Tito Alfarabi Biwarno

1103213012

Hasil dari kode yang digunakan dan dataset adalah NaN karena beberapa hal yang ada. Penjelasannya sebagai berikut.

1. Data Tidak Lengkap atau Tidak Valid

- Dataset **imports-85.data** memiliki nilai `?`, yang secara otomatis diubah menjadi **NaN** selama proses pembacaan data.
- Jika nilai **NaN** tidak sepenuhnya dibersihkan atau diimputasi, nilai ini akan masuk ke dalam perhitungan model, yang menyebabkan propagasi error dan menghasilkan **NaN** pada loss atau prediksi.

Penjelasan Teknis: Operasi matematika seperti penjumlahan atau perkalian dengan **NaN** menghasilkan **NaN**, sehingga model tidak dapat belajar dengan benar.

Bukti: Gunakan `auto_data.isnull().sum()` untuk memverifikasi jumlah nilai yang hilang di setiap kolom.

2. Rentang Nilai yang Tidak Konsisten (Skala Tidak Normal)

- Dataset ini mencakup fitur numerik dengan rentang nilai yang sangat besar dan kecil, seperti panjang kendaraan (dalam inci) dan harga kendaraan (dalam ribuan dolar).
- Target (harga) juga memiliki variasi besar, dari nilai kecil hingga puluhan ribu dolar.

Penjelasan Teknis: Model neural network sangat sensitif terhadap skala data. Jika skala data tidak konsisten, gradien yang dihitung selama pelatihan dapat menjadi terlalu besar (exploding gradient) atau terlalu kecil (vanishing gradient), menyebabkan nilai parameter menjadi tidak stabil.

Bukti: Gunakan `auto_data.describe()` untuk memeriksa rentang nilai setiap fitur dan target.

3. Overfitting Akibat Learning Rate yang Tidak Sesuai

- Learning rate yang terlalu besar membuat model memperbarui bobot dengan langkah-langkah yang terlalu besar, sehingga menyebabkan parameter model menjadi tidak stabil.
- Hal ini memicu exploding gradient dan nilai parameter yang ekstrem.

Penjelasan Teknis: Gradien yang besar membuat loss function menjadi tidak terdefinisi (overflow), yang langsung menghasilkan nilai **NaN**.

Bukti:

- Jika loss menjadi **NaN** dalam beberapa iterasi awal, ini adalah tanda learning rate terlalu besar.
- Coba gunakan learning rate yang lebih kecil, seperti `0.001`.

4. Masalah Distribusi Data (Outlier atau Ketidakseimbangan)

- Beberapa fitur atau target mungkin memiliki outlier yang ekstrem, yang mendistorsi pembelajaran model.
- Misalnya, harga kendaraan yang jauh lebih tinggi dibandingkan sebagian besar data dapat menyebabkan model berfokus hanya pada outlier tersebut.

Penjelasan Teknis: Outlier dalam target (harga) menyebabkan nilai loss menjadi sangat besar, sehingga optimisasi tidak berjalan dengan baik dan nilai loss atau gradien menjadi **NaN**.

Bukti: Gunakan histogram atau boxplot untuk melihat distribusi data, misalnya:

```
import matplotlib.pyplot as plt
plt.hist(y, bins=30)
plt.show()
```

5. Tidak Ada Normalisasi pada Fitur atau Target

- Fitur numerik dan target tidak dinormalisasi ke skala yang sama. Sebagai contoh, beberapa fitur memiliki rentang dalam inci, sedangkan harga berada dalam ribuan dolar.

Penjelasan Teknis: Tanpa normalisasi, model akan kesulitan untuk menentukan bobot yang sesuai untuk setiap fitur. Fitur dengan skala besar mendominasi pembelajaran, menyebabkan gradien meledak atau underflow.

Bukti: Gunakan normalisasi seperti `StandardScaler` untuk memastikan semua fitur berada dalam skala serupa.

Kesimpulan Utama

Hasil **NaN** terjadi karena kombinasi dari:

1. **Nilai hilang (NaN)** dalam dataset.
2. **Rentang nilai fitur dan target yang tidak konsisten**.
3. **Learning rate yang terlalu besar**, menyebabkan exploding gradient.
4. **Distribusi data yang tidak seimbang**, dengan kemungkinan outlier yang signifikan.