Question 1.1:

Training examples to be saved in X can be the passenger’s age, gender and income. These three attributes have correlation with each other and are very important factors to what the passenger will actually buy. The predicted outcomes are the pre-booked items that have been purchased. This way the algorithm will analyze the different features of the passengers and find a pattern with their correspondent purchases. An example would be a man of 50 years old working as a CEO would definitely buy the extra leg room and priority boarding. A good model to use in this test case would be matrix factorization based on collaborative filtering. Just like Netflix recommends movies to its users. This model will analyze passengers’ interactions with the pre booked items by decomposing the user-item interaction into a product of two matrices. First matrix contains the products’ interests and the second matrix the users’ interests. Combining these two matrices we will have a final complete matrix demonstrating what each user will prefer to buy. Hence a recommender system. This is the best choice because this model is the most scalable. Use precision metric to evaluate how many results from the matrix match the results from the testing set.

Question 1.2:  
After having such an important difference between the targets of the model, the latter will have what we data scientists call noise in future predictions. Meaning the model will learn that the priority boarding has a higher probability of purchases than food. Hence, a solution would be using hyper parameter regularization to ensure minimal loss in the cost function. So, when we determine the loss function in use on the given data, we use hyperparameter optimization to choose the qualified hyperparameters in the set that give the minimal associated loss. That way we will ensure that the model’s performance will be intact. Another way to handle this situation is by using feature selection. The latter will detect the correlation between the labels and the desired outcome and by setting a p-value (usually of 0.05), this selection process will choose which features are important to the predictions of the model and which features are just added work. Thus, if there is no correlation between the labels of the model and the food output, then it is best to drop this feature from the equation. If not, then hyperparameter tuning is the solution.

Question 1.3:

Dear concerned data scientist,

This model will certainly be less accurate since there is the case of data overfitting. This continuous training of the model on new data without hyper parameter tuning will lead to this overfitting. Meaning the model’s analysis will be too dependent on the set of data it was trained on, that it will fail to fit on new data or predict future attributes in a reliable way. A solution would be using cross validation. The latter is to split the training set and the test set in a way to reduce the chance of overfitting data. Instead of splitting the data randomly, cross validation will analyze if the model will generalize its results on different data. If yes, then another dataset split will be performed in order to have the perfect balance and in order to prevent overfitting. Multiple cross validations will be implemented and then averaged in order to find the best train test split that ensures that the model is not going to overfit.

Hope this helps,

Hopefully your new Data Scientist Colleague,

Omar.

Question 2.1:

In this test case, accuracy is a poor metric. Since only 0.01% of the bonds are bad then the accuracy will certainly be 99.99% where the data needed is not that abundant. Even a very simple model that always predicts the good bonds would still have 99.99% accuracy. Imagine applying this model to a new dataset with 0.01% of good bonds and the rest are bad. This model caused by overfitting will predict bad bonds as good bonds. Say that having bad bonds is the positive outcome since this model predicts these bad bonds. Then the negative is having good bonds. So false positive is having predicting bad bonds as good bonds and false negative is having good bonds as bad bonds. Hence instead of using the accuracy metric which of course will give us a 99.99% since bad bonds only make 0.01% of 100000 bonds; we could use another metric to evaluate the model. For example, predicting bad bonds as good bonds is of high caution. Since bad bonds have devastating effects on the financial market, the number of false positives should be minimized. Thus, a good metric would be using precision which relies on determining how much of the predicted positives, are actually positive. Hence, there would be no misconception of taking bad bonds as good ones.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Actual | |
|  |  | Bad | Not bad |
| Predicted | Bad | 9.999 | 9.999 |
| Not bad | 0.001 | 99,980.001 |

Question 2.2:

The 3 metrics of choice are: L2 regularization, threshold value and learning rate. First off, data scientists use regularization or “metrics” to prevent model overfitting. Meaning, the metric would add bias if the model contains high variance. First hyperparameter tuning method is ridge regression. The latter uses order of least squares function to check the loss between the real output and the predicted one. In addition, this method adds a penalty value lambda to force the parameters to decrease towards zero. This is a good choice for a metric. However, say there exists independent variables that are highly multicollinear, then the computations of the OLS will be highly unstable. This would lead in a very poor hyperparameter tuning. Second, threshold value metric. This method is quite simple, since the developer choses a certain threshold T, say equal to 0.5 where in this case we are talking about binary classification. This method is used to get rid of possible negative values predicted by the model. Since the logistic regression algorithm has a plot that tends under the y-axis, low values can tend to be lower than 0. Hence, this threshold value is used to assign every predicted outcome less than T (0.5) to be 0, and every output higher than T to be assigned the value of 1. This method is quite unstable since say we have the value y = 0.499999, then based on this function it would assign y to be 0 instead of 1. But logically, y should be 1 and not 0. Here we have the case of a false positive. Hence this metric is not a go-to. Last but not least, the learning rate regularization method. The latter is the best choice of a metric, since this model uses stochastic gradient descent. This method works in finding the minimum local optimum (point) and decreases the weights by a certain learning rate alpha (assigned by the developer) that will subtract the weights until they reach the local optimum. This function is the most efficient since it makes use of the convex shape of gradient descent, and ensures that the weights are modified, resulting in minimizing the loss function for one tuple of data and the cost function for the entire dataset.

Question 3.1:

Using this matrix, the simple mathematical concept to use is Bayesian Probability. The latter is calculating the probability of an event based on the probability of a previous event. In this case it would work out perfectly since the events are related to yesterday and today. For example, the probability P(R|T) (probability of reading knowing that he/she trained yesterday) is 0.5 in the matrix (2nd row 1st column). Hence by applying the Bayesian formula to find the probability of training at any day, we get:

P(R|T) = P(RnT)/P(T). We need to find P(T) in this case. Thus, modify the formula by having an assignment for P(T):

P(T) = P(RnT)/P(R|T).

This formula calculates correctly will result in having the probability of training at any given day.