Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Inteligencia Artificial II (CI5438)

Modelo de Regresión Lineal

Estudiantes:

Giacomo Tosone 14-11085

Rommel Llanos 15-10789

Profesor: Carlos Infante

Contenidos

Sobre el problema

El problema a tratar en este informe consiste en crear una función la cual permita predecir el precio de un automóvil basándose en la data de entrenamiento entregada, asimismo también se deben de tratar las técnicas para la limpieza y ajustes pertinentes de la data de prueba a utilizar.

En vista de esto se tienen 3 problemas fundamentales que se deben resolver y los cuales estaremos profundizando en este informe, estos son:

- Explorar y limpiar la data a trabajar.
- Crear las funciones de regresión lineal y de descenso gradiente.
- Probar el modelo con diferentes porcentajes de la data y ajustando sus parámetros.

Sobre la instalación.

A continuación, se presenta en paso a paso el proceso de instalación de los paquetes necesarios para el funcionamiento de los scripts realizados en python:

- Se requiere tener Python >= 3.11
- Se requieren los siguientes módulos:
    - Pandas: pip3 install pandas
    - Matplotlib: pip3 install matplotlib
    - Sk-learn: pip install -U scikit-learn
    -

Por último el script a correr es el archivo **main.py** ubicado en la carpeta /src

Sobre los sets de datos

El set de datos a trabajar es el archivo **CarDekho.csv,** en el cual se encuentra la data de prueba la cual consiste en información referente a diferentes modelos de autos, así como información referente a su precio, marca, modelo, tipo de gasolina, cantidad de asientos disponibles, la cantidad de dueños previos, kilometraje, entre otros valores, en particular las columnas presentes tienen el siguiente formato:

```
In [50]: df.dtypes

Out[50]: Make                 object
         Model                object
         Price                 int64
         Year                  int64
         Kilometer             int64
         Fuel_Type            object
         Transmission         object
         Location             object
         Color                object
         Owner                object
         Seller_Type          object
         Engine               object
         Max_Power            object
         Max_Torque           object
         Drivetrain           object
         Length              float64
         Width               float64
         Height              float64
         Seating_Capacity    float64
         Fuel_Tank_Capacity  float64
```

Ejemplo 1: Formato del Dataset

Esta data presenta mucha información útil que puede ser utilizada como base para entrenar la función de regresión lineal, pero para esto la data necesita ser limpiada y adaptada a un formato completamente numérico.Por último antes de comenzar a tratar la data se requiere dividirla en solo las columnas que se desean utilizar, para este trabajo se van a ocupar las siguientes columnas:

- Price.
- Make.
- Year.
- Kilometer.
- Fuel_Type.
- Transmission.
- Owner.
- Seating_Capacity.

- Fuel_Tank_Capacity.

Adicional a esto se quería trabajar con **Seller_type**, sin embargo solo un 2% de la data tenía un valor diferente al mas comun, por lo cual la distribución de la data no era lo suficientemente variada para considerar que pueda tener un impacto en el modelo.

```
In [55]: df["Seller_Type"].value_counts()

Out[55]: Seller_Type
         Individual                1997
         Corporate                   57
         Commercial Registration      5
         Name: count, dtype: int64
```

Ejemplo 2: Distribución de Seller_type.

Parte 1: Valores nulos

Ya teniendo los valores a utilizar se requiere hacer un estudio de la data para encontrar valores nulos en esta y así eliminarlos o reemplazarlos, para este trabajo todos los valores nulos presentes van a ser reemplazados por la mediana de la columnna o por el valor más común. En particular las columnas que presentaban valores nulos son Seating_Capacity con 64 valores nulos y Fuel_Tank_Capacity con 113 valores nulos. En el caso de Seating_Capacity el valor del medio es de 5 asientos y este es adicionalmente el valor más común por lo cual los valores nulos se reemplazan por 5.0, para el caso de Fuel_Tank_Capacity el valor del medio es de 50 litros pero el valor más común es 35 litros, por lo cual se opta por usar 35 litros para rellenar los valores nulos.

```
dfx.info()
```

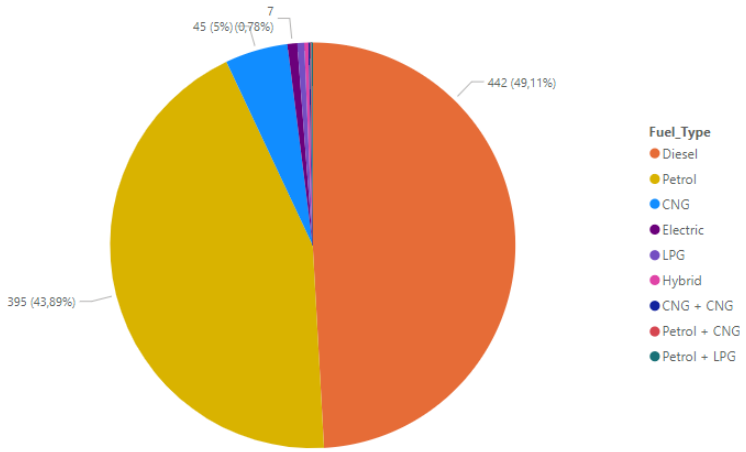
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2059 entries, 0 to 2058
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Make                2059 non-null   object
 1   Year                2059 non-null   int64
 2   Kilometer           2059 non-null   int64
 3   Fuel_Type           2059 non-null   object
 4   Transmission        2059 non-null   object
 5   Owner               2059 non-null   object
 6   Seating_Capacity    1995 non-null   float64
 7   Fuel_Tank_Capacity  1946 non-null   float64
 8   Price               2059 non-null   int64
dtypes: float64(2), int64(3), object(4)
memory usage: 144.9+ KB
```

Ejemplo 3: Valores nulos de la data

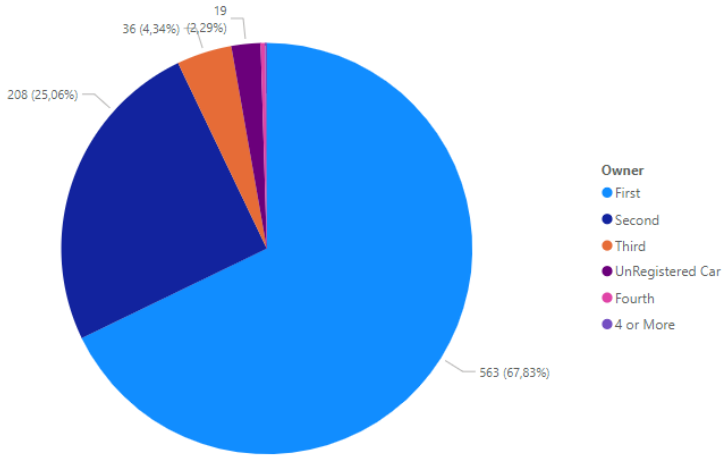
Parte 2: Transformación de columnas.

Ya que la data no presenta valores nulos, el siguiente paso es el de convertir las columnas con valores no numéricos en valores numéricos, para esto se van a adoptar dos diferentes estrategias, para las columnas con pocos valores serán transformadas en n columnas binarias donde 1 representa que presenta el valor y 0 que no, en particular.

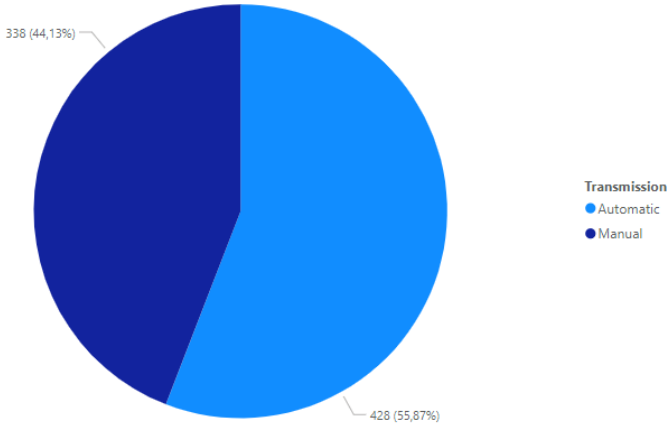
- Fuel_Type: Presenta 9 valores distintos donde Petrol y Diesel ocupan el 95% de la data, por lo cual se agrupan los otros 7 valores en la columnas Others_F la cual solo representa el 5% de la data.



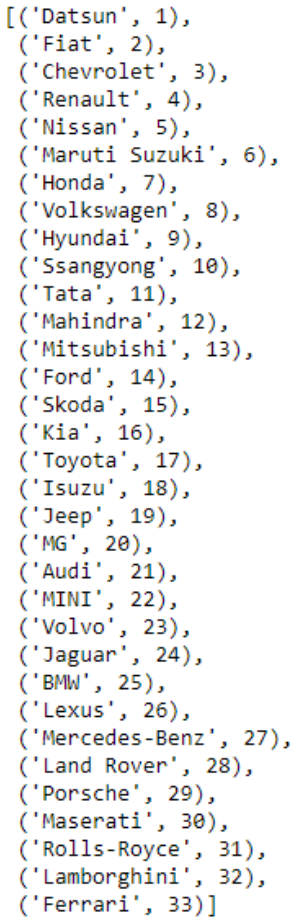
- Owner: Presenta 6 valores de los cuales First y Second representan el 92% de la data, por lo cual los otros 6 valores son agrupados en una tercera columna categórica llamada Others_O, la cual solo representa el 8% de la data.



- Transmission: Este valor representa el tipo de caja del vehículo y solo posee dos valores, siendo Automatic el más común, por lo cual las columnas nuevas serán Automatic y Manual.



- Make: Esta columna representa la marca del vehículo y posee 33 valores posibles, para esta variable se quiere tomar un enfoque diferente y englobar en una sola columna, para esto a cada posible valor se le asigna un entero y se reemplazan los valores en string por dichos enteros en el set de datos, si embargo esto puede generar un sesgo en la data por lo cual para mitigar esto calculó el promedio de precio de cada marca en el dataset y se fueron enumerando de menor a mayor de manera que la marca con el precio promedio más baja sea el 1 y la marca con el promedio más alto es la 33.



Ejemplo 4: Valores de Make en base al promedio

Parte 3: División de la data

Por último el set de datos fue dividio de diferentes maneras, se tiene el archivo clean_data.csv donde se presenta toda la data a utilizar, los archivos 80_data.csv y 20_data.csv los cuales representan el 80% y el 20% de la data limpiada y por último se encuentran 5 archivos adicionales de data donde cada uno representa el 20% de la data a trabajar, todos estos archivos se usan para realizar diferente pruebas e el modelo, en particular:

- clean_data.csv: Se usa para realizar pruebas usando toda la data, este parece ser el método simple que mejor resultados arroja.
- 80 y 20: Estos se usan para realizar pruebas sobre un porcentaje particular de la data pero los resultados tienden a ser peores que usando toda la data.
- División de data en 5: Estos últimos se usan para realizar pruebas a la función por medio del método K_Fold, para hacer la división se utiliza una

librería externa, este último es el método que mejor funciona al hacer muchas iteraciones (500.000) pero se requiere un alfa más bajo que en los casos anteriores.

Sobre las funciones

Para poder aproximar el precio de un vehículo por medio del método de regresión lineal, se necesitan 2 funciones principales así como una cantidad de funciones secundarias las cuales algunas sirven como apoyo directo de las 2 funciones principales y otras sirven como funciones de apoyo para poder probar dichos resultados, los detalles de estas se encuentran en el código debidamente documentados en los archivos main.py y gradient_descent.py siendo main.py el script a correr para poder realizar pruebas y obtener los resultados/gráficas correspondientes.

No profundizaremos en los detalles de cómo funcionan cada función pero sí hablaremos de ciertos detalles de diseño de esta y mostraremos los resultados en otro módulo de este informe.

Regresión lineal

Para esta función se tiene que es de forma lineal donde cada variable en la data a modelar es multiplicado por un peso el cual denota que porcentaje de este valor es importante a la hora de predecir un precio, por lo cual la suma de cada atributo multiplicado por sus pesos es lo que determina el precio desestimado de un vehículo, una consideración a tener en cuenta es que se trabajan con los valores normalizados, por lo cual la data que es pasada a la función es previamente normalizada, adjunto a esto se presentan como variables de entrada de la función gradiente los valores de iteraciones, alpha y epsilon, por lo cual estos pueden ser cambiados entre interacciones.

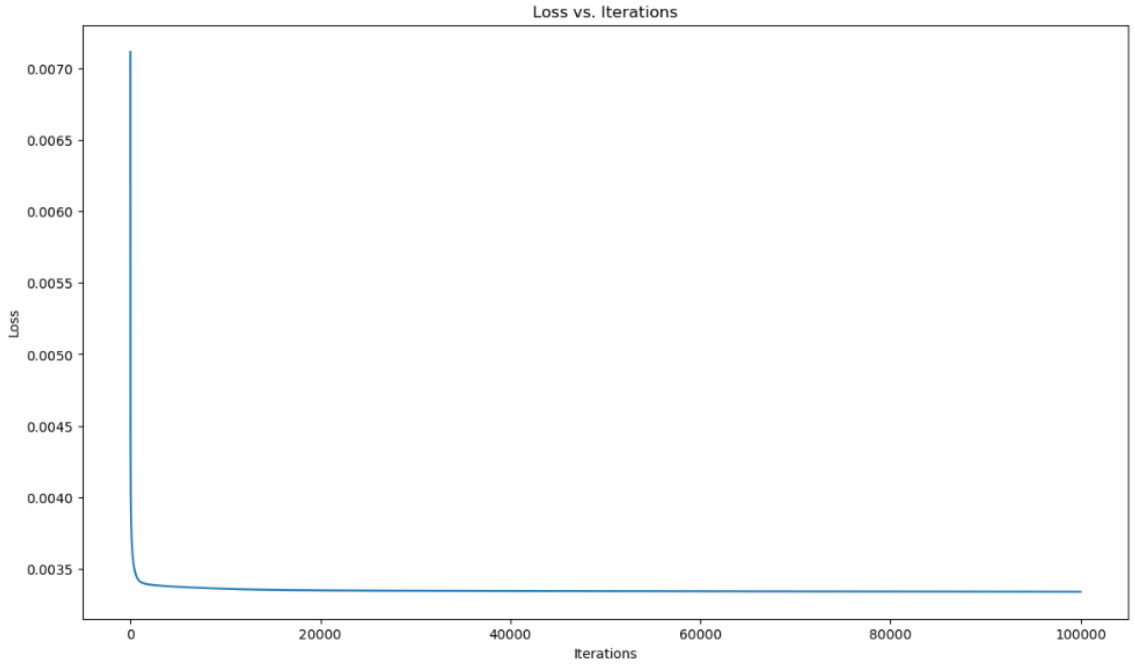
```
w = gradientDescent(X,y,iterations=5000, alpha=0.01, epsilon=2e-3)
```

Descenso Gradiente:

Esta función es la que se encarga de actualizar los pesos en base al epsilon dado, en cada una de las iteraciones de esta (siendo las iteraciones un valor de entrada) este calcula la pérdida de cada columna y procede a actualizar los pesos en base a estos hasta que dicha pérdida es menos al epsilon, punto en el que la función converge y se terminan las iteraciones, si la función converge o terminan el

número de interacciones esta retorna los pesos calculados y con estos se puede armar la función de regresión lineal para predecir predecir precios de vehículos, si la función termina al acabarse el número de interacciones esta es propensa a tener cierto margen de error, en las pruebas realizadas el margen de error más bajo que se pudo obtener era cerca al 50% en promedio.

Otro punto a resaltar es que se trataron 2 maneras diferentes de realizar estas pruebas, una de ellas consiste en calcular el modelo de pruebas y revisar por medio de unas funciones auxiliares la predicción del modelo obtenido sobre la data que se utiliza, si alguno de estos valores su predicción era 4 veces o más del precio original entonces dicha fila era eliminada del modelo de datos, luego de hacer esta depuración de data se realiza un modelo nuevo sin la data “problemática” dicha técnica podia mejor en un 5% las predicciones realizadas.



Ejemplo 5: Gráfica resultante de un entrenamiento

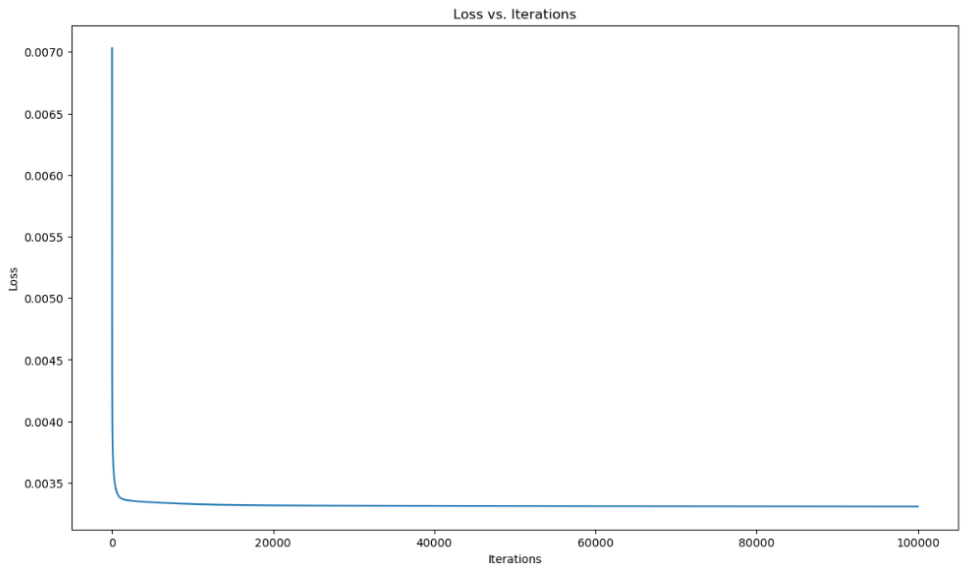
El otro método a utilizar es el uso del K_Fold sobre 5 “chunks” de la data los cuales se usan para entrenar el modelo, esta técnica genera los modelos con menor promedio pero dicho método requiere un alfa bajo (0,0001) y un número de iteraciones alto por encima de los 300.000.

```python
#Valores de cada conjunto
scores = []
weights = []

kf = KFold(n_splits=5, shuffle=False )
kf.get_n_splits(X)
for train_index, test_index in kf.split(X):
    # Se divide la data en conjunto de entrenamiento y de prueba
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Se entrena el modelo
    w = gradientDescent(X_train, y_train, iterations=100000, alpha=0.0001, epsilon=2e-4)
    weights.append(w)

    #Se evalua el modelo
    predictions = predict(X_test,w)

    #Calcula el puntaje de cada conjunto
    s = score(y_test, predictions)
    scores.append(s)
```

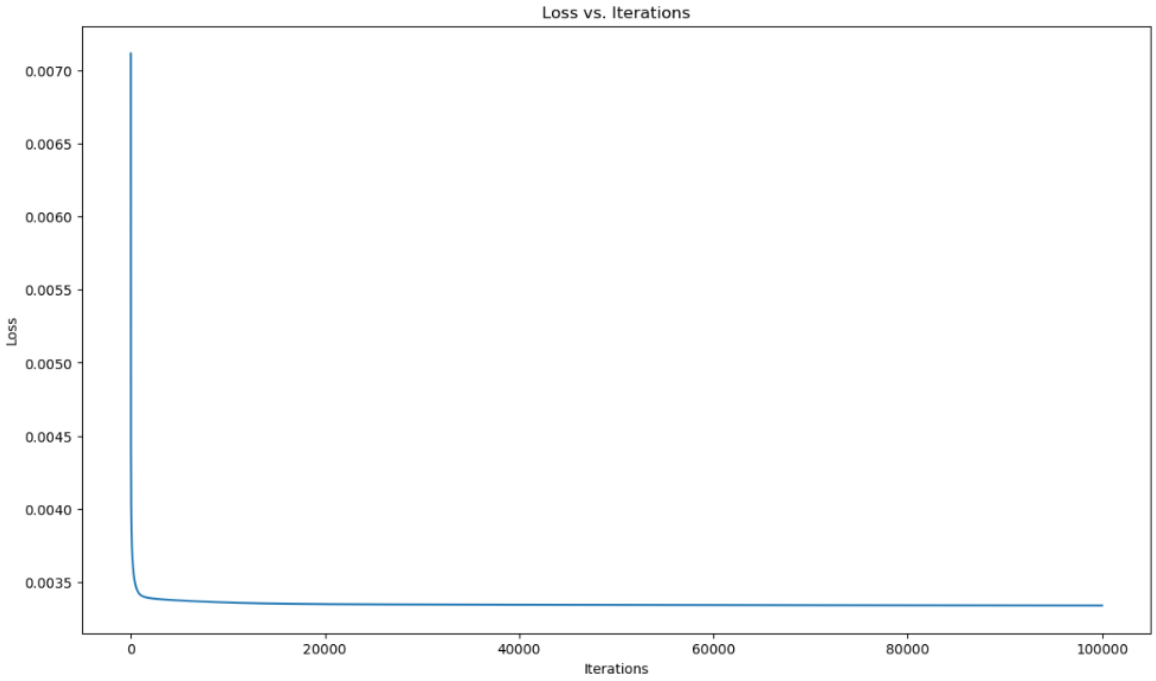
*Ejemplo 6: Función de K_Fold para descenso gradiente***Problemáticas Encontradas**

Resultados obtenidos

En base a las pruebas realizadas se obtuvo los siguientes resultados de los modelos entrenados:
En base a utilizar toda la data el mejor modelo encontrado era el resultante de utilizar un alfa de 0,01 y 100.000 iteraciones el cual daba como resultado los siguientes valores.



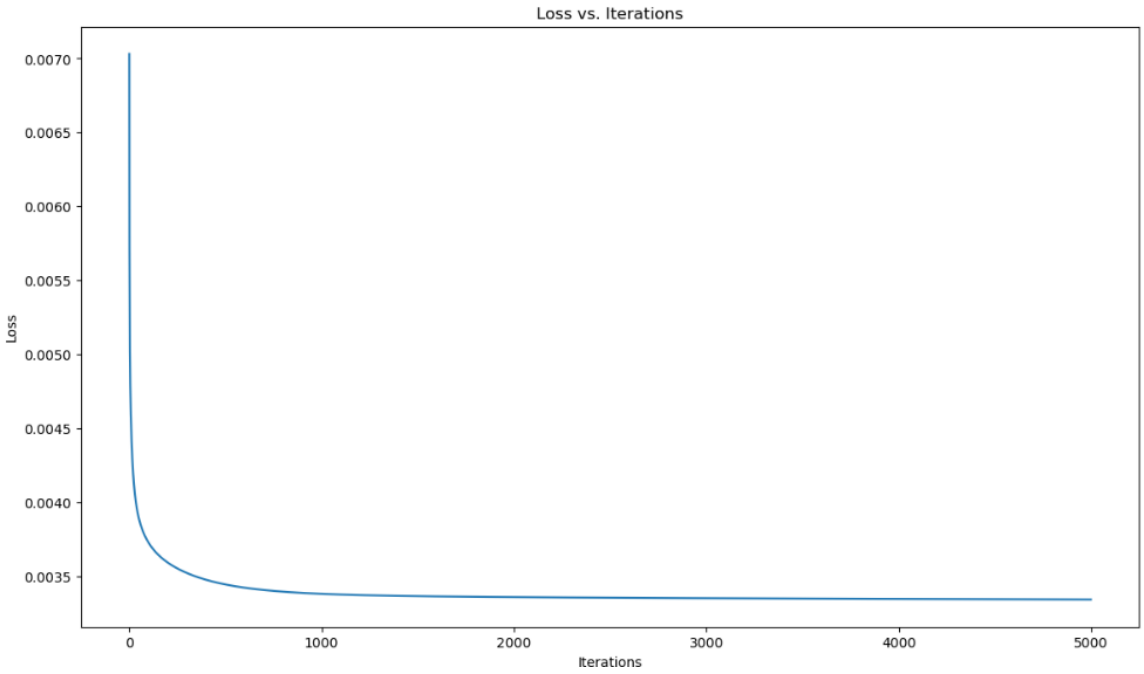
Este resultado los precios que aproxima en promedio tienen una desviación de un 59%, en base a este modelo se calcularon los precios aproximados cuyos valores sean al menos un 300% más grandes que el precio real y se eliminaron del modelo de datos, esto hizo que se eliminaran 23 filas del modelo, este nuevo set de datos fue utilizado para volver a entrenar otro modelo con un alfa de 0,001 y 1000.000 iteraciones, y su resultado fue el siguiente.



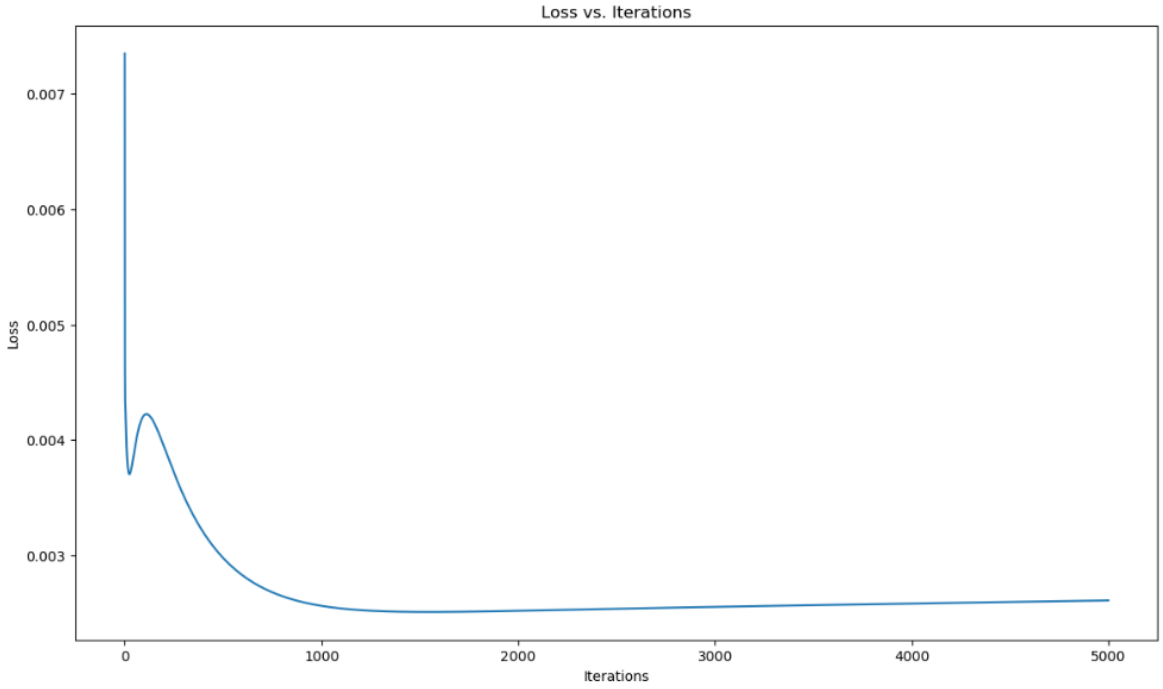
Este nuevo modelo resultante en promedio sus valores aproximados se desvían en un 54% lo cual es una mejora del 5% frente a la primera iteración.

Con el método del K_Fold también se consiguen resultados similares, cuyo modelo resultante de hacer 500.000 iteraciones con un alfa de 0,0001 arroja que en promedio las predicciones de precios se desvían en un 56% frente al valor real.

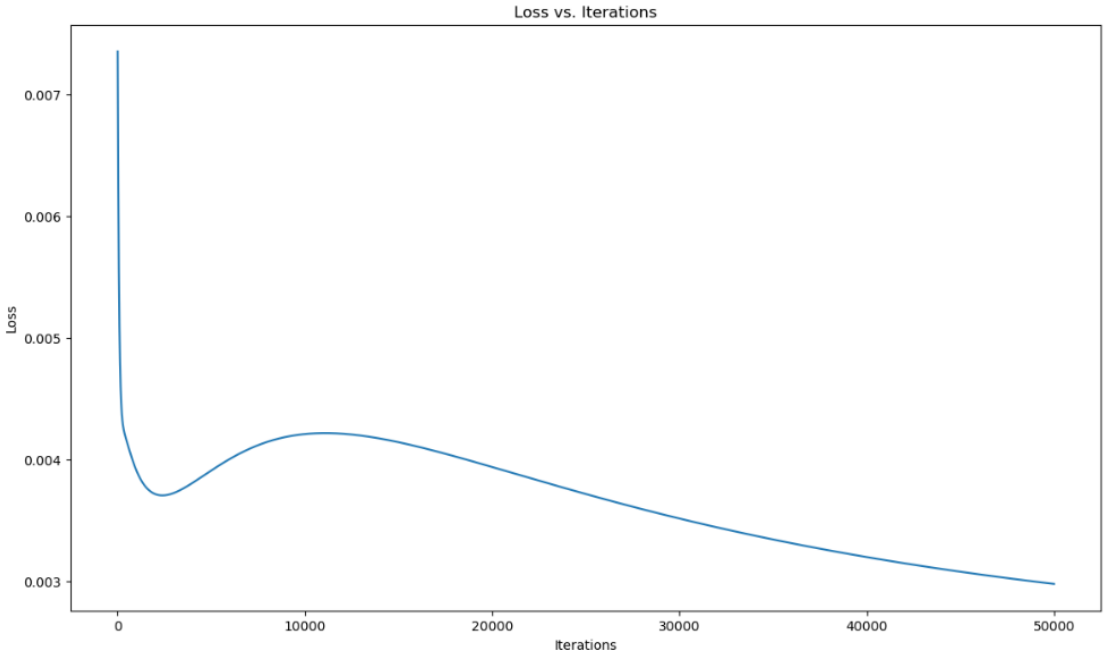
También se realizaron pruebas sobre los diferentes “chunks” utilizando un número bajo de iteraciones, en base a toda la data el modelo resultante presenta en promedio una desviación de la predicción de un 59% teniendo sin embargo los valores medios entre un 31% y 26%.



Utilizando el 80% de la data con un alfa de 0,01 y 5.000 iteraciones, se obtiene un modelo que tiene en promedio un error de predicción del 141% siendo esta su gráfica resultante



En base a esta misma data se quiere repetir el experimento pero con un alfa de 0,0001 y 50000 iteraciones, el promedio de error en la predicción de este nuevo modelo es de un 121% teniendo valores en la mediana de 112% y un 113%, por lo cual al bajar un 20% la cantidad de data, el modelo es mucho menos exacto a pesar de usar un alfa más pequeño, por ultimo su gráfica resultante es la siguiente.



Conclusiones

En base a los resultados obtenidos se pueden ver varias prácticas útiles a seguir a la hora de entrenar un modelo, en primer lugar es muy importante tener el mayor volumen de data posible, al pasar de utilizar un modelo con 2000 dato a uno con 1600 datos el la precisión del modelo decayó a la mitad a pesar del ajuste del alfa.

Otra enseñanza importante es la eliminación de datos anormales en el modelo, el modelo al eliminar solo 23 filas problemáticas mejora su precisión en un 5%, esto es eliminar un 1% de la data para obtener un 5% de precisión en este caso particular.

Un último resultado interesante es la escogencia del alfa y las iteraciones, tener un alfa pequeño es muy importante a la hora de tratar con data de menor tamaño, pero este tiene que venir de la mano del aumento del número de iteraciones, en el caso del k_fold que trataba con conjuntos de 400 valores, su precisión aumentó un 20% al disminuir el alfa y subir el número de iteraciones de 5.000 a 500.000, mientras que cuando se trata con toda la data no se consigue mucha diferencia al pasar de 5.000 a 500.000 iteraciones.