# 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 some 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.

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?

These questions need to be answered separately for each pre-booked item:

# Assumption:

1. Since we are talking about a budget airline, we can assume that people would like to keep the cost as low as possible, and in turn only a small proportion of the customers would pre-book extra items. This would lead to an imbalanced dataset, which would mean that we need to be careful selecting the evaluation metric. E.g. If the dataset contains 10% of customers who bought and 90% of customers who didn’t, and we classified all the customers as “did not buy”, then the Accuracy of the model would be 90%.
2. Since we are talking about a budget airline, we could assume that it operates only 1 type of airplane, which would decrease costs (maintenance, training, purchase of planes/parts).

**Priority boarding (imbalanced dataset):**

Possible predictors:

* Airport Name
* Departure Time
* Day of the Week of the flight
* Transfer/Direct flight
* Country of Departure
* City of Departure
* Season
* Month of Year
* Age of customer
* Nationality

Algorithm: Binary classification

Predicted variable: Yes/No {0, 1}

Evaluation metric: Confusion Matrix and F1 score

**Extra leg room (imbalanced dataset):**

Possible predictors:

* Length of the flight
* Departure Time
* Transfer/Direct flight
* Day of the Week of the flight
* Direction of the flight (E, S, W, N)
* Season
* Month of Year
* Age of customer
* Nationality

Algorithm: Binary classification

Predicted variable: Yes/No {0, 1}

Evaluation metric: Confusion Matrix and F1 score

**Exact seating (balanced dataset):**

This one is tricky. Each seat has different characteristics (window/middle/aisle), position on the plane (front, in front of the wing, behind the wind, etc.), side (left/right)

A prediction can be called good, if it takes into account the features the customer cares about the most, but it does not necessarily have to be an exact seat. It is also possible that the predicted exact seat is already taken. The most sensible solution would be to group seats together based on common characteristics, and predict a group for each customer.

Possible predictors:

* Length of the flight
* Departure Time
* Transfer/Direct flight
* Day of the Week of the flight
* Direction of the flight (E, S, W, N)
* Season
* Month of Year

Algorithm: Multinomial classification

Predicted variable: {0, n} n = seat groups based on their characteristics

Evaluation metric: Accuracy, Confusion Matrix and F1 score

**Food and beverages (imbalanced dataset):**

Possible predictors:

* Country of Departure
* City of Departure
* Length of the flight
* Airport Name
* Departure Time
* Transfer/Direct flight
* Season
* Month of Year
* Age of customer
* Nationality

Algorithm: Multinomial classification

Predicted variable: Food/Beverage items {0, n + 1} n = items on the menu + 1 for no item

If beverages and food items can be selected separately, then we would need two algorithms.

Evaluation metric: Confusion Matrix and F1 score

### Question 1.2

We know that the popularity of the products varies. 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?

How would this influence the recommendations?

Having a low purchase rate for an item would result in imbalanced training/testing datasets. This means, that the customers who did not buy priority boarding/food items would outnumber the customer who did.

Most algorithms are designed to maximize accuracy and reduce error, and they work best when the classes in the dataset are balanced. If left untreated the algorithm could have a high accuracy, while at the same time make bad predictions.

Do we need to handle it somehow?

Yes.

How? (Assuming this question was implied in the previous one)

1. Depending on the amount of data we have, we could under sample the majority classes or oversample the minority classes, or use k-folds cross validation.
2. Ensemble: Keep the customers who bought a product, and resample the customers who didn’t into n different datasets (The ratio of customer who bough to the customer who didn’t can be changed based on performance). After which we combine the customers who bought with each data set, and train an algorithms on each dataset. At the end we combine the results.
3. Use an algorithm that is suited for imbalanced data. (e.g. XGBoost)

### 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 hyper parameter-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.

# Assumption:

1. The Data Scientist at the client is called Markus.
2. He is good at his job, and so far has been very cooperative with us.

First, I would ask about his/her concerns (just to have it on the record):

Dear Markus,

Thank you for your email.

You mentioned that you would have some concerns regarding our training set-up.

I was wondering if you could clarify which aspects of the set-up are you

Best regards,

Denes

Since we assumed that he is good at his job he would most likely point out the following:

* The amount of data used for retraining is from the past 30 days, while the original model is using the data from the past year. This seems a bit inconsistent. Either the original model should be trained on the data from the past 30 days, or the retraining should use the data from the past 30 days.
* He would also point out, that although it is good practice to retrain the model periodically; it would also be nice to set up some kind of monitoring system, which checks the error of the model daily. This could help find a timeframe, after which it is imperative to retrain the model. This could be longer than 1 day.
* Considering that using new data might result in lower performance using the original hyper parameters, it would be advisable to reevaluate these if after retraining the error rate hasn’t decreased.

After he pointed the above out, I would respond with an email like this:

Dear Markus,

Thank you for sharing your concerns with us.

The model saves the error rate into a log file after each day.

We retrain the model after each day in order to see how soon the hyper parameters need to be recalculated. We will monitor the error rate and see after what timeframe would the hyper parameters need to be retrained. Once we established this timeframe we can reduce the model retraining frequency.

Both the training and the retraining processes will be using data from the past 1 year. This was only a miscommunication from our part.

Hope this helps,

Denes

## 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?

Again, this is an imbalanced dataset. If we classify every entry in the entire dataset as good bonds, we would get 99.99% accuracy. And yet out model would not be able to make accurate predictions.

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 | *0* | *0* |
| Not bad | *10* | *99,990* |

### 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 hyper parameters

* 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.

* AIC/ BIC: These are penalized-likelihood criterions. AIC is more liberal, while BIC penalizes model complexity more heavily. These allow comparing different models to each other. If the model complexity (number of predictor variable) can change then I would use these.
* Pseudo R2 and p value: pseudo R2 shows how much of the variance is explained by the model, while the p value shows if the relationship is due to chance or not. Higher R2 the better, lower the p value the better.
* ROC Curve and AUC: ROC is a probability curve and AUC represents degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, better the model.

## 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.

* Markov Chain [x, y] = [5/6. 1/6]

## 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 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.