Q1. Define binary cross entropy as a cost function:

Binary cross-entropy is a cost function commonly used in machine learning and neural network models for binary classification problems, where the goal is to predict one of two possible outcomes (e.g., yes/no, true/false, etc.).
The binary cross-entropy cost function measures the difference between the predicted probability distribution and the true probability distribution of the binary classification problem. It is defined as follows:

$$J(\theta) = -(1/m) * \sum_{i=1}^{m} [y^{(i)} \log(h(\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h(\theta(x^{(i)})))]$$

Where:

- θ represents the model's parameters.
- m is the number of training examples.
- x represents the i-th training example.
- y^(i) is the corresponding true label for the i-th training example (0 or 1).
- h(θ(x^(i)) is the predicted probability that y^(i) is 1 given x^(i) and the model's parameters.

The binary cross-entropy cost function penalises the model heavily when it predicts a probability close to 0 for a true positive example or close to 1 for a true negative example, while rewarding it when the prediction is close to the true label. The goal of the model is to minimise the cost function by adjusting its parameters to improve its predictions.

Q2. Difference between adam optimizer and gradient descent

Adam optimizer and Gradient descent are both optimization algorithms commonly used in training machine learning models. However, there are several differences between the two:

1. Method: Gradient descent is a basic optimization algorithm that computes the gradient of the cost function with respect to the model parameters and takes a step in the opposite direction of the gradient to minimise the cost function. Adam optimizer, on the other hand, is a more advanced optimization algorithm that combines the benefits of two other optimization algorithms, namely, Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp).
2. Learning Rate: Gradient descent uses a fixed learning rate that determines the size of the steps taken during each iteration of the algorithm. The learning rate is typically chosen based on a trial-and-error approach. Adam optimizer, on the other hand, uses an adaptive learning rate that adjusts the step size based on the gradient's magnitudes, resulting in more efficient and faster convergence.
3. Momentum: Adam optimizer also incorporates the concept of momentum, which helps to accelerate the convergence process. It calculates an exponentially decaying average of past gradients and uses this to update the parameters, thereby reducing the variance in the parameter updates.
4. Performance: In practice, Adam optimizer tends to outperform Gradient descent in many cases. Adam optimizer is well-suited for large-scale datasets with high-dimensional input features, where the gradients can be highly varied, and the optimal learning rate is hard to determine. Gradient descent is simpler to implement and is still used when a simple optimization algorithm is needed.

Overall, Adam optimizer is a more advanced and efficient optimization algorithm than Gradient descent, but the choice between the two ultimately depends on the specific problem at hand and the available computational resources.