INSTITUTO POLITECNICO NACIONAL “IPN”

ESCUELA SUPERIOR DE COMPUTO “ESCOM”

LANDEROS CORTES PEDRO JONAS

MACHINE LEARNING

PRACTICA 4

Indice

INDICE DE FIGURAS

Introducción

Regresores

Un regresor es un modelo que aprende a predecir un valor numérico continuo a parte de un conjunto de variables de entrada, llamadas features, que formalmente buscan aproximar una función desconocida.

La manera estándar de entrenar un regreso es “minimizando” una función de perdida “L” sobre un conjunto de datos.

Existen distintos tipos de regresores, como lo pueden ser:

- Lineales: Su predicción es una combinación lineal de las features, son rápidos interpretables y buenos como una línea de partida.
- No lineales: Capturan relaciones complejas como lo pueden ser árboles o redes neuronales.
- Ensamble: Combinan muchos modelos débiles para mejorar el rendimiento.
- Basados en proximidad: Predicen con promedios de vecinos parecidos (KNN).

Métricas de evaluación

Las métricas de evaluación sirven para medir que tan bien se desempeña un modelo en un problema específico. Existen métricas para regresión y para clasificación.

Para regresión existen las siguientes métricas:

- MAE (Mean Absolute Error)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Figure 1. MAE formula

Mide el promedio del valor absoluto de los errores, mide cuantas unidades se equivocan (en promedio) en el modelo.

- MSE (Mean Squared Error)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Figure 2. MSE formula

Es el promedio de los errores al cuadrado y penaliza mas los errores grandes.

- RMSE (Root Mean Squared Error)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Figure 3. RMSE formula

En donde la Raiz cuadrada del MSE se expresa en las mismas unidades que la variable objetivo.

- R^2 (Coeficiente de determinación):

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Figure 4. R^2 formula

En donde se mide la proporción de la varianza de los datos que el modelo logra explicar, y valores cercanos a 1 indican un buen ajusta y valores lejanos o negativos indican mal desempeño.

Ejemplos de métricas de regresión con formulas

Supóngase que existen un dataset con precios de casas (en miles de pesos).

- Valores reales: [100, 150, 200]
- Predicciones del modelo: [110, 140, 195]

Es decir, en la primera casa hubo un error +10, en la segunda -10 y en la tercera -5.

MAE:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$MAE = \frac{|100 - 110| + |150 - 140| + |200 - 195|}{3}$$

$$MAE = \frac{10 + 10 + 5}{3} = \frac{25}{3} \approx 8.33$$

Figure 5. Ejemplo MAE

En promedio, el modelo se equivoca en 8.33 mil pesos, esto quiere decir que si por ejemplo una casa cuesta 100,000 el modelo puede predecir 108,000 o 92,000

MSE:

MSE (Error Cuadrático Medio)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$MSE = \frac{(100 - 110)^2 + (150 - 140)^2 + (200 - 195)^2}{3}$$

$$MSE = \frac{100 + 100 + 25}{3} = \frac{225}{3} = 75$$

Figure 6. Ejemplo MSE

El error cuadrático medio es de 75 (al cuadrado) miles de pesos, esto reafirma que en errores mas grandes se penaliza mucho más, errores de 10,000 a 20,000 hubieran sido más penalizados.

Es importante aclarar que al decir que los errores mas grandes son “mas penalizados” se refiere a que esta métrica sirve para evaluar casos en los que la predicción requiere evitar los errores grandes, como por ejemplo en una dosis de medicamento una predicción con un error grande puede ser muy grave, por lo cual se podría utilizar esta métrica para evaluarla primero.

RMSE:

RMSE (Raíz del Error Cuadrático Medio)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$RMSE = \sqrt{75} \approx 8.66$$

Figure 7. Ejemplo RMSE

El modelo se desvía aproximadamente 8.7 mil pesos del valor real.

R^2

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

1. Promedio real ($\bar{y}$):

$$\bar{y} = \frac{100 + 150 + 200}{3} = 150$$

2. Suma de errores del modelo (SSE):

$$\sum(y_i - \hat{y}_i)^2 = 100 + 100 + 25 = 225$$

3. Suma de la varianza total (SST):

$$\sum(y_i - \bar{y})^2 = (100 - 150)^2 + (150 - 150)^2 + (200 - 150)^2$$

$$= 2500 + 0 + 2500 = 5000$$

4. Sustituyendo:

$$R^2 = 1 - \frac{225}{5000} = 1 - 0.045 = 0.955$$

Figure 8. Ejemplo R^2

Significa que el modelo explica el **95.5 % de la variación** de los precios de las casas.

Desarrollo

Regresores

Para el desarrollo de esta práctica, se utilizaron los siguientes regresores en distintos datasets en ejemplos realizados con Python:

LinearRegression

La regresión lineal es un modelo que busca aproximar una relación lineal entre las variables independientes y la variable dependiente, tiene como forma general:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

donde:

- x_i son las características,
- β_i son los coeficientes que aprende el modelo,
- β_0 es el intercepto,
- $\hat{y}$ es la predicción.

La estimación de los coeficientes se hace con el método de los mínimos cuadrados que busca minimizar el MSE.

La siguiente imagen muestra una representación visual de la regresión lineal.

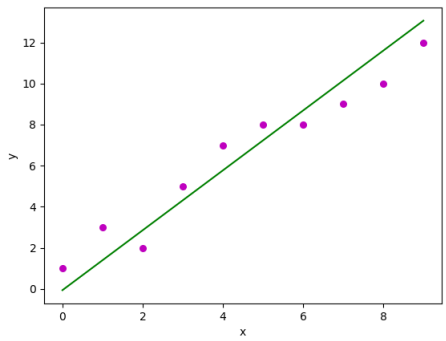

Figure 9. Regresion lineal

KNeighborsRegressor

Es un algoritmo no paramétrico que para cada nuevo punto calcula la distancia a todos los puntos de entrenamiento y selecciona los K puntos mas cercanos, y devuelve como predicción el promedio de los valores reales de esos K vecinos.

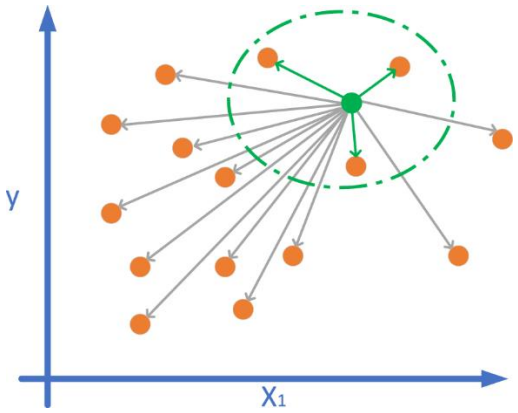

Figure 10. KNeighborsRegressor

RandomForestRegressor

Es un modelo de ensamble basado en arboles de decisión, cada árbol divide los datos de forma jerárquica según las reglas definidas por cada dataset, cada nodo divide el espacio de datos en regiones cada vez más homogéneas y al final se hace una predicción obteniendo el promedio de los valores en la hojas.

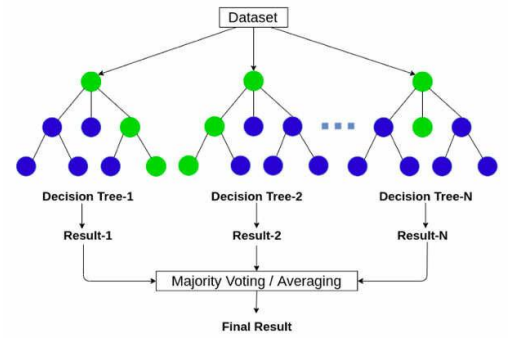

Figure 11. RandomForestRegressor

MLPRegressor

Es un tipo de red neuronal artificial que está compuesta por: Una capa de entrada, una o varias capas ocultas con neuronas artificiales y una capa de salida.

Cada neurona realiza

$$z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$

$$a = f(z)$$

donde f es una **función de activación** (ReLU, sigmoide, tanh).

El modelo aprende ajustando los pesos w mediante la retro propagación y descenso de gradiente.

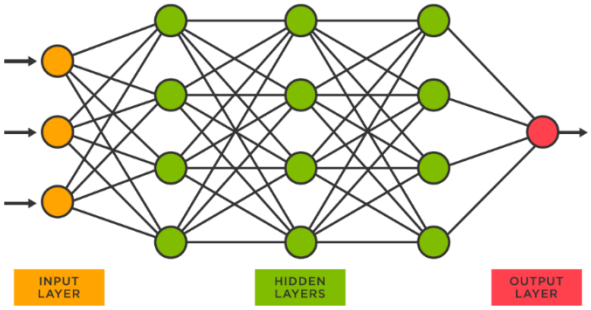

Figure 12. MLPRegressor

Datasets

Iris Dataset

Id, SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm, Species

Boston Housing

CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, MEDV

Diabetes Dataset

Gender, Age, Hypertension, heart_disease, smoking_history, Bmi, HbA1c_level, blood_glucose_level, diabetes

Wine Quality

fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, quality, Id

Car Price Prediction

car_ID, symboling, CarName, fueltype, aspiration, doornumber, carbody, drivewheel, enginelocation, wheelbase, carlength, carwidth, carheight, curbweight, enginetype, cylindernumber, enginesize, fuelsystem, boreratio, stroke, compressionratio, horsepower, peakrpm, citympg, highwaympg, price

Concrete Compressive Strength

cement, blast_furnace_slag, fly_ash, water, superplasticizer, coarse_aggregate, fine_aggregate, age, concrete_compressive_strength

Flujo del programa

1. **Configuración de datasets**
   En DATASETS se define, para cada CSV, la **columna objetivo** y las columnas a **descartar** (por ejemplo, IDs o etiquetas que podrían inducir fuga de información). El script intenta cargar cada archivo desde el directorio actual y avanza solo con los que encuentra.

2. **Preprocesamiento automático por tipo de variable**
   Se detectan **columnas numéricas** y **categóricas** a partir de los dtypes. Luego se construye un ColumnTransformer con dos subpipelines:

- Numéricas: SimpleImputer(strategy="mean") + StandardScaler()
- Categóricas: SimpleImputer(strategy="most_frequent") + OneHotEncoder(handle_unknown="ignore", sparse=False)
  Esto asegura que todos los modelos reciban entradas limpias y en una escala adecuada (especialmente relevante para **KNN** y **MLP**).

3. **Entrenamiento y evaluación por modelo**
   Para cada dataset se separa en **train/test** (test_size=0.2, semilla fija) y, por cada modelo en MODELOS, se arma un Pipeline (preprocess → model). Se entrena con los datos de entrenamiento y se predice en test. Se calculan **MAE**, **MSE**, **RMSE** y **R^2**, y se consolidan en un DataFrame ordenado por R^2 (descendente).

4. **Visualización y exportación**
   Se generan **barras** de MAE, RMSE y R^2 por modelo, y un **diagrama de dispersión** (y_real vs. y_pred) para el **mejor modelo** del dataset. Todo se guarda en una carpeta outputs/<dataset>/. Además, se exporta un CSV con las métricas por dataset y otro **resumen global** con el mejor modelo de cada dataset y su R^2, junto con una barra global de "Mejor R^2 por dataset".

5. **Robustez y reproducibilidad**

- matplotlib usa backend **no interactivo** para evitar problemas de GUI y guardar PNGs sin abrir ventanas.
- Se suprimen *warnings* visuales para una salida limpia.
- Manejo de errores por dataset: si falta un archivo o la columna objetivo, el script **lo salta** y continúa con el resto.
- RANDOM_STATE=42 fija la reproducibilidad en particiones, RF y MLP.

Ejecucion del programa

A continuación se muestran los resultados obtenidos con las métricas de evaluación en los distintos datasets:

Iris dataset:

```
=== Dataset: Iris.csv | target='PetalLengthCm' ===
Métricas guardadas en: outputs\Iris_metricas.csv
                Modelo       MAE       MSE      RMSE        R2
0     LinearRegression  0.227055  0.087596  0.295966  0.973273
1  RandomForestRegressor  0.265764  0.096389  0.310466  0.970590
2   KNeighborsRegressor  0.264667  0.103533  0.321766  0.968410
3         MLPRegressor  0.372069  0.271717  0.521264  0.917094
```

Figure 13. Resultados Iris Dataset

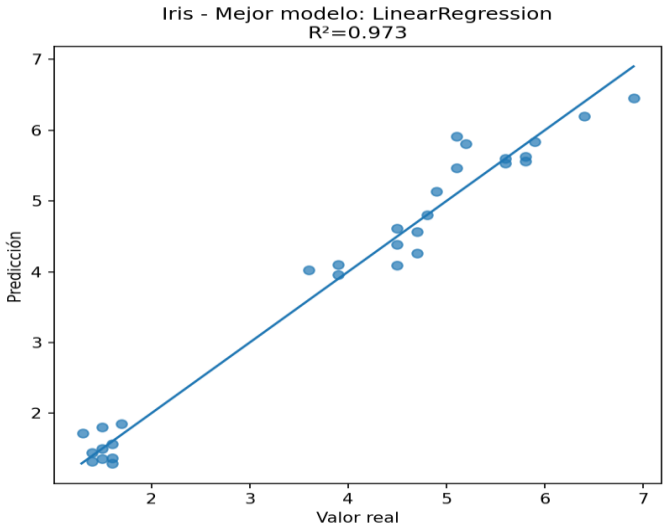

Figure 14. Iris dataset regresion lineal

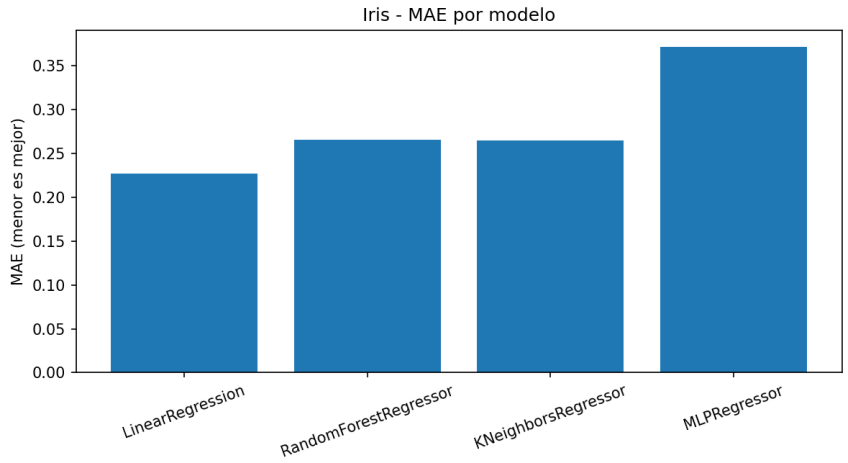

Figure 15. Iris dataset MAE

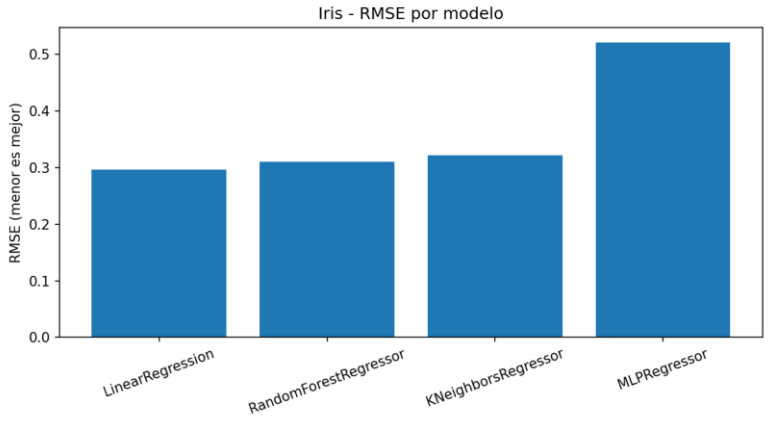

Figure 16. Iris dataset RMSE

Housing dataset:

```
=== Dataset: HousingData.csv | target='MEDV' ===
Métricas guardadas en: outputs\HousingData_metricas.csv
                 Modelo       MAE        MSE      RMSE        R2
0  RandomForestRegressor  2.091722   8.894597  2.982381  0.878711
1          MLPRegressor  2.301767  12.557948  3.543719  0.828756
2    KNeighborsRegressor  2.841176  22.634627  4.757586  0.691348
3       LinearRegression  3.147436  25.002389  5.000239  0.659060
```

Figure 17. Resultados Housing dataset

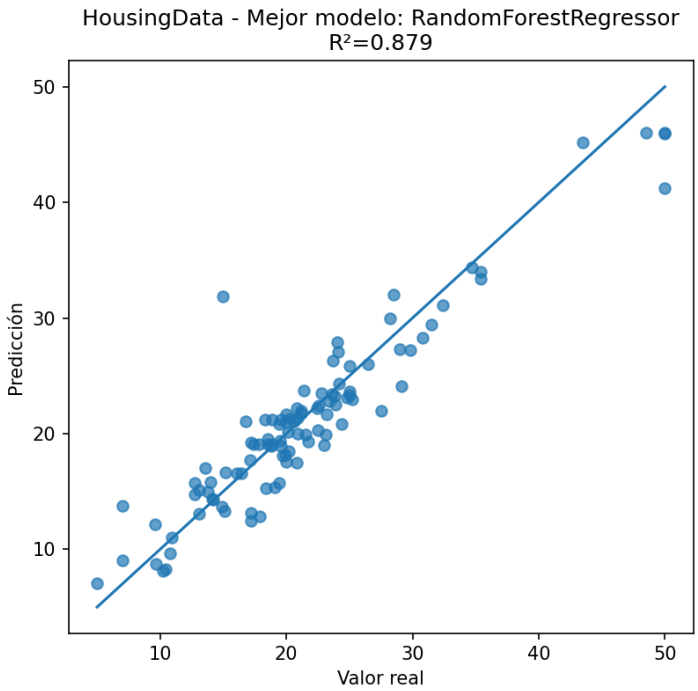

Figure 18. Housing dataset regresion lineal

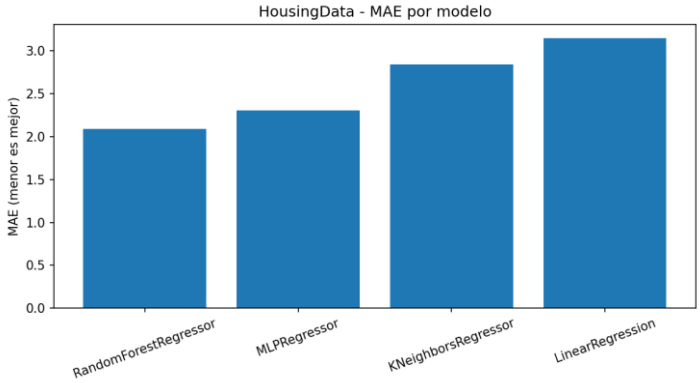

Figure 19. Housing dataset MAE

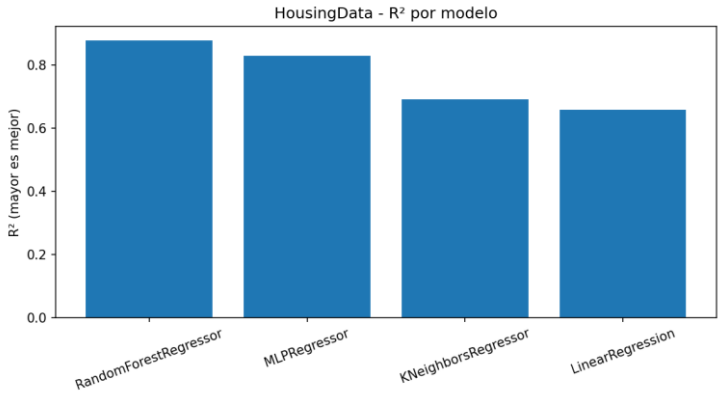

Figure 20. Housing dataset R2

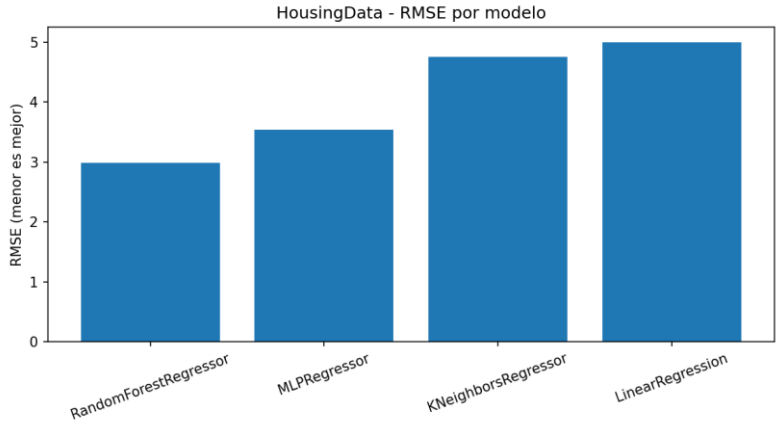

Figure 21. Housing dataset RMSE

Diabetes dataset:

```
=== Dataset: diabetes_prediction_dataset.csv | target='blood_glucose_level' ===
Métricas guardadas en: outputs\diabetes_prediction_dataset_metricas.csv
                Modelo        MAE          MSE       RMSE        R2
0         MLPRegressor  30.801686  1521.048774  39.000625  0.077527
1       LinearRegression  30.724376  1572.467846  39.654355  0.046342
2  RandomForestRegressor  32.692466  1702.538538  41.261829 -0.032542
3    KNeighborsRegressor  33.654090  1819.056878  42.650403 -0.103207
```

Figure 22. Resultados Diabetes dataset

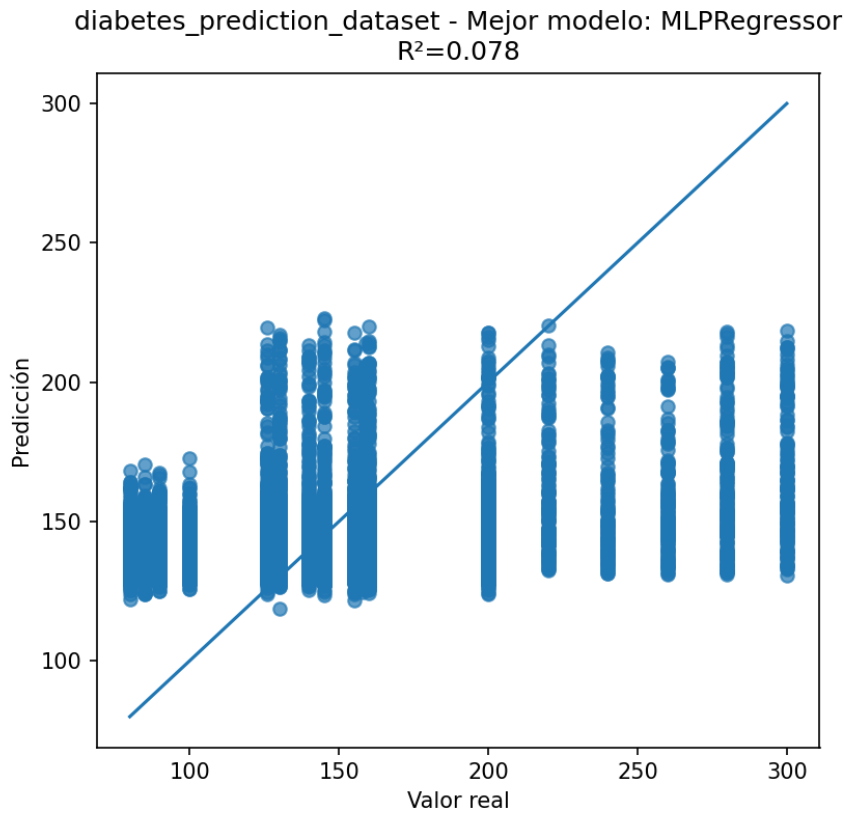

Figure 23. Diabetes dataset regresion lineal

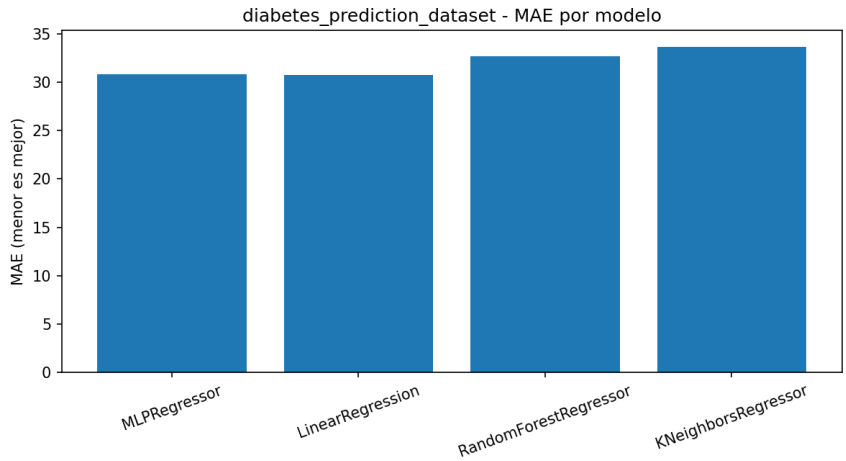

Figure 24. Diabetes dataset MAE

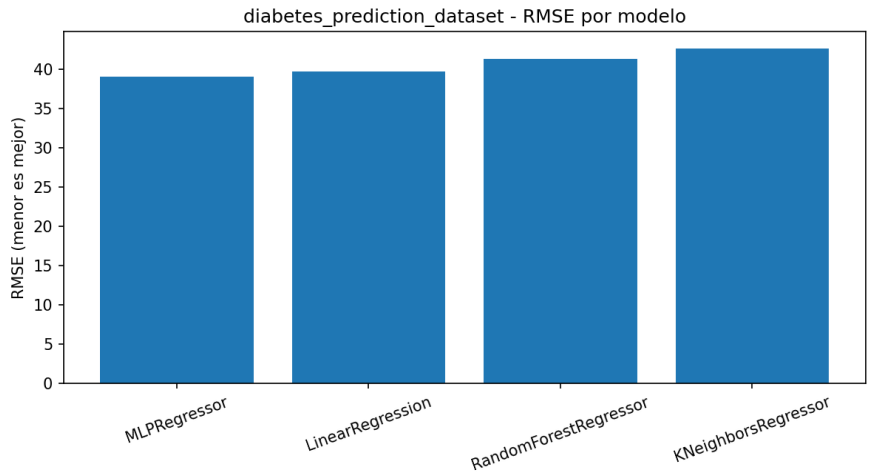

Figure 25. Diabetes dataset RMSE

Wine dataset:

```
=== Dataset: WineQT.csv | target='quality' ===
Métricas guardadas en: outputs\WineQT_metricas.csv
                 Modelo       MAE       MSE      RMSE        R2
0  RandomForestRegressor  0.410320  0.298992  0.546802  0.462702
1       LinearRegression  0.477340  0.380032  0.616468  0.317069
2    KNeighborsRegressor  0.463755  0.382707  0.618634  0.312262
3           MLPRegressor  0.508725  0.454628  0.674261  0.183018
```

Figure 26. Resultados Wine dataset

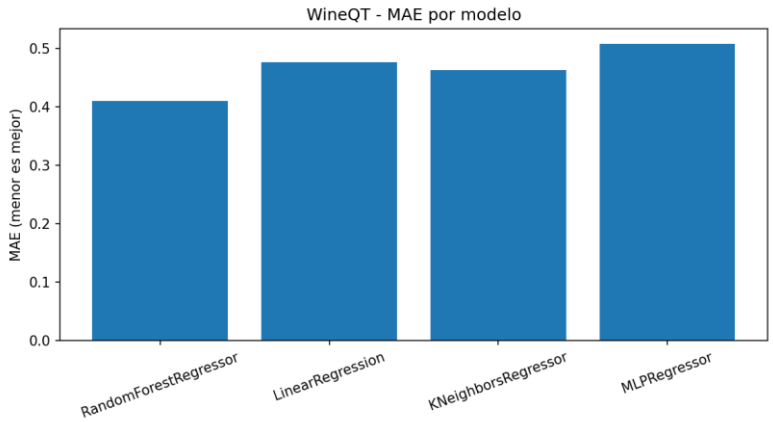

Figure 27.Wine dataset MAE

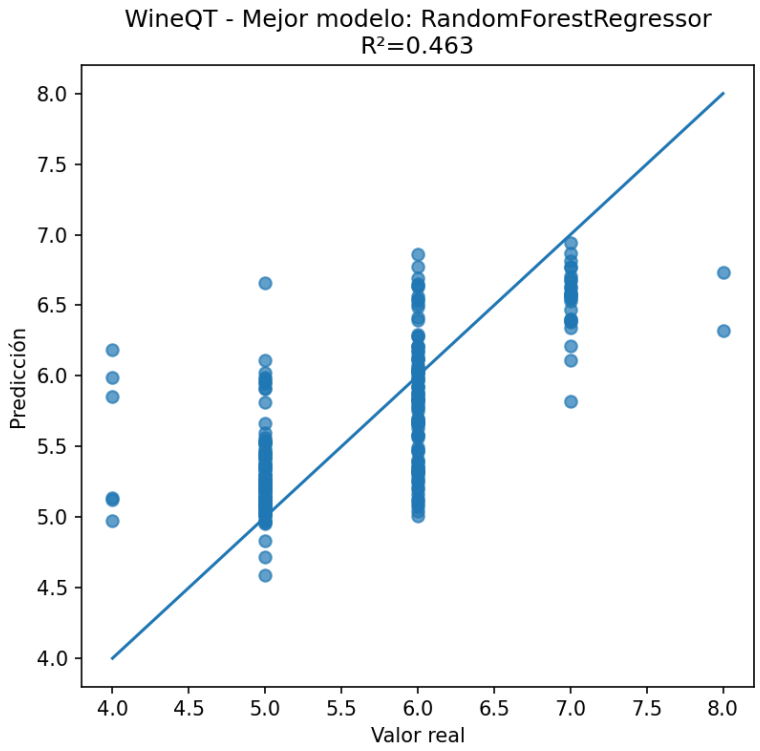

Figure 28. Wine dataset regresion lineal

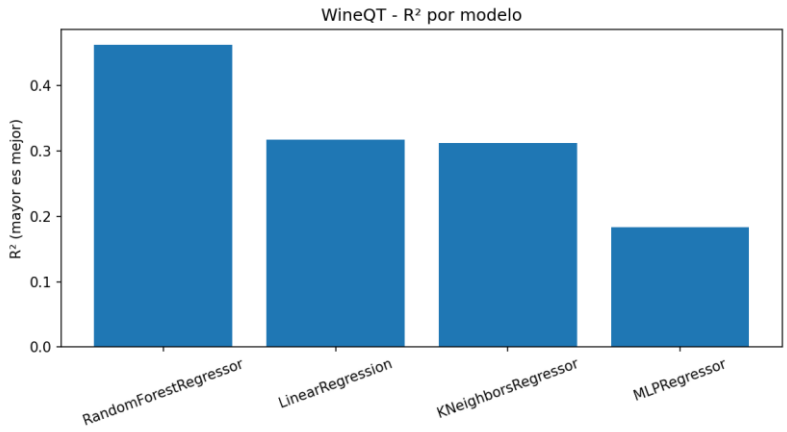

Figure 29. Wine dataset R2

Car dataset:

```
=== Dataset: CarPrice_Assignment.csv | target='price' ===
Métricas guardadas en: outputs\CarPrice_Assignment_metricas.csv
                Modelo          MAE           MSE          RMSE        R2
0  RandomForestRegressor   1327.187309  3.564893e+06   1888.092440  0.954843
1    KNeighborsRegressor   2761.691049  2.208183e+07   4699.130913  0.720285
2       LinearRegression   3700.946767  4.199948e+07   6480.700463  0.467984
3          MLPRegressor  13488.530087  2.608811e+08  16151.812998 -2.304634
```

Figure 30. Resultados Car dataset

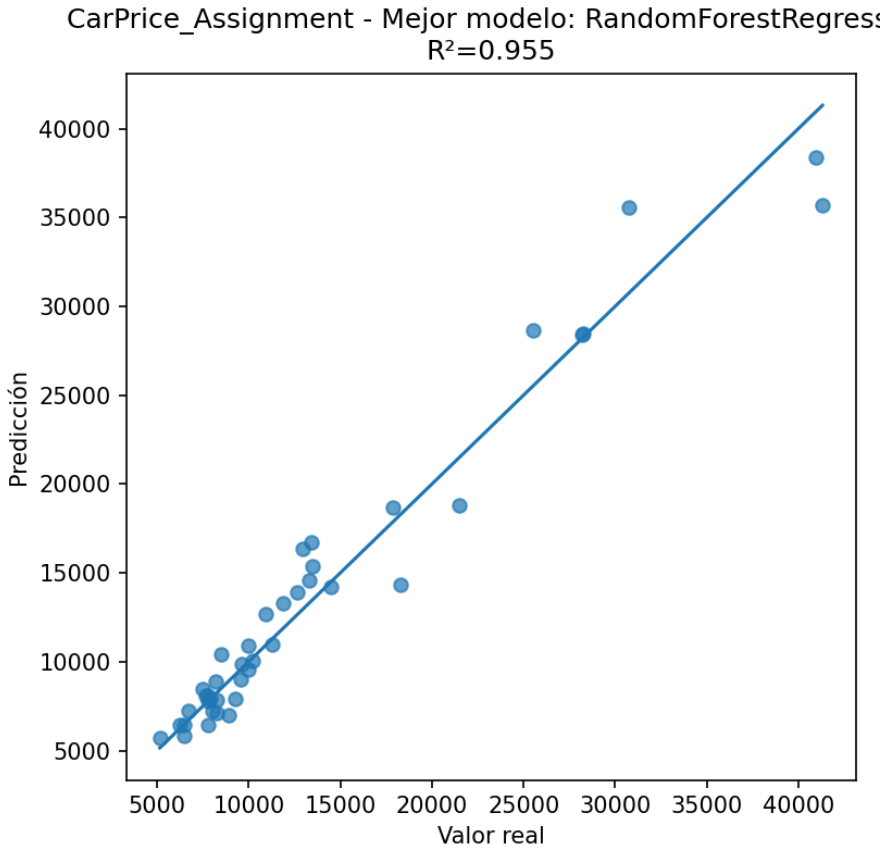

Figure 31. Car dataset Regresion lineal

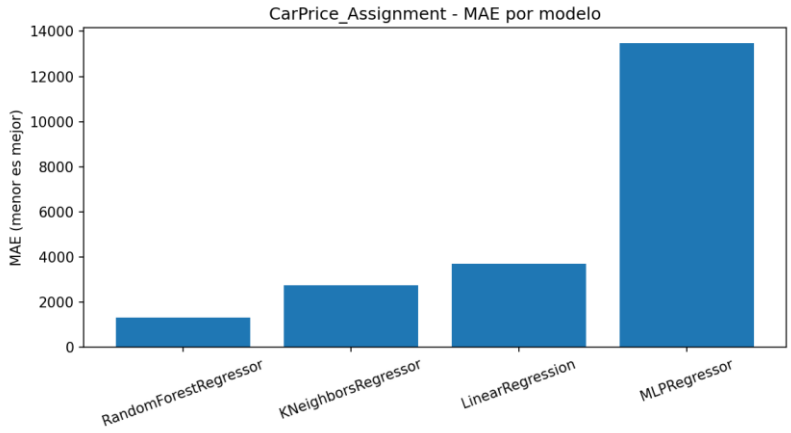

Figure 32. Car dataset MAE

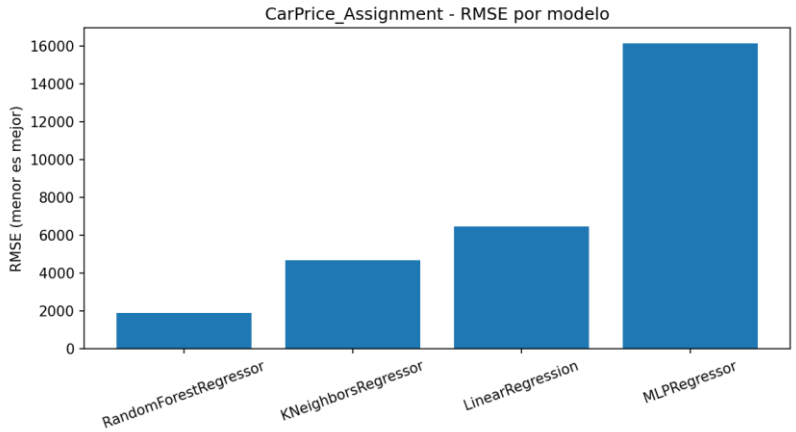

Figure 33. Car dataset RMSE

Concrete dataset:

```
=== Dataset: concrete_data.csv | target='concrete_compressive_strength' ===
Métricas guardadas en: outputs\concrete_data_metricas.csv
                Modelo       MAE        MSE      RMSE        R2
0  RandomForestRegressor  3.792175  30.716122  5.542213  0.880796
1          MLPRegressor  4.639592  34.063625  5.836405  0.867805
2    KNeighborsRegressor  6.800515  73.618692  8.580134  0.714298
3       LinearRegression  7.745559  95.970940  9.796476  0.627553
```

Figure 34. Resultados Concrete dataset

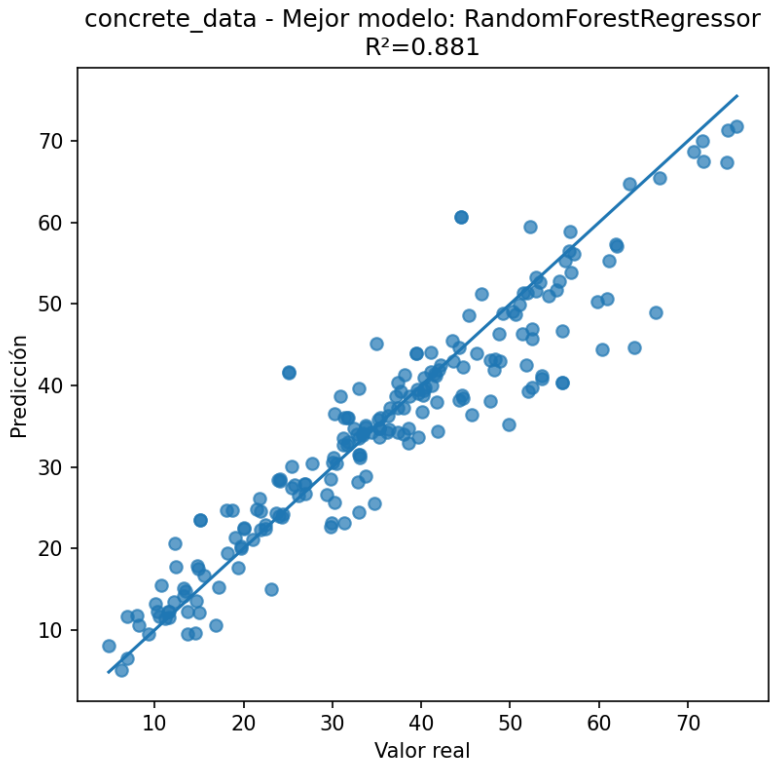

Figure 35. Concrete dataset Regresion lineal

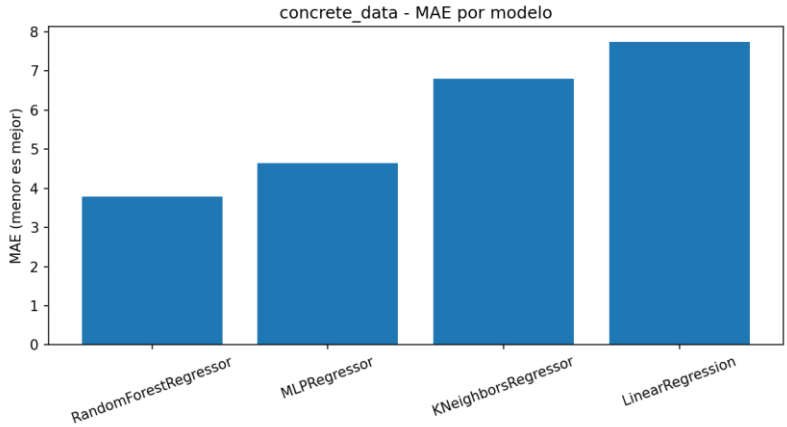

Figure 36. Concrete dataset MAE

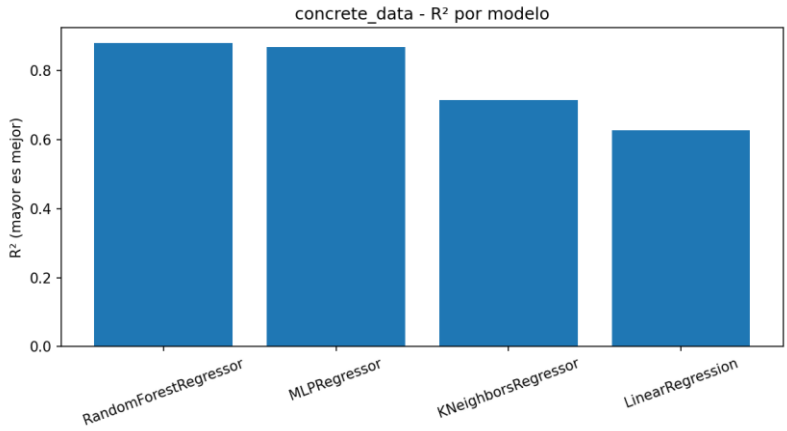

Figure 37. Concrete dataset R2

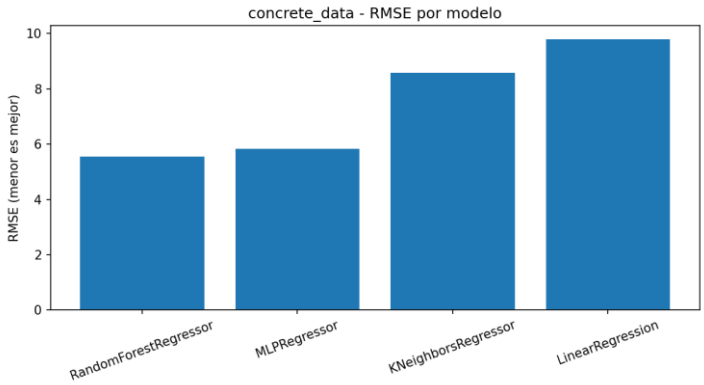

Figure 38. Concrete dataset RMSE

Explicacion de codigo

Bloque de imports y ajustes: silencia *warnings* y fija matplotlib.use("Agg") para exportar figuras.

DATASETS: mapeo {csv: {target, drop}} con comentarios sobre cada archivo (objetivo y columnas a ignorar).

MODELOS: cuatro regresores representativos; RF con n_estimators=300 y n_jobs=-1; MLP con early_stopping=True y capas (64, 32); KNN con k=5; Lineal sin regularización (baseline).

detectar_tipos_columnas(df_X): separa nombres de columnas numéricas y categóricas.

construir_preprocesador(df_X): arma el ColumnTransformer (imputación + escalado/OneHot). handle_unknown="ignore" evita excepciones si aparece una categoría nueva en test.

evaluar_modelos_en_dataset(...):

1. Carga CSV y valida la columna objetivo.
2. Quita columnas a descartar si existen.
3. Separa X/y y construye el **preprocesador** con las columnas de X.
4. train_test_split con semilla fija.
5. Entrena cada Pipeline y calcula **MAE/MSE/RMSE/R^2**; arma el DataFrame de resultados ordenado por R^2 y guarda las parejas (y_true, y_pred) para gráficas de dispersión.

plot_barras_metricas(...): genera PNGs de barras para MAE, RMSE y R^2 en outputs/<dataset>/.

plot_scatter_mejor_modelo(...): selecciona el mejor por R^2 y dibuja **y_real vs. y_pred** con la diagonal de referencia.

main():

- Itera por DATASETS, salta los que no existan, evalúa y **exporta métricas** y **gráficas**.
- Compone un **resumen global** (mejor por dataset) y su gráfica de R^2.
- Imprime en consola las rutas de **todas** las figuras generadas para trazabilidad.

Conclusiones

Con esta práctica pude reforzar los conceptos de aprendizaje supervisado aplicados a problemas de regresión. El programa nos permitió probar varios modelos distintos (lineal, ensamble, vecinos y redes neuronales) sobre diferentes datasets, evaluando su desempeño con métricas objetivas como MAE, RMSE. Gracias al uso de *pipelines* y preprocesamiento automático, el flujo de trabajo se mantuvo ordenado y reproducible.

El aprendizaje principal fue cómo implementar un sistema comparativo basado en reglas y buenas prácticas de ML, que puede servir de base para proyectos más grandes donde se requiera analizar múltiples enfoques antes de tomar decisiones sobre qué modelo usar en producción.