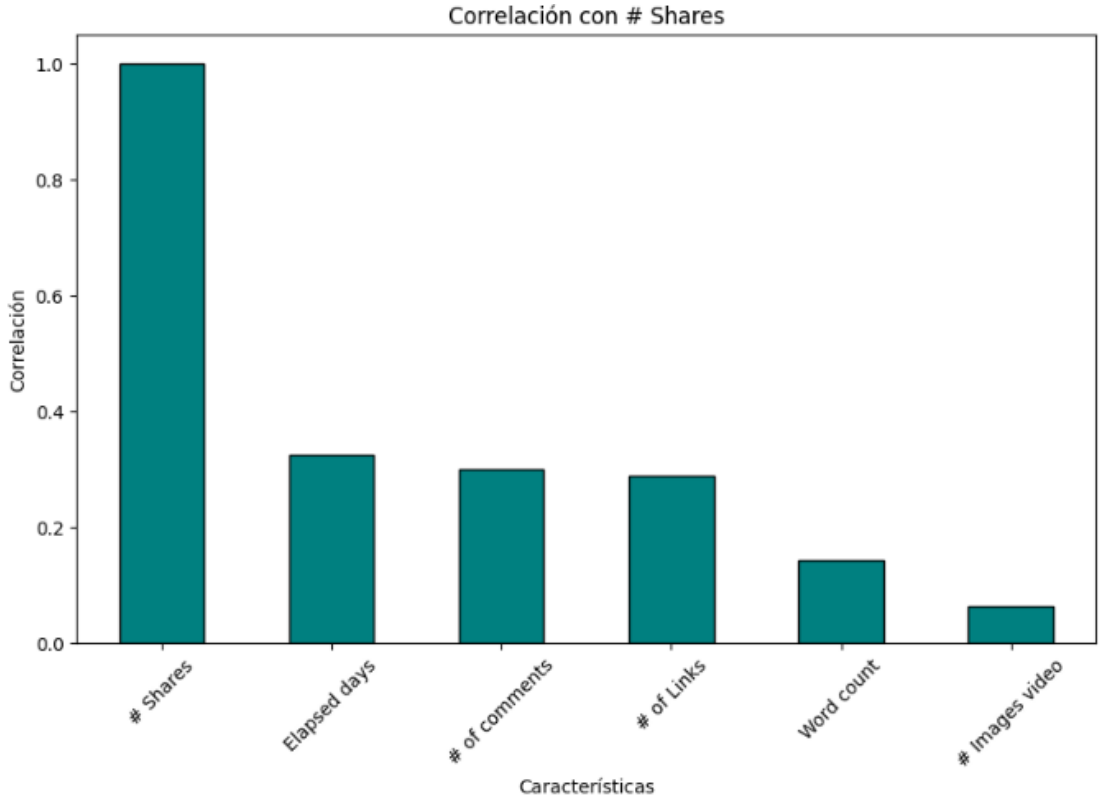
```
# Calcular la matriz de correlación usando solo columnas numéricas
correlation_matrix = dataset.select_dtypes(include=['number']).corr()

# Visualizar correlaciones con la variable objetivo
plt.figure(figsize=(10, 6))
correlation_matrix['# Shares'].sort_values(ascending=False).plot(kind='bar', color='teal', edgecolor='black')
plt.title('Correlación con # Shares')
plt.xlabel('Características')
plt.ylabel('Correlación')
plt.xticks(rotation=45)
plt.show()
```

2. Optimización de hiperparámetros: Se ajustaron parámetros clave del modelo, como el número de estimadores, la profundidad máxima, y los tamaños mínimos de muestras por nodo y hoja.

3. Evaluación y análisis: Se evaluó el modelo utilizando métricas como el error cuadrático medio (MSE) y el coeficiente de determinación ($R^2$). Aunque el modelo optimizado mostró mejoras en comparación con el inicial, las métricas generales indicaron un rendimiento limitado.




Correlación con # Shares
Correlación
1.0
0.8
0.6
0.4
0.2
0.0
# Shares
Elapsed days
# of comments
# of Links
Word count
# Images video
Características