# Data Science Test

## 1. Airline Use Case

Suppose we build a recommendation engine for a budget airline that predicts online purchases of pre-booked items for each passenger, such as priority boarding, extra leg room, exact seating, and food and beverages.

### Question 1.1

There is a confusion at the client about how to set up the machine learning task. What are the training examples (X) and what are the predicted outcomes (y) ? What kind of algorithm to use? What should be the evaluation metric of the model? Please share your suggestions with them for each question.

**Answer: I would use a classification based model and my training examples would consist the next data: User features(age, gender, nationality etc), Product features(cost, quality, category etc) and historical data. Historical data is about previous purchases of users. And my outcome would be a binary value(0,1) that means the pre-booked items is liked or not. My preferred technique is the collaborative filtering which is based on assumption that people like things similar to other things they like and things that are liked by other people with similar taste. And this technique use k-nearest neighbors algorithm that would be suitable for this problem. I think the accuracy is a good choice to evaluate the model because we should determine an item is in liked or disliked group of actual user.**

### Question 1.2

We know that the popularity of the products are varying. E.g. purchase rate for priority boarding is 20 % while for food is 2 %. How would this influence the recommendations? Do we need to handle it somehow?

**Answer: I think it doesn’t affect the recommendations because priority boarding and food are not similar purchase items. This means that if a passenger buys priority boarding, it’s not sure that the passenger will buy food as well. If they would be similar, because of their content or popularity, our recommendation system would suggest them to users, therefore the purchase ratings would be similar. I think we don’t need to handle this.**

### Question 1.3

We settled to use one year’s data of online pre-booked purchase behavior for model training, which we split into 70% training and 30% evaluation sets randomly. Our final model is ready and it performs well on both sets. The plan is to retrain the model (no hyperparameter-tuning, just re-run) every day at 1 am based on data of the previous 30 days.

A data scientist from the client’s team expresses concerns that the production system will not perform as well as indicated by our training setup. Is this concern valid? How would you address his concern? Write an email to him.

**Answer: I think splitting is wrong because we need a 3rd set. I don’t know what is evaluation set but I think we need a training set to train the model, a validation set to validate the model and a test set to test the model. I think validation or test set is missing. On the other hand the data scientist’s concern is valid because if we retrain the model with new data, the performance and accuracy can change. And during the retrain we may find a different algorithm or a new set of features that improves our predictions. For example retrain can be very important related to an airline because the passengers’s habits may change during one year. In this case our model is not going to be accurate because it didn’t consider changing of customers.  
My email: “Dear Mr. Smith! Thank you for your observation, our developer team starts to examine your concern, we will keep you updated regarding the issue. Best regards, Gergő Boros”**

## 2. ML methodology

### Question 2.1

A new classifier model identifies bad bonds in the financial market for a hedge fund. Bad bonds can have devastating effects and must be avoided in the portfolio. 0.01% of all bonds fall into this category and our model has an accuracy of 99.99%. Is this ML model doing a good job? Why?

Fill in the empty confusion matrix below with a possible concrete outcome if there are 100,000 bonds in the market.

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

**Answer: I think this is not a good ML model because 0.01% of dataset is bad bonds and 99.99% other. So this dataset consists highly imbalanced data, therefore the 99.99% accuracy is not good in my opinion. Before training we should apply some techniques to solve imbalance for example upsampling or downsampling. But if I’m mistaken and the model is good, I tried to fill the chart.**

### Question 2.2

On a logistic regression model with binary outcome in {0,1} that is optimized with stochastic gradient descent you have to tune hyperparameters

* learning rate
* L2 regularization
* batch size
* threshold value: the predicted probability above which we assign 1

Choose 3 metrics that you can use to compare the trained model and decide which one is the best for this use case. Explain why.

**Answer:   
1. Classification accuracy**

**Classification accuracy is the number of correct predictions made as ratio of predictions made. They usually use when there are an equal number of observations in each class.**

**2. ROC curve**

**It represents a model’s ability to discriminate between positive and negative classes. An area of 1.0 represents a model that made all predictions perfectly. An area of 0.5 represents a model as good as random.**

**3. Confusion matrix**

**The [confusion matrix](https://machinelearningmastery.com/confusion-matrix-machine-learning/) is a handy presentation of the accuracy of a model with two or more classes. The table presents predictions on the x-axis and accuracy is showed on the y-axis. The cells of the table are the number of predictions.**

**Unfortunately I don’t know which metric would be a good choice.**

## 3. Math

### Question 3.1

The matrix below shows the probability that you are in a state today given we know your state from yesterday. There are two states: each day either you read or you train.

When you read one day you are very likely to continue the book the following day.

When you train you decide with a coin flip whether to go out and train again the next day.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Today | |
|  |  | Reading | Training |
| Yesterday | Reading | 0.9 | 0.1 |
| Training | 0.5 | 0.5 |

Which mathematics concept would you use to calculate the probability of training at any given day? ( The probability of training after an infinite number of days?) You are not required to calculate it.

**Answer: Bayes Theorem**

## 4*.* Coding challenge

The final question is about testing your skills in writing production-ready code. Your solution will be evaluated by the following criteria:

* Completeness of the solution
* Documentation quality
* Organization of the solution

### 4.1 The basics

As a first step, you need to find a toy dataset that you will use during the exercise. Using it a **binary classification problem** should be solved using the **logistic regression** algorithm. You are not required to do any EDA or feature engineering but you should follow principles that are required to build a production-ready model. However the model performance, in this case, is irrelevant, but choosing the proper metric and evaluating the model is necessary.

A requirements.txt is needed to recreate your environment.

### 4.2 Implementation of a custom estimator

The aim of this exercise is to write a custom estimator in Python and more specifically in the Scikit-Learn fashion.

Here are the **primary requirements** of our custom estimator:

* the estimator is used for binary classification tasks;
* the model is a standard logistic regression;
* on top of the model, the threshold is optimised with respect to a specific metric: the Gini impurity of the splits which needs minimizing.

Because our custom estimator should follow the Scikit-Learn fashion, here are some **secondary** **requirements** of its implementation:

* a class named ***ThresholdBinarizer*** should be implemented; more specifically this class should be used to optimise the threshold and needs inheriting from Sklearn *BaseEstimator* & *TransformerMixin*;
* the custom estimator should be implemented within a class named ***custom\_estimator***; as such class should be used for binary classification, it needs inheriting from Sklearn *BaseEstimator* & *ClassifierMixin*.
* The logistic regression model is not needed to be implemented from scratch, you can use the ones built-in Sklearn.
* The final goal is to have a custom estimator and by calling its predict method the class assignment will be done based on the logistic regression model and the optimized threshold determined with Gini impurity metric.

Eventually, we will also take into consideration the quality of your implementation:

* each class & method should be documented;
* your code should be packaged;
* you should provide a script (Python file or iPython notebook) showing your code running and highlighting your estimator’s abilities on a *light toy* *dataset* of your discretion;
* please provide in a CSV format the *light toy dataset* you have chosen *(such dataset should only serve the purpose of testing your implementation - while not serving any aspect of a Machine Learning project, e.g. EDA, feature engineering, modelling, validation, etc.)*;
* you should also provide a *requirements.txt* file mentioning the versions of the Python packages your implementation is based on.

The solutions have to be uploaded to a git repository shared with [norbert.liki@aliz.ai](mailto:norbert.liki@aliz.ai) and [pierre@aliz.ai](mailto:pierre@aliz.ai) in separate folders for the tasks.