a. Polynomial regression takes much longer to train. KNN has almost no training cost. Whereas for polynomial regressions it can be quite computationally expensive to compute the weights vector w. Even if we assume that matrix X is given, taking the transpose is in a O((n1 or n2)(n_fetures)) operation(will go over each row, then will go over each column in each row). Referring to [here]() matrix multiplication and inversion can be generalized to have O(n^3) complexity. Eventually the whole equation will approximately result in O(n^2) + O(n^3) + O(n^3) .. which can be concluded to have runtime complexity of O(n^3). Which is quite expensive.

b. For prediction polynomial regression will likely do better. Again if we assume matrix X is given, to do a prediction for each feature it will be a matrix, vector(or a matrix shaped (degree+1,1)) multiplication. Which will have a computational complexity (n2)((degree+1)^2) which could be approximated to have complexity of O(n^3). Whereas for KNN, computational most expensive step while predicting is the creation of Distances matrix. Where each entry represents the distance between each row vector X_train_i and X_train_ j . To be able to calculate this matrix one must iterate over each row of X_train and then for each row of X_train go over each row X_test and calculate each row. For all of the distance metrics we calculated norms, they have a complexity of n_features. Eventually calculating this distance matrix will result in a (n1)(n2)(n_features) complexity. Which can be approximated to have O(n^3) complexity. So theoretically, we can argue that both models will approximately have the same time complexity. However in practice it could be beneficial to assume that degree size will likely be less than n1,n2 and n_features.
    i. I generalized (n2)((degree+1)^2) to O(n^2) and (n1)(n2)(n_features) to O(n3) to be able to do direct comparisons. My reasoning behind this is that the time complexity of polynomial regression is bound to 2 variables(where one is a quadratic term); degree+1 and n2. Whereas, the time complexity of KNN is bound to 3 variables; (n1)(n2)(n_features). Hence I argued that polynomial regression is likely to have a better runtime assuming that degree +1 will be less than either n1, n2 or n_features. Which will not be the case when degree+1 is greater than n1, n2 and n_features.

c. KNN will likely require a larger file size. KNN will store a matrix sized (n1, n_features) and vector sized (n1, 1). Whereas, polynomial regression will only store a “weights” vector size (degree+1, 1).
    i. Similar to the point I made in “b.” this will not be true for an extremely complicated polynomial function where degree+1 > n1*n_features + n1 .

** I made decisions on points “b.” and “c.” because I assumed the question expected me to do so. However, theoretically my decisions are not absolute.

** My argument for polynomial regression assumes that the input features have a single attribute.