Before jumping on to the questions asked, some results are depicted below :

	Positive	Negative	Total
General	**364**	1277	1641
Price	90	1726	**1816**
Safety	68	**1747**	1815
Transit Location	140	1632	1772

Table 1. Correctly labelled instances

Aspect	Count
General	**238**
Price	63
Safety	64
Transit Location	107

Table 2. Incorrectly labelled instances

Answer 1.

As highlighted in table1, our model most accurately detects :

1. **General** aspect with positive sentiment.
2. **Safety** aspect with negative sentiment.
3. **Price** aspect, all in all.

Answer 2.

As highlighted in table2, our model performs poorly on :

1. **General** aspect.

Answer 3.

According to me, every machine learning library has its own advantages and disadvantages. In cases where there is a need to develop quick solutions to problems in a shorter duration of time, I use Keras. However, internal modifications and optimizations are difficult to carry out in Keras models. In cases when optimization and research is key to the problem, I use PyTorch. In some cases when I want to have clean graphical solutions to problems, when it is easier to visualize the network as a graph, and crisp inputs can be defined, I use Tensorflow. Comparing Tensorflow and PyTorch, I found there are less sophisticated tools for production and deployment in case of PyTorch. **I personally prefer working with Tensorflow**, because of the crisp input parameters it requires, ease of visualization (as graphs), extensive open source contributions and regular updates on state of the art research models. However, sometimes it can be daunting to debug the code due to the presence of multiple edges between nodes and thus, forming these connections in the mind becomes a prerequisite for carrying out efficient debugging.