Computer Hardware: predecir el rendimiento relativo publicado de una computadora a partir de su hardware

Julieta Itzel Pichardo Meza | A01369630@tec.mx

31 de agosto de 2025

ABSTRACT Este documento presenta la implementación manual y evaluación de un modelo de Machine Learning para predecir el rendimiento relativo publicado (PRP) de sistemas computacionales a partir de sus componentes hardware utilizando el dataset "Computer Hardware". Se utilizó un modelo de regresión lineal multivariable entrenado con variables cuantitativas como el tiempo de ciclo de máquina (MYCT), memoria principal mínima y máxima (MMIN, MMAX), memoria caché (CACH) y canales de E/S (CHMIN, CHMAX).

El modelo demostró un excelente desempeño predictivo con un R^2 de 0.804, explicando el 80.4% de la variabilidad del PRP. Los resultados mostraron una capacidad de generalización excepcional, con un error cuadrático medio (MSE) de 0.096 en el conjunto de prueba, superior al obtenido en entrenamiento (MSE: 0.223).

El estudio concluye que el modelo de regresión lineal implementado manualmente constituye una herramienta efectiva y interpretable para la predicción del rendimiento computacional, con potencial aplicaciones en diseño de sistemas y planificación de capacidades.

Keywords: Computer Hardware, Performance Prediction, Linear Regression, Data Processing, Gradient Descent.

1. INTRODUCCIÓN

La predicción del rendimiento computacional es esencial para la selección y diseño de sistemas informáticos. Este trabajo implementa un modelo de regresión lineal manual para predecir el Published Relative Performance (PRP) usando el dataset "Computer Hardware".

El modelo identifica la memoria máxima (MMAX) y memoria caché (CACH) como variables predictoras clave, demostrando alta efectividad con R^2 de 0.804 y excelente capacidad de generalización. El enfoque proporciona transparencia metodológica y interpretabilidad de los resultados, ofreciendo una base para aplicaciones prácticas en evaluación de hardware.

2. DESCRIPCIÓN DEL DATASET

El dataset *"Computer Hardware"* proporciona un registro de los componentes (hardware) que conforman diferentes computadoras, así sumando un total de 208 registros. Este dataset fue publicado en el año de 1987 por Jacob Feldmesser.

La estructura de este dataset consiste en un total de diez features: dos feaures categóricas y ocho features cuantitativas.

A continuación, se describen las 10 columnas que contiene este dataset:

- **VendorName (Categorical)**: Es una expresión regular de la marca manufacturera de la computadora, por ejemplo: “cambex”, “ibm”, “honeywell”, etc.
- **ModelName (Categorical)**: Es una expresión regular que es conformada por varios símbolos únicos, por ejemplo: “32/60”, “470v/b”, “470v/7”, etc.
- **MYCT (Interger)**: Se refiere a “Machine Cycle Time” o “Tiempo de Ciclo de la Máquina” en español. Su unidad de medida es nanosegundos.
- **MMIN (Interger)**: Se refiere a “Minimum Main Memory” o “Mínimo de Memoria Principal” en español. Su unidad de medida es kilobytes.
- **MMAX (Interger)**: Se refiere a “Maximum Main Memory” o “Máximo de Memoria Principal” en español. Su unidad de medida es en kilobytes.
- **CACH (Interger)**: Se refiere a “Cache Memory” o “Memoria en Caché” en español. Su unidad de medida es kilobytes.
- **CHMIN (Interger)**: Se refiere a “Minimum Channels” o “Mínimo de Canales” en español. Su unidad de medida es en números enteros.
- **CHMAX (Interger)**: Se refiere a “Maximum Channels” o “Máximo de canales” en español. Su unidad de medida es en números enteros.
- **PRP (Interger)**: Se refiere a “Published Relative Performance” o “Rendimiento Relativo Publicado” en español. Su unidad es un valor entero.
- **ERP (Interger)**: Se refiere a “Estimated Relative Performance” o “Rendimiento Relativo Estimado” en español. Su unidad es un valor entero.

3. EXTRACCIÓN Y LIMPIEZA DE DATOS

El primer paso en el procesamiento es la carga y la revisión de datos faltantes. En la documentación del dataset indica que ninguna instancia tiene valores vacíos; aún así se realizó una limpieza de valores vacíos para así eliminar estas instancias. Después de este proceso no cambió en tamaño del dataset ni el número de instancias.

Una vez cargado el dataset, se realizó la visualización de las variables que muestra cada columna, para así descartar el uso de las variables categóricas ya que para su uso en un modelo que usa regresión lineal multivariable no son relevantes.

Ya con el dataset limpio de variables categóricas, se puede hacer el análisis de correlación entre las variables restantes para la elección de variables a usar como dependientes e independientes.

A continuación, se muestra la tabla de correlación de las variables cuantitativas:

	125	256	6000	256.1	16	128	198
125	1.000000	-0.337071	-0.379592	-0.340414	-0.300734	-0.255629	-0.306571
256	-0.337071	1.000000	0.757827	0.602788	0.526665	0.293877	0.798310
6000	-0.379592	0.757827	1.000000	0.600680	0.568594	0.562388	0.865576
256.1	-0.340414	0.602788	0.600680	1.000000	0.588128	0.423550	0.704642
16	-0.300734	0.526665	0.568594	0.588128	1.000000	0.541762	0.608841
128	-0.255629	0.293877	0.562388	0.423550	0.541762	1.000000	0.621309
198	-0.306571	0.798310	0.865576	0.704642	0.608841	0.621309	1.000000

Figura 1. Tabla de correlación entre variables cuantitativas

En la tabla se puede observar la correlación de todas las variables entre sí.

Para la elaboración del modelo, se eligió a la variable PRP, "Rendimiento Relativo Publicado" (o como señala en la tabla, la columna "198") como la variable dependiente, o variable a predecir en este modelo. Por esta razón, se buscan las features relacionadas con PRP que tangan un coeficiente de correlación más cercano a 1 para la construcción del modelo.

Así mismo, los coeficientes de las features con relación a PRP fueron las siguientes:

- MMAX (6000) = 0.865576
- CHMAX (128) = 0.621309
- CACH (256.1) = 0.704642
- CHMIN (16) = 0.608841
- MMIN (256) = 0.798310
- MYCT (125) = -0.306571

Se puede observar que todas las variables excepto MCYT tienen correlaciones positivas con el rendimiento, por lo que la hace descartarse automáticamente. De la misma manera, los predictores más cercanos a 1 son MMAX y MMIN.

Analizando los dos predictores más altos de este análisis se observa que la correlación MMAX y MMIN también es de un valor alto: 0.757827, lo que puede presentar un problema de multicolinealidad ya que ambas variables miden conceptos similares. Al incluir ambas en el modelo puede hacer el modelo inestable al inflar la varianza de los coeficientes de la regresión lineal multivariable, por lo que se elimina la feature de MMIN como predictor dentro del modelo, así dejando a MMAX como el principal predictor del modelo.

Debido a la redundancia de información de features con alta correlación con PRP como lo son CACH y CHMIN, se elige como predictor secundario a CACH por su coeficiente de correlación, así descartando a CHMIN.

En este modelo se elige solamente tener dos predictores con los cuales trabajar las predicciones, por lo que el modelo solo utilizará MMAX y CACH como predictores. La fórmula del modelo se asienta de la siguiente manera:

$$PRP = \beta_0 + \beta_1 * MMAX + \beta_2 * CACH + \varepsilon$$

Ecuación 1. Modelo de regresión multivariable final.

Donde:

- PRP es el Rendimiento relativo publicado.
- β_0 es el bias o término de intercepto en la ecuación cuando todos los valores de las variables independientes son 0.
- MMAX es la memoria principal máxima en kilobytes.
- β_1 es el coeficiente de cambio para MMAX.
- CACH es la memoria caché en kilobytes.
- β_2 es el coeficiente de cambio para CACH.
- ε es el error residual del modelo, es decir, la diferencia no explicada entre el PRP y su valor predicho.

4. TRANSFORMACIÓN DE DATOS

Teniendo un nuevo dataset en donde solamente se conservaron las features de MMAX, CACH y PRP, ya se pueden hacer las normalizaciones correspondientes a estas columnas.

Para este modelo, todas las variables fueron normalizadas utilizando el método de estandarización Z-score para asegurar que cada feature tuviera media cero y desviación estándar de uno.

La transformación de los datos es necesaria para garantizar la estabilidad numérica del algoritmo de descenso de gradiente y permitir la comparación directa de los coeficientes que se hayan generado en el modelo.

Para cada variable predictora *Xi* como para *PRP,* la normalización se realizó mediante la siguiente fórmula:

$$X_{normalizada} = \frac{X - \mu x}{\sigma x}$$

Ecuación 2. Ecuación utilizada para normalización de datos con Z-score.

Donde:

- μX es la media de la variable X.
- σX es la desviación estándar de la variable X.

A continuación la implementación de esta fórmula para la normalización de la feature de MMAX (o la columna *"6000"* dentro del dataset):

```
# Normalize '6000' column
mean_6000 = df['6000'].mean()
std_6000 = df['6000'].std()
df['6000_normalized'] = (df['6000'] - mean_6000) / std_6000

```

Figura 2. Implementación de Z-score para normalizar la columna de MMAX en Python.

Como se observa en la implementación, se hace la normalización de los datos para después agregarlos como otra feature dentro del dataframe ya existente.

Ese mismo proceso ocurre con las instancias de CACH y PRP; creando así nuevas columnas con sus respectivos nombres: *"6000_normalized"*, *"256.1_normalized"* y *"198_normalized"*.

Después de la normalización de todas las features, se hicieron las siguientes visualizaciones de ambas variables *Xi* contra la variable dependiente PRP:

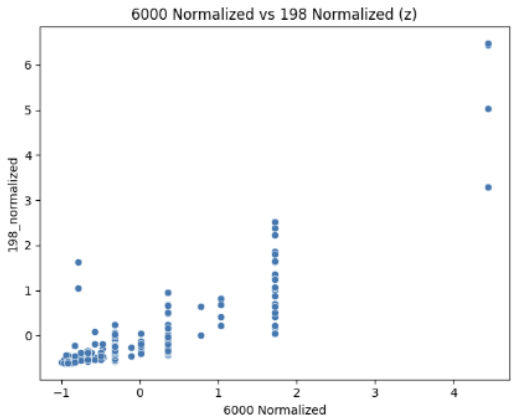

Figura 3. Scattered plot de MMAX normalizado vs PRP normalizado.

Para esta comparación de MMAX contra PRP, se puede observar que los puntos se distribuyen a lo largo de una línea diagonal con una inclinación positiva, lo que da a entender que tiene una relación semejante a la lineal visible; además de contar con una dispersión moderada alrededor de la tendencia central.

Esta gráfica cuenta con un bias bajo ya que los puntos se agrupan alrededor de una línea clara, por lo que una relación lineal puede capturar bien la tendencia principal.

En cuanto a la varianza mostrada en los datos es moderada, ya que la dispersión

es visible pero no es excesiva (aunque si se puede distinguir que existe ruido en la gráfica); esto es un indicador de que MMAX explica una porción significativa de la variabilidad, pero no en su totalidad.

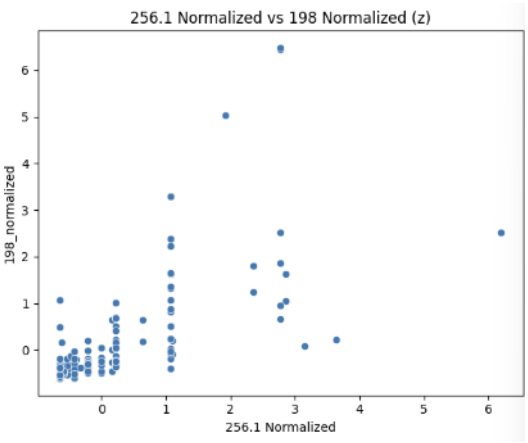

Figura 4. Scattered plot de CACH normalizado vs PRP normalizado.

En el caso de la comparación de CACH con PRP, también se puede observar un comportamiento lineal positivo como en la gráfica anterior; aunque en este caso la gráfica muestra una distribución más dispersa de los datos, sobre todo en los valores extremos.

Para la interpretación del bias, es moderado ya que la tendencia lineal es visible, pero es menos definida, lo que es un indicador de que CACH por sí solo explica menos variabilidad que MMAX.

En cuanto a la varianza de los datos para CACH, estos cuentan con alta varianza ya que hay una dispersión alrededor de la tendencia, provocando valores extremadamente altos como se mencionó anteriormente. Esto sugiere que hay otros factores además de CACH que influyen en el valor de PRP.

Al incluir ambas variables en el modelo se puede encontrar un balance óptimo de bias-varianza, ya que MMAX proporciona el poder predictivo principal y CACH añade información complementaria a este poder predictivo.

Después de haber visualizado y normalizado los datos del dataframe para su mejor uso en el modelo de regresión múltiple, se puede empezar la división del dataset para la parte del entrenamiento y las pruebas posteriores.

El primer paso para ello es mezclar el dataframe de manera aleatoria, después se calcula el tamaño para cada conjunto, teniendo la siguiente distribución:

- 80% para entrenamiento.
- 20% para test.

Una vez calculado el tamaño de acuerdo con el tamaño del dataset, se hace el corte aplicando slicing directo con la ayuda de Numpy.

Para finalizar este paso, se imprimen los valores del tamaño de cada uno de estos nuevos datasets:

```
Dataset original: 208 registros
Train set: 166 registros (79.8%)
Test set: 42 registros (20.2%)
```

Figura 5. Información descriptiva de los datasets creados para la fase de entrenamiento y testeo.

Como último paso de la trasformación de los datos, se le agrega la columna de bias a los dos dataframes que se acaban de crear: *"df_train"* y "*df_test"*.

En este caso se agrega el bias ya que esto permite que el modelo tenga flexibilidad para ajustar su intercepción por la misma razón de que es un modelo de regresión lineal; además de que como se cuenta

con variables normalizadas, el bias captura la media original de la variable a predecir. Adicionalmente, añadir el bias ayuda a representar el rendimiento base de PRP cuando todas las características estén en sus valores promedio.

En seguida se muestra el dataset *"df_train"* ya con la feature de *"bias"* incluida:

	bias	6000_normalized	256.1_normalized	198_normalized
0	1	-0.325506	-0.643972	-0.441851
1	1	-0.836235	-0.430171	-0.534967
2	1	-0.751114	-0.483622	-0.379774
3	1	1.717409	-0.643972	1.066620
4	1	0.355466	1.066435	-0.032143

Figura 6. Dataframe *"df_train"* con la nueva columna *"bias"*.

5. CONSTRUCCIÓN Y EVALUACIÓN DEL MODELO

La construcción del modelo de regresión lineal multivariable consiste en varias implementaciones matemáticas como sigue:

5.1 Fundamento matemático

Anteriormente en el punto 3 se planteó la ecuación a usar en el modelo de regresión lineal multivariable:

$$PRP = \beta_0 + \beta_1 * MMAX + \beta_2 * CACH + \varepsilon$$

Ecuación 1. Modelo de regresión multivariable final.

Donde:

- PRP es el Rendimiento relativo publicado.
- β_0 es el bias o término de intercepto en la ecuación cuando todos los valores de las variables independientes son 0.
- MMAX es la memoria principal máxima en kilobytes.
- β_1 es el coeficiente de cambio para MMAX.
- CACH es la memoria caché en kilobytes.
- β_2 es el coeficiente de cambio para CACH.
- ε es el error residual del modelo, es decir, la diferencia no explicada entre el PRP y su valor predicho.

En el código se construye una matriz X que contiene una columna en 1's para el bias (θ_0) seguida de las columnas con las características normalizadas que se optimizarán durante el entrenamiento [θ_0, θ_1, θ_2].

5.2 Función de costo (MSE)

En el código se calcula manualmente mediante dos bucles anidados: el primero calcula las predicciones $\Sigma\theta_j x_j$ para cada muestra, y el segundo suma los errores cuadrados $\Sigma(y\_pred - y\_real)^2$ para obtener el MSE dividiendo entre m.

Esta es la fórmula en la que se basa este proceso en el código:

$$J(\theta) = (1/2m) * \Sigma(h\theta(x^{(i)}) - y^{(i)})^2$$

Ecuación 3. Ecuación de la función de costo utilizada.

Donde:

- $h\theta(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \ldots + \theta_n x_n^{(i)}$ es la predicción.
- m es el número de muestras.

5.3 Cálculo de gradientes

En el código, para cada parámetro θ_j, se acumula el producto $\Sigma(\text{error} * x_j)$ a través de todas las muestras.

Esta ecuación es la que es implementada exactamente en la fórmula del cálculo del gradiente:

$$\partial J/\partial \theta_j = (1/m) * \Sigma(h\theta(x^{(i)}) - y^{(i)}) * x_j^{(i)}$$

Ecuación 4. Ecuación del cálculo del gradiente.

Donde:

- ∂J es la derivada parcial de la función de costo J.
- $\partial \theta_j$ respecto al parámetro θ_j.
- m es el número total de muestras de entrenamiento.
- Σ es la suma desde $i=1$ hasta. $i=m$ (abarca todas las muestras).
- $h\theta(x^{(i)})$ es la predicción del modelo para la muestra i.
- $y^{(i)}$ es el valor real (target) de la muestra i.
- $x_j^{(i)}$ es el valor de la característica j en la muestra i.

5.4 Actualización de parámetros

En el código Cada parámetro *theta[j]* se actualiza restando el gradiente multiplicado por la tasa de aprendizaje: *theta[j] = theta[j] - alpha * gradientes[j]*.

Esto ejecuta el paso fundamental del descenso de gradiente.

La actualización de parámetros en el código se hace en base a la siguiente regla:

$$\theta_j = \theta_j - \alpha * (\partial J/\partial \theta_j)$$

Ecuación 5. Regla de actualización de parámetros.

Donde:

- α es la tasa de aprendizaje.

5.5 Normalización y desnormalización de datos

En el código las características se normalizan antes del entrenamiento. Para predicciones, se desnormaliza el resultado aplicando la transformación inversa para volver a la escala original.

La ecuación en la que se basa el proceso de normalización es la siguiente:

$$X_{normalizada} = \frac{X - \mu x}{\sigma x}$$

Ecuación 2. Ecuación utilizada para normalización de datos con Z-score.

Donde:

- μX es la media de la variable X.
- σX es la desviación estándar de la variable X.

Adicionalmente la ecuación en la que se basa el proceso de desnormalización es la siguiente:

$$Y_{real} = Y_{normalizada} * \sigma Y + \mu y$$

Ecuación 6. Ecuación utilizada para desnormalización de datos con Z-score.

Donde:

- $Y_{normalizada}$ es el valor predicho por el modelo en escala normalizada.
- σY es la desviación estándar de los valores reales de Y.
- μy es el valor promedio de los valores reales de Y.

5.6 Métrica R²

En el código se calcula manualmente comparando la suma de cuadrados residual (errores de predicción) contra la

suma de cuadrados total (variabilidad de los datos).

La manera de interpretar R^2 es que si su valor es cercano a 1 indica buen ajuste del modelo.

La ecuación para calcular este valor es la siguiente:

$$R^2 = 1 - (SS\_residual \,/\, SS\_total)$$

Ecuación 7. Ecuación utilizada para calcular el error cuadrático de los resultados del modelo.

Donde:

- $SS\_residual$ es el cálculo de $\Sigma(y\_real - y\_pred)^2$.
- SS_total es el cálculo de $\Sigma(y\_real - \bar{y})^2$.

5.7 Convergencia

En el código cada época se verifica el cambio máximo en los parámetros. Si todos los cambios son menores que la tolerancia (1e-6), el algoritmo se detiene, indicando que encontró un mínimo local de la función de costo.

Dentro del algoritmo, el modelo se detiene cuando:

$$max(|\theta_j^{(k+1)} - \theta_j^{(k)}|) < tolerancia$$

Ecuación 7. Validación del valor de convergencia implementado en el modelo.

Donde:

- $\theta_j^{(k)}$ es el valor del parámetro *j* en la iteración actual.
- $\theta_j^{(k+1)}$ es el valor del parámetro *j* después de la actualización.
- $|\theta_j^{(k+1)} - \theta_j^{(k)}|$ representa la magnitud del cambio en el parámetro *j* entre iteraciones.
- max () es la operación que asegura que todos los parámetros hayan convergido.
- *tolerancia* es el umbral de convergencia, en este caso 1e-6 (0.000001).

6. VISUALIZACIÓN Y ANÁLISIS DE RESULTADOS

6.1 Output del modelo

Después de que el código acaba de ejecutarse, se imprime la siguiente información en la terminal:

```
RESULTADOS FINALES:
====================================
Épocas ejecutadas: 1000
Parámetros finales:
  Bias (θ₀) = 0.003318
  Coef MMAX (θ₁) = 0.727416
  Coef CACH (θ₂) = 0.282962
Error MSE final = 0.223249

EVALUACIÓN DEL MODELO:
====================================
Error MSE training: 0.223249
Error MSE test: 0.095699

R² en training: 0.804321

EJEMPLO DE PREDICCIÓN:
====================================
Para MMAX=8000, CACH=512:
PRP predicho: 661.93
```

Figura 7. Resultados finales al ejecutar el modelo de regresión.

6.2 Interpretación del output del modelo

A continuación, se hace la interpretación de cada uno de los parámetros calculados en el modelo:

- **Bias (θ_0) = 0.003318**: Este parámetro indica que el rendimiento base es prácticamente 0 cuando los parámetros se encuentran en su media (que es 0 ya que están normalizados).
- **Coef MMAX (θ_1) = 0.727416**: Este parámetro indica que memoria máxima es el predictor más poderoso del rendimiento en

el modelo, a comparación con los demás parámetros.

- **Coef CACH (θ_2) = 0.282962**: Este parámetro, por su valor, es el parámetro secundario que contribuye a las predicciones hechas en el modelo.

En comparación, se puede sacar la importancia de un parámetro con respecto a otro, por ejemplo:

$$0.727416 / 0.282962 \approx 2.57$$

Ecuación 8. Valor numérico de importancia de la variable MMAX contra CACH.

Este valor indica que la variable MMAX es 2.57 veces más importante que la variable CACH.

Ahora, se interpretan los valores del rendimiento del modelo:

- **Error MSE Training = 0.223249**: Este error es moderado en los datos de entrenamiento considerando que la desviación estándar de PRP es de 1 en escala normalizada.
- **Error MSE Test = 0.095699**: Este error indica que el modelo tiene una excelente generalización, ya que el error sí se aproxima a cero. Esto también es un indicador de que el modelo funciona mejor con datos nuevos que con los datos de entrenamiento.
  Algunas de las posibles razones por las cuales acontezca esto son:
  - Una división favorable del dataset.
  - El training set podría tener más variabilidad.
  - En general, hay un buen equilibro entre bias y varianza en los datos.
- **R^2 = 0.804321**: El valor mostrado indica que el modelo tiene un muy buen poder predictivo, contando con 80.43% de accuracy.
- **Ejemplo de predicción**: Para MMAX=8000, CACH=512, el PRP predicho fue de 661.93, la cual es una predicción acorde con los recursos altos provenientes de las variables independientes.

6.3 Cálculo de errores en unidades originales

Para calcular el error en unidades originales se implementó el siguiente código:

```
# Error en unidades ORIGINALES:
error_train_real = np.sqrt(error_train) * 100
error_test_real = np.sqrt(error_test) * 100
print(error_train_real)
print(error_test_real)
```

Figura 8. Cálculo de error en unidades originales (desnormalizadas), en Python.

Como resultado, imprime los siguientes valores:

```
47.24923779950612
30.93533018973207
```

Figura 9. Resultado de calcular el error MSE en unidades reales.

En este output, el primer valor hace referencia a que el valor predicho de PRP puede tener un valor de ±47.2 unidades reales de PRP en la parte del entrenamiento.

El segundo valor hace referencia al valor predicho de PRP puede tener un valor de ±30.9 unidades de PRP reales en la parte del testing.

Al interpretar estos valores, se puede decir que el modelo generaliza mejor de lo que aprende, por lo que se considera un modelo con “Overfitting”.

6.4 Visualización de valores predichos

A continuación, se muestra una grafica en donde se hace la comparación de los valores reales contra los valores predichos por el modelo:

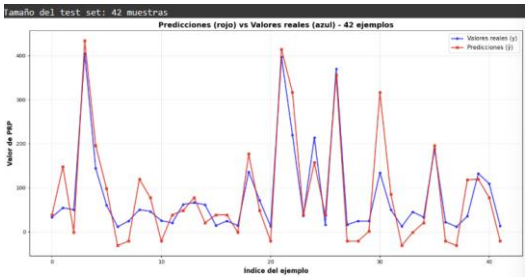

Figura 10. Predicciones (rojo) vs Valores reales (azul) con una muestra de 42 ejemplos.

En este gráfico se puede observar el fenómeno de “Overfitting” anteriormente señalado, ya que aunque se ajusta de manera considerable en varios puntos, existen otros en donde las predicciones son exageradamente grandes o pequeñas a comparación de su valor real.

6.5 Conclusiones

El modelo hecho cuenta con varios puntos a favor como los son:

- Cuenta con una excelente generalización (ya que el valor de MSE_test < MSE_train).
- Tiene un alto poder predictivo ($R^2 = 0.804$).
- Tiene coeficientes intuitivos (ambos positivos, MMAX es el más relevante).
- Tiene un balance de bias-varianza óptimo

Así mismo hay puntos a considerar, como la diferencia en el error MSE en train-test, ya que se tendría que hacer una investigación acerca del mejor rendimiento en test y cómo se puede arreglar; además de incluir otra variable dentro de las variables independientes del modelo para poder explicar el 19.57% de variabilidad todavía no explicada dentro del modelo.