First we give an algorithm that produces a subgraph whose expected number of edges has the desired value. For this, we simply choose k nodes uniformly at random from G. Now, for $i < j$, let X_{ij} be a random variable equal to 1 if there is an edge between our i^{th} and j^{th} node choices, and equal to 0 otherwise.

Of the $n(n-1)$ choices for i and j, there are $2m$ that yield an edge (since an edge (u, v) can be chosen either by picking u in position i and v in position j, or by picking v in position i and u in position j). Thus $E[X_{ij}] = \frac{2m}{n(n-1)}$.

The expected number of edges we get in total is

$$\sum_{i<j} E[X_{ij}] = \binom{k}{2} \cdot \frac{2m}{n(n-1)} = \frac{mk(k-1)}{n(n-1)}.$$

We now want to turn this into an algorithm with expected polynomial running time, which always produces a subgraph with at least this many edges. The analogous issue came up with MAX 3-SAT, and we use the same idea here: For this we use the same idea as in the analogous MAX 3-SAT: we run the above randomized algorithm repeatedly until it produces a subgraph with at least the desired number of edges.

Let p^+ be the probability that one iteration of this succeeds; our overall running time will be the (polynomial) time for one iteration, times $1/p^+$. First note that the maximum number of edges we can find is $e = \frac{k(k-1)}{2}$, and we're seeking $e' = e \cdot \frac{2m}{n(n-1)}$. Let e'' denote the greatest integer strictly less than e'. Let p_j denote the probability that we find a subgraph with exactly j edges. Thus $p^+ = \sum_{j \geq e'} p_j$; we define $p^- = \sum_{j<e'} p_j = 1 - p^+$. Then we have

$$\begin{aligned}
e' &= \sum_j jp_j \\
&= \sum_{j<e'} jp_j + \sum_{j \geq e'} jp_j \\
&\leq \sum_{j<e'} e''p_j + \sum_{j \geq e'} ep_j \\
&= e''(1-p^+) + \binom{k}{2} p^+
\end{aligned}$$

from which it follows that

$$(e'' + \binom{k}{2})p^+ \geq e' - e'' \geq \frac{1}{n(n-1)}.$$

Since $e'' \leq \binom{k}{2}$, we have $p^+ \geq \frac{1}{k(k-1)n(n-1)}$, and so we are done.

[1] ex553.136.7