Internship Technical Report

Société Générale and Convertize

Yanis Tazi

October 2018

Contents

1 Introduction

After studying 5 years in a row, I decided to take a gap year in order to acquire some work experience. I was hesitating between Finance and Data Science and I also wanted to see the difference between working within a large global company and a Start-Up. Therefore, I decided to find my internships accordingly. This is why, I interned first as a Data Scientist in a Start-Up in London and after that, I moved to New-York in Trading in the Delta One Desk at Société Générale.

2 Convertize

June-December 2017 , London

2.1 Overview

Convertize is a neuromarketing digital company based in London. It is a start up specialised in improving conversion rates for ecommerce websites. The technics used vary from neurosciences to Data Science and Machine Learning in order to understand the traffic of the website and to improve the customers experience during the visit on the website. I have been hired as an intern in Research for a new A/B Testing approach in order to write a white paper available on Amazon [4] and as a Data Scientist.

2.2 Projects

At Convertize, I have been working on three main projects and a few less important projects. In this section, I will explain in details the 3 main projects that I have set up and quickly go through the other ones.

2.2.1 Recommender System

I-Content-Based Recommendation

Content-based systems[9] generate recommendations based on similarities between items name and description. We do not need any assumptions on the customers. Therefore, this is a good starting point when we do not have data about the customer and when we want to recommend similar products.

Let's go through a complete implementation that I set up in Python to see how it works:

MINES ParisTech PSL

Convertize THINK OPTIMIZATION

SOCIETE GENERALE Corporate & Investment Banking

Yanis Tazi - Convertize

Content-Based Recommendation

Libraries to use :

pandas allow us to define, read and modify dataframes. sklearn is a machine learning library useful in Machine Learning with available implemented functions such as Tf-Idf

In [4]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
```

Get the dataset with description of the products

We will do the demo with a fake dataset from Kaggle because I do not have the right to use the real ones.

In [16]:
```
product_df= pd.read_csv("/Users/yanis/Downloads/sample-data.csv")
#let's have a quick look at the dataset:
print product_df.head(10)
print 'length:'+ str(len(product_df))
```

```
   id                                        description
0   1  Active classic boxers - There's a reason why o...
1   2  Active sport boxer briefs - Skinning up Glory ...
2   3  Active sport briefs - These superbreathable no...
3   4  Alpine guide pants - Skin in, climb ice, switc...
4   5  Alpine wind jkt - On high ridges, steep ice an...
5   6  Ascensionist jkt - Our most technical soft she...
6   7  Atom - A multitasker's cloud nine, the Atom pl...
7   8  Print banded betina btm - Our fullest coverage...
8   9  Baby micro d-luxe cardigan - Micro D-Luxe is a...
9  10  Baby sun bucket hat - This hat goes on when th...
length:500
```

In [17]:
```
product_df = product_df.dropna(axis=0, how='any')
product_df = product_df.reset_index(drop=True, inplace=False)
#check the length after removing nan values to have a clean dataset(in this demo it is not going to change)
print 'length:'+ str(len(product_df))
```

```
length:500
```

The only difference is that in the demo, instead of having name of the product and descriptions , we only have the description but the idea of combining product name + description will be done by combining id + description here. It will weaken a bit the algorithm since we do not have the name of the products anymore.

Figure 1: Content-Based Implementation -Part1

In [19]:
```
product_df['combined']=product_df['id'].astype(str)+' - '+product_df['description']
product_df['combined'].head(5)
#Here, we can see the combination between 'id' - 'description'.
#In the products dataset at Convertize, instead of using the 'ids' we use the product names.
```

Out[19]:
```
0    1 - Active classic boxers - There's a reason w...
1    2 - Active sport boxer briefs - Skinning up Gl...
2    3 - Active sport briefs - These superbreathabl...
3    4 - Alpine guide pants - Skin in, climb ice, s...
4    5 - Alpine wind jkt - On high ridges, steep ic...
Name: combined, dtype: object
```

TF-IDF and Cosine Similarity

In [20]:
```
tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 3), min_df=0, stop_words='english')
```

In [24]:
```
product_matrix = tf.fit_transform(product_df['combined'])
product_matrix.shape
#For each product, we have the weight associated to its description based on the words used in all products.
#It will be explain in details in the internship report
```

Out[24]:
```
(500, 53505)
```

In [64]:
```
# Now we use the cosine similarity which is a way of measuring the similarity between different description of p
roducts
cosine_similarities = linear_kernel(product_matrix, product_matrix)
rec_table = {}

for col, row in product_df.iterrows():
    similar_indices = cosine_similarities[col].argsort()[:-12:-1]
    # we recommend up to 5 products
    similar_items = [(cosine_similarities[col][i], product_df['id'][i]) for i in similar_indices]
    rec_table[row['id']] = similar_items[1:]
print rec_table.items()[2]
# We read the result like this: For item 3, the 3 most similar items are: 2,299,495 and their respective scores
#Let's check those items:
print product_df['description'][product_df['id'].isin( [3,2,299,495])]
```

```
(3, [(0.41732199122567909, 2), (0.11330253415873627, 299), (0.10993729003809452, 495), (0.10885373158003085, 30
0), (0.10115764349025325, 156), (0.09974338308099448, 318), (0.099010898196287306, 155), (0.091910144105251751,
165), (0.091466012528305191, 164), (0.090029185870147257, 258)])
1      Active sport boxer briefs - Skinning up Glory ...
2      Active sport briefs - These superbreathable no...
298    Active boy shorts - We've worn these versatile...
494    Active briefs - These featherweight, quick-wic...
Name: description, dtype: object
```

Figure 2: Content-Based Implementation -Part2

Recommendations:

In [67]:
```
def recommend(item_id, num):
    if num <= 10:
        i = 1
        print("I will recommend using the cosine similarity score up to " + str(
            num) + " products similar to item_id " + str(item_id) +  ".")
        previous_rec = 0
        for rec in rec_table[item_id][:num]:

            # make sure that we do not display a produt with the same description twice

            if previous_rec != rec[0] and rec[0] != 0:
                print("-Recommendation " + str(i) + ":"  + " (the score is:" + str(
                    rec[0]) + " and the item_id is:" + str(rec[1]) + ").")
                i = i + 1
            previous_rec = rec[0]
    else:
        print ("We can only give you up to 10 similar products ! Please reduce the number of similar products")

recommend(item_id=420, num=4)
```

```
I will recommend using the cosine similarity score up to 4 products similar to item_id 420.
-Recommendation 1: (the score is:0.882391408185 and the item_id is:114).
-Recommendation 2: (the score is:0.878619603106 and the item_id is:113).
-Recommendation 3: (the score is:0.660392436093 and the item_id is:398).
-Recommendation 4: (the score is:0.240852824499 and the item_id is:136).
```

Figure 3: Content-Based Implementation -Part3

Through this Jupyter notebook example, I have been showing the basic steps to construct a recommendation system based only on the content of the products.

II-Collaborative Filtering

It is a way of recommendation based on user's behavior and similarities between users. There are two main approaches:
-User-based which measure the similarities between the targeted user and other users.
-Item-based measuring the similarities between items a targeted user has rated or interacted with and other items.

It is one of the mostly used personalised recommendation algorithm. There are 3 main assumptions: People with similar preferences will be interested by the same items Preferences and interests are stable over time. We can predict choices according to past preferences.

To be able to implement those algorithms, we need to have access to a database referring to the preferences of the users:

	Items	1	2	3	...	i	...	j	...	N-1	N
Users											
1						***R(1,i)***		***R(1,j)***			
2						-		-			
3						*R(3,i)*		-			
...						-		-			
k						-		*R(k,j)*			
...						*R(,)*		-			
l						***R(l,i)***		***R(l,j)***			
...						-		-			
F-1						-		-			
F						***R(F,i)***		***R(F,j)***			

Figure 4: Data base Users/Ratings

In this rating matrix, each row represents a user and each column an item. We usually have a very sparse matrix and to fix this, we sometimes use a dictionary or a list for coding optimisation. Since the matrix is really sparse, I have introduced a more flexible ratings notation in the sense that even if people do not rank a products, we might be able to guess the ratings in order to have more ratings per users and therefore to be able to have better predictions. To do that, as long as a user puts a product in the basket, we rate the products with a 4 stars and when the user buys the product but does not rate it , we give a 5. I am conscient that it is not reflecting precisely the user's preferences but it is a simple and accurate way enough to handle the sparsity of the matrix.

II-1-Item-based Filtering

In order to compute the similarity between 2 items (i and j), we only look at the users who have rated both items. Again, different similarity measures are available: Cosine, Pearson and Adjusted Cosine Similarity.

-Adjusted Cosine Similarity: It is a modified version of the Pearson Similarity. In this computation, we take into account that users have different ratings behaviour, i.e some of them give high score ratings while others lower the scores. To partially solve this issue, we subtract the average rating of all items for each user to the pair of items. This method also allows us to predict any user-item rating pairs using a weighted sum of a number of similar items(threshold number or similarity bigger than a threshold) to the targeted item.

$$\boxed{sim(i,j) = \frac{\sum_{u \in U}[(R(u,i) - \overline{R(u)}) * (R(u,j) - \overline{R(u)})]}{\sqrt{\sum_{u \in U}(R(u,i) - \overline{R(u)})^2} * \sqrt{\sum_{u \in U}(R(u,i) - \overline{R(u)})^2}}}$$

and the predicted rating from product k for user u is:

$$\boxed{P(u,k) = \frac{\sum_{n(simitems)} sim(k,n) * R(u,n)}{\sum_{n(simitems)} |sim(k,n)|}}$$

with n defined as number of similar items that we want to take into account for the rating predictions.

The way I introduced n at Convertize is : $n = \bigcup_{k=0}^{9} B_k$, where $B_k = A_k - A_{k+1}$ and $A_{k+1} = A_k - max(A_k)$. It simply means that we take the 10 most similar items in the comparison (an other way is to set up a threshold similarity and to take only those with higher similarities). However, we might end up with no items in this situation compared to the other one. This is why the first method is a reliable way to always have a predicted value.

II-2-User-based Filtering

This approach is based on the comparison of the targeted user's behavior with other user's behavior in order to find his nearest neighbours and to be able to predict his preferences according to his neighbours preferences. Let's compute the similarity between users using the Pearson Correlation:

$$\boxed{sim(u,v) = \frac{\sum_{i \in I}[(R(u,i) - \overline{R(u)}) * (R(v,j) - \overline{R(v)})]}{\sqrt{\sum_{i \in I}(R(u,i) - \overline{R(u)})^2} * \sqrt{\sum_{i \in I}(R(v,i) - \overline{R(v)})^2}}}$$

As we can see, this is equivalent to the item-based filtering . The only difference here is that we change the focus.

The advantage with the item-based filtering method is that we can compute if offline.

Problems of the collaborative filtering: -Sparse matrix
-Early rater
-People with out of the box behavior

II-3-Hybrid Filtering

The idea is to use the best of both approach. Collaborative filtering is meaningful when users have rated many items in common. This is rare because they need to rate exactly the same item and it happens that users have seen similar items but not exactly the same ones. The idea behind this hybrid system is to incorporate in the rating the content of the items and not the item itself. In collaboration with content, we need not only the item itself but also the content itself. Therefore, this is a much harder approach to implement because we need to identify specific content for each item via content-based and be able to find similar content in other items in order to reduce the sparsity problem.

Let's illustrate with an example:

	Target item and content description	Content description 1	Content description 2	Content description 3	Content description 4	Targeted item
Users						
i		$w_{i,1}$	$w_{i,2}$	$w_{i,3}$	$w_{i,4}$	***R(i,item)***
j		$w_{j,1}$	$w_{j,2}$	$w_{j,3}$	$w_{j,4}$	***R(j,item)***
k		$w_{k,1}$	$w_{k,2}$	$w_{k,3}$	$w_{k,4}$	***R(k,item)***
l		$w_{l,1}$	$w_{l,2}$	$w_{l,3}$	$w_{l,4}$	***R(l,item)***
Targeted user *		$w_{*,1}$	$w_{*,2}$	$w_{*,3}$	$w_{*,4}$	$R(u_*, i_*)$

Figure 5: Data base Users/Content-Ratings

We want to predict the rating of the targeted item for the targeted user. To do so, we find some key words(4 in the example) in the content description via the content-based algorithm. The weight indicates how important they are to a user based on its historical behavior. Now, we use the Pearson correlation coefficient in order to find the correlation between users and the targeted user. This method allows us to use much more items in order to find similarity between users. Indeed, in collaborative filtering if the users have not rated a specific item in common , we are not able to take into account their ratings for the prediction. However, it can happen that they rate similar items so with the content approach , we are able to tackle the problem and still take them into account.Also, we solve the problem found in content based approach because we now use user impressions with similarities between users and not only description of items. The problem that is still present is the new item because we need ratings in order to rate the new item for the targeted user.

I proposed a new rating formula based on the weights determined for the targeted user in proportion to the average of the weights given by the other users and their ratings:

$$R(u_*, i_*) = [\sum_{c \in Content} \frac{w_{u*,c}}{\sum_{u \in (Users \setminus user*)} w_{u,c}}] * [\sum_{u \in (Users \setminus user*)} R(u, i_*)]$$

2.2.2 Dynamic Pricing Model

Figure 6: Queing Theory

Dynamic Pricing is a strategy which consists in adjusting product prices in response to real time supply and demand. In most of the e-commerce websites, prices are updated in order to optimise the revenues. For example, Amazon updates its prices every 10 mn. With dynamic pricing, profitability has been optimized since websites are able to adapt to customer needs. However, this idea has emerged decades ago in the retail industry but was not widely used due to the reticence of retailers because of the taken risk. Indeed, a slight drop uncontrolled in the price of a product can have large effects on the profit. Nowadays, with the large amount of data gathered and the robustness of algorithms, it is relatively simple to create good dynamic pricing models.

In order to set up a pricing model, we need to simulate customer's arrival on a website. An intuitive way of doing so, is to think of it as a Poisson Process with parameter λ . Indeed, we can imagine the arrival of the customer as a waiting line model.A poisson Process [6] with parameter λ models the probability of arrival of one customer during an infinitesimal period δ as $\lambda * \delta$ and the probability of no arrival as $(1 - \lambda * \delta)$. We assume that δ is defined such as at maximum one customer can arrive during the period $[t, t + \delta]$ $\forall t > 0$ .Also, in our model , we define λ as a geolocalized constant. At Convertize, I have decided to define λ as a constant updated every day representing the arrival rate in one period :

$$\lambda = \frac{number of Leads of the day}{number of Periods of the day}$$

Also, I said geolocalised because we have noticed that customers from different regions are not willing to pay the same price for an identical item. Based on some tests that we have set up before the dynamic pricing, Dutch people tend to pay more than French people for identical items and that's why we have decide to geolocalise the constant λ. Unfortunately , I can not explain in details for privacy reasons how we have done this process but it will not stop us from understanding the dynamic pricing model that we are going to use. Let's from now assume that λ is a constant in our model. I am

going to introduce some other notations:
-N the limited product quantity during a period T
-r the internal cost of selling the product(think of it as the delivery and storing costs)
-$\gamma = \frac{numberofSales}{numberofLeads}$
Now, let's define the probability of buying the product at Price p: $\mathbb{P}(Price = p)$. Intuitively, if we defined $g(p)$ as the probability modelling a customer willing to buy the product at price p, this probability can be expressed in terms of $g(p)$ as:

$$\boxed{\mathbb{P}(Price = p) = \int_p^{+\infty} g(x)dx = 1 - G(p) = \exp(-\gamma * p) \quad , \gamma > 0}$$

A few points on this equation:
-$\mathbb{P}(Price = p)$ decreases with p which is a rational behavior
-$P(Price = 0) = 1$ because if the price is null everyone will buy and the probability tends to 0 when the price goes to $+\infty$ because no one would pay the product.
-When γ increases , the probability of buying a product at price p decreases. So when we have a good conversion rate (high γ), we are able to lower the price of the product and still match the desired profit. That's why choosing γ as the ratio between sales and leads is a good trade off.
Our objective is to **maximise** the expected revenue at time t that we can obtain by time T: $\mathbb{E}(i,t)$, $i \in [0,N]$ and $t \in [0,T]$ with:
-$\mathbb{E}(0,t) = 0 \ \forall t \in [0,T]$: without products to sell, we can not expect revenue
-$\mathbb{E}(i,T) = 0 \ \forall i \in [0,N]$:: at time T , we are not able to sell products anymore

$$\mathbb{E}(i,t) = [(1 - \lambda * \delta) * \mathbb{E}(i, t+\delta)] \tag{1}$$
$$+[(\lambda * \delta) * G(p) * \mathbb{E}(i, t+\delta)] \tag{2}$$
$$+[(\lambda * \delta) * (1 - G(p)) * (\mathbb{E}(i-1, t+\delta) + p - r)] \tag{3}$$

To compute $\mathbb{E}(i,t)$, we need to consider 3 different cases when we have i products at time t :

(1): No customer appears between time t and $t+\delta$ with probability defined earlier $(1-\lambda*\delta)$ with an expected revenue at time $t+\delta$: $\mathbb{E}(i,t+\delta)$ because no products will be sold since no one appeared.

(2): A customer appears between time t and $t+\delta$ with probability $\lambda*\delta$ but does not buy at price p with probability $[1-\mathbb{P}(Price=p)=G(p)]$ with the same expected revenue as in (1) at time $t+\delta$.

(3): This time, a customer appears with the same probability $\lambda*\delta$ and buys a product at price p with probability $[\mathbb{P}(Price=p)=1-G(p)]$ so this time, the expected revenue at time $t+\delta$ has changed to: $\mathbb{E}(i-1,t+\delta)+p-r$

Finally, we need to find the value p_* that maximise $\mathbb{E}(i,t)$, i.e $f_*(i,t)$ such that:

$$\begin{aligned} f_*(i,t) &= \max_p\{[(1-\lambda*\delta)*\mathbb{E}(i,t+\delta)]+[(\lambda*\delta)*G(p)*\mathbb{E}(i,t+\delta)] \\ &\qquad +[(\lambda*\delta)*(1-G(p))*\mathbb{E}(i-1,t+\delta)+p-r)]\} \\ &= \max_p\{[(1-\lambda*\delta)*f_*(i,t+\delta)]+[(\lambda*\delta)*G(p)*f_*(i,t+\delta)] \\ &\qquad +[(\lambda*\delta)*(1-G(p))*f_*(i-1,t+\delta)+p-r)]\} \end{aligned} \tag{4}$$

$$\Rightarrow \boxed{\frac{f_*(i,t+\delta)-f_*(i,t)}{\delta} = -\lambda \min_p\{(\exp(-\gamma*p)*(f_*(i,t)-f_*(i-1,t)-p+r)\}} \tag{5}$$

Let's denote : $q(p)=\exp(-\gamma*p)*(f_*(i,t)-f_*(i-1,t)-p+r)$.
Therefore, since $\exp(-\gamma*p)>0\ \forall\gamma,p>0$: the solution of (5) is obtained for p_* solution of $q'(p_*)=0$.

$$\Rightarrow -\gamma*(f_*(i,t)-f_*(i-1,t)+r-p_*)-1=0 \qquad \Rightarrow p_*(i,t)=\frac{1}{\gamma}+f_*(i,t)-f_*(i-1,t)+r \tag{6}$$

Replacing p_* in (5):

$$\boxed{\frac{\partial f_*(i,t)}{\partial t} = -\frac{\gamma}{r} * \exp(-\gamma * [f_*(i,t) - f_*(i-1,t) + \frac{1}{\gamma} + r]) \quad \forall i \geq 1}$$

This differential equation has solutions:

$$\boxed{f_*(i,t) = \frac{1}{\gamma} * \ln(\sum_{j=0}^{i} \frac{\gamma^j * (T-t)^j * \exp(-j * (1+\gamma * r))}{j!})}$$

And we replace in (6):

$$\boxed{p_* = \frac{1}{\gamma} * [\ln(\frac{\sum_{j=0}^{i} \frac{\gamma^j * (T-t)^j * \exp(-j*(1+\gamma*r))}{j!}}{\sum_{j=0}^{i-1} \frac{\gamma^j * (T-t)^j * \exp(-j*(1+\gamma*r))}{j!}}) + 1] + r}$$

The limits of this model is that we have fixed the parameters γ *and* λ and therefore this means that the consumer's behavior is constant during the period T. This is why, for results reflecting the reality, we need to keep T low(at Convertize , we have chosen 6 hours) to cut a day into four parts.

2.2.3 A/B Testing

A/B testing also known as 'split testing' is a common technic used in Marketing and Business Intelligence in order to compare two variations of the same website to find which one performs better.
The important statistical notion to understand A/B Testing is the Statistical Significance. In simple words, Statistical Significance is the probability that the difference between conversion rates of the two variations (A vs B) is the result of real changes in consumer behaviour.

In fact, this is a statistical robust way of proving that our results are reliable. This is why, marketers and online retailers use this level to decide for the 'best variation'.An other interpretation is a measure of quantification of the level of certainty.The most widely used threshold of significance level is 95% for marketers which means that there is 95% that our results are not due to chance but that they really represent consumer's behaviour instead.

In a statistical test[3], we need to confront two hypothesis : H_0 also called the Null Hypothesis vs H_1 the alternative hypothesis. In order to reject or fail to reject H_0, we need to know some parameters : sample size, mean and the variance for each variation pages.

	H_0 is True	H_0 is False
Decision: Fail to reject H_0	Correct	Type II Error
Decision: Reject H_0	Type I Error	Correct

Table 1: Statistical Hypotheses

The Statistical significance is the Probability of NOT rejecting H_0 when it is true i.e the Probability of NOT committing the Type I Error or False Positive.On the other hand, not rejecting H_0 when H_0 is false is called the Type II Error or False Negative .In A/B Testing, Type I Error is the probability that we want to minimise first because it means that we made a claim that is not true. However, in medicine it is often the Type II Error that we are concerned with. For example, in cancer detection, False Positive or Type I Error is to tell to a patient that he/she has cancer when it is not true which will give anxiety to the patient while False Negative or Type II Error would result in the non-treatment of the disease because the we will reject the hypothesis of the patient having cancer which is worse. Since both are errors, we would like to find a way to reduce both:

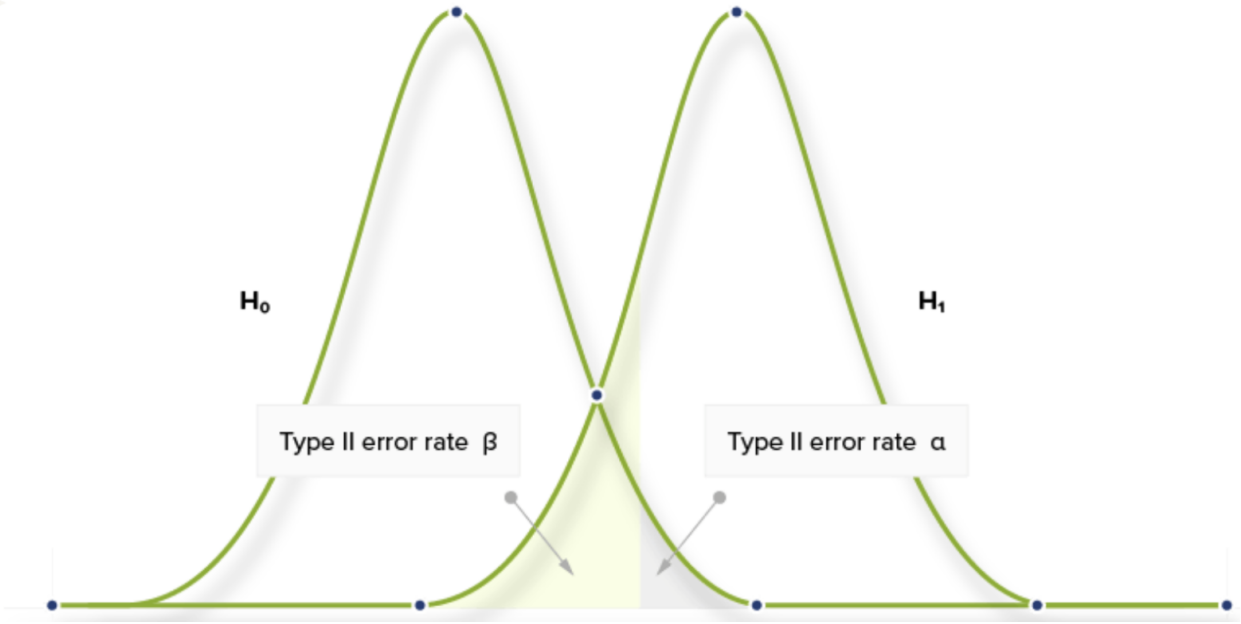

Figure 7: Graph representing Type I and Type II Errors

As we can see, when the parameters are fixed, those two errors are antagonist. Therefore, reducing one results in increasing the other one.Therefore, we need to find a trade off and decide which one to minimise keeping in mind that the more we want to minimise it, the more it will increase the other error when the other values are frozen. The way to reduce both is to have a more robust model with bigger sample size but for marketers and online retailers, increasing sample size has costs.

Let's now set up the model:

I-Classical Model

Hypotheses :

- H_0: There are no differences between the 2 groups, i.e the conversion rates are the same for the 2 variations.
-H_1: There are differences between the 2 groups , i.e the conversion rates are different

Mathematical notations :

$CR_A = \frac{C_A}{N_A}$; $CR_B = \frac{C_B}{N_B}$ where:

-CR_A , CR_B are the conversion rates for variations A resp B
-C_A , C_B are the number of people that converted for variations A resp B
-N_A , $N_B$are the number of visitors(that converted or not) for variations A resp B

Finally, we can reformulate the hypothesis as follow:
-H_0: $CR_A = CR_B$
-H_1: $CR_A \neq CR_B$

Find the right test :

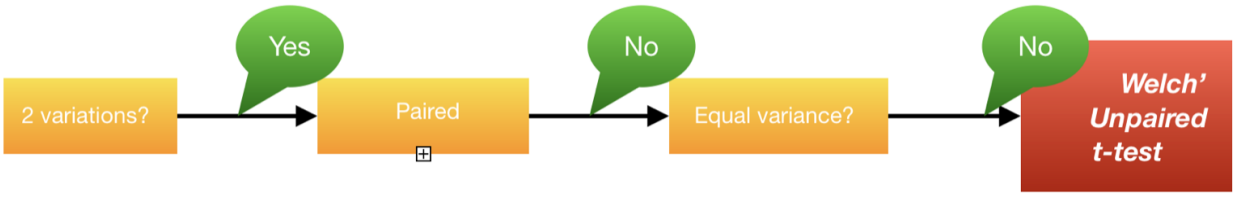

Figure 8: Rules to find the right test

Welch's t-test, also called unequal variance t-test, is a statistical way of testing if two samples of unequal sizes and variances have the same mean. It is a variation of the well-known Student's t-test when the two samples have unequal variances . Using Welch or a Student Test when the variances are equal lead to the same results.

In order to be able to use the Welch t-test, one last hypothesis is primordial: we need to assume that our distributions have Normal distributions. They are often used in Social Sciences to represent real-valued random variables whose distributions are not perfectly known. Also, the probability that a user convert is represented as a Bernoulli distribution of probability $\mathbb{P}(X_A = 1) = CR_A$ and $\mathbb{P}(X_A = 0) = 1 - CR_A$. Equivalently, we defined for variation B: $\mathbb{P}(X_B = 1) = CR_B$ and $\mathbb{P}(X_B = 0) = 1 - CR_B$. Therefore, with this representation, we can assume the convergence to Normal Distribution for large sample sizes.

Finally, we need to distinguish between a one-sided vs two-sided test. Here, since the alternative hypothesis is simply that the conversion rates are different , we need to use a two-sided approach since we do not assume that a conversion rate is only greater or only lower than the other one.

To compute the significance level α in order to get the Statistical Significance $(1 - \alpha)$:

-t-value: $t = \frac{CR_A - CR_B}{\sqrt{\frac{S_A^2}{N_A} + \frac{S_B^2}{N_B}}}$

-Degree of freedom ν: $\nu = \frac{(\frac{S_A^2}{N_A} + \frac{S_B^2}{N_B})^2}{\frac{S_A^4}{N_A^2 * (N_A - 1)} + \frac{S_B^4}{N_B^2 * (N_B - 1)}}$

with $S_{A,B} = \sqrt{\frac{1}{N_{A,B} - 1} * [C_{A,B} * (1 - CR_{A,B})^2 + (N_{A,B} - C_{A,B}) * CR_{A,B}^2]}$

We can therefore use the value of t and ν to get the Significance Level.

II-New Model

Motivations and issues to solve :

Originally, the significance level was computed for fixed sample size but with A/B testing , marketers want to calculate their significance level over time since the beginning of their experiment. All the parameters including the sample size are evolving over time.

During my internship at Convertize, I have decided to compute the significance level every running day in order to give our clients more robust results. This is why, to get accurate results , we are waiting for fixed horizons before displaying the significance level. The algorithm first wait seven running days and at least X visitors(I can not give the exact figures) before computing the significance level. Indeed, I have noticed that there is too much volatility at the beginning of the tests and that the results were not accurate before seven running days. After those seven initialisation days, we give the significance level at the end of every running day, a running day meaning 24 hours of testing .Therefore, because the sample size evolves over time, the Significance Level also changes over time.

One of the most common error in A/B testing is to calculate significance level and to directly draw conclusions. The significance calculation makes an assumption that most of the marketers violate all the time without even noticing: the sample size was already fixed. In fact, the significance level changes with the sample size. Intuitively, higher the confidence level is , higher the sample size should be. This is true , but not enough. When the sample size changes , the conversion rate of the two variation is also changing and this also has effects on the significance level. An other problem also visible in the graphic above is called regression to the mean due to the novelty effect of a variation. Smaller is the difference between the two conversion rates, lower will be the significance level. This is what is happening with the 'regression to the mean' because sometimes the two variations tends to become closer and closer and therefore this can have an impact on our significance level. Understanding those concepts and introducing them in the Significance Level algorithm allow us to make reliable computation.
This example illustrates a pattern that I have noticed in my tests with 2 conversion rates of variations of a webpage (red and grey) . It shows us that we need to wait even if a variation seems better than the other before drawing conclusions.

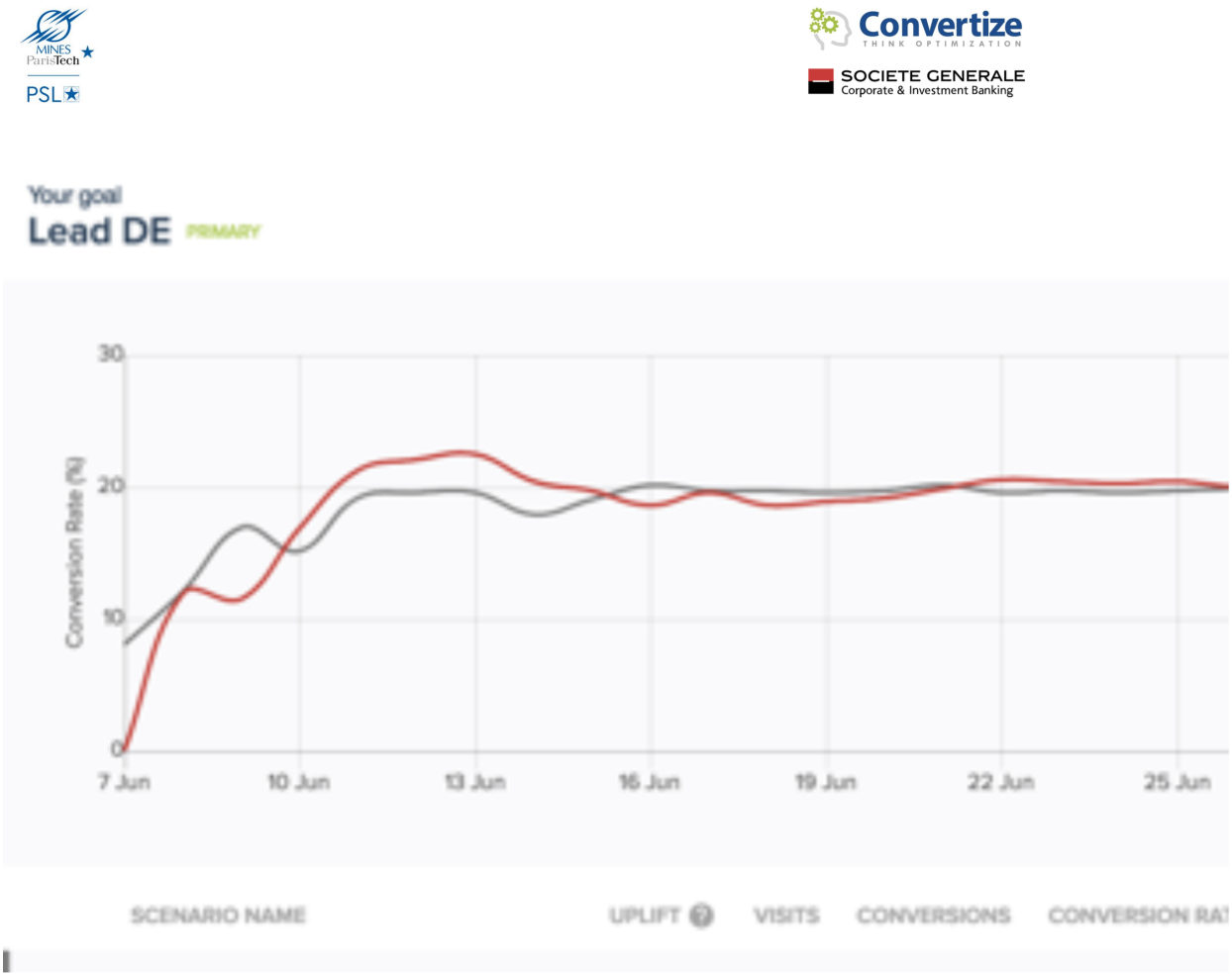

Figure 9: Regression to the mean problem

An other problem is the so-called : 'Saturday is not Tuesday'. As its name suggests, there is no reason to believe that the conversion rate for users who visit a page on a Tuesday would be the same as the one for users who visit the page on a Saturday. This should be taken into account in the algorithm . Online marketers are not taking this problem into account in the computation of the Significance Level. To introduce this issue in our computation and the fact that behaviour of consumer changes over time, we will not compute anymore the conversion rate of a variation as : $CR = \frac{C}{N}$.

Modifications in the Conversion Rate :

Since we are going to compute the significance level every running day, I will set up a weighted average in order to highlight the days before and therefore to give much more weight to what has happened the days just before than what happened a month ago. This will be much more representative of the reality. To illustrate my choices, let's assume that I am a retailer selling online clothes and I want to compare the Conversion Rates of page A selling winter clothes VS page B selling summer clothes. Let's also assume that we are running our tests since January and we are in May now. While the overall conversion rate might be higher for variation A than for variation B because in overall, the proportion of people buying winter clothes during the winter was high, this tendency is changing over time. This is why introducing a weighted conversion rate and giving much more weight to the previous days will give us much accurate results than just averaging the conversions. Indeed, if we give the same weight to the daily conversion rate, it means that the first day of the experiment will have the same impact on the significance level as the last running days which is a wrong assumption. In fact, a lot of events happen over time and more close we are to the last running day more the weight of the conversion rate should be important.

$CR'_{A,B} = \sum_{i=1}^{lastRunningDay}[\frac{i}{\sum_{j=1}^{lastRunningDay} j} * CR_{A,B}(i)]$ with $CR_{A,B}(i)$ the real conversion rate of variation A resp B for the day i.

Here, I gave the basic model of the modification of the weights because of confidentiality issues but at Convertize, we went deeper in the reflexion to set up the weights based on average weights of the conversion rates . Indeed, because of sales, seasons and week-end, the behaviour of the consumers can change and this affect the significance of our results and therefore, the algorithm is programmed in a way to handle those outliers days.

2.2.4 Others

I have been working on other smaller projects involving neuroscience and neuromarketing notions in order to set up a predictive notification system using the concepts of urgency, scarcity and social proof to improve conversion rates. I worked also on a gamification project introducing rankings and awards aspects . In order to get more intuition about neurosciences, the CEO suggested me to read some books [1] ,[5] .

3 Société Générale

January-June 2018 , New-York

3.1 Overview

Société Générale Corporate and Investment Banking operates at all financial market levels around the world in more than 40 countries with more than 12,000 employees. I interned in the Delta One Trading Desk within the Equity Derivatives Team in order to learn Trading and to apply Machine Learning and Data Engineering to Financial Markets. I worked in the front-office surrounded by traders and I assisted them and automated some daily tasks while working on the development of a prediction model and the automation of deal pricing.

3.2 Projects

3.2.1 Modified Dutch Auction Prediction

The goal of this Project was to predict:
-Price of the Modified Dutch Auctions[2] at the expiration called the Aggregated Price
-Proration representing the percentage of shares that the company wanted to buy back that was actually bought
-Reversal for the days after the expiration representing the direction of the stock from a day to another

In Trading, a Modified Dutch Auction is set up by the company in order to buy back a predetermined value of its shares within a price range for a predetermined period of time(usually 1 or 2 months). It is a signal to show to the market that its shares are undervalued and therefore this tend to increase the price of the shares according to laws of supply and demand.This usually happens when the price of the stock has significantly dropped over the last few years. It gives to shareholders a range of price to liquidate part of their shares at a price where they usually can make profit. Therefore, it happens that Modified Dutch Auction are subjected to proration since investors can determine the price for which they want to sell them so the number of tender shares by far exceeds the predetermined number of shares that the company is willing to buy back. Once all tenders are received, the purchase price for the accepted tender shares is the lowest price per share from among the specified offer range at which the shares have been tendered that will allow to purchase the maximum number of shares. This is tricky for share tenders because they need to find a trade off between maximising their profit ,therefore tend shares at a high price , and being accepted, therefore tend shares at the lowest price where the maximum number of shares is accepted.

I-Creating a Modified Dutch Auction Database

This was a long process gathering informations from different sources: mostly from Bloomberg and internal Société Générale APIs. In order to get the right features for my different Modified Dutch Auctions, I spent a lot of time with the Traders in order to understand what are the variables that for sure impact the price and the reversal of those deals and what are the other variables that might potentially have an impact. After a listing of those variables (approximately 70), it was the time to historically gather informations about those Modified Dutch Auction . After having the deals, we need to fill the dataframe with the features input. With Bloomberg open API (pybbg in Python) , we have access to a lot of historical data in order to fill the data frame . Thanks to the SG APIs, we were also able to gather some of the missing columns. Finally, we have had access to a consistent data frame with about 65 columns and X deals. Unfortunately, because of a lot of confidentiality in a Trading desk I can not show you the data frame and the name of the features that the Traders consider important for those deals but basically each row contains informations about a certain deal with both qualitative and quantitative features. The qualitative features have been transformed using dummies variables in order to work with an homogeneous data frame containing only numerical features.

II-Aggregate Price Prediction :

II-1-Data Preparation :

We need to make our data frame suitable for our analysis in the sense that we need to take into account in our model only the variables that we will have access to at the time of the prediction. Therefore, I have first deleted all the variables which I can not rely on . For example , let's say I need to predict the Aggregate Price at day t therefore , I can not rely on all the features available for $day > t$. Also, I need to identify the target variable called 'AGG PP' in my data frame. Finally, we do feature standardization for the feature in the data to have zero mean and unit variance in order to get all the data on the same scale and this implicitly weights all feature equally in their representation.

II-2-Data Visualization

Let's first plot the correlation matrix with the associated python code:

```
In [ ]: import numpy as np
def correlation_matrix(what='numerical'):
    #I can't show you clean_and_same_format because it cleans the dataframe for the aggregated price with colum names
    df=clean_and_same_format(what)
    if what=='numerical':
        df=pd.DataFrame(StandardScaler().fit_transform(df),columns=df.columns)
        df=df.corr()
    else:
        df=df.corr('spearman')
    fig=plt.figure()
    ax1=fig.add_subplot(111)
    cmap=cm.get_cmap('jet',30)
    cax=ax1.matshow(df,interpolation='nearest',vmin=-1,vmax=1,cmap=cmap)
    ax1.grid(True)
    plt.title('DUTCH Correlation')
    labels=df.columns.tolist()
    ticks=np.arange(0,len(df.columns),1)
    ax1.set_xticks(ticks)
    ax1.set_yticks(ticks)
    ax1.set_xticklabels(ticks,fontsize=7)
    ticks=(np.arange(0,len(df.columns),1)).tolist()
    res=['']*len(ticks)
    for i in range(len(res)):
        res[i]=str(ticks[i])+ ' : '+labels[i]
    ax1.set_yticklabels(res,fontsize=7)
    fig.colorbar(cax)
    plt.show()

correlation_matrix()
```

Figure 10: Python Correlation Code

A great way to start the data visualization process is to plot the general correlation matrix in order to understand the relationship between the different features and the target in our dataset. I have blurred the name of the features but it does not change the way to understand the matrix.

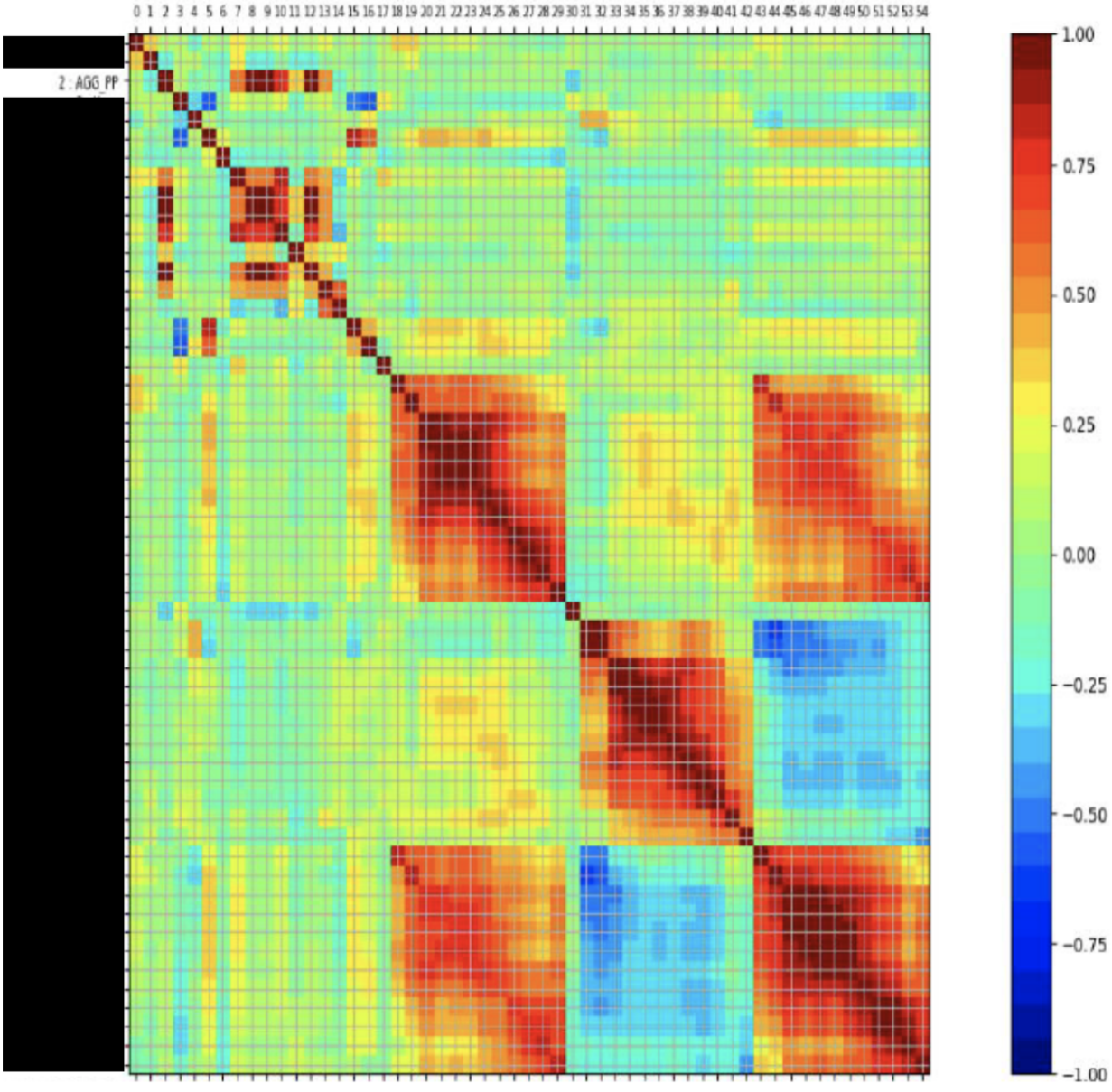

Figure 11: Correlation Matrix of Modified Dutch Auction

An other useful representation of the data is the scatter plot. Here, I will show you the scatter plot of part of my data frame :

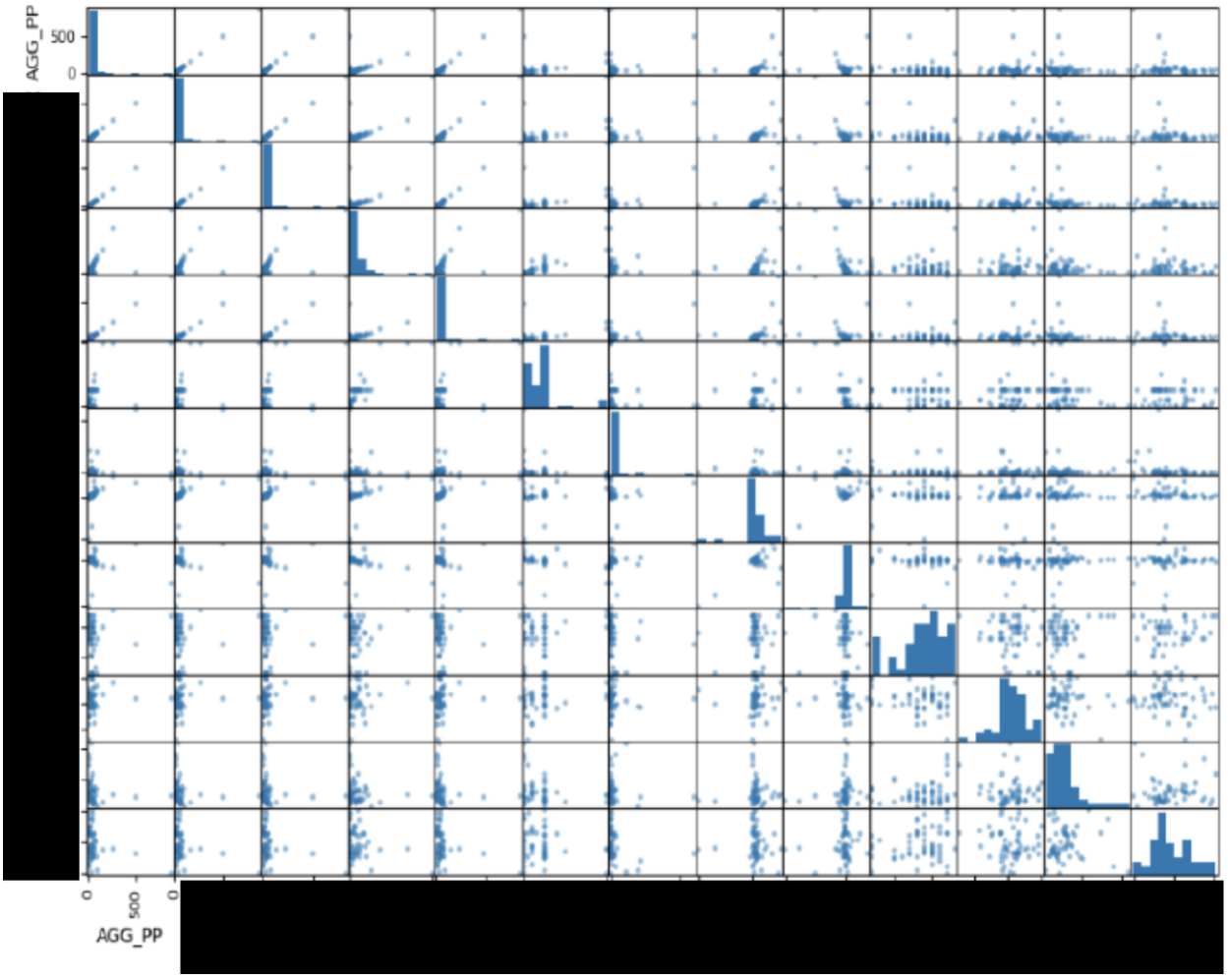

Figure 12: Correlation Matrix of Modified Dutch Auction

This data visualization step was primordial for this study because one of the constraint was to keep the variables intact because the traders wanted to know the importance of each feature that I will use in the prediction model at the end. Therefore, I was not able to use a PCA in order to reduce the dimensionality of my dataset because by orthogonal transformation I would have lost the meaning of my features . Therefore, this visualization step allow me to understand which variables were correlated to my target and therefore which ones to potentially keep in my models. I have now reduced my model to only about 10 features.

II-3-Models

Since we are predicting[7][8] the Aggregate Price, a continuous variable, we want to use a Regression Model for Continuous Variables. Also, because we know the target, each example in the model is a pair of (Input,Target) so this is a supervised learning model. The common models for this specific task are: Linear Regression, Ridge Regression, Lasso Regression, Gradient Boosting Regression and Random Forest Regression.

a- Linear Regression :

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 \Leftrightarrow \min_{\beta \in \mathbb{R}^p} \|y - x\beta\|_2^2 \Rightarrow \hat{\beta} = (x^T x)^{-1} x^T y$$

For the problem to be well defined, we need the number of target that we want to estimate to be much greater than the number of features.

Advantages:

-Very simple method and intuitive to use and understand
-Produces easily interpretable solutions

Disadvantages:

-Sensitive to outliers when the dataset is small
-Useful to describe only linear relationship between variables

b- Ridge Regression :

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + lambda \sum_{j=1}^{p} \beta_j^2 \Leftrightarrow \min_{\beta \in \mathbb{R}^p} \|y - x\beta\|_2^2 + \lambda \|\beta\|_2^2$$

It is a regularised Linear Regression with lower coefficients because of the penalisation term.
λ control the strength of the penalty terms. Indeed, $\lambda = 0$ corresponds to the Linear Regression and higher λ is, the more we shrink the coefficient and the penalty is important.

Advantages:

-Better compromise bias and variance
-Reduces overfitting

Disadvantages:

-Not able to shrink a coefficient to 0 unlike Lasso.

c- Lasso Regression :

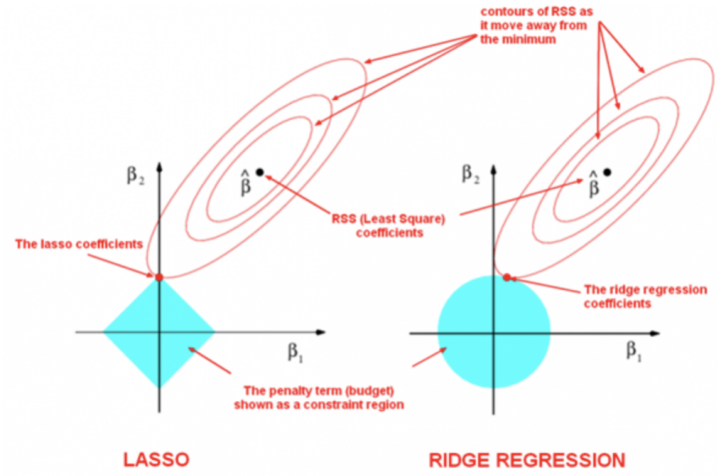

Figure 13: Geometrical Explanation of coefficient shrinkage

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^{p} \|\beta_j\|_1 \Leftrightarrow \min_{\beta \in \mathbb{R}^p} \|y - x\beta\|_2^2 + \lambda \|\beta\|_1$$

The difference between Lasso and Ridge is the shape of the constraint region. This subtile difference can have real consequences in the model because the shape in Ridge lets us select either no coefficient or all the coefficient whereas with Lasso , it performs both shrinkage and variable selection.
Advantages:

-Performs easily shrinking coefficients and therefore eliminates inputs not contributing
-Computational efficiency in high dimensions

Disadvantages:

-Not good for low features dataset.

d- Gradient Boosting :

Gradient Boosting is a new method introduced in the last 20 years and was originally designed for classification problems but rapidly extended to regression problems. The idea is to 'boost' weak learners into a strong one. A weak predictor model can be any model that performs just a little better than a random model.
$F(x) = \sum_{m=1}^{M} \gamma_m h_m(x)$ with $h_m(x)$ the weak learners that are chosen at each stage to minimise the loss function (I chose the squared error in my model) with respect to the model F_{m-1}.
$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) = argmin_h \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h(x))$
$\Leftrightarrow F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla L(y_i, F_{m-1}(x_i))$.
Finally,$\gamma_m = argmin_\gamma \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) - \gamma \frac{\partial L(y_i, F_{m-1}(x_i)}{\partial F_{m-1}(x_i)}) \forall m \geq 1$.

In summary , the algorithm works as follow: we use simple models to analyse the errors. Those errors are then difficult to model and we focus on them to improve the next predictions. At the end, we combine all the weak predictors with the weight optimised.

Advantages:

-Learn non-linear relationships
-Robust to outliers

Disadvantages:

-Easier to overfit
-Training is longer since the trees are built sequentially

II-4-Train,Evaluate and Fine tune the Models

In order to evaluate our model, we need to split our dataset intro three parts: Training,Validation and Testing set. Usual conventions are 80 to 85% for Training-Validation and the remaining for Testing. Indeed, Training our model into more than that usually lead to a problem called overfitting. On the contrary, if our Test set is too big compared to the Training set , this can lead to underfitting the model. Cross-validation is a resampling procedure used to evaluate the robustness of our model . The process is exactly the same as Training/Testing and there is a parameter k called that splits the Training data into k parts to Train the model on (k-1)parts and evaluate on the remaining part. The figure below illustrates the notion of overfitting and underfitting and the fact that having a balanced model allow us to generalise well on unseen data.

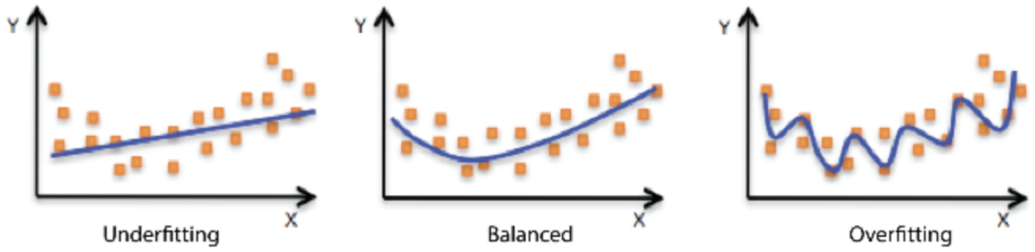

Figure 14: Intuitive Explanation of Overfitting and Underfitting

Let's see the performances of the models:

	Linear Regression	Lasso Regression $\lambda = 0.0001$	Ridge Regression $\lambda = 0.0001$	Gradient Boosting
Error Metrics	$R^2//RMSE$	$R^2//RMSE$	$R^2//RMSE$	$R^2//RMSE$
Test size(%)				
10	0.9701//0.6	0.9675//0.6	0.9772//0.6	0.8746//1.6
20	0.9765//1.3	0.9888//1.26	0.9788//1.28	0.5742//3.4
25	0.9904//1.3	0.9924//1.1	0.9912//1.2	0.5448//2.8

Figure 15: Performances on Testing after Cross-Validation

Let's find the parameter α that maximizes the performance on the validation set:

```
In [ ]: test_size=[0.1,0.2,0.25]
for i in range(3):
    X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size=test_size[i], random_state=0)
    ridge = Ridge(alpha=alpha_to_keep)
    ridge.fit(X_train,y_train)
    y_pred = ridge.predict(X_test)
    ridge_mse = mean_squared_error(y_pred,y_test)
    ridge_rmse = np.sqrt(ridge_mse)
    print '%.4f' % ridge.score(X_test, y_test) + '//'+ '%.4f' %ridge_rmse

alpha is the parameter that we want to use the penalise the coefficients in the regression(corresponds to our lambda)

In [3]: alphas =np.logspace(-4,-1,20)
print alphas

[ 0.0001      0.00014384  0.00020691  0.00029764  0.00042813  0.00061585
  0.00088587  0.00127427  0.00183298  0.00263665  0.00379269  0.00545559
  0.0078476   0.01128838  0.01623777  0.02335721  0.03359818  0.0483293
  0.06951928  0.1        ]

In [9]: scores = [ridge.set_params(alpha=alpha).fit(X_train, y_train).score(X_test,y_test)for alpha in alphas]
alpha_to_keep = alphas [scores.index(max(scores))]

Finally, we have trained our model for different values of alpha and we will keep the model with the highest alpha.
```

Figure 16: Fine-Tuning of α in Ridge Regression

To understand the results, we need to understand the metric that we have used:
-R^2 also called coefficient of determination is a statistical measure to evaluate how close the data are to the fitted regression line. It gives the percentage variation in the output explained by the inputs. It also corresponds to the improvement from the regression model compare to the mean model:
$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$

-$RMSE$ is the root mean squared error. It is the standard deviation of the prediction error. Basically, it measures how far the points are from the regression line:
$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$.

As we saw when we chose the model, large values of λ corresponds to strong regularisation while low values tend to converge to the coefficients of Linear Regression. This value of λ is close to 0 and this is why in the Performance comparison , we tend to see similar results between Lasso, Ridge and Linear Regression. Notice however that the results are closer between Ridge and Linear Regression. We will understand why when we will analyse the importance of the features in our model.Before, we can conclude than keeping the Lasso Model seems to be the good choice based on the performance analysis. However, even the Ridge Model and the Linear Model are giving great results!

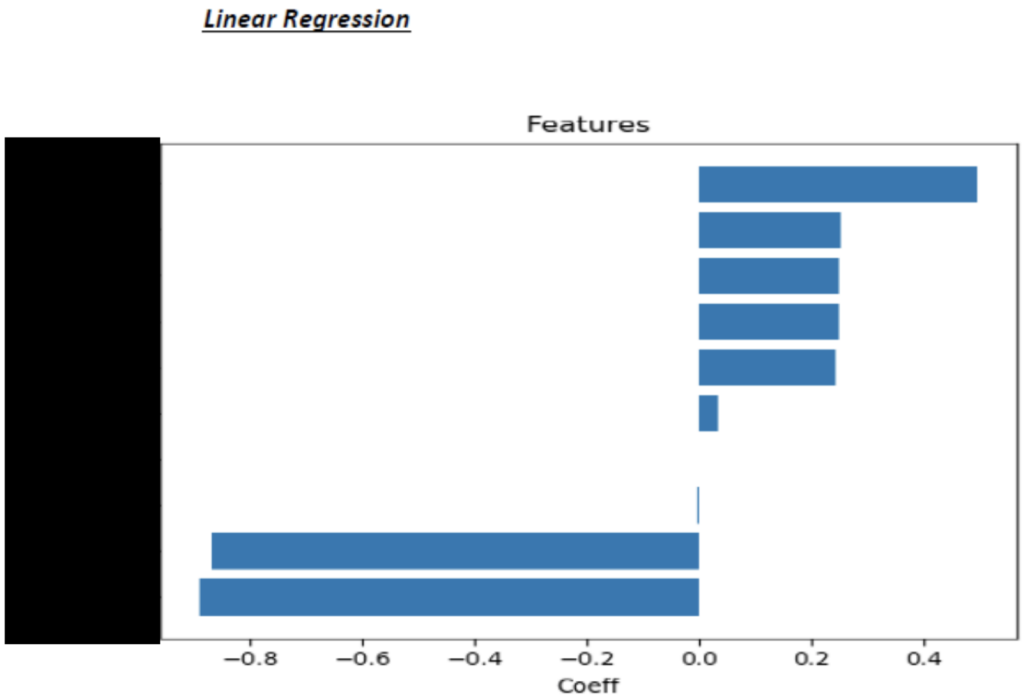

Figure 17: Features Importance Linear Regression

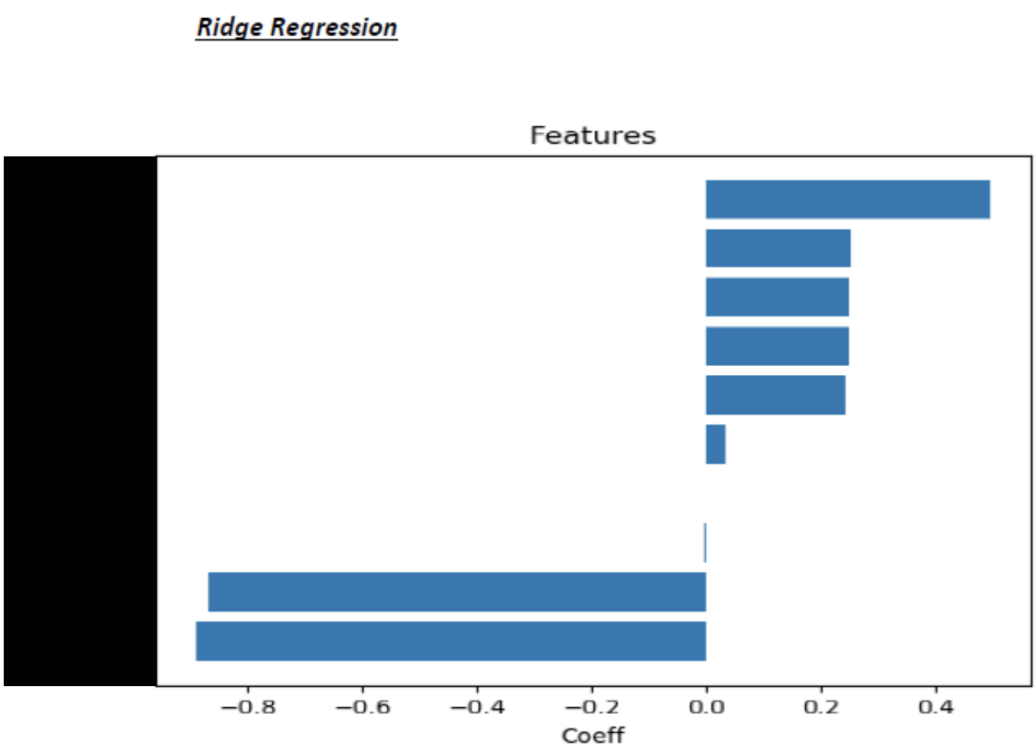

Figure 18: Features Importance Ridge Regression

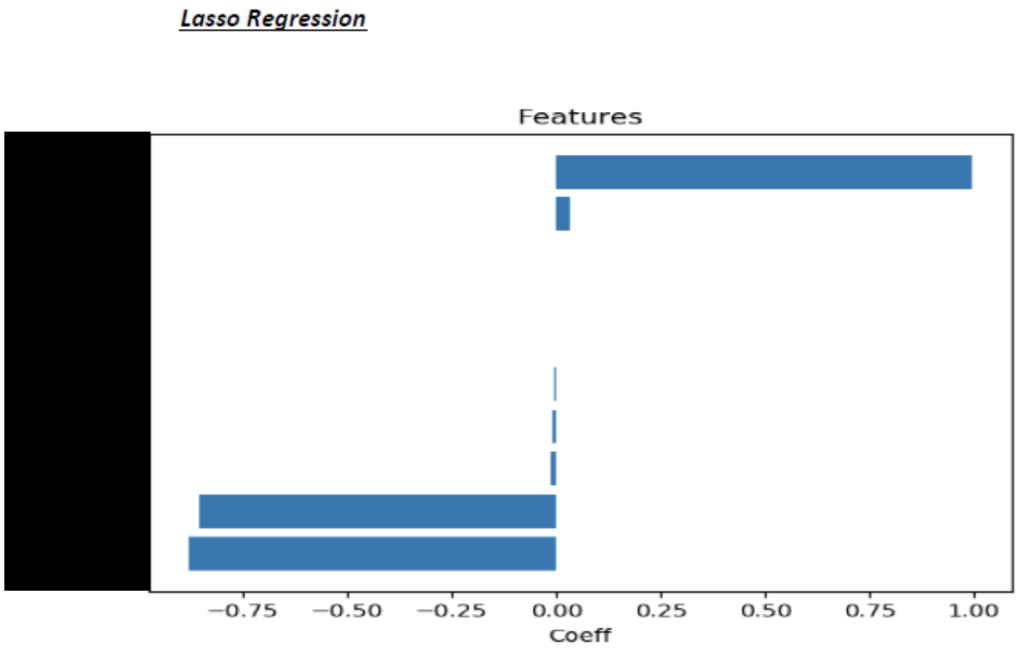

Figure 19: Features Importance Lasso Regression

Those results are quite interesting! As we can see Lasso Regression clearly penalises coefficient to zero while Linear Regression and Ridge Regression have almost the same coefficients(not exactly I checked) .This also validates what we have showed in the figure explaining how Lasso could shrink coefficients to 0 and not Ridge.

Now, let's save the model:

```
In [ ]: df_pred = pd.DataFrame(columns=col_to_keep) #The new dataframe contains only the columns used in the model
        data_deal_1 = [?,?,?,?,?,?,?,?,?,?,?] #This are numerical values but I can not give the values
        data_deal_2 = [?,?,?,?,?,?,?,?,?,?,?
        df_pred.loc[0] = data_deal_1
        df_pred.loc[1] = data_deal_2
        model = pickle.load(open(filename,'rb'))
        result = model.predict(df_pred)
        print result
```

Figure 20: Saving the model

Before believing my model, the Traders challenged me on those 2 deals to check my results:

Methods	Lasso Regression	Ridge Regression
	Real Price Predicted Price %Error	Real Price Predicted Price %Error
Deal 1	20.5 20.86 1.76	20.5 20.88 1.85
Deal 2	192 199 3.65	192 199.2 3.75

Figure 21: Results comparison

As we can see, the model gives really promising results on those 2 new deals!

III-Proration Prediction

In order to predict the proration , we will follow the process established in the first part. However, we will see that the prediction for the proration will be much more challenging. The good thing is that we have already prepared the data frame so we will not have to go through this long process.

III-1-Data Preparation

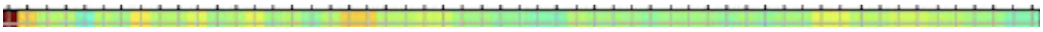

Figure 22: Correlation Proration(first square) with other variables

This is the row of the matrix corresponding to the correlation between the Proration and the other features.As we can see, the features are not well correlated to the Proration, so using linear models will be quite difficult. To emphasise this, I will plot the graph points of the most correlated variables to the Proration.

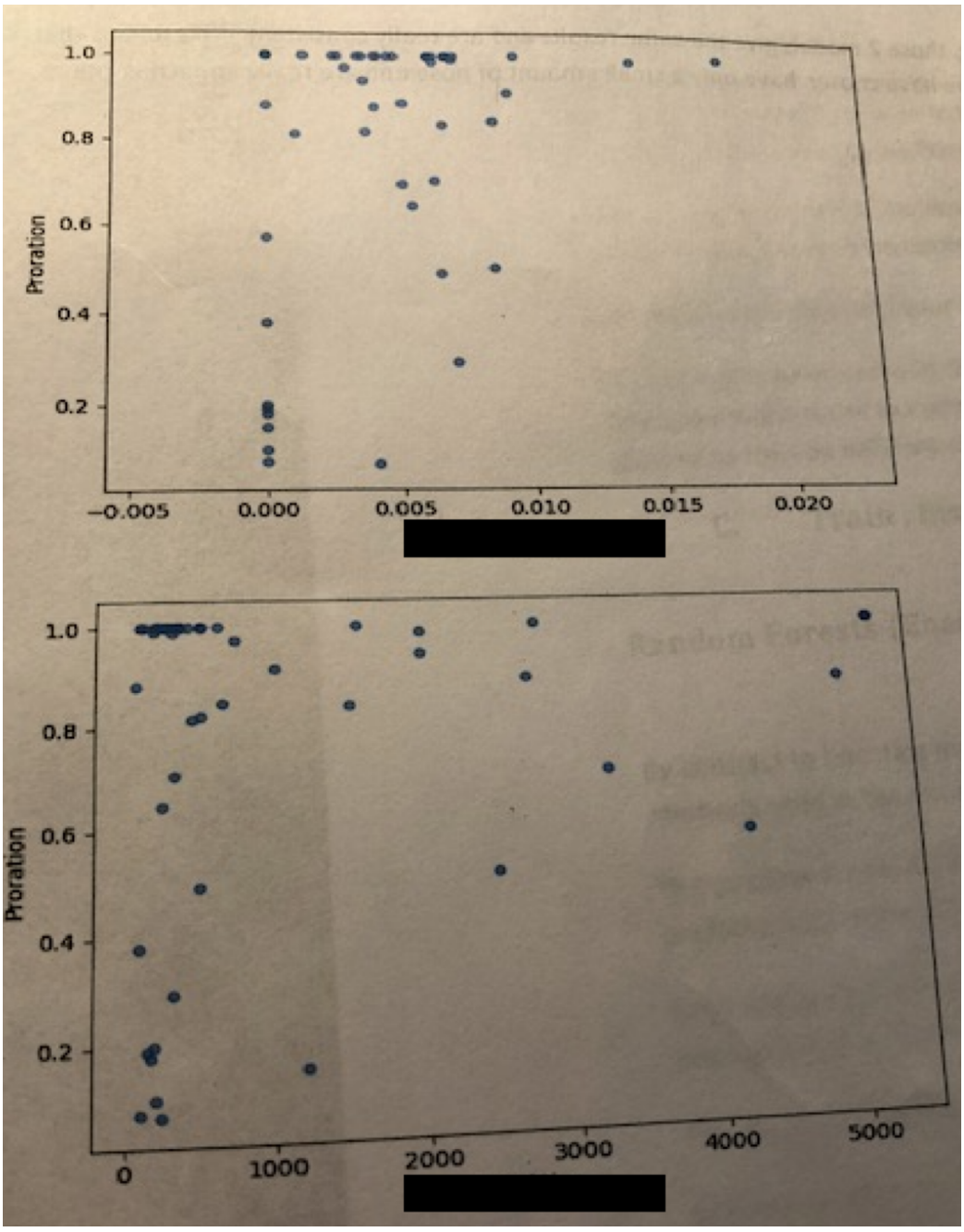

Figure 23: Proration with respect to

III-2-Model

Random Forests :

By contrast to boosting methods (cf Gradient Boosting), Random Forests are a subset of averaging methods used in the ensemble methods. We build different estimators independently(the decision trees) and average them to get the predictions. An important parameter in random forest is the number of trees we want to build before averaging their results. In this prediction, I will try to fine-tune the model to chose the optimal number of trees The algorithm of a decision-tree works as follow:
We build the tree greedily from top to bottom and each split has the goal to maximise information gain where the information gain is the difference between the error before split and error after split.
Our goal is to find a tree $f(x)$ such that: $\min_f \sum_{i=1}^{n}(f(x_i) - y_i)^2$.
To do that, we find the best split at each node and repeat the process until the maximal number of features and the maximal depth of the tree are not reached. Finally , select the feature to be the root of the subtree and remove it.

Advantages:

-No overfitting and less variance when using RandomF instead of Decision Trees
-Good model for non linear problems

Disadvantages:

-Difficult to interpret when there are a lot of trees.

III-3-Train, Evaluate and Fine Tune the Models

We will keep a simple model in order to predict the Proration: After tuning, I found that a Random Forest was the model giving the best performances. Here is the feature importance and one of the trees that I plotted:

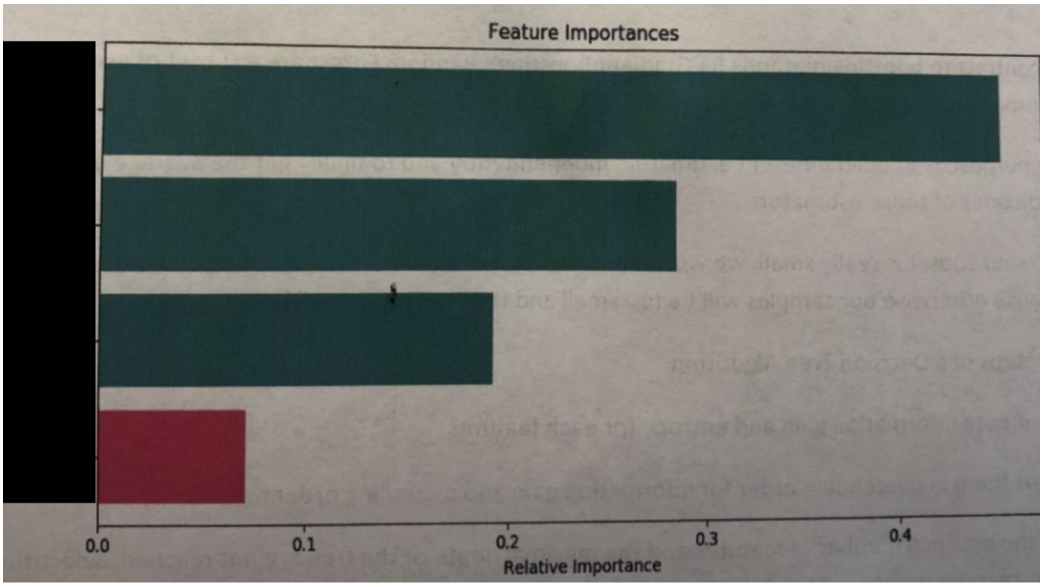

Figure 24: Feature Importance

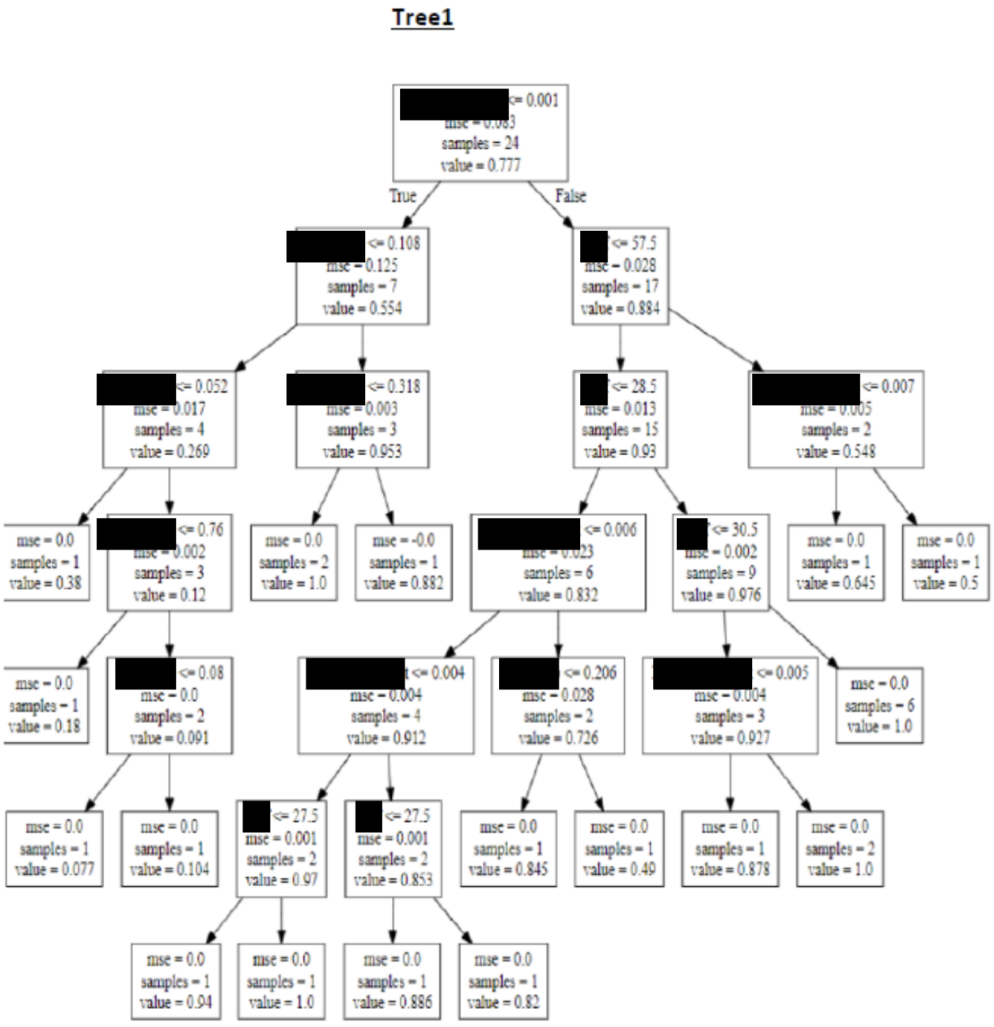

Figure 25: First Tree

Let's now evaluate the performance to explain why I kept 22 Trees:

	Random Forest 10 Trees	Random Forest 22 Trees
Error Metrics	R^2//$RMSE$	R^2//$RMSE$
Test size(%)		
20	0.32//0.29	0.81//0.11

Figure 26: Model Comparison

III-4-Predictions

We will do the predictions again on the same deals as before :

Methods	Lasso Regression
	Real Proration Predicted Proration %Error
Deal 1	1 1 0
Deal 2	1 0.962 3.95

Figure 27: Proration Evaluation

Again, the results were very satisfactory which means that the model seems to be general and well adapted for this prediction.

IV-Reversal Predictions

After the expiration of the Modified Dutch Auction, we want to be able to predict the Reversal during 5 days. In order to give reliable results , instead of predicting a continuous stock price for this task, I have decided to take a binary approach . I will instead predict for those 5 days if the stock is going to increase or decrease compared to the day before. This is actually what we need to take our positions in the market because we are going to buy or sell some shares based on what we predict. We are not looking for a precise price.

IV-1- Data Preparation

Here, we had to change the data frame in order to incorporate other informations because the Reversal happens at the end of the Modified Dutch Auction. Therefore, we know the Proration , how the stock evolved during the auction and other risk informations that I took from SG API about some hedging and indexes parameters that the Traders wanted to introduce in the analysis.

IV-2 Data Visualization

I am again going to plot the correlation matrix for the target and some variables that seems to be important to determine the reversal. The variable 'Dutch Expi+1d' is the binary target that we want to predict.

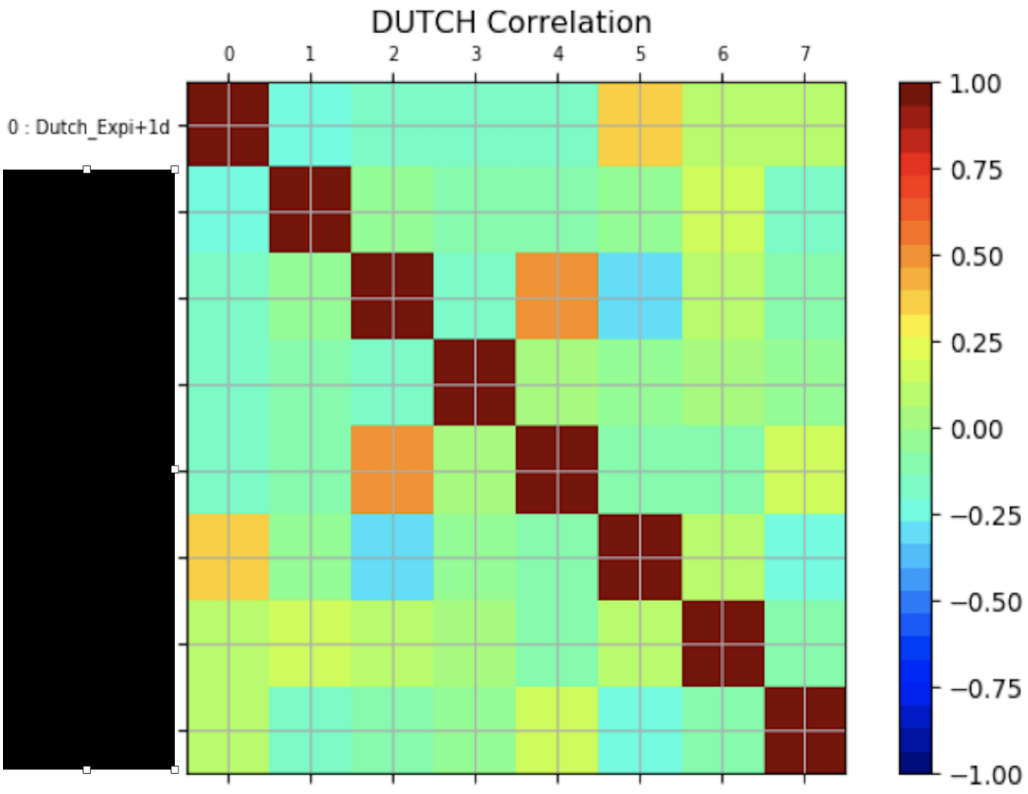

Figure 28: Reversal Correlation Matrix

IV-3 Models

Now that we have a binary target, we can not apply the previous models useful for regression tasks. We will introduce 4 new Models and the Classification version of Random Forests.

a- Gaussian Naïve Bayes Classifier :

It is a supervised learning algorithm based on Bayes' Theorem using a "naïve assumptions": independence between each pair of features. A naÏve Bayes classifier considers each of these features to contribute independently to the probability of the class. This is where the correlation matrix can also be useful in the feature selection for this model.
Let's denote y the binary classification output and a vector $x = (x_1, ..., x_n)$ representing the n independent features of our model.
Using Bayes' Theorem:$\mathbb{P}(y \mid x = (x_1, ..., x_n)) = \frac{\mathbb{P}(y)\mathbb{P}(x|y)}{\mathbb{P}(x)}$.
$\mathbb{P}(y \mid x = (x_1, ..., x_n)) = \frac{\mathbb{P}(y)\prod_{i=1}^{n}\mathbb{P}(xi|y)}{\mathbb{P}(x_1,...,xn)}$ with $\mathbb{P}(x_1, ..., x_n)$ a constant.
$\Rightarrow \mathbb{P}(y \mid x = (x_1, ..., x_n)) \propto \mathbb{P}(y)\prod_{i=1}^{n}\mathbb{P}(xi \mid y)$
Therefore, $\hat{y} = argmax_y P(y)\prod_{i=1}^{n} P(x_i \mid y)$.
The last common assumption is that: $(x_i \mid y) \sim \mathcal{N}(\mu_y, \sigma_y^2)$.
Advantages:

-Simple and intuitive

Disadvantages:

-Naïve because independent variable is a very strong assumption
-Quite big dataset to have a significant estimation of the probability

b- k-Nearest Neighbours Classifier :

It's the easiest algorithm to understand, yet powerful in most of the cases. In classification, the output is simply the majority vote of the k chosen neighbours if the weight are uniforms. A common approach is to chose a distance weight approach which gives proportionally inverse weights to the distance from the point the we want to predict.
Advantages:

-Good detection of linear and non-linear patterns

Disadvantages:

-Very sensitive to outliers when k is small

c- Logistic Regression :

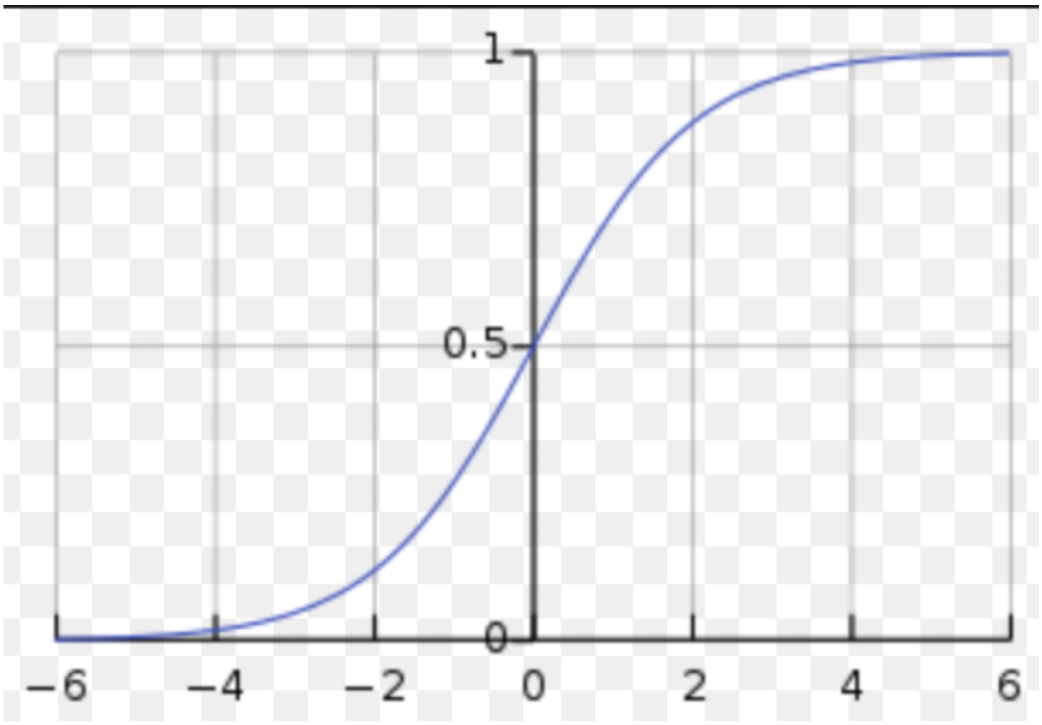

Figure 29: Logistic Regression Function

Logistic Regression is quite similar to Linear Regression but the biggest difference lies in their usage. While Linear Regression is used to predict continuous output, Logistic Regression is used for the purpose of classification. Since we are prediction a binary value 0 or 1 for the classification purpose, we need to modify the function in order to get values between 0 and 1.
We use a sigmoid function :
$h = g(z) = \frac{1}{1+\exp(-z)}$ with $z = \theta_0 + \sum_{i=1}^{n} \theta_i x_i$.
Also , we need to modify the cost function that we minimise using gradient descent in order to get the values of the parameters . Indeed, the classic cost function:$C(h_\theta(x), y) = \sum_{i=1}^{n} (h_\theta(x_i) - y_i)^2$ is not convex and therefore might not converge .
Hence, we modify the cost function: $C(h_\theta(x), y) = -\frac{1}{n}[\sum_{i=1}^{n}(y_i \log(h_\theta(x_i)) + (1 - y_i) \log(1 - h_\theta(x_i))]$. Intuitively, since $h_{\theta(x_i)}$ represents $\mathbb{P}(y_i = 1 \mid \theta, x)$,this cost function penalises a lot the false positive and the false negative in the sense that if $h_{\theta(x_i)} = 0$ while $y_i = 1$, the cost goes to $+\infty$ and similarly when $h_\theta(x_i) = 1$ and $y_i = 0$.

Advantages:

-Well suited: designed for binary categorical variable and all explanatory variables
-Form of the gradient similar to linear regression so very easy to compute

Disadvantages:

-Sensitive to outliers

d- Support Vector Machine :

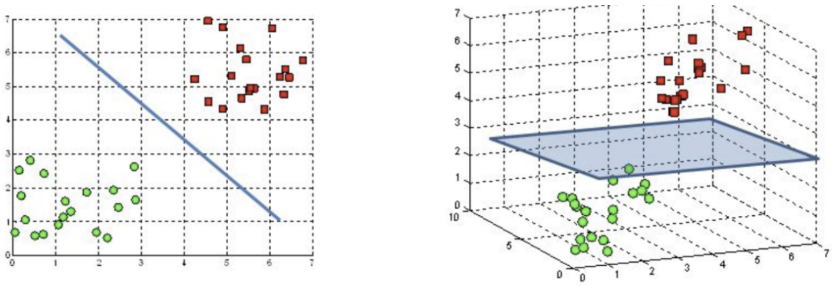

Figure 30: Support Vector Machine

Support Vector Machine also called Large Margin Classifier is a supervised learning algorithm designed to chose the hyperplane that maximise the distance from the two classes of our training set. To do that, we maximise the distance between the nearest points of both classes and the hyperplane: $\min_\theta C \sum_{i=1}^{m} y_i * cost_1(\theta^T f_i) + (1 - y_i) * cost_0(\theta^T f_i) + \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$ with $f_i = similarity(x, l_i)$.
There are different types of function for the similarity but we can think about the kernel (similarity function) of the measure of how close our input x is to l_i (1 if very close and 0 otherwise). l_i are simply the training inputs : l_1 is x_1 ;...;l_m is x_m.
We can easily relate SVM to Logistic Regression and the parameter C as the inverse of λ because our cost function in SVM is in the form : $C * A + B$ whereas it is : $A + \lambda * B$ in Logistic Regression. Therefore, large C results in lower bias and high variance which is the result of small λ.

Advantages:

-It depends only on a subset of points because of the maximisation distance between the closest points
-Robust to noise

Disadvantages:

-Kernel selection
-High complexity to define the hyperplane

e- Random Forest Classifier :

It is the equivalent of Random Forest Regression for classification problems .

IV-4-Train,Evaluate and Fine tune the Models :

We follow exactly the same process as before and fine-tune the models(γ,C for SVM , k for Nearest-Neighbors and the number of trees for Random Forests) to get better results.
Since we will follow the same processes for all the reversal , I will show the performances and not rewrite the steps for everyday.
The important thing to note is that I trained my Gaussian Naïve Bayes with less features than the other models because of the assumption of independent variables so I had to remove the dependent features before training the model.

1-Performance and Predictions for Reversal Day 1

Models	Gaussian NB	kNN k=7	kNN k=5	SVC $\gamma = 2\ C = 1$	RF
Error Metrics	R^2	R^2	R^2	R^2	R^2
Test size(%)					
10	0.6	0.78	0.4	0.6	0.8
20	0.6	0.84	0.5	0.625	0.8
25	0.64	0.9	0.5	0.6	0.875

Figure 31: Models comparison for Reversal day 1

The chosen metrics for kNN is inversely proportional distance.

Models	kNN, $k = 2$
	Real Reversal Predicted Reversal
Deal 1	0 0
Deal 2	1 1

Figure 32: Results for Reversal day 1

2-Performance and Predictions for Reversal Day 2

Models	Gaussian NB	kNN	SVC $\gamma = 2\ C = 1$	RF
Error Metrics	R^2	R^2	R^2	R^2
Test size(%)				
10	0.6	0.4	0.6	0.6
20	0.5	0.5	0.5	0.8
25	0.6	0.5	0.6	0.9

Figure 33: Models comparison for Reversal day 2

Models	RF
	Real Reversal **Predicted Reversal**
Deal 1	0 0
Deal 2	1 1

Figure 34: Results for Reversal day 2

3-Performance and Predictions for Reversal Day 3

Models	Gaussian NB	kNN	SVC $\gamma = 2\ C = 1$	RF
Error Metrics	R^2	R^2	R^2	R^2
Test size(%)				
10	0.4	0.5	0.6	0.8
20	0.5	0.625	0.625	0.875
25	0.4	0.7	0.7	0.9

Figure 35: Models comparison for Reversal day 3

Models	RF
	Real Reversal Predicted Reversal
Deal 1	0 0
Deal 2	0 0

Figure 36: Results for Reversal day 3

4-Performance and Predictions for Reversal Day 4

Models	Gaussian NB	kNN	SVC $\gamma = 2\ C = 1$	RF
Error Metrics	R^2	R^2	R^2	R^2
Test size(%)				
10	0.8	0.5	0.4	0.8
20	0.8	0.625	0.4	0.875
25	0.825	0.7	0.5	1

Figure 37: Models comparison for Reversal day 4

Models	RF	Gaussian NB
	Real Reversal **Predicted Reversal**	**Real Reversal** **Predicted Reversal**
Deal 1	1 0	1 0
Deal 2	0 0	0 0

Figure 38: Results for Reversal day 4

Since the 2 models had good accuracy scores and they both predict the wrong binary value , it seems that the Reversal on day 4 was unlikely to happen based on the models that we constructed. I introduced 2 models here that had good results on the validation and testing set and that produced both the wrong output.

5-Performance and Predictions for Reversal Day 5

Models	Gaussian NB	kNN $k = 5$	SVC $C = 1$	RF
Error Metrics	R^2	R^2	R^2	R^2
Test size(%)				
10	0.4	0.8	0.6	0.8
20	0.4	0.8	0.5	0.825
25	0.4	0.8	0.5	0.95

Figure 39: Models comparison for Reversal day 5

Models	RF	Gaussian NB
	Real Reversal Predicted Reversal	Real Reversal Predicted Reversal
Deal 1	0 0	0 0
Deal 2	0 0	0 0

Figure 40: Results for Reversal day 5

Conclusions

The models that we have used are working quite well on those two deals. The data analysis part gave the traders all the key elements to understand the impact of the features in the different targets that they wanted to predict. This model has been set up on SG Prediction APIs to be used by the Traders and to give them a good vision on what is going to happen in the deal. They also have the possibility to plot the matrix correlations as well as the feature importances for all the models that they use. In order to use the model, they just need to enter the name of the deal and the deal id. The rest of the job is already done...

3.2.2 Others

After this successful project , I was asked to complete it by adding a detection algorithm in order to detect the Dutch Auction before their announcement on Bloomberg. To do so, I used web scrapping algorithms and internal stock options websites with their RSS Feed in order to scan twice daily before and after the close of the market if there is an announcement of a Dutch Auction. To do so, I introduced a few keywords and everytime that the keyword is detected, the corresponding article is given back to me. Therefore, by the end of the day we need to manually recheck those ten articles to see whether or not there is a real Modified Dutch Auction appearing.

I have also been working on a Data Engineering project in order to create a Pricing Tool in Python for the team and an algorithm to generate the Pay and Receive Returns list automatically and send them to the brokers every morning.

4 Conclusion

This gap year has been full of benefits for me because I finally found my path. Indeed, my first internship in London made me realize that I wanted to stay in the area of Machine Learning . Having the opportunity to work in Data Science not only in ecommerce sector but also in finance during my internships allow me to strengthen my skills and to see a wide variety of useful technics mandatory to become a good Data Scientist. Finally, since my earliest childhood I was attracted by medicine, therefore, when I discovered the biomedical area of artificial intelligence , I knew that this was the perfect combo for me. It's the perfect mix between strong mathematical and computer science skills and the medical sector. To start working in this area, I moved back to Paris and enrolled to the Data Challenge launched by Institut Gustave Roussy. It was a breast MRI cancer and lesion classification evaluated base on the AUC ROC curve. A part of the code is available on my github . I used Convolutional Neural Networks and Transfer Learning in order to train my algorithms and I finished second of the challenge. I have been therefore looking for my end-of-studies internship in Artificial Intelligence Research in the Biomedical sector and I have just been accepted in the laboratory Memorial Sloan Kettering Cancer Center in New-York. After this internship, I wish to pursue a PhD in Artificial Intelligence to pursue my research and I hope to have an impact in the future in the health industry.

References

[1] Nir Eyal. *Hooked*. 2013.

[2] Investopedia. Modified dutch auction.

[3] Ning-Zhong Shi Jian Tao. *Statistical Hypothesis Testing Theory and Methods*. 2008.

[4] Philippe Aimé Jochen Grünbeck, Yanis Tazi. *A/B Testing - The Hybrid Statistical Approach: Using Frequentist and Bayesian approach*. 2018.

[5] Daniel Kahnemann. *Thinking Fast and Slow*. 2011.

[6] MIT. Poisson process.

[7] Andrew Ng. Machine learning.

[8] Jerome Friedman Trevor Hastie, Robert Tibshirani. *The elements of Statistical Learningelements-of-statistical-learning*. 2017.

[9] Wikipedia. Recommender systems.