



POLITÉCNICA

UNIVERSIDAD POLITÉCNICA DE MADRID

**PREDICCIÓN:
REGRESIÓN Y SERIES TEMPORALES**

Métodos clásicos para predicción

Alcalde Benitez Patricia Fuxi
Vulpe Beatriz Noelia

Curso 2023-2024

Contents

1	Introducción	1
2	Regresión	1
2.1	Procesamiento y limpieza	1
	Valores duplicados en <code>song_id</code>	3
	Tratamiento de columnas categóricas	3
	Tratamiento de <code>song_name</code>	4
2.2	Entrenamiento de modelos y evaluación	5
	Reducción de dimesionalidad	9
	Feature Selection	11
2.3	Entrenamiento del conjunto de datos de menor dimensionalidad	12
	Reducción de dimensionalidad	13
	Feature selection	13
2.4	Resultados	14
3	Visualización de datos	14
3.1	Relaciones estadísticas	14
	Wordcloud de los títulos	14
	Matriz de correlación	15
	Joinplot	16
3.2	PCA	16
3.3	T-SNE	20
3.4	ISOMAP	20
4	Series temporales	21
4.1	Descripción del conjunto de datos	21
	Influencia del sexo y las provincias	22
4.2	Análisis de la serie temporal	22
	Descomposición: estacionalidad y tendencia	24
	Autocorrelación: estacionariedad	24
4.3	Predicción del año 2021	25
	Entrenamiento de modelos y evaluación	25
	Resultados y conclusiones	36
4.4	Predicción a futuro: año 2022	37

1 Introducción

El enfoque principal de este trabajo radica en la evaluación y comparación de la capacidad predictiva de diversos modelos dentro del contexto específico de los conjuntos de datos proporcionados. La finalidad última es la selección de la metodología que demuestre un rendimiento óptimo al enfrentarse a datos no observados en el conjunto de prueba.

Para lograr este objetivo, se llevará a cabo una descripción detallada del proceso de trabajo, respaldada por el uso de gráficos que proporcionen una visualización clara de las decisiones tomadas y los resultados obtenidos. Se busca no solo presentar los resultados finales, sino también proporcionar una comprensión profunda de las elecciones metodológicas, los enfoques adoptados y cualquier consideración clave que influya en la interpretación de los resultados.

Abordaremos dos problemas distintos, cada uno asociado a un conjunto de datos específico. Utilizaremos los conjuntos de datos `train_ap1_mcp_23_24_train.csv` y `test_ap1_mcp_23_24_test.csv` para llevar a cabo la tarea de regresión, específicamente para predecir la popularidad de canciones. Además, emplearemos el conjunto de datos `nacimientos_2016_2021.csv` para realizar la predicción de series temporales de nacimientos.

La organización del trabajo se estructurará en torno a estos datasets, siendo los dos primeros apartados destinados al problema de regresión y el último focalizado en la serie temporal.

2 Regresión

Contamos con un conjunto de datos compuesto por 15 columnas: `song_name`, `song_id`, `song_duration_ms`, `acousticness`, `danceability`, `energy`, `instrumentalness`, `key`, `liveness`, `loudness`, `audio_mode`, `speechiness`, `tempo`, `time_signature` y `audio_valence`. La tarea consiste en predecir la columna `popularity` en función de estas características.

Exploraremos diversas interrogantes a lo largo de este análisis, tales como: **¿La duración de una canción impacta en su nivel de popularidad?** **¿Existe una tendencia que sugiere que las canciones instrumentales son más populares que aquellas que no lo son?** **¿Podría el tempo de una canción influir en su éxito?** Estas cuestiones guiarán nuestro proceso de indagación y búsqueda de patrones significativos.

El propósito fundamental de este ejercicio es desarrollar un modelo de regresión que optimice la predicción de la variable popularidad basándonos en la información provista por nuestro conjunto de datos.

2.1 Procesamiento y limpieza

Como primer paso, analizaremos la **naturaleza de los datos**, presencia de **datos faltantes** (nulos o Nan) o símbolos no representativos, para decidir si necesitan algún procesamiento.

		Data Type	Distinct Values	Null Values	Missing Values
<code>song_name</code>		object	21808	0	0
<code>song_id</code>		int64	23721	0	0
<code>popularity</code>		int64	98	0	0
<code>acousticness</code>		float64	3438	0	0
<code>danceability</code>		float64	1035	0	0
<code>song_duration_ms</code>		int64	17279	0	0
<code>energy</code>		float64	1754	0	0
<code>instrumentalness</code>		float64	4472	0	0
<code>key</code>		int64	12	0	0
<code>liveness</code>		float64	1593	0	0
<code>loudness</code>		float64	12586	0	0
<code>audio_mode</code>		int64	2	0	0
<code>speechiness</code>		float64	1435	0	0
<code>tempo</code>		float64	18864	0	0
<code>time_signature</code>		int64	5	0	0
<code>audio_valence</code>		float64	1568	0	0

(a) Comprobación de valores faltantes

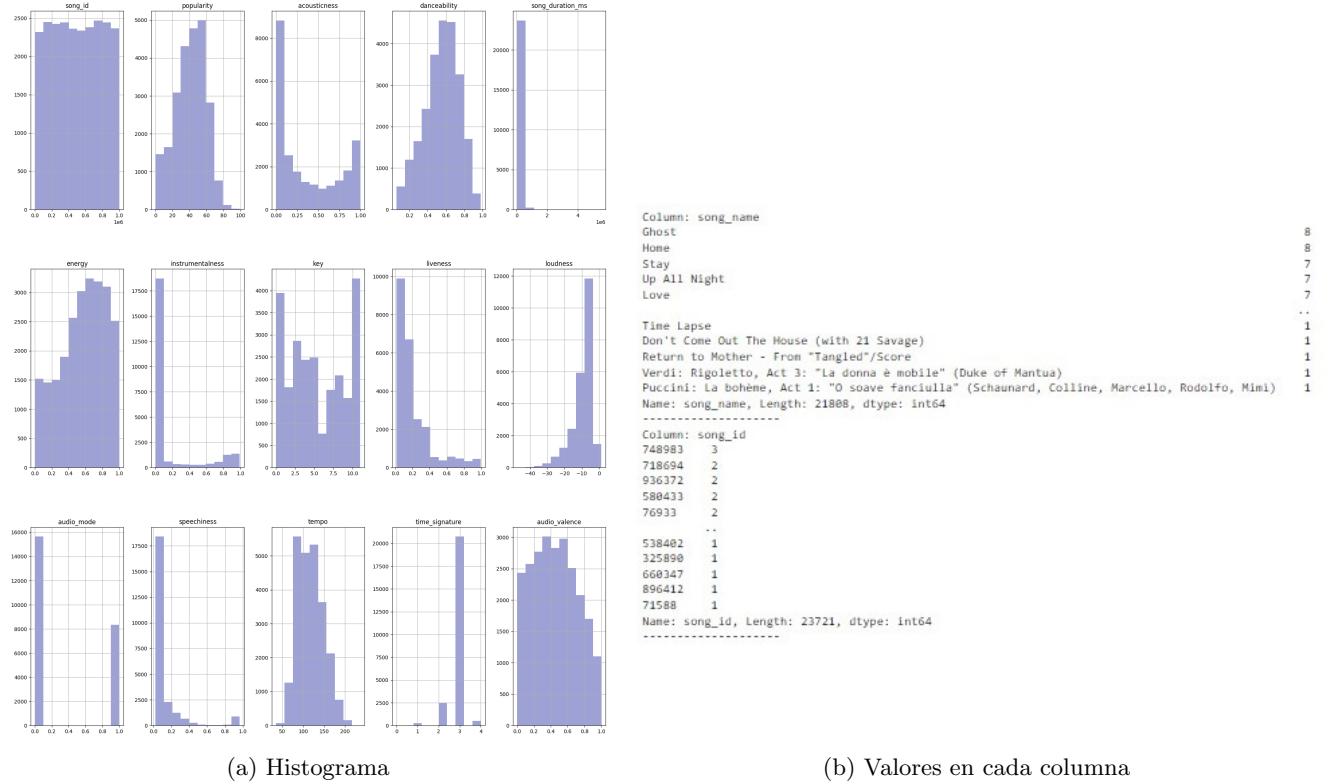
(b) Comprobación de ?

	<code>song_id</code>	<code>popularity</code>	<code>acousticness</code>	<code>danceability</code>	<code>song_duration_ms</code>	<code>energy</code>	<code>instrumentalness</code>	<code>key</code>	<code>liveness</code>	<code>loudness</code>	<code>audio_mode</code>	<code>speechiness</code>	<code>tempo</code>	<code>time_signature</code>	<code>audio_valence</code>
count	24000.000000	24000.000000	24000.000000	24000.000000	24000000e+04	24000.000000	24000.000000	24000.000000	24000.000000	24000.000000	24000.000000	24000.000000	24000.000000	24000.000000	24000.000000
mean	500840.386825	41.052417	0.370186	0.652322	3.388258e+05	0.569191	0.149253	5.330167	0.214456	-9.596249	0.347825	0.121792	117.845065	2.997083	0.452112
std	288298.975063	18.108720	0.356401	0.188490	1.275382e+05	0.284864	0.303700	3.481212	0.197887	6.038361	0.478228	0.188332	30.926825	0.397899	0.259177
min	18.000000	0.000000	0.000001	0.050000	1.721300e+04	0.002683	0.000000	0.000000	0.015200	-47.490000	0.000000	0.022500	34.866000	0.000000	0.000000
25%	251669.600000	29.000000	0.038700	0.433000	1.830988e+05	0.381000	0.000000	3.000000	0.097000	-11.848250	0.000000	0.038700	92.938000	3.000000	0.236000
50%	500290.000000	43.000000	0.233000	0.570000	2.089300e+05	0.601000	0.000041	5.000000	0.128000	-7.818000	0.000000	0.050000	115.838000	3.000000	0.443000
75%	752274.750000	55.000000	0.728000	0.689000	2.658218e+05	0.787000	0.036500	8.000000	0.264000	-5.500000	1.000000	0.107000	139.162260	3.000000	0.685000
max	999979.000000	99.000000	0.990000	0.980000	5.552217e+06	0.999000	0.993000	11.000000	0.996000	1.342000	1.000000	0.985000	238.735000	4.000000	1.000000

Figure 2: Estadísticas descriptivas

Contamos con un conjunto de datos que abarca **24000 canciones**, caracterizadas mayormente por atributos numéricos y sin la presencia de valores faltantes. Al examinar ciertas columnas, como `key`, `audio_mode`, y `time_signature`, notamos que poseen un número reducido de valores únicos, sugiriendo la posibilidad de tratarlas como atributos categóricos durante el entrenamiento de nuestros modelos.

Adicionalmente, nos enfrentamos a una característica no numérica, `song_name`, que podría resultar de interés más adelante, ya que podría contener variables adicionales que influyen en la popularidad de una canción. A continuación, realizaremos un análisis más detallado de los valores presentes en nuestras columnas.



(a) Histograma

(b) Valores en cada columna

Al revisar los recuentos de valores de las columnas, identificamos algunas observaciones destacadas:

- Instancias de canciones con el mismo nombre:** Se observa la presencia de varias canciones con nombres comunes como "Ghost", "Home", y "Love". Esta observación sugiere que el nombre de la canción podría ser una variable influyente en su lanzamiento, lo que posiblemente contribuiría a mejorar los resultados de nuestros modelos.
- Valores duplicados en la columna song_id:** Se identificaron duplicados en la columna `song_id`. Este hallazgo es relevante ya que un `song_id` debería referenciar una única canción. Antes de abordar estas duplicaciones, es crucial analizar las muestras en cuestión, ya que podría haber casos legítimos de canciones distintas compartiendo el mismo identificador. Si tras el análisis se confirma que las duplicaciones no son apropiadas, consideraremos la eliminación de la columna `song_id`, originalmente pensada para ser utilizada como índice.

Valores duplicados en song_id

	song_name	song_id	popularity
1094	Entre deux eaux	290857	0
1586	18 Days	824810	48
1736	Sahel	609889	30
2238	It's Christmas Time Again	110928	50
3170	Mesmerize	823581	68
...
23689	Ffunny Ffriends	572408	50
23809	Touch	294106	41
23819	Vacation	380788	67
23857	Inspiration Information	688045	50
23942	Quit Breaking My Heart (Reprise) - Reprise	390055	12

279 rows × 16 columns

Figure 4: Filas con valores duplicados.

song_name	song_id	popularity	acousticness	danceability	song_duration_ms	energy	instrumentalness	key	liveness	loudness	audio_mode	speechiness	tempo	time_signature	audio_valence
Lately	609889	81	0.0335	0.820	243080	0.812	0.000	3	0.1320	-5.843	0	0.0438	92.024	3	0.573
Sahel	609889	30	0.8380	0.497	384075	0.271	0.883	8	0.0804	-14.980	0	0.0337	153.035	2	0.194
<hr/>															
Knew U	290857	51	0.128	0.822	191147	0.742	0.00120	2	0.160	-5.384	0	0.2400	98.384	3	0.700
Entre deux eaux	290857	0	0.541	0.474	194026	0.305	0.00000	3	0.522	-13.282	0	0.0987	79.448	3	0.621

Figure 5: Muestras de canciones con el mismo id

Se puede observar que hay canciones diferentes compartiendo el mismo id, por lo que surge la consideración de eliminar `song_id`, en lugar de utilizarlo como índice, como se había planeado inicialmente. Esta medida no solo abordaría posibles ambigüedades en la asignación de identificadores, sino que también simplificaría la estructura del conjunto de datos, facilitando su análisis y modelado subsiguientes.

Dado que esta columna no se incluiría como característica de entrada en el entrenamiento de los modelos, se ha decidido eliminar esta columna y optar por una indexación numérica estándar.

Tratamiento de columnas categóricas

Hemos optado por tratar las columnas `key`, `audio_mode`, y `time_signature` como variables categóricas, reconociendo su naturaleza discreta. En esta etapa, procedemos a aplicar el método `pd.get_dummies()` para llevar a cabo la codificación one-hot de estas columnas.

Este proceso resulta en la creación de un total de 19 nuevas columnas, cada una representando una categoría única presente en las columnas originales. Este enfoque nos permite gestionar de manera efectiva la información categórica, facilitando su integración en futuros análisis y modelos.

Tratamiento de song_name

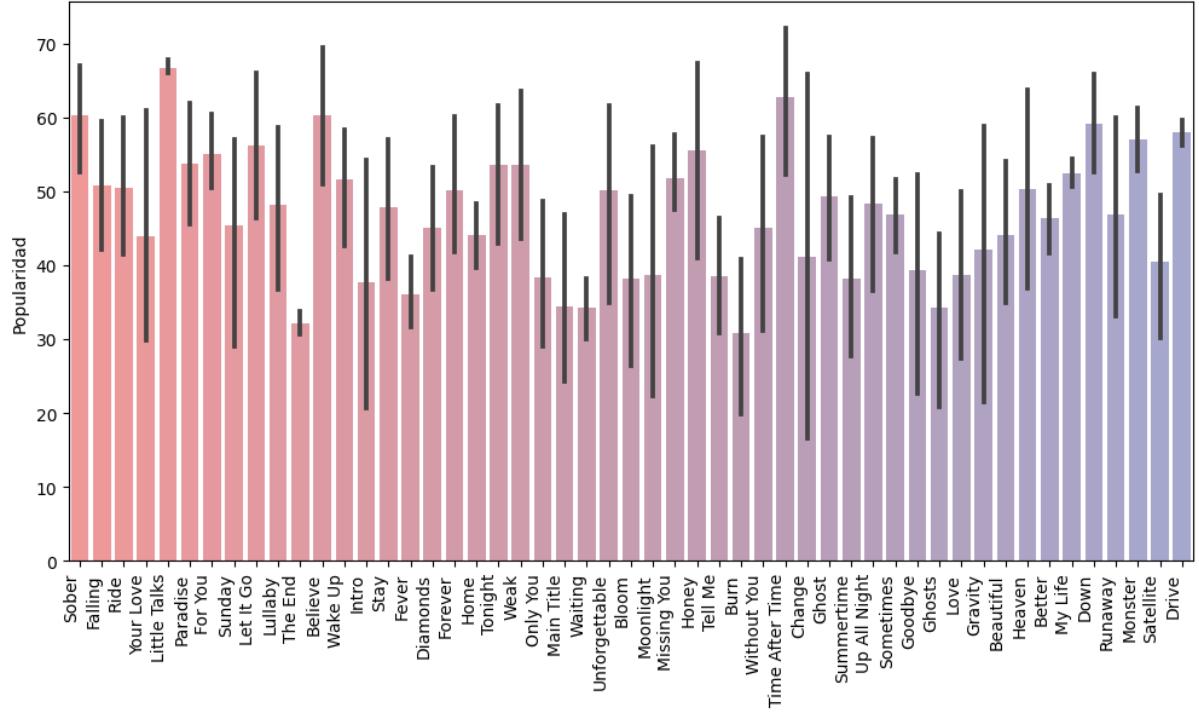


Figure 6: Nombres más duplicados y su popularidad

Las canciones con títulos presentes en la lista de nombres más recurrentes (≥ 5) manifiestan una tendencia hacia una popularidad situada en el rango de 30-70. A pesar de la variabilidad considerable en la popularidad dentro de canciones con el mismo nombre, consideramos que el nombre podría ejercer una influencia.

Con esta hipótesis, exploraremos la posibilidad de que la longitud del título pueda impactar en la popularidad de una canción. Para abordar esta cuestión, introduciremos una nueva columna denominada `title_length`, que reflejará la longitud de cada título.

En la misma línea, exploraremos con el word embedding para la columna `song_name`. La presencia de palabras específicas en los títulos podría ejercer una influencia significativa en la popularidad de una canción. Para cuantificar esta relación, transformaremos esta columna de cadenas de texto en una representación numérica que emplearemos en nuestro proceso de entrenamiento. Este enfoque nos permitirá capturar y modelar de manera efectiva la influencia semántica de las palabras clave presentes en los títulos de las canciones. Primero limpiaremos nuestro texto (quitar tildes, paréntesis, signos de puntuación, números, símbolos y pasaremos todo a minúsculas) y aplicaremos un modelo de spacy para generar vectores de 300 valores.

Realizado todo esto, eliminamos finalmente la columna `song_name`, teniendo al final un total de 330 columnas para nuestros entrenamientos.

2.2 Entrenamiento de modelos y evaluación

Antes de avanzar con el entrenamiento y evaluación de nuestros modelos, hemos dividido nuestro conjunto de prueba utilizando la función `train_test_split` para crear dos subconjuntos: uno de **prueba** y otro de **validación**. Este último se utilizará para aplicar las medidas de error pertinentes, ya que no disponemos de los valores reales del conjunto de prueba.

Además, previo al inicio de esta sección, decidimos emplear la biblioteca PyCaret. Esta herramienta nos permite explorar de manera eficiente un amplio conjunto de modelos en un corto período de tiempo mediante la función `compare_models()`. Luego de realizar la comparación, hemos identificado ciertos modelos que muestran un Error Absoluto Medio (MAE) más bajo. Estos modelos destacan por su rendimiento superior, y planeamos utilizarlos también en nuestro análisis.

A continuación, procederemos a entrenar un total de **8 modelos**, que incluyen tanto modelos seleccionados previamente como aquellos identificados por PyCaret con un menor Error Absoluto Medio (MAE).

Posterior al entrenamiento, evaluaremos el rendimiento de cada modelo utilizando varias métricas, como `cross_val_score` con '`neg_mean_squared_error`', `R2`, `MSE` y `MAE`. Con estos resultados en consideración, avanzaremos en el entrenamiento de los modelos que hayan demostrado un mejor desempeño, y exploraremos la posibilidad de mejorar aún más su rendimiento mediante la aplicación de técnicas de **reducción de dimensionalidad** y **selección de características**.

SGD Regressor

La técnica base por antonomasia para predecir valores es la regresión lineal y a pesar de funcionar para modelos muy sencillos no se descartó la idea de probarla.

Al tener un conjunto relativamente grande se nos ocurrió que podría ser efectivo el **Descenso de Gradiente Estocástico (SGD)**, este método se fundamenta en la técnica de descenso de gradiente un método de optimización que permite obtener los valores de los parámetros de una función que la hacen **mínima** y que nos permite obtener unos resultados como si se tratase de una regresión lineal pero de forma **escalable**.

Antes de empezar a entrenar el modelo, utilizamos `GridSearchCV`, una técnica que nos ayuda a buscar los mejores hiperparámetros. Los hiperparámetros que quisimos ajustar para obtener los mejores resultados fueron: `alpha`, `learning_rate` y `penalty`.

Los resultados obtenidos fueron los siguientes: '`alpha': '0.1'`', '`'learning_rate': 'adaptive'`', '`'penalty': 'elasticnet'`'.

```
cv_score: -237.6030526727619
R2 Error: 24.524517409863 %
Error MSE: 246.96279707550474
Error MAE: 12.505259394645595
```

Figure 7: Resultados con `SGDRegressor()`

Random Forest

Este método consiste en la combinación de varios árboles de decisión con el fin de alcanzar una solución.

Los **árboles de decisión** son algoritmos que infieren grafos en forma de árbol donde cada nodo sirve para tomar una decisión. En cada nodo se establece una bifurcación en función de los valores de un atributo concreto y al final de cada camino siempre hay una **etiqueta** (un nodo hoja). Una de las principales ventajas reside en su **alta explicabilidad** (Explainable-AI). Aunque se suelen utilizar para problemas de clasificación, también pueden resultar muy útiles para problemas de regresión.

Al igual que en el Descenso de Gradiente necesitamos, en primer lugar, obtener los mejores hiperparámetros de acuerdo con nuestro conjunto de datos de entrenamiento. De nuevo, haremos uso de la función `GridSearchCV` para llevar a cabo esta búsqueda exhaustiva.

Esta vez, el proceso de búsqueda lleva mucho tiempo, por ende, hemos almacenado los resultados. Los hiperparámetros que obtuvimos fueron los siguientes: `'criterion': 'squared_error'`, `'n_estimators': 200`.

```
cv_score: -204.62183520075087
R2 Error: 36.812042449014314 %
Error MSE: 206.75687259956558
Error MAE: 11.194867342509921
```

Figure 8: Resultados con `RandomForestRegressor()`

SVM Regressor

El siguiente modelo que hemos entrenado son las **Máquinas de Soporte Vectorial (SVM)**. Se basan en la **optimización**: se persigue la obtención del **hiperplano que minimice el error** y que consiga la **mayor separación** entre los ejemplos con el objetivo de alcanzar la máxima generalidad posible. Además, implementan el **kernel trick**.

No es novedad el uso de técnicas de búsqueda exhaustiva de hiperparámetros para lograr los mejores resultados. Esta ocasión, los mejores hiperparámetros son: `'kernel': 'poly'`, `'degree': 3`, `'gamma': 'auto'`, `'C': 10.0`.

```
cv_score: -289.69933834810615
R2 Error: 8.571089427797263 %
Error MSE: 299.16389685235384
Error MAE: 13.337021268820852
```

Figure 9: Resultados con `SVR()`

MLP Regressor

Este modelo es un tipo de Red Neuronal cuya arquitectura básica se fundamenta en una **capa de entrada, capas ocultas y la capa de salida**. Además se incorporó la técnica de **back-propagation** para adaptar los pesos propagando los errores hacia atrás.

Nuevamente, buscamos los mejores hiperparámetros con `GridSearchCV()`: `'activation': 'logistic'`, `'learning_rate': 'adaptive'`, `'solver': 'sgd'`.

```
cv_score: -410.113746114068
R2 Error: -40.354096988105994 %
Error MSE: 459.25165608307
Error MAE: 16.447451058608678
```

Figure 10: Resultados con `MLPRegressor()`

Extra Trees Regressor

Es un algoritmo de regresión basado en **ensambles** que utiliza **árboles de decisión**. A diferencia de Random Forest, construye árboles considerando todas las características en cada división y utilizando umbrales de división y observaciones seleccionadas de forma aleatoria. La aleatoriedad adicional busca reducir la varianza del modelo, mejorando su capacidad de generalización en nuevos datos.

En estos casos, dado el tiempo de ejecución bajo en comparación con los modelos anteriores, la búsqueda de hiperparámetros se ha realizado manualmente. En este modelo, obtuvimos los siguientes hiperparámetros: `'n_estimators': 400`, `'max_depth':None` y `'min_samples_split': 2`.

```
cv_score: -202.46934349302845
R2 Score: 37.85562602060472 %
MSE: 203.34217011635562
MAE: 11.001233940972224
```

Figure 11: Resultados con `ExtraTreesRegressor()`

Light Gradient Boosting Machine

Es un algoritmo eficiente de regresión basado en **Gradient Boosting**. Su enfoque de construcción de árboles, conocido como "hoja-first", prioriza la expansión de las hojas más beneficiosas para la reducción de la función de pérdida. Diseñado para ser ligero y rápido, el LGBM Regressor implementa técnicas de regularización y ofrece un rendimiento eficaz en términos de velocidad y eficiencia computacional.

Realizando pruebas con `n_estimators`, `learning_rate`, `max_depth` y `min_child_samples`, obteniendo: `'n_estimators' :500`, `'learning_rate': 0.1`, `'max_depth':-1`, `'min_child_samples':20`

```

cv_score: -208.08284666662547
R2 Score: 35.48741564122485 %
MSE: 211.09117468424856
MAE: 11.31048441631293

```

Figure 12: Resultados con `LGBMRegressor()`

Gradient Boosting Regressor

El LGBM Regressor y el Gradient Boosting Regressor comparten la base del Gradient Boosting para la regresión, pero difieren en aspectos clave. El *LGBM Regressor* destaca por su *eficiencia computacional* gracias a su enfoque "*hoja-first*" y está diseñado para gestionar grandes conjuntos de datos de manera eficiente. En contraste, el *Gradient Boosting Regressor* puede ser más lento en algunos escenarios, ya que sigue un enfoque "*profundidad-first*". Además, mientras que ambos modelos implementan técnicas de regularización, sus estrategias específicas difieren, con el LGBM Regressor utilizando penalizaciones en las hojas y el *Gradient Boosting Regressor* aplicando *restricciones a la profundidad* de los árboles.

Contando con estas diferencias, decidimos probar su rendimiento.

```

cv_score: -212.05165471209642
R2 Score: 34.21228518957605 %
MSE: 215.2635200892515
MAE: 11.599396569926826

```

Figure 13: Resultados con `GradientBoostingRegressor()`

Extreme Gradient Boosting

Es un algoritmo de regresión basado en **ensamblado** que ha demostrado ser altamente efectivo en una variedad de problemas. Utiliza una técnica de **Gradient Boosting** y está diseñado para ser extremadamente eficiente y preciso. XGBoost destaca por su **velocidad** de entrenamiento, capacidad para manejar conjuntos de **datos grandes y complejo**, y su robustez frente al sobreajuste.

```

cv_score: -226.17030968817153
R2 Score: 30.418226024328575 %
MSE: 227.67803446616142
MAE: 11.789893539007753

```

Figure 14: Resultados con `XGBRegressor()`

Para realizar una comparación más efectiva de los resultados obtenidos con cada método, procederemos a representar gráficamente dichos resultados, incluyendo tanto los errores como los puntajes obtenidos mediante validación cruzada.

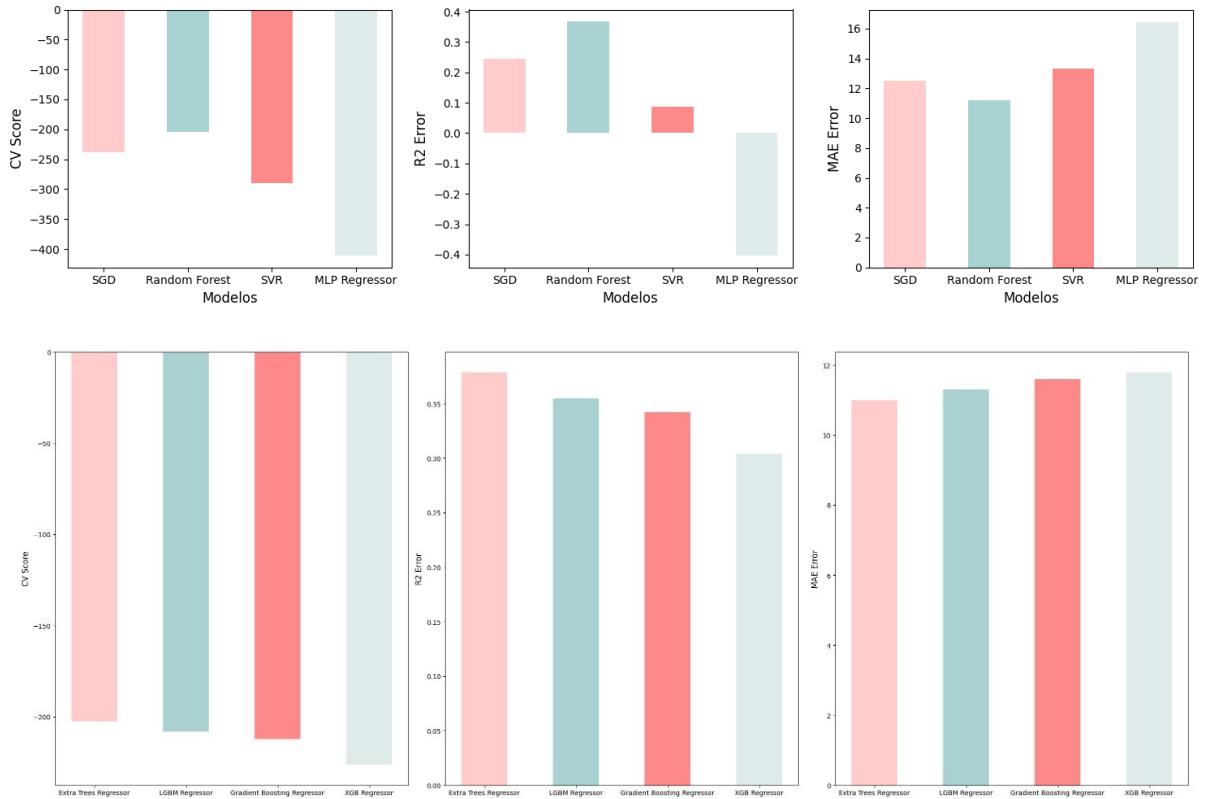


Figure 15: Comparación de resultados

A lo largo de las próximas secciones, nos enfocaremos en potenciar aún más el rendimiento de nuestros tres modelos más prometedores hasta la fecha: **ExtraTrees**, **LGBM Regressor** y **RandomForest**. Al afinar y ajustar estos modelos, buscamos maximizar su capacidad predictiva y lograr un rendimiento aún más robusto en la tarea de regresión.

Reducción de dimensionalidad

Para reducir dimensiones en nuestro extenso conjunto de datos, optaremos por técnicas eficientes como **Análisis de Componentes Principales** (PCA) y **Descomposición de Valor Singular Truncada** (Truncated SVD).

Estas opciones se destacan por su capacidad para preservar la información esencial mientras reducen el número de dimensiones de manera eficaz. La elección se basa en su **eficiencia** en tiempo de ejecución, especialmente beneficiosas en comparación con enfoques más intensivos computacionalmente en conjuntos de datos grandes, como t-SNE o UMAP.

Resultados PCA y Truncated SVD

Primero, realizaremos un análisis exploratorio con PCA para evaluar el número óptimo de dimensiones al cual convendría reducir nuestro conjunto de datos. Nos guaremos por la condición de que se considere adecuado reducir a n dimensiones cuando podamos explicar al menos el 90% de la varianza. Este enfoque nos permitirá determinar una reducción de dimensionalidad efectiva, preservando la información esencial del conjunto de datos.

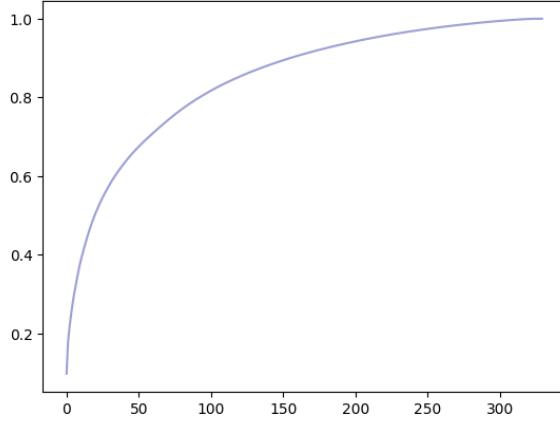
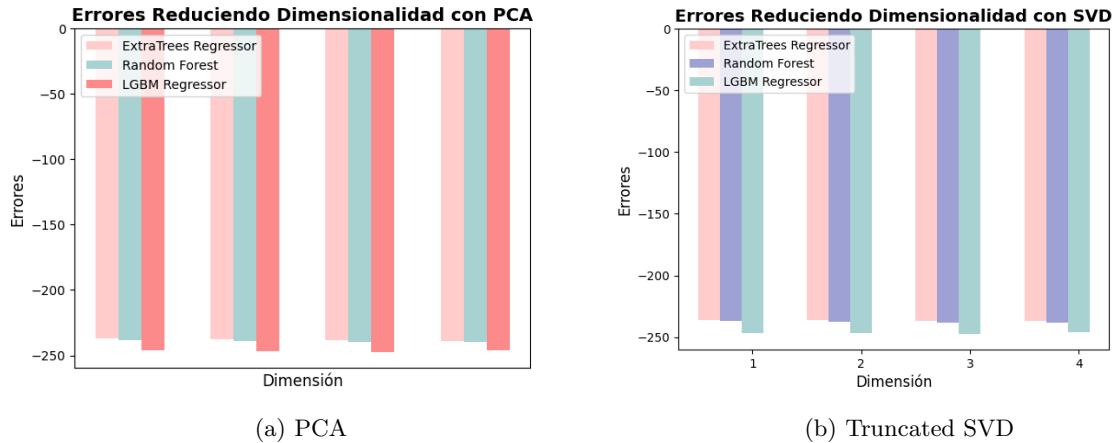


Figure 16: Explicabilidad de la variaza

Al utilizar 150 componentes, hemos logrado explicar el 90% de la varianza en nuestro conjunto de datos. Por ende, nos adentraremos en un análisis detallado de los resultados derivados de esta reducción de dimensionalidad. Considerando también la eficiencia temporal, hemos decidido realizar pruebas adicionales en un rango de 150 a 300 componentes, aumentando de 50 en 50. Esto nos proporcionará una perspectiva más completa con un total de 4 pruebas.

Al igual que en secciones previas, con el objetivo de facilitar la comparación, presentaremos visualmente los resultados obtenidos, enfocándonos específicamente en los resultados de la validación cruzada.



Como se evidencia en los resultados, los valores de la validación cruzada han disminuido. Es crucial destacar que, al aplicar el **error cuadrático negativo**, hemos observado un deterioro en el rendimiento de los modelos.

Esto puede atribuirse al hecho de que, al reducir la dimensionalidad, se **sacrifica información** detallada que estaba presente en el conjunto de datos original. La **complejidad de predecir la popularidad** de las canciones como problema implica que la pérdida de características relevantes al reducir dimensiones puede afectar negativamente la capacidad predictiva. La popularidad de una canción depende de una interacción compleja de diversas características

musicales y contextuales, y la reducción de dimensionalidad podría no capturar completamente esta complejidad, afectando así la calidad de las predicciones.

Feature Selection

La selección de características se fundamenta en evaluar la importancia relativa de cada característica en relación con el objetivo de la predicción, utilizando métodos estadísticos, algoritmos de aprendizaje automático o técnicas específicas diseñadas con este propósito.

En la primera etapa, realizaremos un análisis empleando el modelo Lasso para obtener una perspectiva sobre la importancia de las características. Posteriormente, considerando la eficiencia computacional, aplicaremos un único método de selección de características: Select K-best.

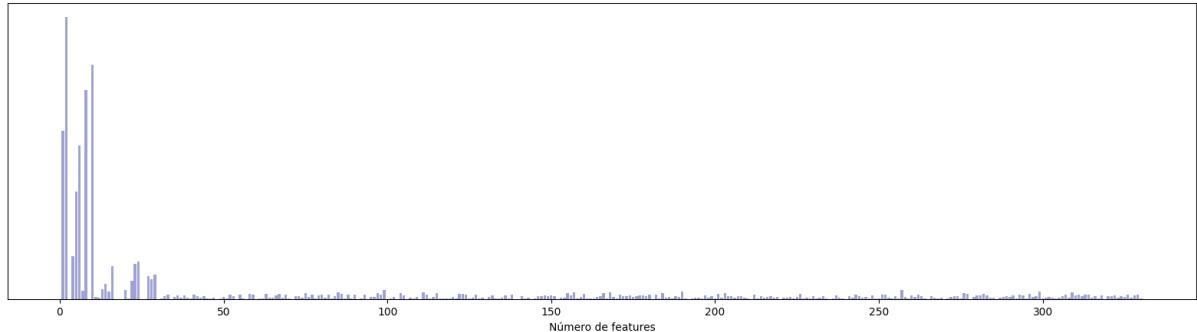


Figure 18: Importancia de las características

Considerando un umbral de importancia establecido en 0.1 para filtrar la relevancia proporcionada por Lasso, realizaremos pruebas utilizando el método SelectKBest. Dado que el filtro nos ha proporcionado un total de 163 características consideradas importantes, llevaremos a cabo una selección de 150 características para evaluar posibles mejoras.

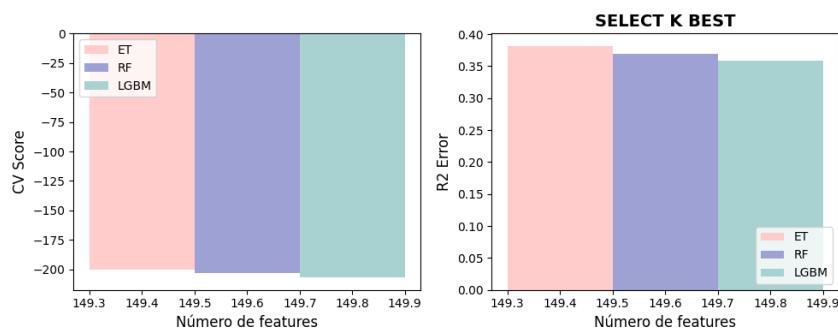


Figure 19: Select k-best

Se puede observar que hemos conseguido mejorar nuestro modelo inicial de **Extra Trees Regressor**, pasando de un **cv_score: -202.469** y **R2 Score: 37.855%** a **-200.579** y **38.192%**. Utilizaremos este modelo para predecir la popularidad de una nueva selección de canciones.

2.3 Entrenamiento del conjunto de datos de menor dimensionalidad

Con el objetivo de respaldar la importancia de integrar el título y su semántica en la predicción de la popularidad, hemos realizado una nueva serie de entrenamientos utilizando el conjunto de datos sin incluir las dimensiones del word embedding del título. Además, consideramos relevante resaltar la variación en el rendimiento predictivo al emplear la codificación one-hot para las columnas que consideramos categóricas.

En esta sección, detallaremos los resultados obtenidos con los primeros cuatro modelos en nuestro conjunto de datos original, manteniendo la columna 'title-length'. Dada la diferencia de más de 310 componentes en las columnas consideradas, se percibió como poco influyente la eliminación de dicha columna. Este enfoque nos permitirá destacar de manera efectiva la contribución del título y la codificación categórica en la predicción de la popularidad musical.

A continuación, analizaremos los resultados de cada sección, destacando las diferencias y similitudes en el rendimiento de los modelos.

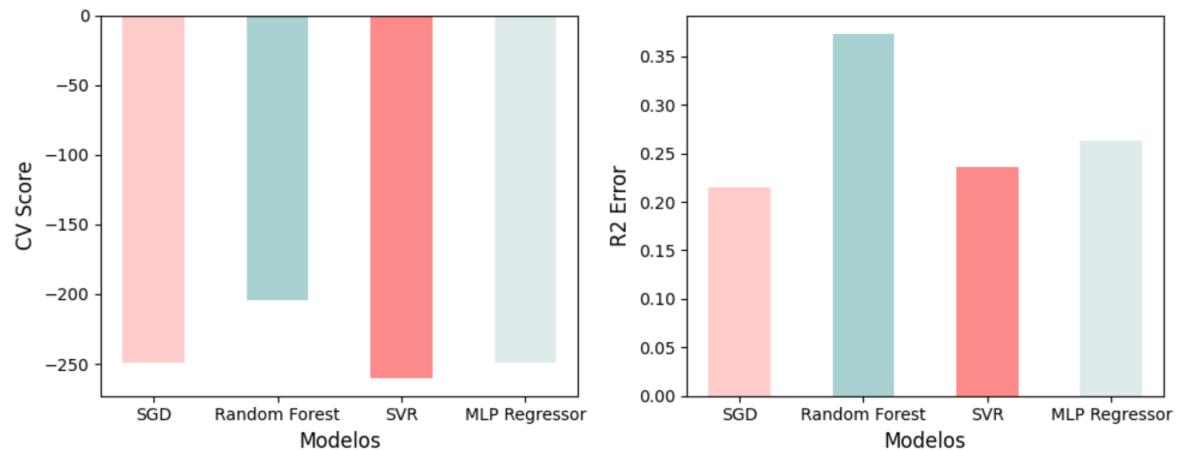
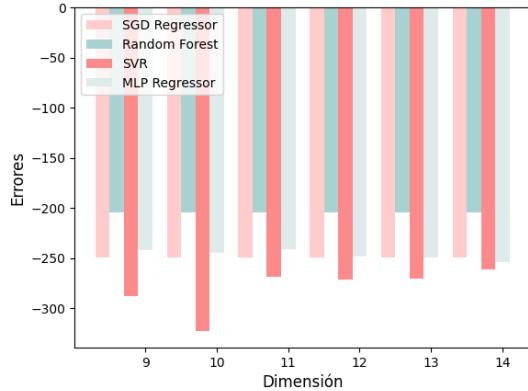


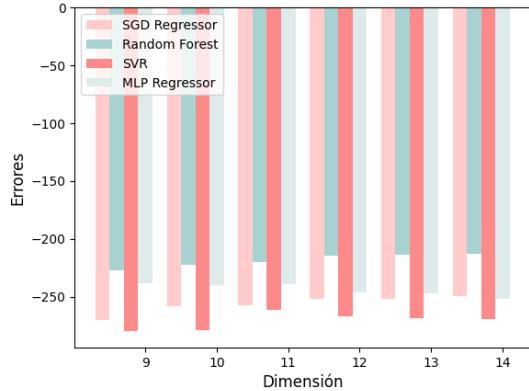
Figure 20: Resultados con datos completos

En términos generales, los cuatro modelos muestran un rendimiento mejorado en este conjunto de datos, destacando que el mejor modelo en ambas versiones del dataset es **Random Forest**. Al comparar los resultados de los dos Random Forest, se observa que trabajar con el conjunto de datos original proporciona una mejora, ya que el **MAE** disminuye de **11.194** a **11.176**.

Sin embargo, es relevante señalar que **ningún modelo logra superar los resultados del ExtraTrees Regressor** en nuestro conjunto de 330 características, que inicialmente obtuvo un MAE de **11.001**. Esta diferencia de desempeño es notable. Además, es importante tener en cuenta que el modelo Extra Trees presenta un **tiempo de ejecución significativamente inferior** en comparación con el Random Forest.



(a) PCA



(b) Truncated SVD

Reducción de dimensionalidad

La implementación de la **reducción de dimensionalidad**, en este caso, también conlleva un **deterioro** en el rendimiento de los modelos. Con nuestro mejor modelo, **Random Forest**, aumenta el **MAE** inicial de **11.176** a **11.401** con **PCA** y **11.425** con **Truncated SVD**. Esto sugiere que la predicción de este problema en particular requiere una amplia variedad de características para lograr una predicción más precisa.

Feature selection

Este patrón se refleja también en los resultados de la selección de características. En esta ocasión, al disponer de un conjunto reducido de características, decidimos explorar dos técnicas adicionales de selección.

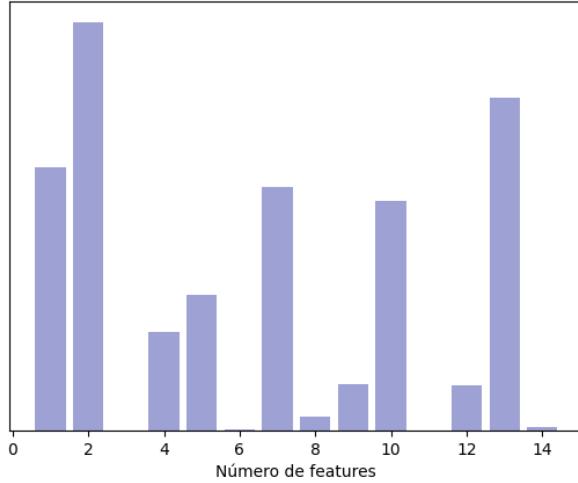
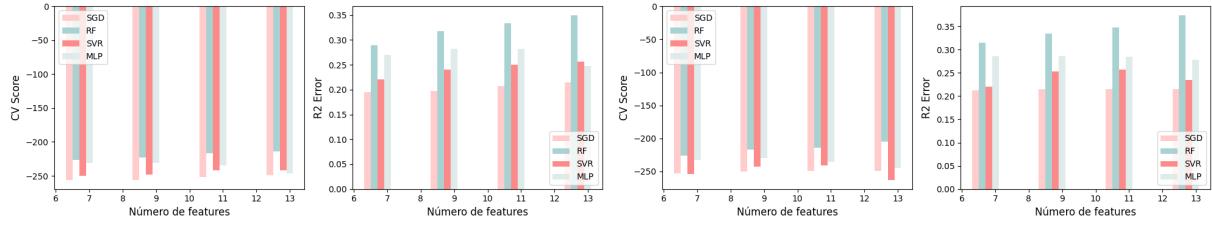


Figure 22: Importancia de características

En el análisis de importancia, el modelo Lasso señaló aproximadamente unas 7 características como relevantes. Además, se observa que la característica 14 (`title.length`) no es considerada importante, confirmando nuestra suposición inicial.



(a) Select from model

(b) Sequential feature selection

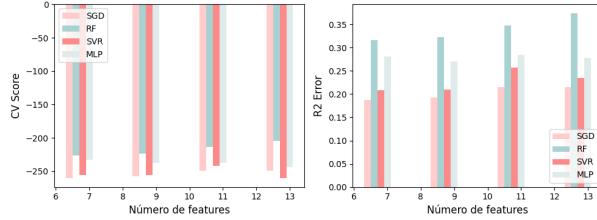


Figure 24: Select k-best

Concluyendo este apartado, hemos destacado la importancia del título al observar las diferencias entre los mejores modelos en los dos conjuntos de datos. Asimismo, hemos llegado a la conclusión de que la reducción de características no conduce a una mejora en los resultados.

2.4 Resultados

Nuestro mejor modelo es: un ExtraTreesRegressor con `n_estimators=400`, `max_depth=None` y `min_samples_split=2`, aplicado a un `SelectKBest(150)` con `StandardScaler()`.

3 Visualización de datos

3.1 Relaciones estadísticas

Wordcloud de los títulos

Las "nubes de palabras" o "word clouds" emergen como una herramienta visual poderosa y accesible, capaz de **resaltar** las palabras más **prominentes** en un conjunto de datos de texto.

Estas proporcionan una visión instantánea e intuitiva de las palabras clave o términos más relevantes en un conjunto de datos de texto. Son especialmente útiles para resumir grandes cantidades de información de manera visualmente atractiva. Además, las nubes de palabras pueden capturar la esencia y el tema principal de un corpus de texto, lo que facilita la identificación de patrones, tendencias o en este caso, la destacada influencia de palabras en la popularidad de las canciones a través de sus títulos.



(a) Wordcloud de las más populares



(b) Wordcloud de las más populares

Como se puede observar, las canciones menos populares presentan una temática clásica, evidenciada por la frecuente aparición de palabras como 'allegro', 'symphony', 'act II', 'Instrumental', 'Overture', que son comunes en el dominio clásico.

En contraste, el resto de las canciones destacan por las colaboraciones ('feat'), remix y nombres que incluyen temáticas como 'Time', 'Love', 'Girl', 'Man', 'Heart'.

Matriz de correlación

La matriz de correlación es un conjunto de valores numéricos dispuestos en forma de matriz, donde cada entrada representa la correlación entre dos variables. Este valor de correlación, que oscila entre -1 y 1, cuantifica la fuerza y la dirección de la relación lineal entre las variables. Un valor cercano a 1 indica una correlación positiva, mientras que un valor cercano a -1 sugiere una correlación negativa. Esto nos permite identificar asociaciones fuertes o débiles entre las variables.

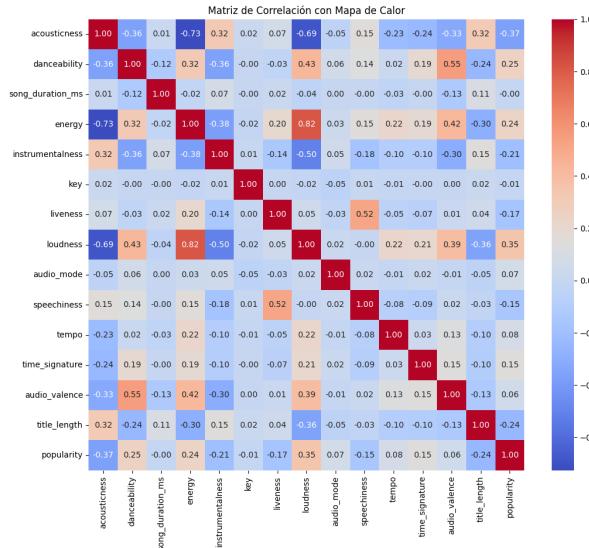
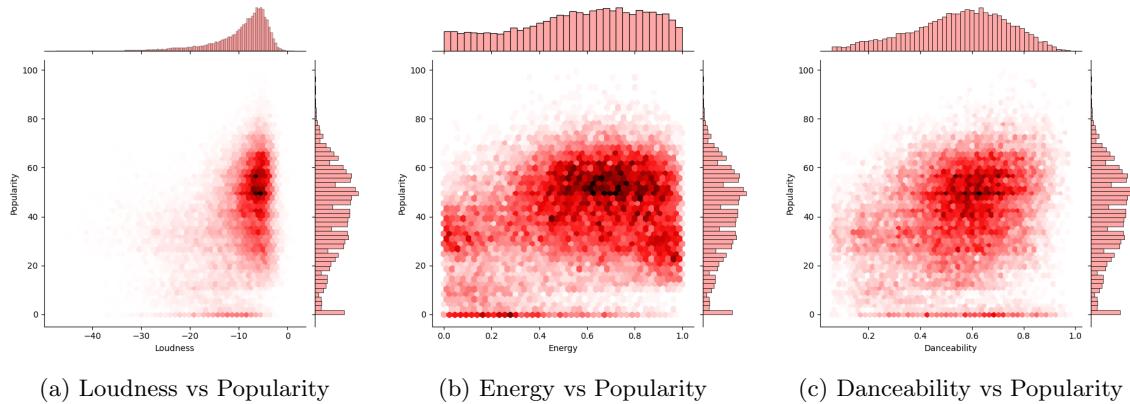


Figure 26: Matriz de correlación

Tenemos características como: `loudness`, `energy` y `danceability` que parecen tener una correlación más alta con `popularity` que el resto de características

Joinplot

El joinplot muestra la dispersión de puntos entre las dos variables, lo que permite observar patrones, tendencias o agrupaciones en los datos. Esto nos puede indicar la naturaleza de las canciones generalmente populares. Más adelante, con las técnicas de reducción de dimensionalidad, trataremos de indagar más en estos aspectos.



3.2 PCA

Aplicamos PCA para reducir a dos dimensiones y obtener las 2 componentes principales. Además añadiremos flechas asociadas a cada componente principal que nos aportarán información sobre la influencia de cada una.

Excluyendo la distinción de colores basada en la popularidad, se observa una división en dos grupos notables en la visualización, siendo el grupo mayoritario de forma ovalada. Este patrón sugiere que uno de los componentes principales tiene una influencia más significativa que el otro en la distribución de las canciones en el espacio bidimensional. Esta inferencia se ve respaldada por las flechas que indican la dirección y magnitud de los componentes principales, mostrando diferencias notables entre ellos. La presencia de dos grupos claramente definidos y la asimetría en la distribución sugieren que ciertos atributos o combinaciones de atributos pueden estar contribuyendo de manera distintiva a la variabilidad observada en el conjunto de datos.

Se destaca que la Componente Principal 1 es responsable de explicar casi el 70% de la varianza en los datos. Este hallazgo está en consonancia con la observación del gráfico previo, donde al representar las flechas asociadas a los componentes principales, se evidenciaba que la flecha correspondiente a la derecha era considerablemente más extensa. Esta diferencia en magnitudes sugiere que la Componente Principal 1 tiene una importancia más significativa en la variabilidad

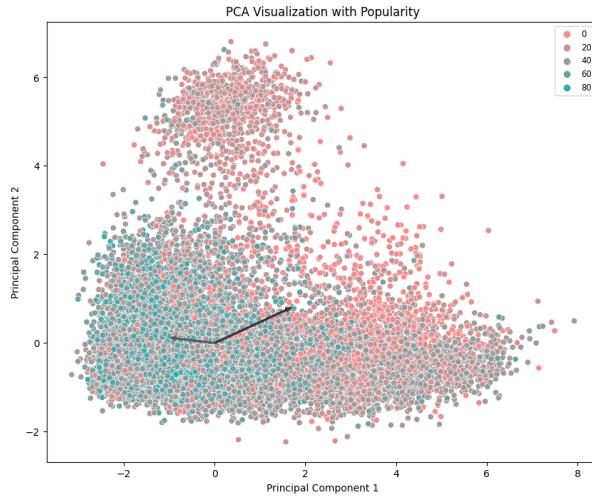


Figure 28: PCA

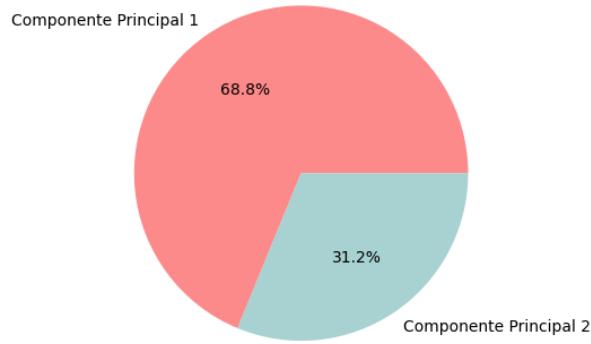


Figure 29: PCA pie

del conjunto de datos, respaldando la conclusión de que ciertos atributos o combinaciones de atributos relacionados con la Componente Principal 1 contribuyen de manera destacada a la estructura del espacio bidimensional observado.

Coeficientes de carga PCA

Los coeficientes de carga, también conocidos como "loadings" en inglés, son los pesos asignados a cada variable original en la construcción de estos componentes principales. Estos coeficientes capturan la contribución relativa de cada variable a la variabilidad total del conjunto de datos. Un coeficiente de carga elevado indica una fuerte influencia de una variable específica en la formación de un componente principal.

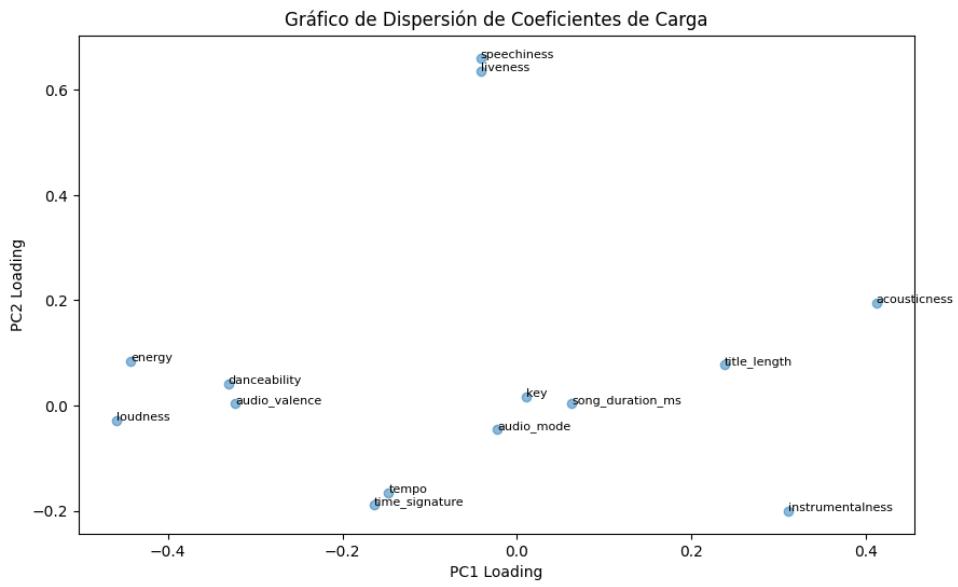


Figure 30: Gráfico de dispersión de coeficientes de carga

Las variables que más influyen en la **componente 1** son **acousticness**, **instrumentalness** y **title_length**. Por otro lado, las que más influyen en la **segunda componente** son **speechiness** y **liveness**.

Segmentación de los PCA en clusters

Además, al considerar la información de popularidad codificada por colores en la visualización de PCA, se observa una densidad de valores de popularidad significativamente mayores en el lado inferior izquierdo del gráfico. Para profundizar en este patrón, emplearemos un modelo de clustering k-means.

Estableceremos el número de clusters **n_clusters** en 3, teniendo en cuenta tanto la estructura como la información de colores en la gráfica. Este enfoque nos permitirá explorar y categorizar posibles agrupamientos o patrones latentes en los datos.

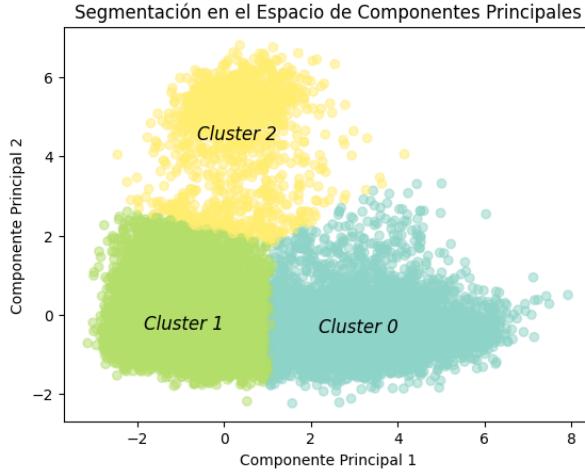
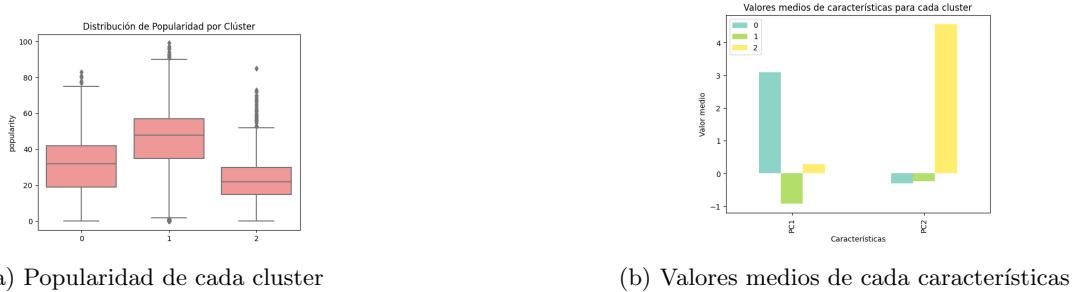


Figure 31: Segmentación en el espacio de componentes principales

Una vez obtenidos estos clusters, que se ajustan a nuestras observaciones previas, visualizaremos la distribución de la popularidad en cada cluster. Esto nos permitirá destacar y validar las hipótesis formuladas en relación con la concentración de popularidad en el lado inferior izquierdo de la gráfica PCA.

Una vez obtenidos estos clusters, que se ajustan a nuestras observaciones previas, visualizaremos la distribución de la popularidad en cada cluster. Esto nos permitirá destacar y validar las hipótesis formuladas en relación con la concentración de popularidad en el lado inferior izquierdo de la gráfica PCA.

Por otro lado, graficar los centroides de cada cluster en un espacio bidimensional puede proporcionar información sobre la posición central de los grupos y ayudar a visualizar la separación entre los clusters.



Podemos ver que el cluster 1 tiene una distribución de popularidad más alta. Además, en el espacio bidimensional, el cluster de interés (cluster 1) se caracteriza principalmente por la Componente Principal 1 (PC1). De acuerdo con el análisis anterior, las variables que más influyen en la Componente 1 son `acousticness`, `instrumentalness` y `title_length`.

Considerando además el análisis del título de las canciones, donde se observa una clara influencia del género musical (clásica - menos popular), se podría argumentar que la importancia de `acousticness` e `instrumentalness` es lógica en este contexto. Estas características podrían estar contribuyendo significativamente a la formación del cluster 1, que se asocia con canciones menos populares y con características acústicas e instrumentales más prominentes.

3.3 T-SNE

T-SNE es una técnica de reducción de dimensionalidad que se destaca por su capacidad para visualizar agrupamientos y relaciones locales en los datos. Usando una perplexity de 60 obtenemos los siguientes componentes.

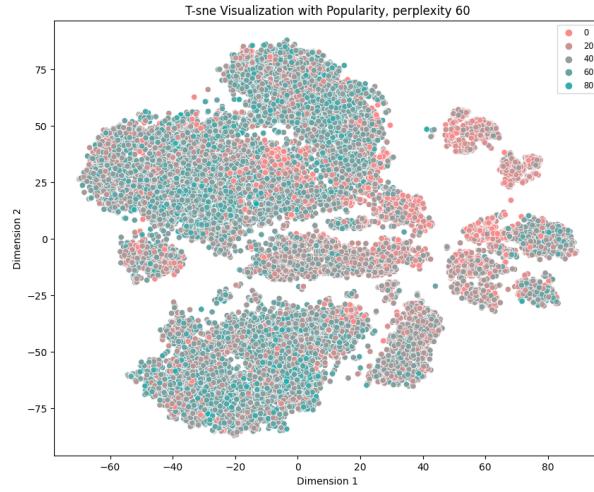


Figure 33: T_SNE con perplexity 60

Utilizando esta técnica podemos también observar ciertas características en cuanto a la forma y el la codificación de color, pero dada su agrupación con mayor numero de clusters, consideramos que k-means no devolvería insights muy cercanos a lo que se ve visualmente.

3.4 ISOMAP

ISOMAP es una técnica de reducción de dimensionalidad que busca preservar las distancias geodésicas entre todos los pares de puntos en un conjunto de datos. Al hacerlo, ISOMAP captura la estructura intrínseca de los datos, especialmente en casos donde las relaciones no lineales son cruciales.

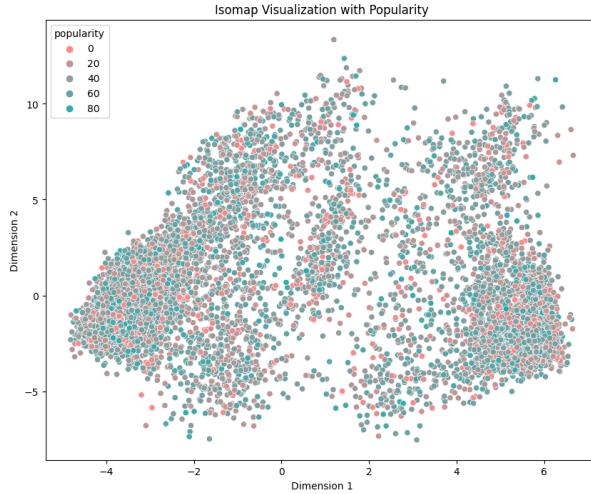


Figure 34: ISOMAP

Por otro lado, la visualización con ISOMAP no nos trae aparente caracterización de la popularidad.

4 Series temporales

4.1 Descripción del conjunto de datos

Nuestro conjunto de datos consta de cinco columnas: `PROI`, `SEXO`, `MESPAR`, `ANOPAR` y `count`. Al iniciar nuestra evaluación, nos enfocaremos en comprender la naturaleza de los datos. Esto incluye revisar varios aspectos, como la tipología de los datos presentes, la identificación de valores faltantes y únicos de las muestras, y adicionalmente, analizar las métricas descriptivas asociadas a cada una de las columnas.

	Data Type	Distinct Values	Missing Values	PROI	SEXO	MESPAR	ANOPAR	count
<code>PROI</code>	int64	52	0	7488.000000	7488.000000	7488.000000	7488.000000	7488.000000
<code>SEXO</code>	int64	2	0	26.500000	3.500000	6.500000	2018.500000	295.92054
<code>MESPAR</code>	int64	12	0	15.009333	2.500167	3.452283	1.707939	420.07736
<code>ANOPAR</code>	int64	6	0	1.000000	1.000000	1.000000	2016.000000	14.00000
<code>count</code>	int64	1041	0	25%	13.750000	1.000000	3.750000	2017.000000
				50%	26.500000	3.500000	6.500000	2018.500000
				75%	39.250000	6.000000	9.250000	2020.000000
				max	52.000000	6.000000	12.000000	2021.000000
								2902.00000

(a) Valores faltantes y tipología de los datos

(b) Estadísticas descriptivas

Afortunadamente, contamos con un DataFrame **completo** y **sin datos faltantes**. Todas las columnas contienen valores enteros, alineándose con el rango permitido por el tipo de datos `int64`, por lo que **no encontramos valores no numéricos o no representativos**.

El periodo temporal del conjunto de datos se extiende desde 2016 hasta 2021, proporcionando **información detallada** sobre **7,488 nacimientos**. Al analizar la distribución de **provincias**, observamos una equidad notable, evidenciada por un promedio y mediana de 26.5, lo que sugiere una **representación balanceada de datos** para cada provincia en estudio.

Respecto al **sexo** de los recién nacidos, la columna 'SEXO' utiliza una **codificación numérica**, donde 1 indica Varón y 6 indica Mujer. Las estadísticas revelan una distribución uniforme de los nacimientos a lo largo del año, ya que la media y mediana del mes de parto (MESPAR) se establecen en 6.5. Sin embargo, el **reuento de nacimientos** presenta una **variabilidad significativa**, con un mínimo de 14 y un máximo de 2,902, subrayando la amplitud en la frecuencia de nacimientos registrada.

A continuación, exploraremos aspectos adicionales para obtener una comprensión más profunda y relevante de los datos, buscando identificar patrones demográficos clave en el conjunto de datos.

Influencia del sexo y las provincias

Aprovecharemos la información temporal proporcionada por las columnas **MESPAR** y **ANOPAR** para crear una nueva columna **fecha**. Visualizaremos la cantidad de nacimientos a lo largo de los años, utilizando la distinción de colores según características específicas. Esta representación nos ayudará a identificar fácilmente la contribución de cada característica.

¿Existen diferencias notables en las tendencias de natalidad?

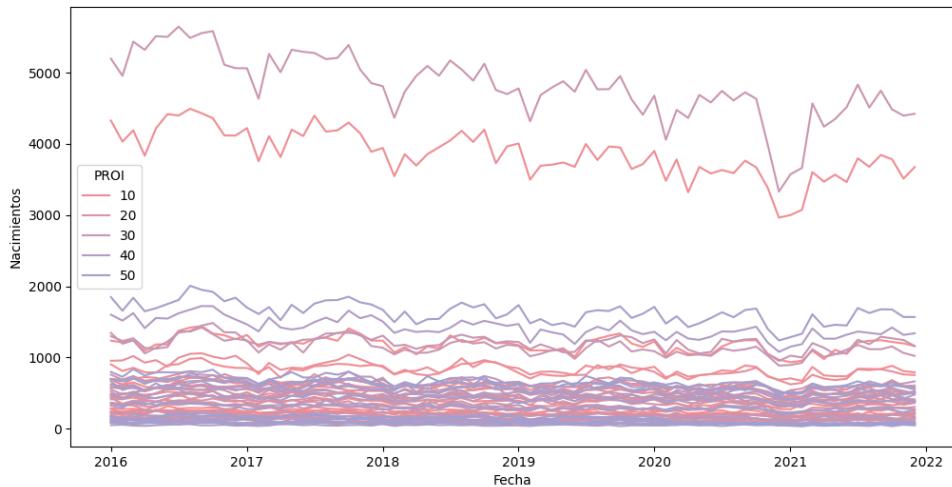


Figure 36: Evolución temporal de nacimientos por provincia

Es evidente que las distinciones por sexo o provincia no ejercen una influencia significativa en las características de la serie temporal. En todos los casos, la evolución muestra una forma y tendencia similar a lo largo del tiempo, indicando que la separación no modifica la estructura temporal de la serie. La única variación observable radica en la cantidad total de nacimientos.

En resumen, la **distinción entre sexos o provincias no altera la forma de la onda temporal**. Esto sugiere que el estudio temporal puede llevarse a cabo prescindiendo de esta división, ya que no aporta cambios sustanciales en la dinámica general de la serie.

4.2 Análisis de la serie temporal

En este apartado, analizaremos lo que revelan las gráficas de series temporales, destacando tendencias, patrones y eventos a lo largo del tiempo.

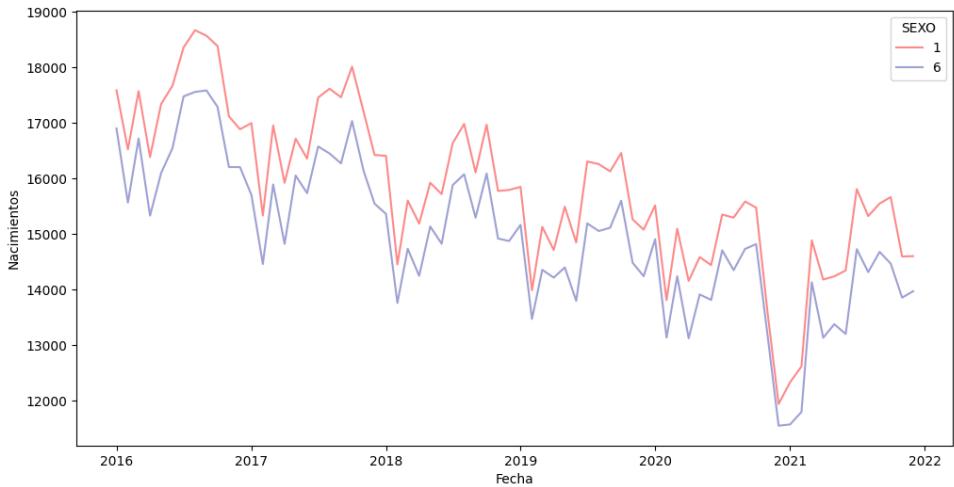


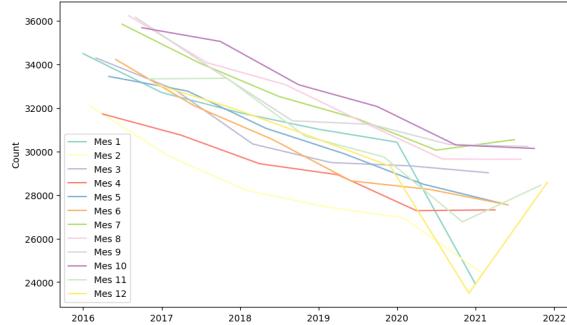
Figure 37: Evolución temporal de nacimientos por sexo



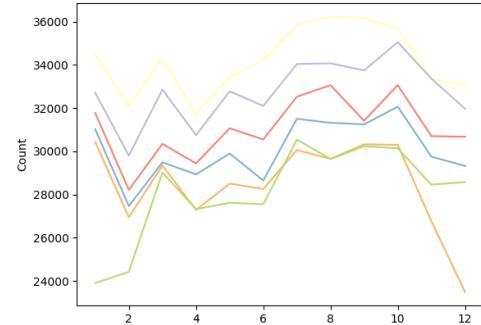
Figure 38: Nacimientos en el tiempo

Notamos **indicios de estacionalidad** en los datos, lo cual significa que hay patrones que se repiten regularmente en la serie temporal. Es interesante destacar que, al marcar el inicio de cada año, se observa un **aumento considerable en los nacimientos alrededor de la mitad del año**, especialmente en los últimos meses de verano. Además, se aprecia una **tendencia descreciente** en la serie temporal.

El número de nacimientos aumenta a lo largo de los meses en un año, pero disminuye en general al analizar varios años. Se destaca un pico en los nacimientos hacia los últimos meses de verano. Esto confirma lo observado en la gráfica general de la serie temporal.



(a) Evolución por mes



(b) Evolución por año

Descomposición: estacionalidad y tendencia

Una serie temporal puede poseer ciertas características que dificultan su comprensión y predicción. Por ello, es importante poder descomponer la serie temporal en sus "componentes atómicos". Una descomposición habitual consiste en escribir la serie temporal como:

$$X(t) = T(t) + S(t) + R(t).$$

Aquí $T(t)$ es la llamada **tendencia**, la evolución natural de la serie temporal independientemente de la estacionalidad, $S(t)$ es el **componente estacional** debido a los cambios periódicos y $R(t)$ es el **componente remanente**.

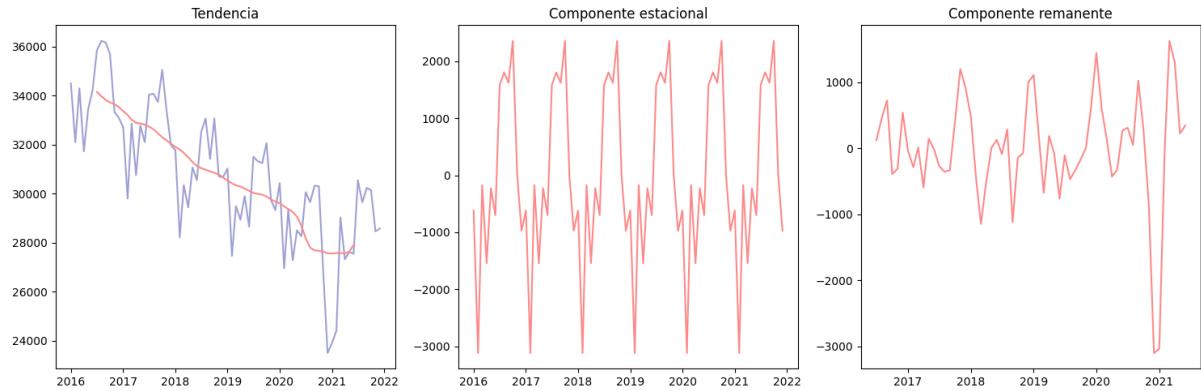


Figure 40: Descomposición

Con este análisis de la descomposición podemos confirmar la tendencia decreciente y la estacionalidad de esta serie temporal.

Autocorrelación: estacionariedad

Una serie temporal estacionaria tiene propiedades estadísticas constantes en el tiempo, lo que facilita el modelado y análisis.

La **prueba ADF** (Augmented Dickey-Fuller) se emplea para evaluar si una serie temporal tiene una raíz unitaria, indicando no estacionariedad. El p-valor de la prueba es crucial para decidir si rechazar la hipótesis nula de raíz unitaria. En este caso, el **p-valor de 0.58** no permite

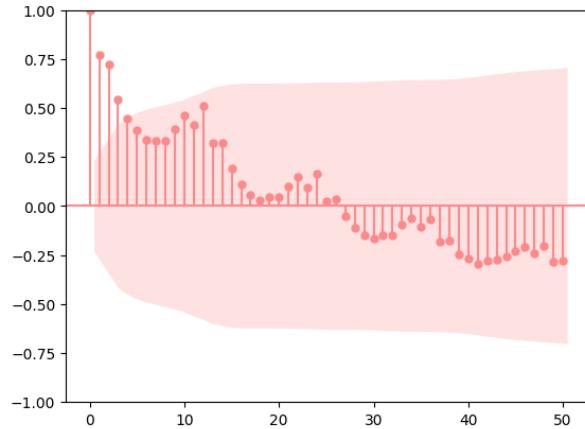


Figure 41: Autocorrelación

rechazar la hipótesis nula, sugiriendo no estacionariedad.

Esto se puede confirmar con una segunda prueba, conocida como **KPSS** (Kwiatkowski-Phillips-Schmidt-Shin), que con un **p-valor resultante inferior a 0.01** indica no estacionariedad. Este hallazgo sugiere que **abordar la no estacionariedad podría mejorar los resultados**, y se explorará esta posibilidad en modelos subsiguientes.

4.3 Predicción del año 2021

Antes de entrenar nuestros modelos, es fundamental dividir los datos en conjuntos de entrenamiento y prueba. Además, estableceremos una semilla para garantizar la reproducibilidad, facilitar la depuración efectiva del código y asegurar una comparación confiable entre los modelos.

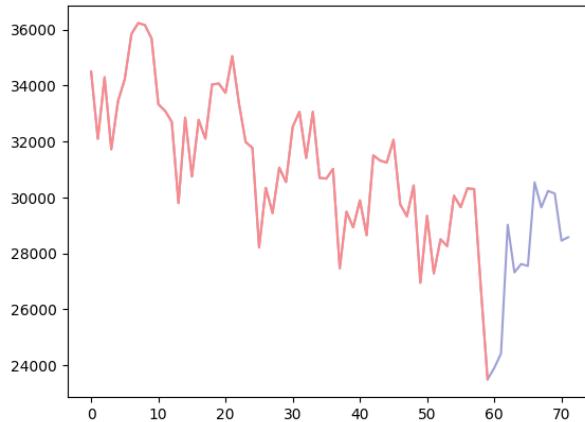


Figure 42: División de los datos

Entrenamiento de modelos y evaluación

Naïve Forecasting

El Naïve forecasting, es una técnica de pronóstico simple pero eficaz que se utiliza en el análisis

de series temporales. Su enfoque básico reside en el supuesto de que **el futuro se comportará de manera similar al pasado inmediato**. En otras palabras, se espera que el **siguiente valor** de la serie temporal sea el **mismo** que el **último valor observado**.

Los pasos básicos para una predicción simple son simples. Se toma el último valor conocido de la serie temporal y se utiliza como pronóstico para el siguiente período. Este método no requiere cálculos adicionales ni ajustes de parámetros y es particularmente útil cuando no hay información adicional disponible sobre tendencias, patrones estacionales o factores externos que puedan afectar la serie temporal.

Sin embargo, este enfoque tiene muchos otros defectos:

- **no se utiliza ningún conocimiento sobre los valores anteriores de la serie temporal**, sólo el último valor, ignorando mucha información.
- este método **sólo permite hacer previsiones a muy corto plazo**: sólo se puede predecir el siguiente valor de la serie temporal. Para el resto de valores futuros no se conoce el valor inmediatamente anterior, por lo que debe utilizarse la predicción y, por tanto, se devuelve una predicción constante.

Podemos aplicar este modelo intentando predecir todos los datos del 2021

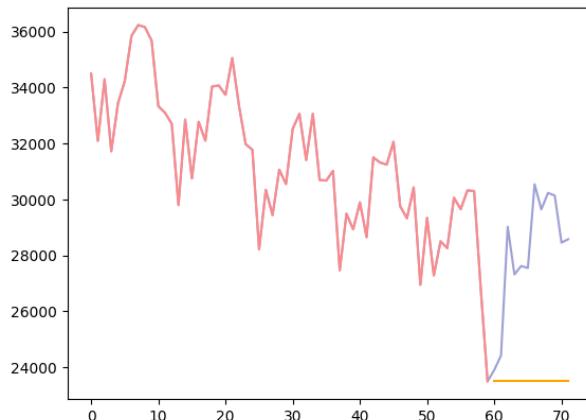


Figure 43: Naïve

Mediante esta técnica obtenemos una **predicciones constantes** para todo los valores de 2021 ya que solo tiene en cuenta el **último valor de 2020**. Una forma de mejorar esto sería prediciendo únicamente el último valor (también conocido como roll-up), de esta forma, solo se tiene en cuenta el anterior valor.

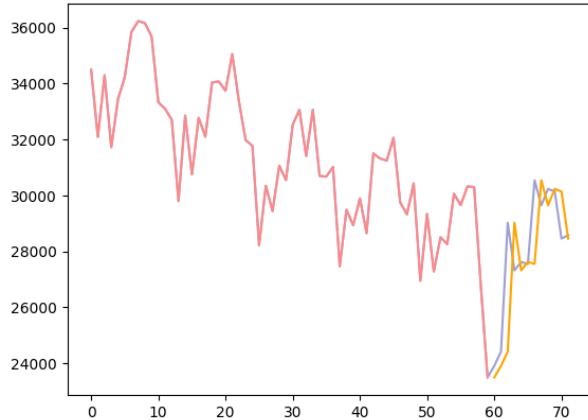


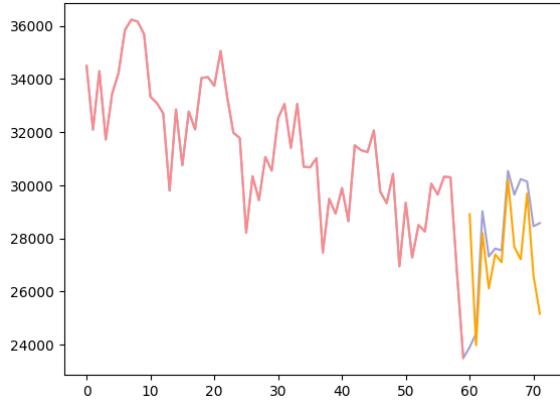
Figure 44: Naïve con roll-up

Regresión lineal ignorando las componentes estacionales

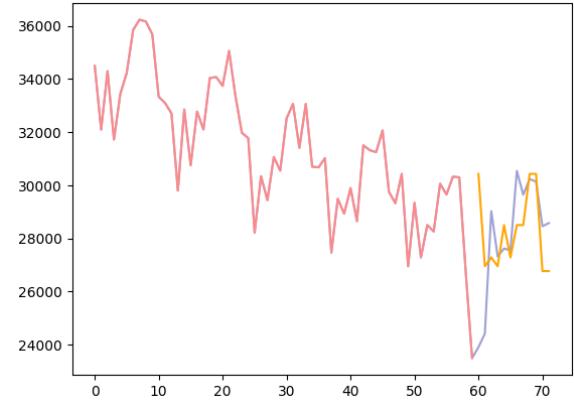
Otro enfoque simple para abordar el problema de la predicción en series temporales consiste en **olvidar por completo el componente temporal** de la serie y tratarlo como un simple problema de regresión.

Para ello, fijamos una ventana con $W > 0$ y creamos el conjunto de datos para el entrenamiento recogiendo los W datos anteriores.

Para llevar a cabo esta técnica utilizaremos dos modelos: una **regresión lineal tradicional** y un **árbol**.



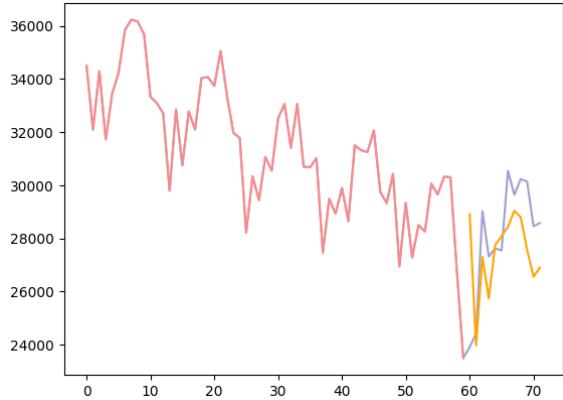
(a) Linear Regression



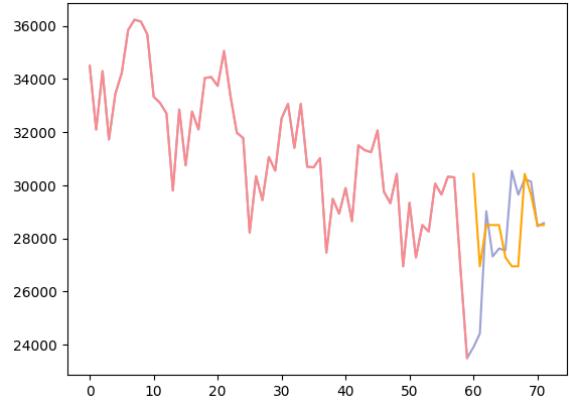
(b) Tree Regressor

¿Mejorarían los resultados si predecimos sólo el último valor, es decir, con roll-up?

Obtenemos unos errores medios absolutos de 1599, 1623.16, 1643.85 y 1583.6 respectivamente. En el caso de la regresión lineal al utilizar roll-up aumenta ligeramente el error y ocurre la situación inversa con el tree regressor. No obstante, la diferencia con ambos métodos es casi despreciable.



(a) Linear Regression con roll-up



(b) Tree Regressor con roll-up

Suavizado Exponencial

La idea de este método es aprovechar la idea de seguir la tendencia. Sin embargo, ahora, a diferencia del enfoque ingenuo, no sólo nos fijaremos en el último valor conocido de la serie temporal, sino también en algunos de los **valores anteriores**, calculando una **media ponderada**. Suponemos que queremos predecir $X(t)$ para un $t > 0$ y que los valores $X(t')$ para $t' < t$ son conocidos

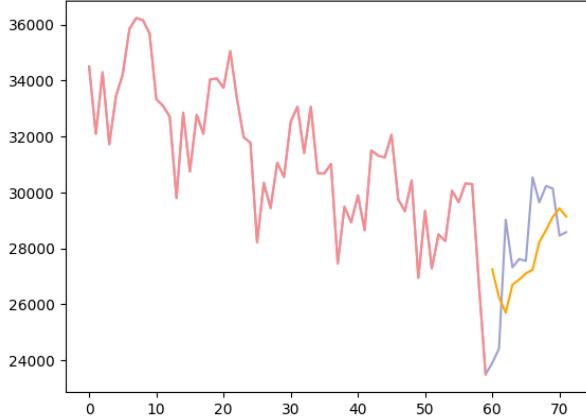


Figure 47: Suavizado Exponencial

A continuación representaremos en una gráfica las distintas técnicas utilizadas hasta el momento y los errores alcanzados con cada una de ellas.

El que **peor** funciona es el **Naïve** y el que **mejor** el **Naïve con roll-up**. El resto consiguen unos errores bastante similares.

Modelos ARMA

1. La idea de un **modelo AR** es suponer que el siguiente valor de la serie temporal se calcula como una **suma ponderada** de los **p** **valores anteriores** de la serie más un **término**

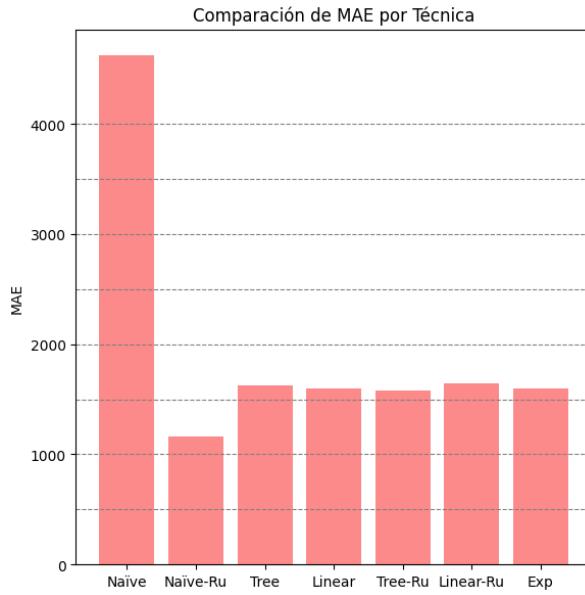


Figure 48: Errores alcanzados con cada una de las técnicas

normal aleatorio (ruido blanco). Este modelo incluye parte de los modelos previos, del suavizado exponencial o la regresión lineal. De este modo, el supuesto de AR es que la serie temporal puede predecirse como

$$\hat{X}(t) = \phi_1 X(t-1) + \phi_2 X(t-2) + \dots + \phi_p X(t-p) + \epsilon(t),$$

donde $\phi_i \in \mathbb{R}$ son constantes y $\epsilon(t)$ es una variable aleatoria con distribución normal con media 0 (y desviación típica fija).

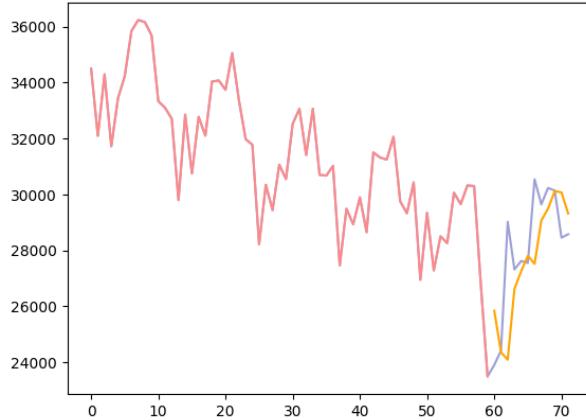
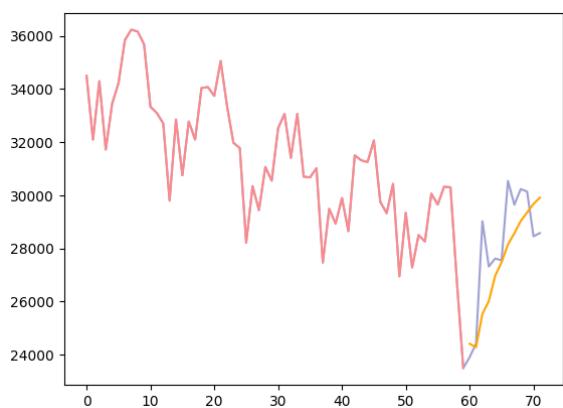
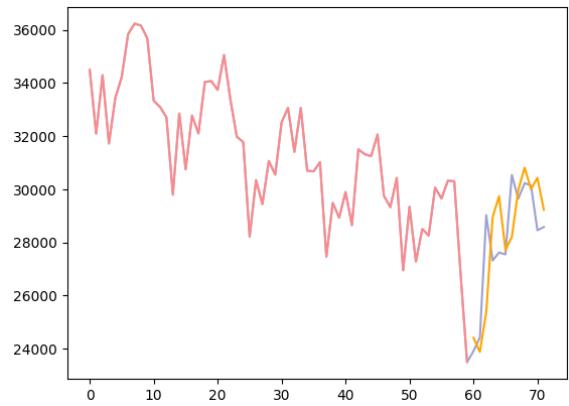


Figure 49: Modelo AR

Sin embargo, Los **coeficientes óptimos** ϕ_i pueden **estimarse a partir de los datos actuales** para ajustar lo mejor posible la serie. Los enfoques típicos plantean el problema como un problema de mínimos cuadrados o el cálculo de un estimador de máxima verosimilitud(MLE). Además, ¿se podrían mejorar los resultados con el **roll-up**?



(a) Modelo AR con ajuste automático de parámetros



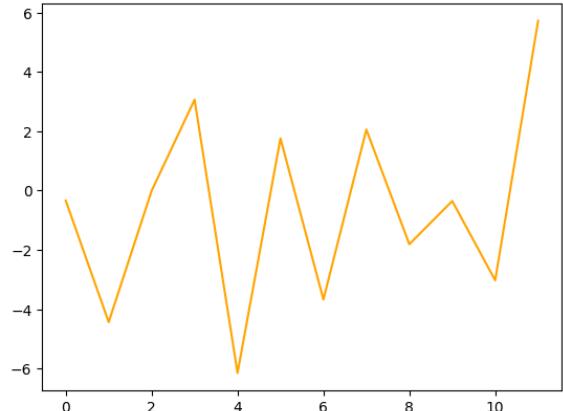
(b) Modelo AR con ajuste automático de parámetros y roll-up

Si nos fijamos en los errores (1246.8, 1179.7 y 1217.14), funcionan bastante mejor que cualquiera de los que hemos utilizado antes. Sin embargo, las tres diferentes técnicas utilizadas con el mmodelo AR no parecen diferir.

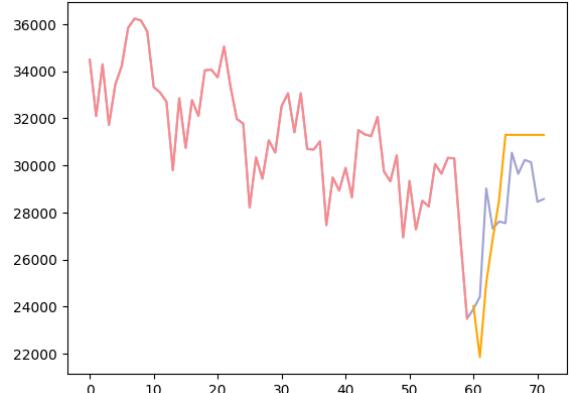
2. A diferencia de los modelos AR, los **modelos MA** suponen que cada valor de la serie temporal no depende del valor anterior, sino que es una **combinación aleatoria de varios ruidos blancos**. Sin embargo, los ruidos blancos están anidados de forma que sus valores resuenan en los siguientes valores de la serie. Escrito explícitamente, un modelo MA tiene la forma

$$\hat{X}(t) = \theta_1\epsilon(t-1) + \theta_2\epsilon(t-2) + \dots + \theta_q\epsilon(t-q),$$

donde $\theta_i \in \mathbb{R}$ son constantes fijas y $\epsilon(t')$ son variables normales independientes con media 0 y desviación típica fija (ruidos blancos). Al igual que con los modelos AR, se puede hacer un ajuste automático de los parámetros



(a) Modelo MA



(b) Modelo MA con ajuste automático de parámetros

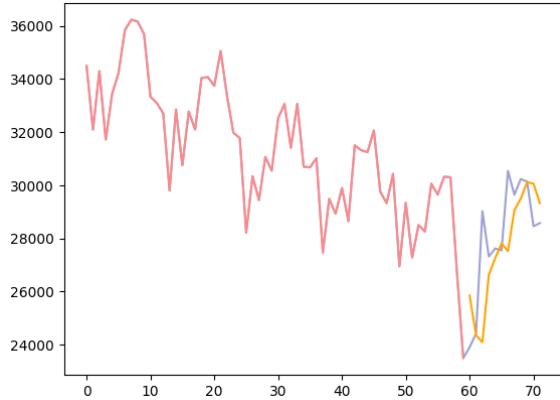
3. Modelos ARMA. En este tipo de modelos, mezclamos lo mejor de ambos mundos AR y

MA y consideramos un modelo combinado de la forma

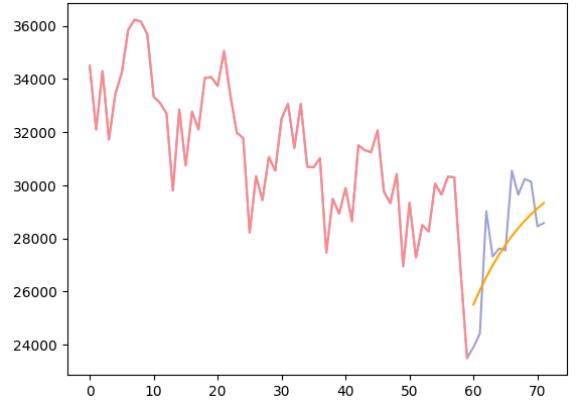
$$\hat{X}(t) = \underbrace{\mu}_{\text{Mean}} + \underbrace{\phi_1 X(t-1) + \phi_2 X(t-2) + \dots + \phi_p X(t-p)}_{AR} + \underbrace{\epsilon(t) + \theta_1 \epsilon(t-1) + \theta_2 \epsilon(t-2) + \dots + \theta_q \epsilon(t-q)}_{MA}.$$

Aquí $\phi_i, \theta_i, \mu \in \mathbb{R}$ son constantes y $\epsilon(h)$ son ruidos blancos (de media 0 y desviación típica fija). Obsérvese que el modelo también incluye una constante μ que le permite ajustarse a un "nivel subyacente" general.

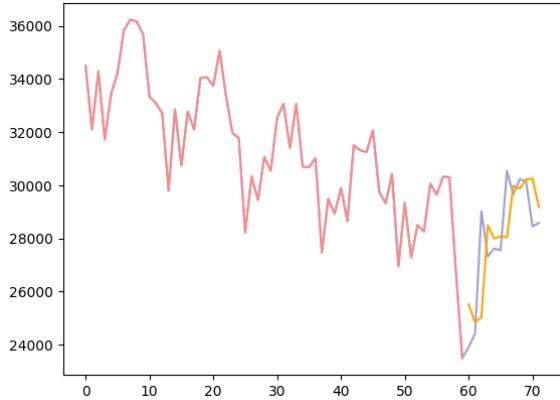
Al igual que en los dos casos anteriores, entrenaremos el modelo base, con ajuste de parámetros y con roll-up.



(a) Modelo ARMA



(b) Modelo ARMA con ajuste automático de parámetros



(c) Modelo ARMA con ajuste automático de parámetros
y roll-up

De nuevo, representaremos en una gráfica las distintas técnicas utilizadas hasta el momento y los errores alcanzados con cada una de ellas.

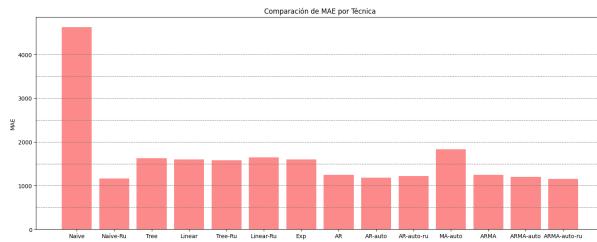


Figure 53: Errores alcanzados con cada una de las técnicas

Vemos que el que menor error consigue sigue siendo el **Naïve con roll up**. También se puede concluir que los modelos ARMA funcionan en su generalidad mejor que los primeros que hemos entrenado, a excepción del modelo MA con ajuste automático de parámetros que funciona bastante peor. Dentro de los modelos ARMA, los que a priori funcionan mejor son: **AR con ajuste automático y ARMA con ajuste automático y roll-up**.

Modelos ARIMA

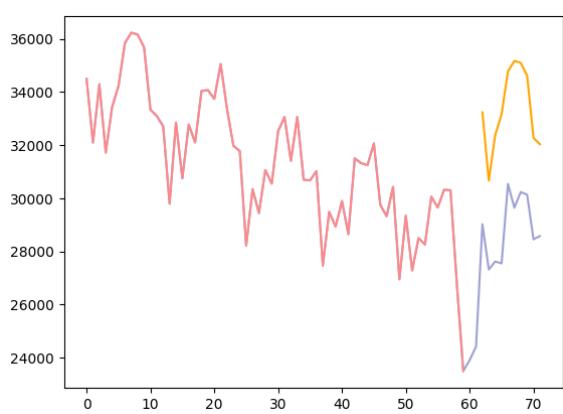
Un modelo ARIMA (AutoRegressive Integrated Moving Average) no es más que un **modelo ARMA** en el que la serie temporal se **diferencia** d veces para **alcanzar la estacionariedad**, $D^d X$. Por lo tanto, la forma general de un modelo ARIMA es

$$\widehat{D^d X}(t) = \mu + \phi_1 D^d X(t-1) + \phi_2 D^d X(t-2) + \dots + \phi_p D^d X(t-p) + \epsilon(t) + \theta_1 \epsilon(t-1) + \theta_2 \epsilon(t-2) + \dots + \theta_q \epsilon(t-q).$$

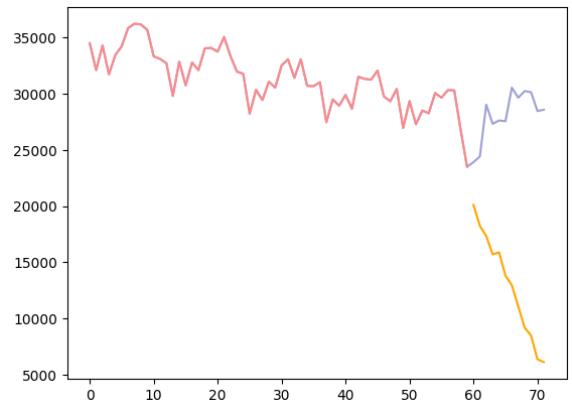
Aquí $\phi_i, \theta_i, \mu \in \mathbb{R}$ son constantes y $\epsilon(h)$ son ruidos blancos (de media 0 y desviación típica fija). Los parámetros p (longitud del modelo AR), q (longitud del modelo MA) y d (grado de diferenciación) se denominan parámetros del modelo ARIMA.

Como en nuestro análisis hemos visto la posibilidad de que nuestro conjunto de datos sea no estacionario, no descartamos probar el funcionamiento de estos modelos. Desgraciadamente, no hemos obtenido buenos resultados con ninguna de las 3 formas

Si usamos la función `auto_arima()` de la librería `pmdarima` podemos llevar a cabo una búsqueda automática del mejor modelo ARIMA para nuestro conjunto de datos. Esta función realiza una búsqueda exhaustiva de modelos ARIMA y selecciona el mejor modelo basándose en criterios como el AIC (Akaike Information Criterion) y el BIC (Bayesian Information Criterion).



(a) Modelo ARIMA



(b) Modelo ARIMA con ajuste automático de parámetros y roll-up

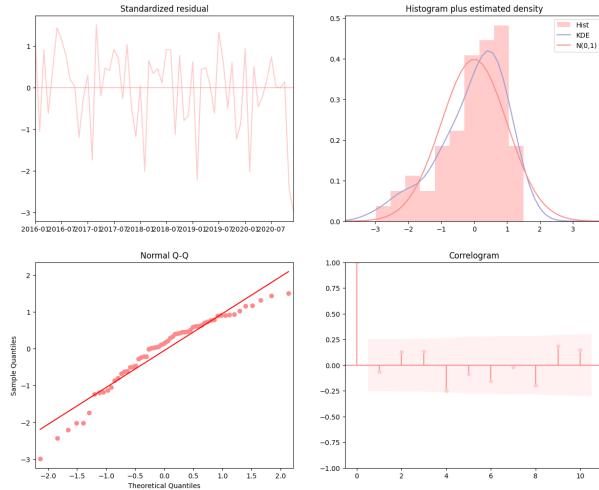


Figure 55: Diagnóstico del mejor modelo ARIMA

Cuatro gráficos resultan de la función `plot_diagnostics`. Los **residuos estandarizados**, el **histograma más la estimación KDE**, la **gráfica de cuantiles normales (Q-Q)** y un **correlograma**.

- **Standardized residuals:** No hay patrones evidentes en los residuos, con valores que tienen una media de cero y una varianza uniforme. En nuestro caso los valores giran en torno a cero pero son algo inconsistentes.
- **Histogram plus estimated density:** La curva KDE debería ser muy similar a la distribución normal (etiquetada como $N(0,1)$ en la gráfica). Podemos ver que sí se parecen ambas curvas.
- **Normal Q-Q:** La mayoría de los puntos de datos deberían ubicarse en la línea recta. Nosotros tenemos la mayoría de los puntos cercanos a la recta, pero no están completamente alineados.
- **Correlograma (gráfica ACF):** El 95% de las correlaciones para un rezago mayor que cero no deberían ser significativas. El área gris es la banda de confianza, y si los valores caen fuera de esta, son estadísticamente significativos. En esta ocasión, todos los valores se encuentran dentro de la región sombreada, es decir, no son significativos.

Modelo SARIMA

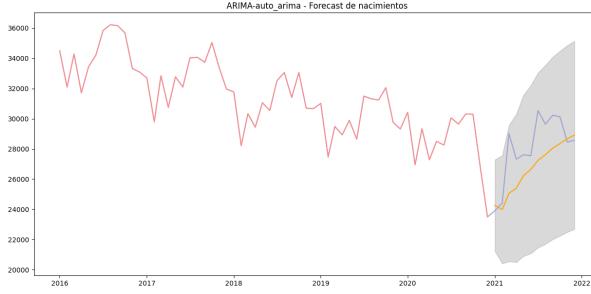


Figure 56: Resultados con el mejor ARIMA

En el entrenamiento anterior no se ha tenido en cuenta la estacionalidad, lo que haremos será utilizar un modelo **SARIMA** que incluye la **descomposición estacional**.

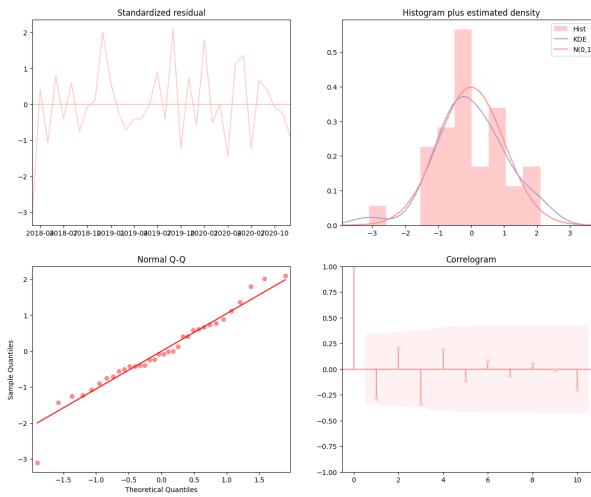


Figure 57: Diagnóstico del mejor modelo SARIMA

- **Standardized Residual:** Los valores fluctúan más que antes, el intervalo de valores es más grande.
- **Histogram plus KDE estimate:** La curva no ha cambiado demasiado con respecto a ARIMA.
- **Normal Q-Q:** ahora los puntos están mucho más alineados con la recta.
- **Correlogram (ACF plot)** Todos los valores se encuentran en la región sombreada.

Los resultados son bastante malos, simplemente visualmente se ve que los errores son inmensos.

Modelo SARIMAX

El modelo **SARIMAX**, que representa "Seasonal Autoregressive Integrated Moving Average with Exogenous Factors", es una extensión del modelo ARIMA que incorpora tanto **componentes estacionales** como **variables exógenas** para mejorar la capacidad de predicción en series temporales.

Para entrenar SARIMAX príremos de nuestros 2 mejores modelos que se hayan conseguido con ARIMA : AR y ARMA con roll-up. Por tanto, probaremos con un order de (1,0,1) o (3,0,0).

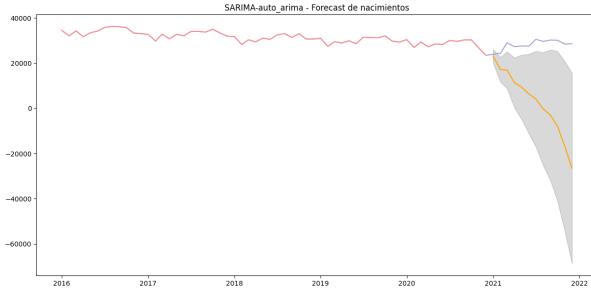


Figure 58: Resultados del mejor modelo SARIMA

Primero , haremos una búsqueda manual de los mejores parámetros tanto estacionales como no estacionales. Los resultados que obtenemos son los siguientes:

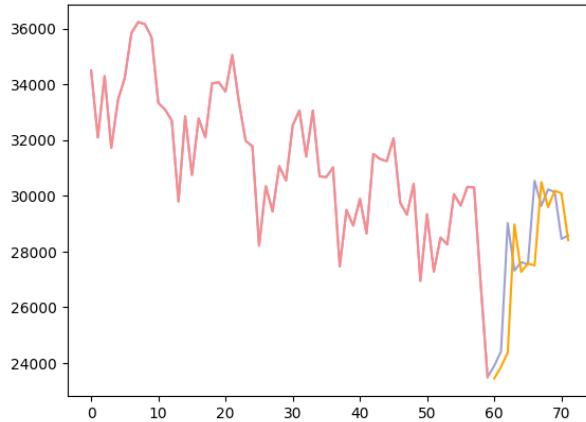


Figure 59: Resultados con SARIMAX

Ahora probaremos con la librería `pmdarima`.

- **Standarized Residual:** Los valores fluctúan más que antes, el intervalo de valores es más grande.
- **Histogram plus KDE estimate:** La curva no ha cambiado demasiado con respecto a ARIMA.
- **Normal Q-Q:** ahora los puntos están mucho más alineados con la recta.
- **Correlogram (ACF plot)** Todos los valores se encuentran en la región sombreada.

Al igual que con SARIMA, los resultados son catastróficos.

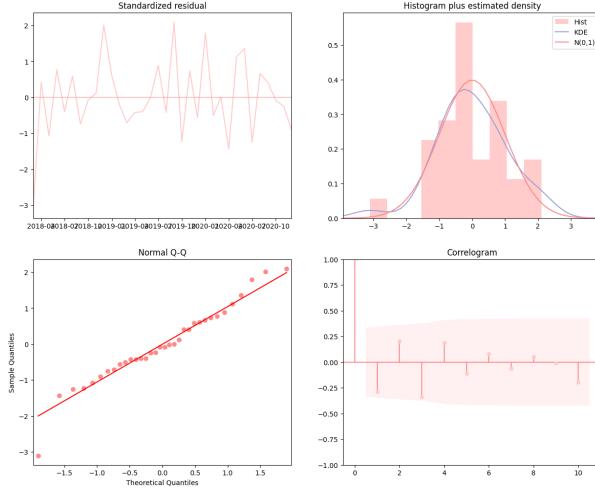


Figure 60: Diagnóstico del mejor modelo SARIMAX

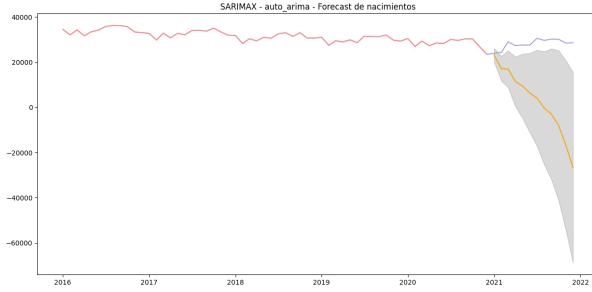


Figure 61: Resultados del mejor modelo SARIMAX

LSTM

Por último, hemos querido entrenar una LSTM (Long Short-Term Memory) que diferencia de las redes neuronales convencionales, las **LSTM** poseen unidades especiales de memoria que les permiten **retener** y acceder a **información relevante a lo largo de secuencias extensas**. Este mecanismo de memoria a largo plazo se traduce en una capacidad única para capturar patrones temporales complejos y aprender dependencias a distintas escalas temporales.

Resultados y conclusiones

Tras el proceso de entrenamiento de todos los modelos, es el momento de sacar conclusiones.

En términos generales, la eficacia de los modelos es **bastante baja**. Esto se debe a que a finales de 2020 y principios de 2021 ocurre un **cambio en el patrón**. Hay un **descenso anómalo** repentino de los nacimientos mucho mayor que en los años anteriores. Esto afecta tanto a las predicciones de prueba de 2021, debido a que el modelo no ha conseguido aprender lo suficiente el cambio de patrón.

A pesar de esto, los que mejores resultados han obtenido son: **Naïve con roll-up**, **ARMA con ajuste automático de parámetros** y **SARIMAX con roll-up**.

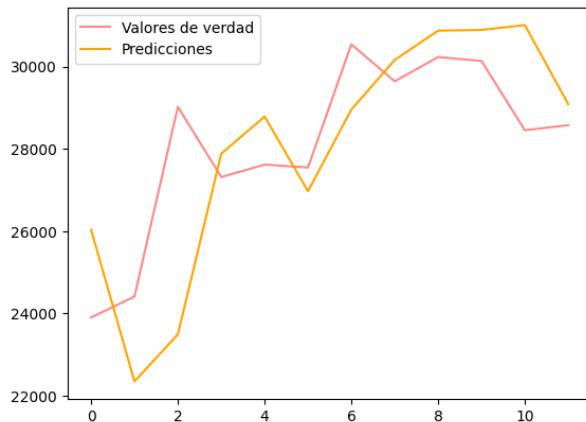


Figure 62: Resultados con LSTM

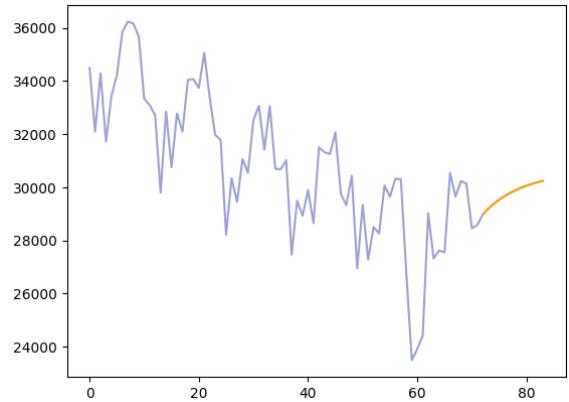


Figure 63: Resultado final

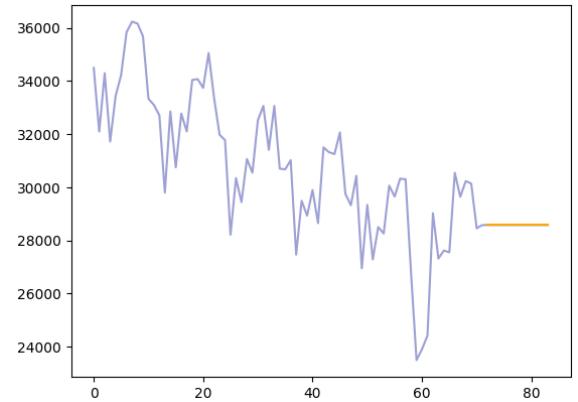
4.4 Predicción a futuro: año 2022

Para la predicción a futuro del número de nacimientos en 2022 utilizaremos las 3 técnicas más efectivas anteriormente.

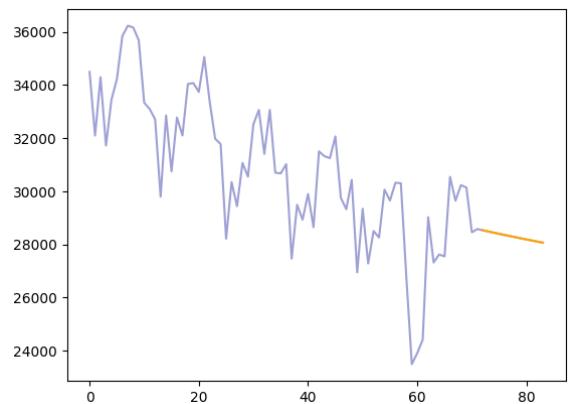
En este caso, el Naïve con roll-up nos da lo mismo ya que al tratar de acceder al valor anterior este siempre se va a corresponder con el último valor del 2021, ya que habíamos inicializado esas predicciones a cero.



(a) ARMA-2022



(b) Naïve-2022



(c) SARIMAX-2022