**Bloom Filter Benchmarking:**

Benchmarking refers to the performance evaluation of the bloom filter under certain conditions.

**In this Benchmarking evaluation, we have included:**

* How the insertion time differs with increasing number of inputs, exceeding the expected number.
* How the search time differs with increasing number of inputs, exceeding the expected number.
* How the false positive rate differs with increasing number of inputs, exceeding the expected number.
* How the compression rate differs with increasing number of inputs, exceeding the expected number.

1. **How the insertion time differs with increasing number of inputs, exceeding the expected number:**

The insertion time is plotted against the increasing number of inputs. As the size of the input increases, the insertion time grows linearly. This is expected since each insertion requires computing multiple hash functions and updating the bit array. The efficiency of insertion of the Bloom filter for using large data tasks is accomplished by the fact that the insertion time for 1,000,000 elements is taken approximately 1.75 seconds. This gives an understanding for the performance of the bloom filter insertion for large datasets.

A group of graphs with numbers

Description automatically generated with medium confidence

1. **How the search time differs with increasing number of inputs, exceeding the expected number:**

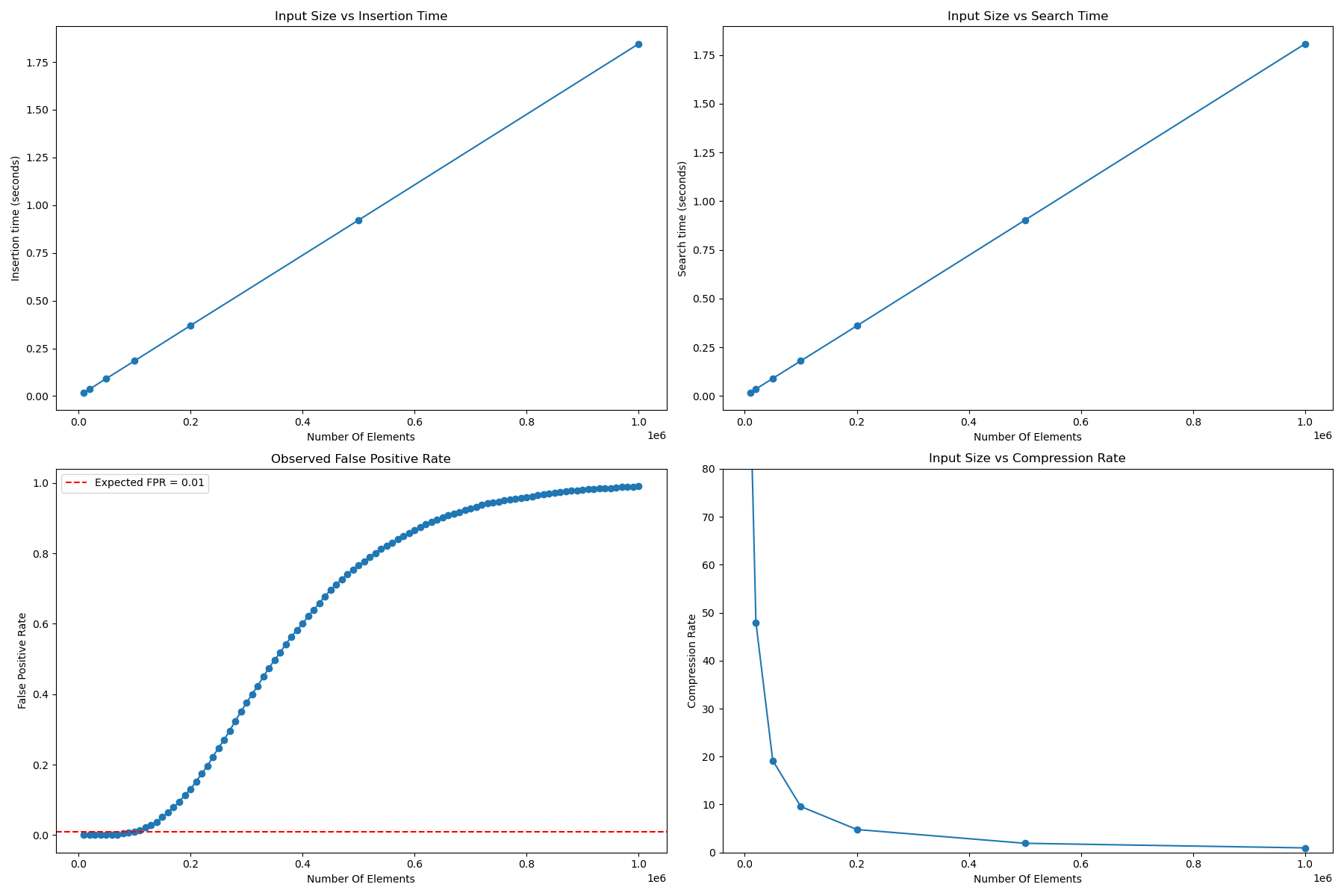
Just like the benchmarking done for insertion, the search times grows linearly with the increasing number of elements. The results for search operation is similar to the insertion time which is 1.75 seconds for an input size of 1000000. This ensures the Bloom filter's efficiency for large-scale data operations.

**A group of graphs with numbers

Description automatically generated with medium confidence**

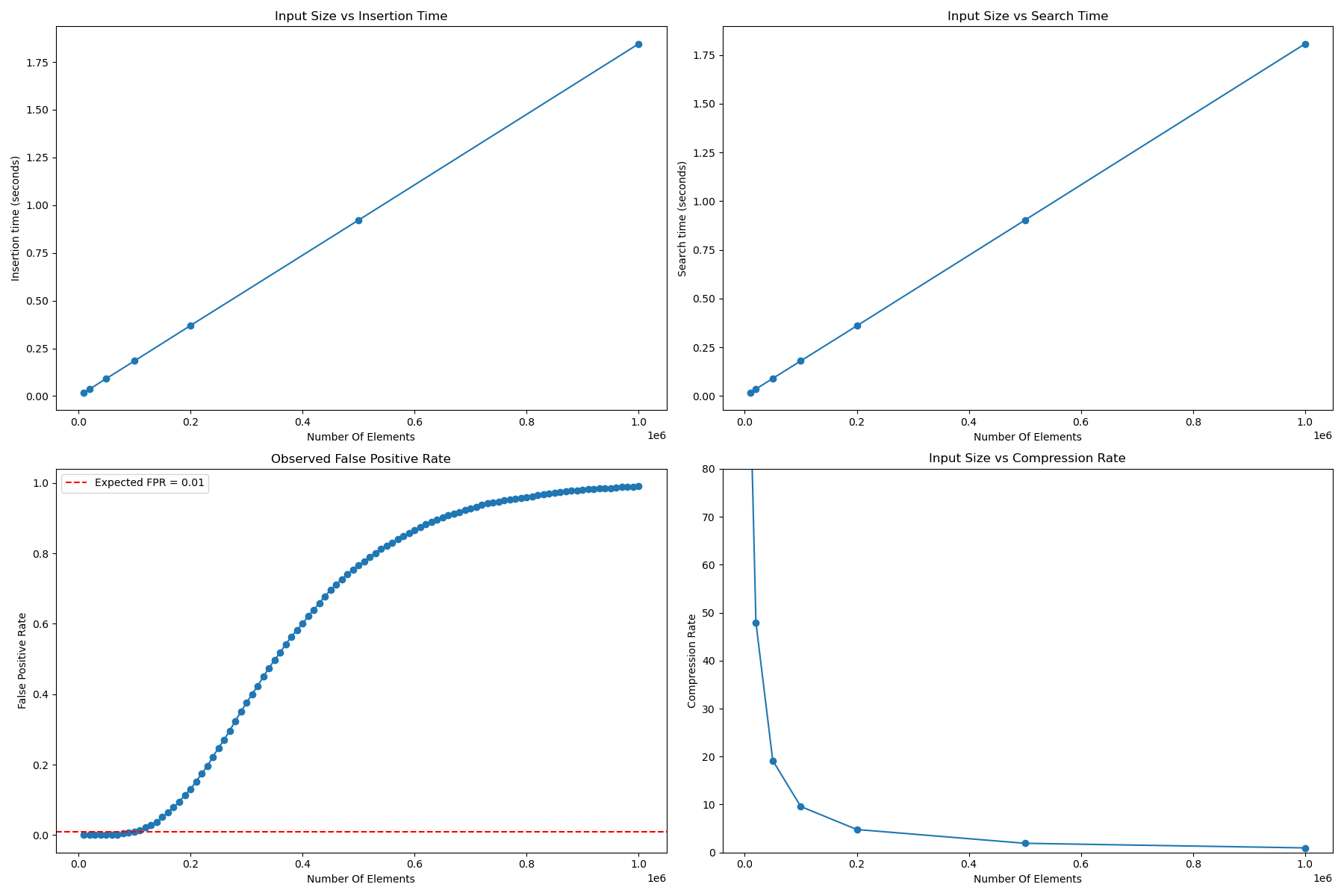
1. **How the false positive rate differs with increasing number of inputs, exceeding the expected number:**

As the number of elements approaches the capacity of the Bloom filter, the false positive rate increases more exponentially. This is due to a higher number of bits being set to 1, increasing the chance that a random element will hash to a set of bits that are all 1s. This is illustrated in the figure below.

****

1. **How the compression rate differs with increasing number of inputs, exceeding the expected number:**

The compression rate decreases with increasing number of inputs.

****