1. Which Linear Regression training algorithm can you use if you have a training set with millions of features?

Normal Equation, Stochastic Gradient Descent, Batch Gradients Descent, and Mini-batch Gradient Descent can be used irrespective of the number of features. Among all of these, I would prefer to use Stochastic Gradient Descent because it is faster in dealing with large features. The Normal Equation and the Singular Value Decomposition (SVD) approach become slow when the number of features is large, but would still handle large training data efficiently.

2. Suppose the features in your training set have very different scales. Which algorithms might suffer from this, and how? What can you do about it?

All algorithms except Normal Equation and SVD may suffer from having features with different scales. The solution to this is using the Standard Scaler or Feature Scaler classes.

3. Can Gradient Descent get stuck in a local minimum when training a Logistic Regression model?

Yes, getting stuck in a local minimum is one of the pitfalls of Gradient Descent. This occurs when the algorithm being run stops too early or the random initialization starts the algorithm from one side of a non-convex function.

4. Do all Gradient Descent algorithms lead to the same model, provided you let them run long enough?

No, they do not, despite gradually reducing the learning or in the case of a convex optimization problem. They may produce similar models when the learning rates are small, but will not result in the same model.

5. Suppose you use Batch Gradient Descent and you plot the validation error at every epoch. If you notice that the validation error consistently goes up, what is likely going on? How can you fix this?

The validation going back up indicates that the model has begun to overfit the data. A solution to this is Early Stopping, where the training stops as soon as the validation error reaches the minimum.

6. Is it a good idea to stop Mini-batch Gradient Descent immediately when the validation error goes up?

Not necessarily, but rather the solution is to stop only after the validation error has been above the minimum for some time (i.e., when one is quite confident there would not be a better model performance) and then rolling back the parameters to the point when the validation error was at a minimum.

7. Which Gradient Descent algorithm (among those we discussed) will reach the vicinity of the optimal solution the fastest? Which will actually converge? How can you make the others converge as well?

Stochastic Gradient Descent picks random instances in the training set at every step and computes the gradients based only on a single instance, thereby making the algorithm much faster than others. Nevertheless, Batch Gradient Descent will converge with ample training time. Stochastic Gradient Descent and Mini-batch Gradient Descent would converge when the learning rate is reduced gradually.

8. Suppose you are using Polynomial Regression. You plot the learning curves and you notice that there is a large gap between the training error and the validation error. What is happening? What are three ways to solve this?

A large gap between the two curves represents an overfitting model, i.e., the model performs significantly better on the training data than on the validation data.
One solution is to feed the overfitting model with more training data. Another solution is to regularize the model thereby reducing the degrees of freedom. Finally, reducing the number of polynomial degrees can fix the issue of an overfitting model.

9. Suppose you are using Ridge Regression and you notice that the training error and the validation error are almost equal and fairly high. Would you say that the model suffers from high bias or high variance? Should you increase the regularization hyperparameter α or reduce it?

This represents an underfitting model. This means it suffers from a high bias. The solution to this model is to reduce the regularization hyperparameter.

10. Why would you want to use:

a. Ridge Regression instead of plain Linear Regression (i.e., without any regularization)?

Ridge Regression prevents the learning algorithm from overfitting the data, unlike linear regression without regularization. This is accomplished by forcing the learning algorithm to fit the data and keeping the model weights as small as possible using regularization.

b. Lasso instead of Ridge Regression?

This is in a case where only a few features are suspected to be useful. Lasso regression tends to reduce useless features' weights down to zero.

c. Elastic Net instead of Lasso?

This is a middle ground between Lasso and Ridge Regression, where the mix ratio can be controlled. This prevents erratic behavior observed in Lasso Regression when the number of features is greater than the number of training instances or when several features are strongly correlated.

11. Suppose you want to classify pictures as outdoor/indoor and daytime/nighttime. Should you implement two Logistic Regression classifiers or one Softmax Regression classifier?

Two logistic regressions would be needed since the outdoor/indoor and daytime/nighttime classes are not exclusive. Softmax regression classifier predicts only one class at a time, hence it is used only with mutually exclusive classes.