# 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.

### 

### Solution 1.1:

For the above problem our Training example (X) should contain passenger’s implicit information like history of purchases as well as passenger’s explicit information like rating on food and beverages given by passenger.

The outcome (Y) should be the Item id that passenger will likely to buy.

Matrix Factorization would be my first priority in building a recommendation engine and on top of this we can use Recall or Mean Reciprocal Rank as our evaluation metrics.

### 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?

### Solution 1.2:

This scenario signifies the imbalance in our dataset. E.g. purchase rate of priority boarding is more than for food, therefore our recommendation system will be more likely to predict boarding more often.

What we can do is try some re-sampling methods like under-sampling or over-sampling in data to remove this imbalanced classification, and then build recommendation on top of that.

### 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.

### Solution 1.3:

This is surely a valid concern, it is a very bad idea to just re-run the same model without seeing the distributions of the new data. Our new data might be biased which could affect our previous approach and performance using older data.

New data should have similar past distributions to give same results.

## 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?

### 

### Solution 2.1:

Here Accuracy is not the right metric to decide the performance of our model. Since this is a problem of imbalanced classification, so our model will likely to give prediction on the class that has very high weights in the data.

So we have to use some other metrics like F1 score to correctly measure the performance of our model.

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* | *9* |
| Not bad | *1* | *99981* |

### 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.

### Solution 2.2

Three metrics that I have chosen for the above case would be **Accuracy**, **F1 Score**, and **AUC**.

Whereas best metric for this use case would be **AUC(Area Under the ROC curve )** as It provides an aggregate measure of performance across all possible classification thresholds.

## 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.

### Solution 3.1:

Mathematics concept I would use to calculate the probability of training at any given day would be **Conditional Random Fields( Discriminative Hidden Markov model) .**

## 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 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 the 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.

### Solution 4.1

<https://github.com/anushuk/Data-Science-test>

### 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.

### 

### Solution 4.2

<https://github.com/anushuk/Data-Science-test>

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.