INTELIGENCIA ARTIFICIAL

PROYECTO FINAL

Francisco Javier Romera Hidalgo

New Technology School

Tokio.

New
Technology
School

Francisco Javier Romera Hidalgo

RESPONSABLE DEL DESARROLLO DE LA APLICACIÓN

Fecha: 26, 11, 2022

INTELIGENCIA ARTIFICIAL
PROYECTO FINAL
Francisco Javier Romera Hidalgo

Tokio.

DESARROLLO

1. Define el problema y sus objetivos.
2. Define que datos vas a obtener y cárgalos.
3. Realiza la limpieza de los datos oportuna explicando el porqué de cada acción.
4. Explora los datos y comparte tus observaciones.
5. Escoge un modelo y justifícalo.
6. Realiza una representación final de los resultados que obtengas.

CONCLUSIONES

Debes explicar claramente si tu proyecto trata sobre la minería de datos, machine learning, deep learning, etc. Y deberás redactar unas conclusiones donde se indique si se ha llegado a obtener algún resultado que satisfaga alguna de las preguntas iniciales.

Definición del problema y objetivos

Este proyecto de **Machine Learning** trata sobre la predicción del precio de coches, basado en un dataset previamente recopilado.

Una compañía de automoción china, Geely Auto, aspira a ingresar al mercado estadounidense estableciendo su unidad de fabricación allí y produciendo automóviles localmente para dar competencia a sus contrapartes estadounidenses y europeas.

Han contratado a una empresa de consultoría para comprender los factores de los que depende el precio de los automóviles. Específicamente, quieren comprender los factores que afectan el precio de los automóviles en el mercado estadounidense.

Sobre la base de varias encuestas de mercado, la firma consultora ha reunido un gran conjunto de datos de diferentes tipos de automóviles en todo el mercado estadounidense.

Objetivo de negocio

Estamos obligados a modelar el precio de los coches con las variables independientes disponibles. Será utilizado por la gerencia para comprender cómo varían exactamente los precios con las variables independientes. En consecuencia, pueden manipular el diseño de los automóviles, la estrategia comercial, etc. para cumplir con ciertos niveles de precios. Además, el modelo será una buena manera para que la administración comprenda la dinámica de precios de un nuevo mercado.

Defino los datos que voy a cargar

Los datos han sido recopilados de la plataforma Kaggle por una empresa externa. Importo las librerías necesarias para cargar los datos, analizarlos (Numpy y Pandas) y visualizarlos (Seaborn y Matplotlib). También importo la librería warnings para evitar que me visualice algunos errores sin importancia.

Por último, importo las librerías de Machine Learning, principalmente SKLearn (PCA, train_test_split, etc.).

```
:
# Análisis de datos
import numpy as np
import pandas as pd

# Visualización
import seaborn as sns
from matplotlib import pyplot as plt
import warnings
pd.set_option('display.max_columns',None)
warnings.filterwarnings('ignore')
%matplotlib inline

# Machine Learning
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.decomposition import PCA
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,LabelEncoder,OneHotEncoder

from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.linear_model import HuberRegressor
from sklearn.ensemble import RandomForestRegressor
```

Cargo los datos

Una vez tengo las librerías instaladas **cargo los datos** con pandas:

df = pd.read_csv("./CarPrice_Assignment.csv")

Analizo los datos

Analizo los datos con los métodos *.info()* y *.describe()*

```
# Aquí veo que tengo 205 registros, no hay nulos y tengo varios tipos de datos.
df.describe(include='all').style.background_gradient(cmap='Blues').set_properties(**{'font-family':'Segoe UI'}) # Include="all" p
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheigh
count	205.000000	205.000000	205	205	205	205	205	205	205	205.000000	205.000000	205.000000	205.00000
unique	nan	nan	147	2	2	2	5	3	2	nan	nan	nan	na
top	nan	nan	toyota corona	gas	std	four	sedan	fwd	front	nan	nan	nan	na
freq	nan	nan	6	185	168	115	96	120	202	nan	nan	nan	na
mean	103.000000	0.834146	nan	nan	nan	nan	nan	nan	nan	98.756585	174.049268	65.907805	53.72487
std	59.322565	1.245307	nan	nan	nan	nan	nan	nan	nan	6.021776	12.337289	2.145204	2.44352
min	1.000000	-2.000000	nan	nan	nan	nan	nan	nan	nan	86.600000	141.100000	60.300000	47.80000
25%	52.000000	0.000000	nan	nan	nan	nan	nan	nan	nan	94.500000	166.300000	64.100000	52.00000
50%	103.000000	1.000000	nan	nan	nan	nan	nan	nan	nan	97.000000	173.200000	65.500000	54.10000
75%	154.000000	2.000000	nan	nan	nan	nan	nan	nan	nan	102.400000	183.100000	66.900000	55.50000
click to expand output; double click to hide output			nan	nan	nan	nan	nan	nan	nan	120.900000	208.100000	72.300000	59.80000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

En un vistazo rápido puedo observar que:

- Hay **2 opciones** para el **tipo de combustible** o el **número de puertas**.

- Hay **7 opciones** de **motor** y **cilindrada**.

- El **nombre** de **coche más usado** es **Toyota Corona**.

- **Predominan coches** de **gasolina**, **4 puertas**, con el **motor delante** y **4 cilindros**.

- El **precio medio** de un coche es **13276.71**.

- El **precio más bajo** de un coche es **5.118**.

- El **peso máximo en vacío** es **4.066**.

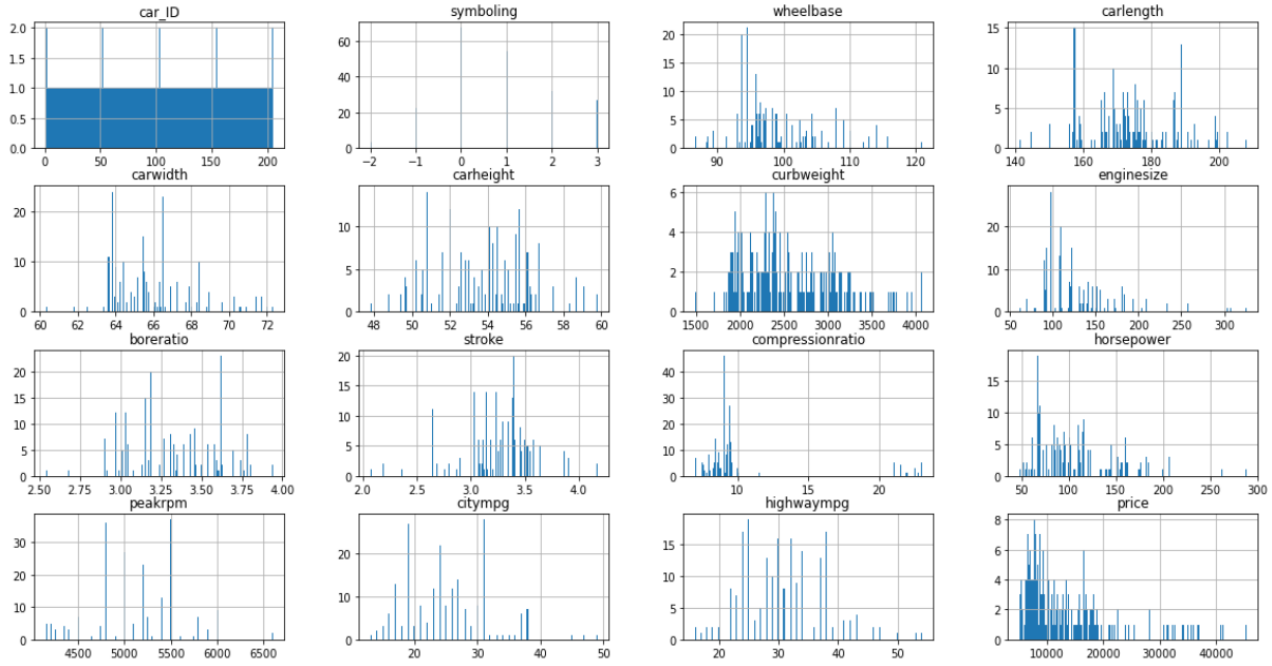
- Hay **205 registros**.

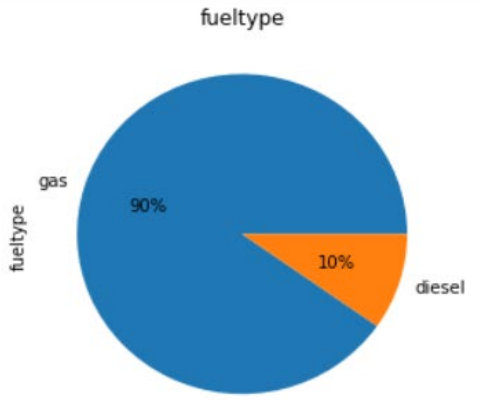
- **No** hay **nulos**.

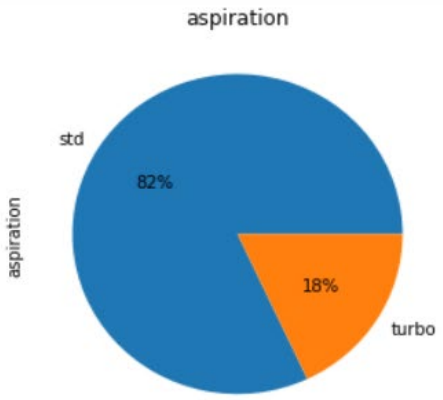
- Tiene **26 columnas**.

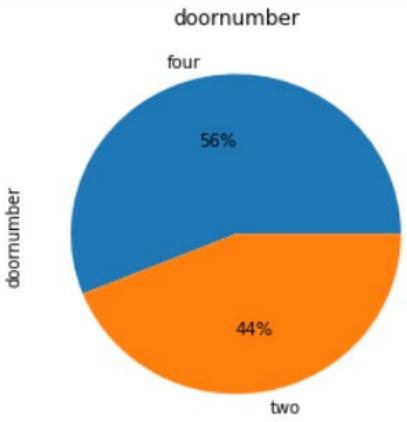
Visualizo los datos

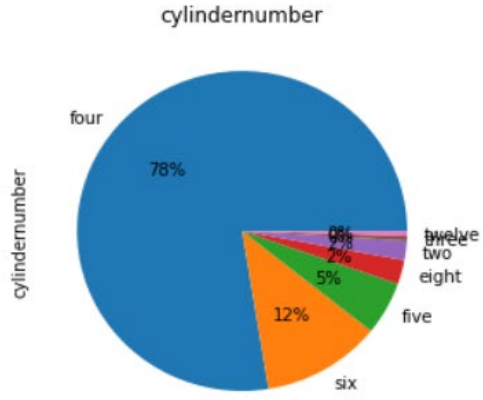
Comienzo por hacer un histograma de los datos del dataframe y un gráfico de pastel con porcentajes sobre las columnas.

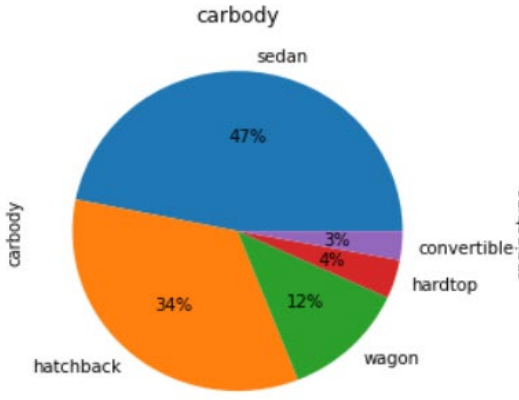


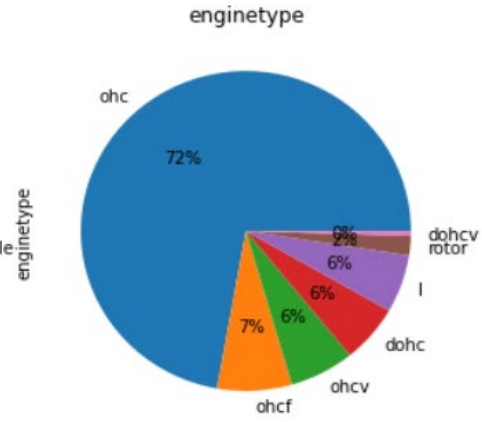

fueltype
fueltype
gas
diesel
90%
10%

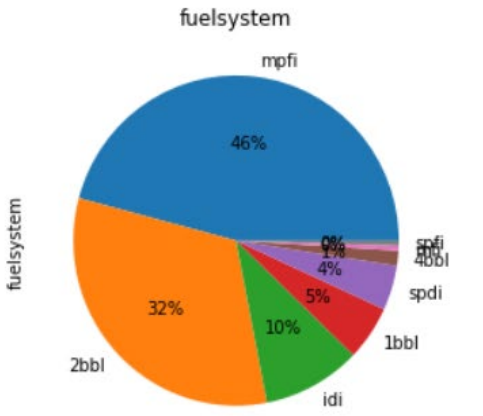

aspiration
aspiration
std
turbo
82%
18%

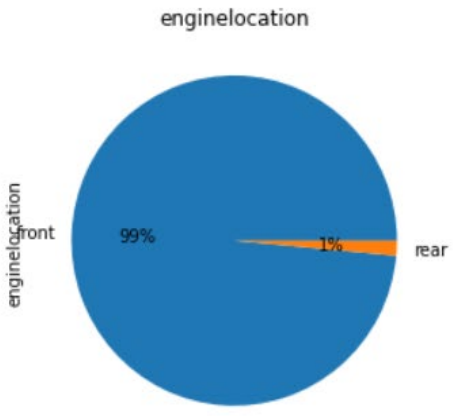

doornumber
doornumber
four
two
56%
44%

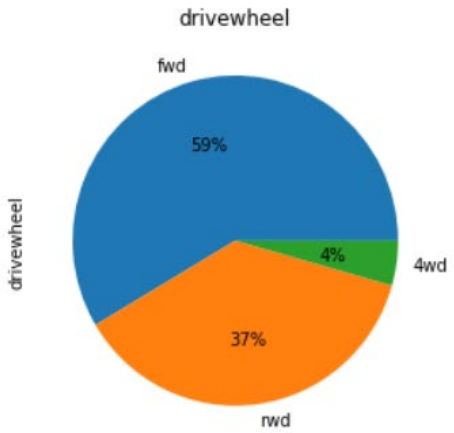

cylindernumber
cylindernumber
four
six
five
eight
two
78%
12%
5%


carbody
carbody
sedan
hatchback
wagon
hardtop
convertible
47%
34%
12%
4%
3%


enginetype
enginetype
ohc
ohcf
ohcv
dohc
l
rotor
dohcv
72%
7%
6%
6%
2%


fuelsystem
fuelsystem
mpfi
2bbl
idi
1bbl
spdi
46%
32%
10%
5%
4%


enginelocation
enginelocation
front
rear
99%
1%


drivewheel
drivewheel
fwd
rwd
4wd
59%
37%
4%

Una vez he echado un vistazo rápido a los datos, veo que atributos están relacionados directamente con el precio. Ésto lo hago mediante la siguiente línea:

df[['columna', 'price']].groupby(['columna'], as_index=False).mean().sort_values(by='price', ascending=False)

Quiere decir que de mi dataframe, coja una columna que quiera comparar y el precio, los agrupe, me haga la media y me los ordene de mayor a menos precio en tan sólo una línea de código.

Muestro un ejemplo:

```
df[['enginelocation', 'price']].groupby(['enginelocation'], as_index=False).mean().sort_values(by='price', ascending=False)
```

	enginelocation	price
1	rear	34528.000000
0	front	12961.097361

Limpieza de datos

En este caso, la limpieza se trata de buscar y eliminar nulos, buscar alguna característica irrelevante y eliminarla o transformar todos los datos a numérico si fuera posible.

CODIFICACIÓN DE DATOS

DE CATEGÓRICOS A ORDINALES

ANTES

	cylindernumber	enginesize	fuelsystem	boreratio	stroke	compressionratio
:	four	130	mpfi	3.47	2.68	9.0
:	four	130	mpfi	3.47	2.68	9.0
'	six	152	mpfi	2.68	3.47	9.0
:	four	109	mpfi	3.19	3.40	10.0
:	five	136	mpfi	3.19	3.40	8.0

DESPUÉS

```
df['cylindernumber'] = df['cylindernumber'].map({'two':2, 'three':3, 'four':4, 'five':5, 'six':6, 'eight':8, 'twelve':12})
df.head()
```

width	carheight	curbweight	enginetype	cylindernumber	enginesize	fuelsystem	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	hig
64.1	48.8	2548	dohc	4	130	mpfi	3.47	2.68	9.0	111	5000	21	
64.1	48.8	2548	dohc	4	130	mpfi	3.47	2.68	9.0	111	5000	21	
65.5	52.4	2823	ohcv	6	152	mpfi	2.68	3.47	9.0	154	5000	19	
66.2	54.3	2337	ohc	4	109	mpfi	3.19	3.40	10.0	102	5500	24	
66.4	54.3	2824	ohc	5	136	mpfi	3.19	3.40	8.0	115	5500	18	

Paso **datos categóricos** a **variables ficticias** con ***.get_dummies()***.

Lo que hago aquí es pasar mis columnas (*fueltype, aspiration, carbody, drivewheel, enginelocation, enginetype, fuelsystema)* a tipo **numérico**.

Por último agrupo los nombres de los coches en marcas y vuelvo a usar *.get_dummies()* para transformarlo a ordinal.

Así ya tendríamos nuestro dataframe transformado completamente.

	car_ID	symboling	doornumber	wheelbase	carlength	carwidth	carheight	curbweight	cylindernumber	enginesize	boreratio	stroke	compressionratio	h
0	1	3	2	88.6	168.8	64.1	48.8	2548	4	130	3.47	2.68	9.0	
1	2	3	2	88.6	168.8	64.1	48.8	2548	4	130	3.47	2.68	9.0	
2	3	1	2	94.5	171.2	65.5	52.4	2823	6	152	2.68	3.47	9.0	
3	4	2	4	99.8	176.6	66.2	54.3	2337	4	109	3.19	3.40	10.0	
4	5	2	4	99.4	176.6	66.4	54.3	2824	5	136	3.19	3.40	8.0	

horsepower	peakrpm	citympg	highwaympg	price	fueltype_diesel	fueltype_gas	aspiration_std	aspiration_turbo	carbody_convertible	carbody_hardtop
111	5000	21	27	13495.0	0	1	1	0	1	0
111	5000	21	27	16500.0	0	1	1	0	1	0
154	5000	19	26	16500.0	0	1	1	0	0	0
102	5500	24	30	13950.0	0	1	1	0	0	0
115	5500	18	22	17450.0	0	1	1	0	0	0

carbody_hatchback	carbody_sedan	carbody_wagon	drivewheel_4wd	drivewheel_fwd	drivewheel_rwd	enginelocation_front	enginelocation_rear	enginetype_dohc
0	0	0	0	0	1	1	0	1
0	0	0	0	0	1	1	0	1
1	0	0	0	0	1	1	0	0
0	1	0	0	1	0	1	0	0
0	1	0	1	0	0	1	0	0

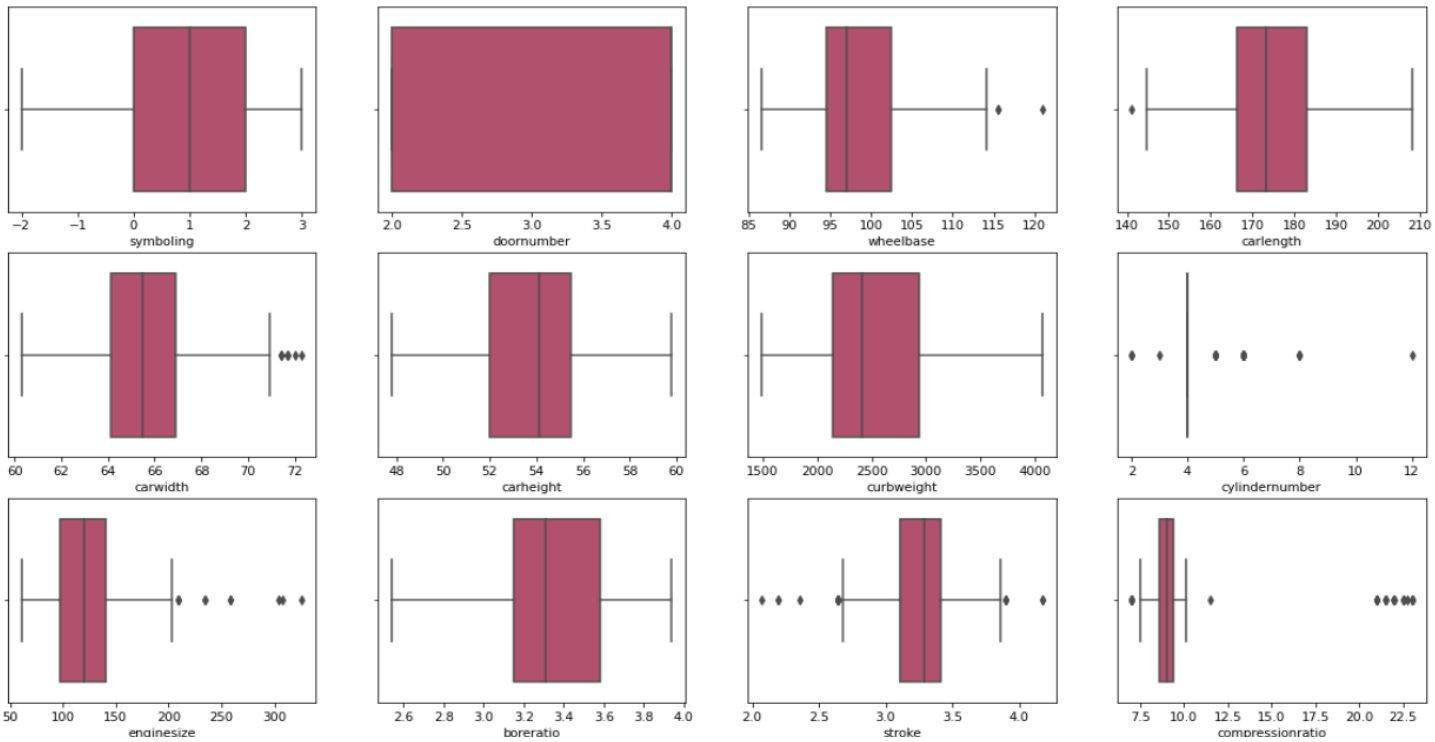
enginetype_ohcv	enginetype_rotor	fuelsystem_1bbl	fuelsystem_2bbl	fuelsystem_4bbl	fuelsystem_idi	fuelsystem_mfi	fuelsystem_mpfi	fuelsystem_spdi
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0

CarName_porsche	CarName_renault	CarName_saab	CarName_subaru	CarName_toyota	CarName_toyouta	CarName_vokswagen	CarName_volkswagen
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

CarName_Nissan	CarName_alfa-romero	CarName_audi	CarName_bmw	CarName_buick	CarName_chevrolet	CarName_dodge	CarName_honda	CarName_isuzu
0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0

Muestro un examen de la forma general de los datos graficados para características importantes, incluida la simetría y las desviaciones de los supuestos.

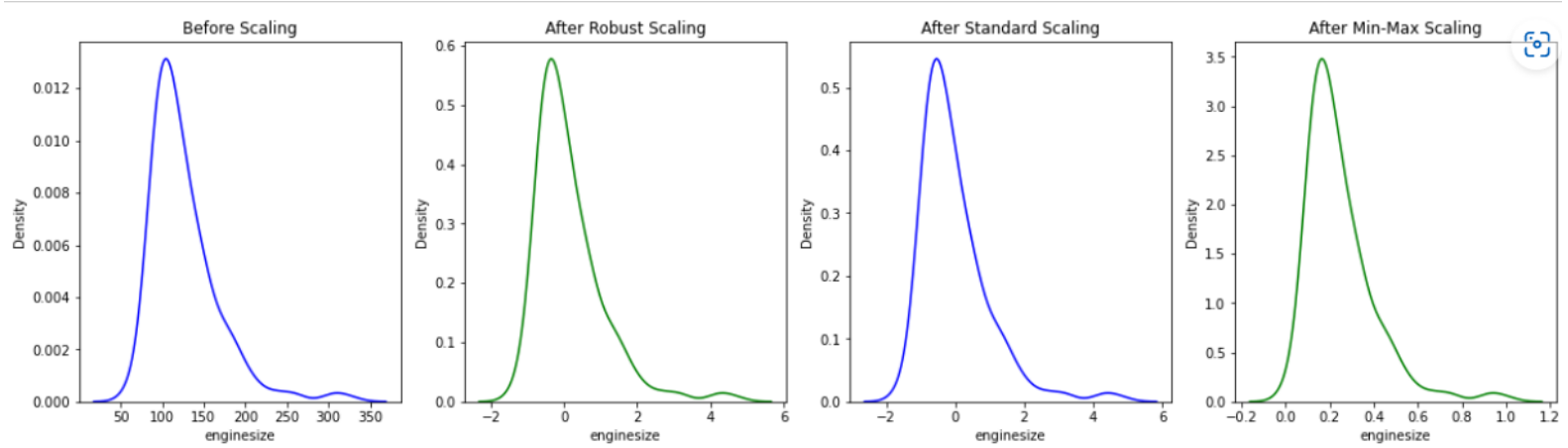
También muestro un examen de los datos en busca de observaciones inusuales que están muy alejadas de la masa de datos. Estos puntos a menudo se denominan valores atípicos. Hay dos técnicas gráficas para identificar valores atípicos, diagramas de dispersión y diagramas de caja, junto con un procedimiento analítico para detectar valores atípicos cuando la distribución es normal (prueba de Grubbs). Véase ejemplo:



No se puede descartar una característica como la relación de compresión, ya que indica la eficiencia de combustión de un motor, por lo que para esto, mientras modelo, usaré un regresor Ridge para verificar la comparación de rendimiento, ya que es inmune a la multicolinearidad.

StandardScaler sigue la Distribución Normal Estándar (SND). Por lo tanto, hace media = 0 y escala los datos a la varianza unitaria.

En presencia de valores atípicos, *StandardScaler* no garantiza escalas de características equilibradas, debido a la influencia de los valores atípicos al calcular la media empírica y la desviación estándar. Esto conduce a la contracción en el rango de los valores de la característica.



Explicación de los valores atípicos de la relación de compresión y por qué debemos mantenerlos

Cada motor tiene una relación de compresión específica. La mezcla de aire y combustible se comprime en el cilindro para crear un encendido, cuya fuerza depende de la relación de compresión: el volumen del cilindro cuando el pistón está en la parte inferior de su carrera frente al volumen del cilindro cuando el pistón está en la parte superior de su carrera. Por cierto, la cilindrada del motor se refiere a la capacidad total de todos los pistones durante un ciclo completo.

Las relaciones de compresión suelen oscilar entre 8:1 y 10:1. Una relación de compresión más alta, por ejemplo, de 12: 1 a 14: 1, significa una mayor eficiencia de combustión. Sin embargo, cuando se trata de motores diesel, la ausencia de bujías requiere una relación de compresión más alta, de aproximadamente 14: 1 a hasta 22: 1. Usan aire caliente para vaporizar y luego encender el combustible.

MODELADO

El proceso de modelado significa entrenar un algoritmo de aprendizaje automático para predecir las etiquetas de las características, ajustarlo para la necesidad comercial y validarlo en datos de espera. El resultado del modelado es un modelo entrenado que se puede usar para inferencia, haciendo predicciones en nuevos puntos de datos.

Análisis de componentes principales

El análisis de componentes principales, o PCA, es una técnica estadística para convertir datos de alta dimensión en datos de baja dimensión seleccionando las características más importantes que capturan la máxima información sobre el conjunto de datos.

Las características se seleccionan en función de la varianza que causan en la salida. La característica que causa la mayor varianza es el primer componente principal

La característica responsable de la segunda varianza más alta se considera el segundo componente principal, y así sucesivamente.

Ventajas de PCA

2 ventajas principales de la reducción de dimensionalidad:

- El tiempo de entrenamiento de los algoritmos se reduce significativamente con un menor número de características.
- No siempre es posible analizar datos en altas dimensiones. Por ejemplo, si hay 100 entidades en un conjunto de datos.

El número total de diagramas de dispersión necesarios para visualizar los datos sería 100(100-1)2 = 4950. Prácticamente no es posible analizar los datos de esta manera.

Aquí muestro como el dataframe pasa de 73 columnas a sólo 29.

```
pca = PCA(n_components=0.99)
x_reduced = pca.fit_transform(x_clean)
```

```
print("Número de características antes de aplicar PCA --> {} y después --> {}".format(x_clean.shape[1], x_reduced.shape[1]))
```

```
Número de características antes de aplicar PCA --> 73 y después --> 29
```

Ya tengo mi dataframe preparado para entrenar. Lo siguiente será separar los datos de entrenamiento y prueba con train_test_split(). También crearé 2 funciones, la primera para **evaluar los errores** de entrenamiento y de **R2 Score**, y la segunda para mostrar las predicciones de mis modelos en una gráfica.

```
X_train_clean, X_test_clean, y_train_clean, y_test_clean = train_test_split(x_clean, y_clean, test_size=0.2, random_state=42)
X_train_r, X_test_r, y_train_r, y_test_r = train_test_split(x_reduced, y_clean, test_size=0.2, random_state=42)
```

```
clean_evals = dict()
reduced_evals = dict()
def evaluate_regression(evals, model, name, X_train, X_test, y_train, y_test):
    train_error = mean_squared_error(y_train, model.predict(X_train), squared=False)
    test_error = mean_squared_error(y_test, model.predict(X_test), squared=False)
    r2_train = r2_score(y_train, model.predict(X_train))
    r2_test = r2_score(y_test, model.predict(X_test))
    evals[str(name)] = [train_error, test_error, r2_train, r2_test]
    print("El error de entrenamiento es: " + str(name) + "{} -> El error de prueba de ".format(train_error) + str(name) + " es: {
    print("El R2 score del entrenamiento de " + str(name) + " es: {} ".format(r2_train*100) + "y el de prueba es: {}".format(r2_t
```

```
def plot_predictions(model, X_test, y_test):
    y_pred = model.predict(X_test)
    df = pd.DataFrame({"Y_test": y_test , "Y_pred" : y_pred})
    plt.figure(figsize=(10,6))
    plt.plot(df[:20])
    plt.legend(['Actual' , 'Predijo'])
```

Ahora probaré algunos modelos que pueden ir bien como Regresión Linear o de Lazo y muestro la gráfica

Regresión Linear

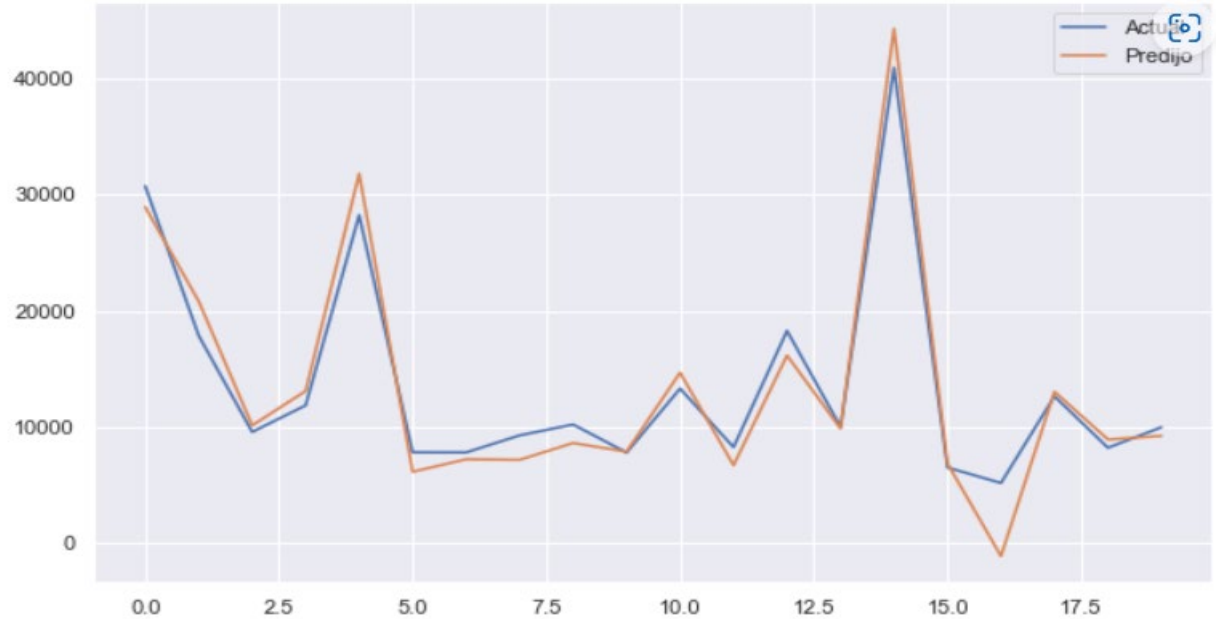
```
# Creo el modelo de Regresión Linear y uso la función para evaluarlo
lr = LinearRegression().fit(X_train_clean, y_train_clean)
evaluate_regression(clean_evals,lr, "Linear Regression", X_train_clean, X_test_clean, y_train_clean, y_test_clean)
```

```
El error de entrenamiento de Linear Regression es: 1261.6483025697448 -> El error de prueba de Linear Regression es: 2863.45615
40306877
El R2 score del entrenamiento de Linear Regression es: 97.33096217243008 y el de prueba es: 89.61367681213584
```

```
# Reduzco las dimensiones
reduced_lr = LinearRegression().fit(X_train_r, y_train_r)
evaluate_regression(reduced_evals,reduced_lr, "Reduced Linear Regression", X_train_r, X_test_r, y_train_r, y_test_r)
```

```
El error de entrenamiento de Reduced Linear Regression es: 2030.9189477917653 -> El error de prueba de Reduced Linear Regressio
n es: 3508.9102401698547
El R2 score del entrenamiento de Reduced Linear Regression es: 93.08386763624078 y el de prueba es: 84.40356731722852
```

```
# Muestro la gráfica del modelo de Regresión Linear y lo que predijo
plot_predictions(lr, X_test_clean, y_test_clean)
```



Modelo de Regresión Lasso

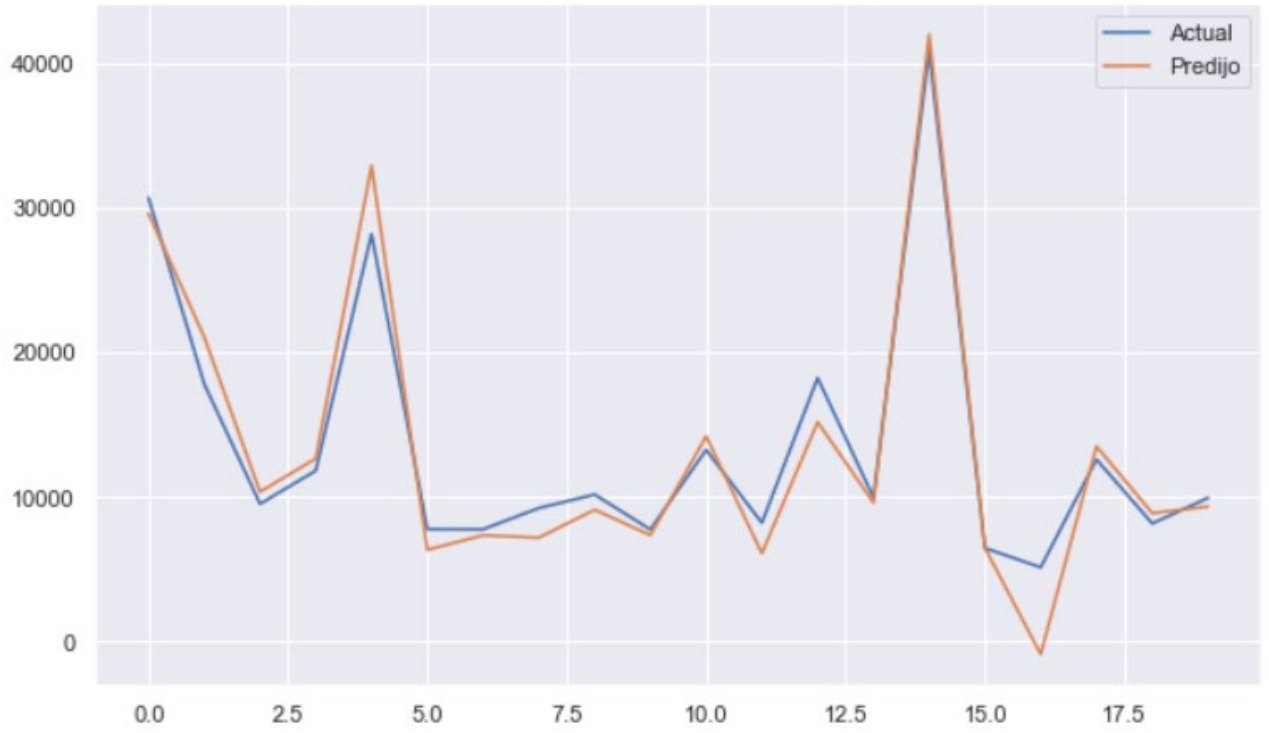
```
las = Lasso().fit(X_train_clean, y_train_clean)
evaluate_regression(clean_evals,las, "Lasso Regression", X_train_clean, X_test_clean, y_train_clean, y_test_clean)
```

```
El error de entrenamiento de Lasso Regression es: 1265.4872738760653 -> El error de prueba de Lasso Regression es: 2929.7048343
044685
El R2 score del entrenamiento de Lasso Regression es: 97.31469464598985 y el de prueba es: 89.12752312270594
```

```
# Reduzco las dimensiones
reduced_las = LinearRegression().fit(X_train_r, y_train_r)
evaluate_regression(reduced_evals,reduced_las, "Reduced Lasso Regression", X_train_r, X_test_r, y_train_r, y_test_r)
```

```
El error de entrenamiento de Reduced Lasso Regression es: 2030.9189477917653 -> El error de prueba de Reduced Lasso Regression
es: 3508.9102401698547
El R2 score del entrenamiento de Reduced Lasso Regression es: 93.08386763624078 y el de prueba es: 84.40356731722852
```

```
# Muestro la gráfica de las predicciones con Lasso Regression y Reduced Lasso Regression
plot_predictions(las, X_test_clean, y_test_clean)
```



Regresión de Cresta (Ridge Regression)

La regresión de cresta se utiliza para resolver el problema de la multicolinealidad cuando las variables independientes están altamente correlacionadas entre sí, y la matriz de correlación será singular y no podemos obtener un parámetro único.

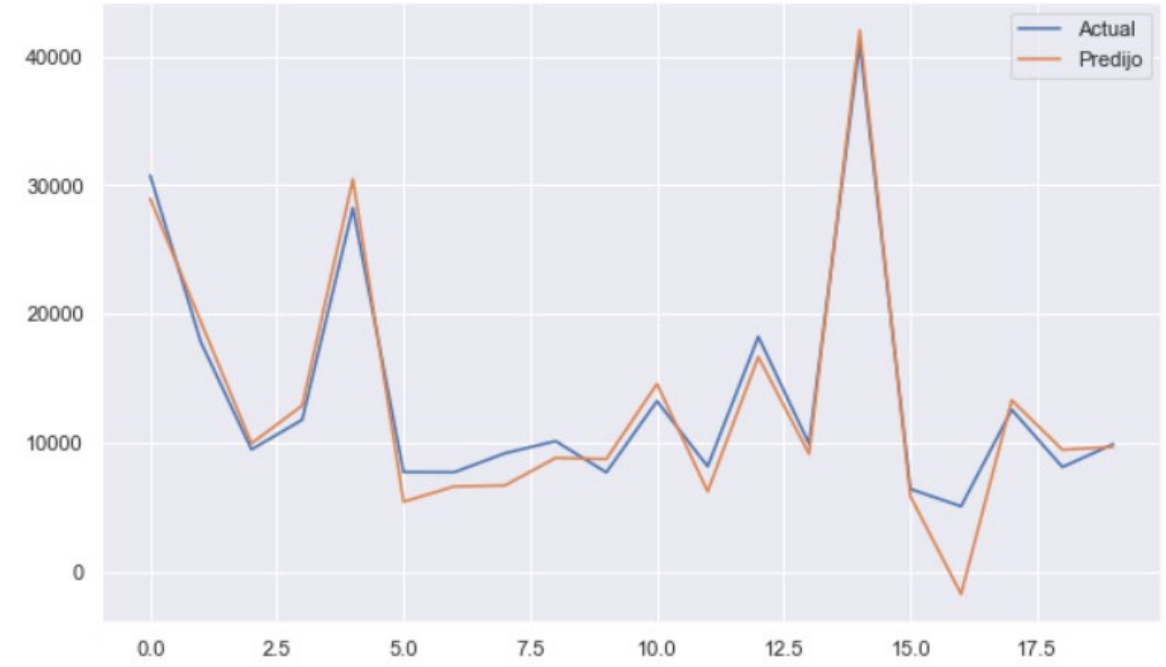
```
rlr = Ridge(alpha=0.9).fit(X_train_clean, y_train_clean)
evaluate_regression(clean_evals,rlr, "Ridge Regression", X_train_clean, X_test_clean, y_train_clean, y_test_clean)
```

```
El error de entrenamiento de Ridge Regression es: 1431.9020309845355 -> El error de prueba de Ridge Regression es: 2921.9394355
6573
El R2 score del entrenamiento de Ridge Regression es: 96.56200909320904 y el de prueba es: 89.18508334136021
```

```
reduced_rlr = Ridge(alpha=0.9).fit(X_train_r, y_train_r)
evaluate_regression(reduced_evals,reduced_rlr, "Reduced Ridge Regression", X_train_r, X_test_r, y_train_r, y_test_r)
```

```
El error de entrenamiento de Reduced Ridge Regression es: 2085.2818391190044 -> El error de prueba de Reduced Ridge Regression
es: 3447.0768834745268
El R2 score del entrenamiento de Reduced Ridge Regression es: 92.70865521092833 y el de prueba es: 84.94839899439896
```

```
plot_predictions(rlr, X_test_clean, y_test_clean)
```



Regresión robusta (Huber Regressor)

La función de pérdida de Huber tiene la ventaja de no estar fuertemente influenciada por los valores atípicos sin ignorar por completo su efecto.

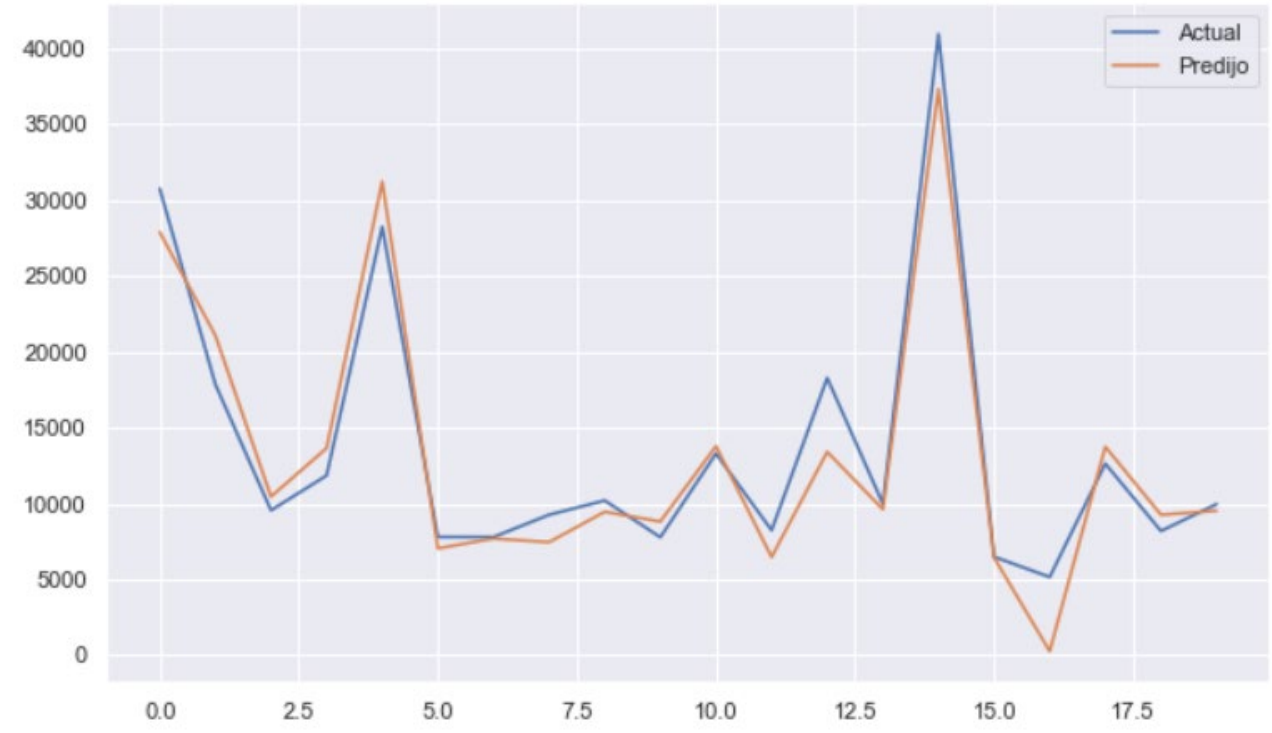
```
huber = HuberRegressor().fit(X_train_clean, y_train_clean)
evaluate_regression(clean_evals,huber, "Robust Regression", X_train_clean, X_test_clean, y_train_clean, y_test_clean)
```

```
El error de entrenamiento de Robust Regression es: 1564.0281259930696 -> El error de prueba de Robust Regression es: 2949.16166
7819833
El R2 score del entrenamiento de Robust Regression es: 95.89826847163405 y el de prueba es: 88.98263041833603
```

```
huber_r = HuberRegressor().fit(X_train_r, y_train_r)
evaluate_regression(reduced_evals,huber_r, "Reduced Robust Regression", X_train_r, X_test_r, y_train_r, y_test_r)
```

```
El error de entrenamiento de Reduced Robust Regression es: 2241.4649503791597 -> El error de prueba de Reduced Robust Regressio
n es: 3505.9964767766946
El R2 score del entrenamiento de Reduced Robust Regression es: 91.57554105917794 y el de prueba es: 84.42945880105532
```

```
plot_predictions(huber, X_test_clean, y_test_clean)
```



RandomForest

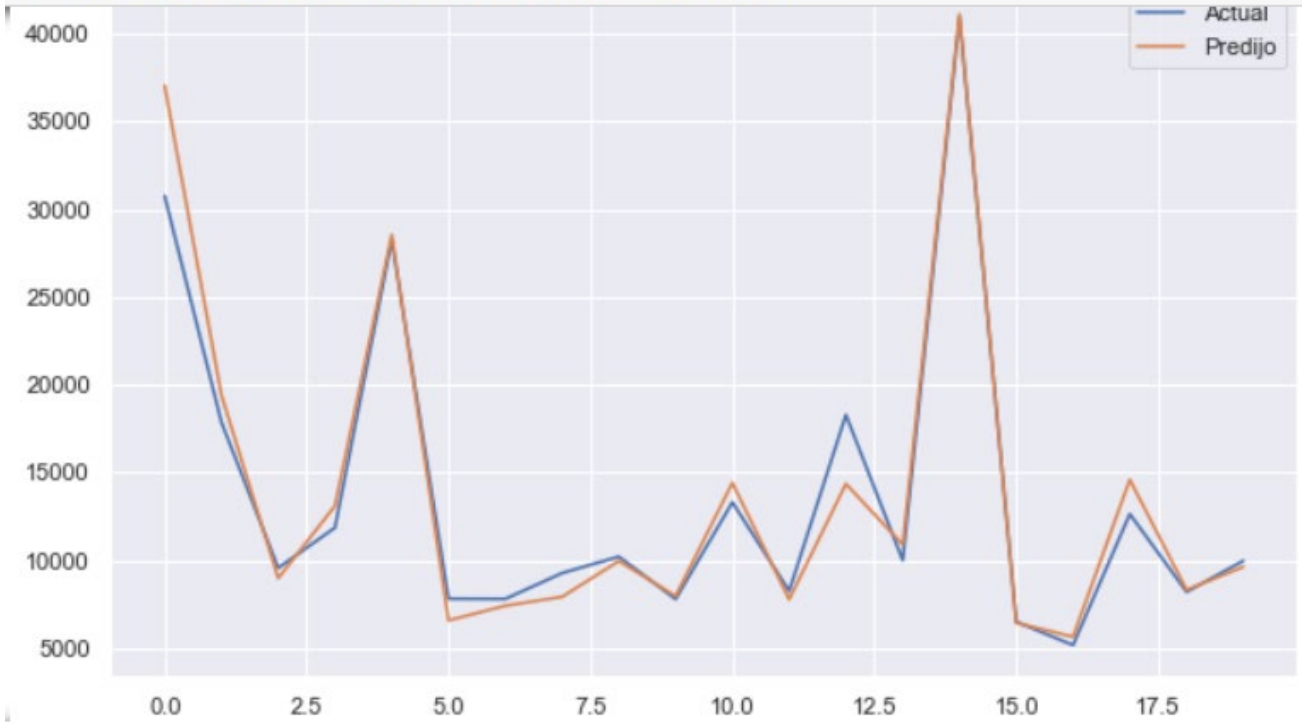
```
rf = RandomForestRegressor(n_estimators=100).fit(X_train_clean, y_train_clean)
evaluate_regression(clean_evals,rf, "RandomForest Regression", X_train_clean, X_test_clean, y_train_clean, y_test_clean)
```

```
El error de entrenamiento de RandomForest Regression es: 879.9080183090031 -> El error de prueba de RandomForest Regression es:
1881.5767931152975
El R2 score del entrenamiento de RandomForest Regression es: 98.70176694088816 y el de prueba es: 95.51539028676163
```

```
rf_r = RandomForestRegressor(n_estimators=100).fit(X_train_r, y_train_r)
evaluate_regression(reduced_evals,rf_r, "Reduced RandomForest Regression", X_train_r, X_test_r, y_train_r, y_test_r)
```

```
El error de entrenamiento de Reduced RandomForest Regression es: 997.9168607788808 -> El error de prueba de Reduced RandomFores
t Regression es: 2775.5858771470266
El R2 score del entrenamiento de Reduced RandomForest Regression es: 98.33019085755706 y el de prueba es: 90.24134206039261
```

```
plot_predictions(rf, X_test_clean, y_test_clean)
```

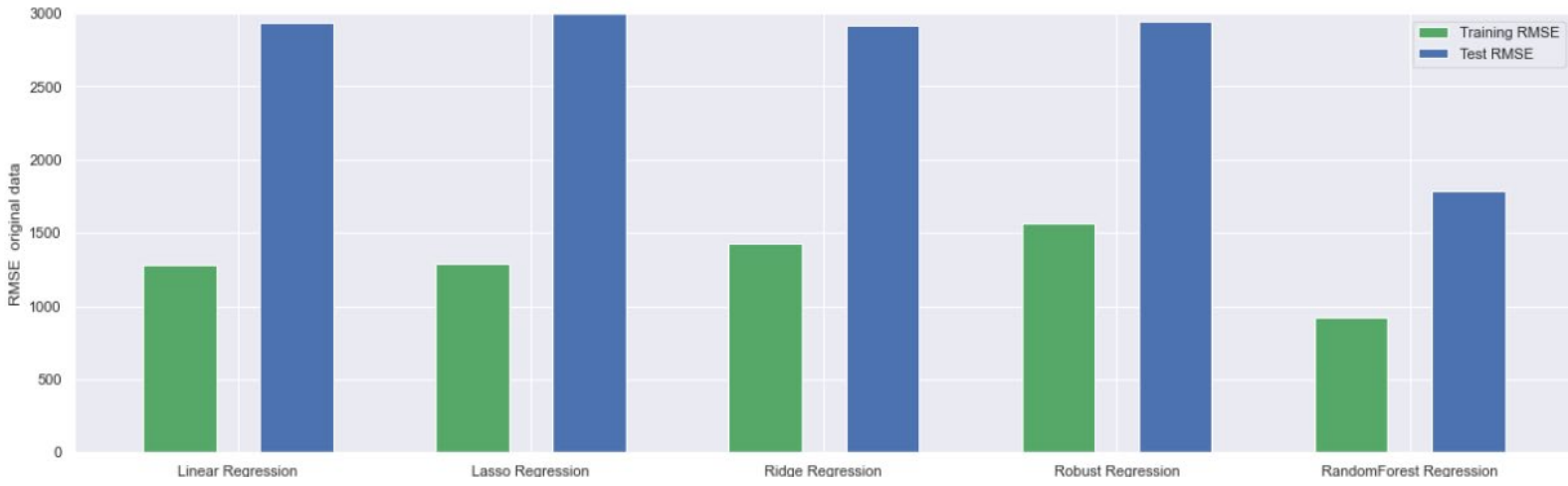


Veo que Random Forest es el que me da el R2 Score más alto, con 98,33% ,seguido de Lasso con 93,08%

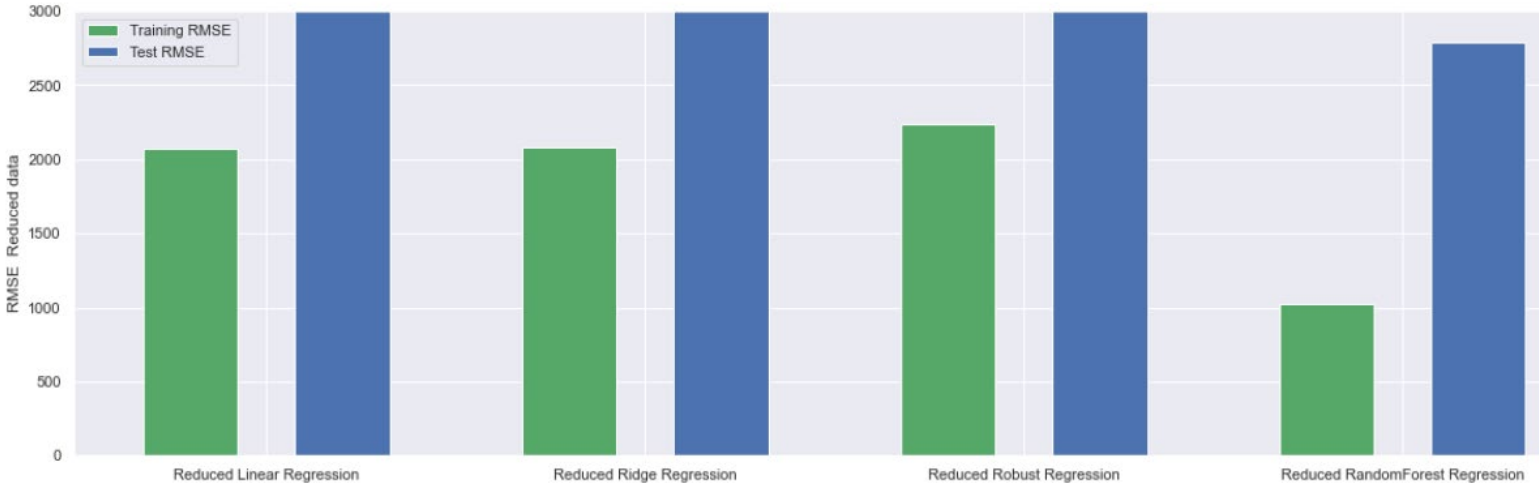
Por último, creo un gráfico de barras para representar los errores de los modelos.

```python
def visualize_errors(evals, data):
    keys = [key for key in evals.keys()]
    values = [value for value in evals.values()]
    fig, ax = plt.subplots(figsize=(20, 6))
    ax.bar(np.arange(len(keys)) - 0.2, [value[0] for value in values], color='g', width=0.25, align='center')
    ax.bar(np.arange(len(keys)) + 0.2, [value[1] for value in values], color='b', width=0.25, align='center')
    ax.legend(["Training RMSE", "Test RMSE"])
    ax.set_xticklabels(keys)
    ax.set_xticks(np.arange(len(keys)))
    plt.ylim(0, 3000)
    plt.ylabel("RMSE  " + str(data))
    plt.show()
```

```python
visualize_errors(clean_evals, 'original data')
```



```
visualize_errors(reduced_evals, 'Reduced data')
```



CONCLUSIÓN

He utilizado varios modelos de regresión para ajustar los datos y parece que todos lograron hacerlo, esto indica que la etapa de preprocesamiento de datos fue un éxito.

También noté que el PCA podría preservar la varianza en los datos, reduje el número de características de 73 a 29 y aún así podría administrar un rendimiento justo en nuestros modelos con solo una ligera diferencia del conjunto de datos original sin tener en cuenta el enorme sobreajuste con modelos de regresión lineal y de lazo en los datos reducidos.