Purchasing Intention Data Analysis

By Guillaume Chevrollier and Lisa Cluzel

Final Project of Python for Data Analysis

2021-2022

Overview

Introduction : The project and the dataset

1) Data exploration

2) Modeling

3) API

Conclusion

The project

This project consists in the final achievement of the course 'Python for Data Analysis' being part of our degree at ESILV.

Our team, composed of Guillaume and Lisa, was given a dataset to work with.
The goal was to **explore and analyze our dataset** by applying the knowledge and skills we had acquired along the course.

First, we started by discovering the data, with some reading, exploration and visualization on our Notebook in Python. Then, we moved on to some dimensionality reduction tools.
Afterwards, we preprocessed the data for the upcoming modeling. We improved our results by changing models, hyperparameters and using gridsearchs. We visualized our performances.
Finally, we implemented a Flask API.

The dataset : Online Purchasing Intention

Our dataset represents some information about visitors on a website. Our goal is to study the **relationship between the data and the purchase**.

Each visit, or session, is an instance. The dataset contains 12330 of them.
Of the 12,330 sessions in the dataset, 84.5% (10,422) were negative class samples that did not end with shopping, and the rest (1908) were positive class samples ending with shopping.

The data consists of 10 numerical and 8 categorical attributes. The **'Revenue'** attribute can be used as the class label. It represents whether a visitor made a purchase or not.

Indeed, this is a **classification problem**.

Link to the dataset here

The dataset : Online Purchasing Intention

The 17 features can be splitted up into 4 groups:

- informations about the visitor himself
- information about the moment of the visit
- information given by Google Analytics
- information about the content of the website that was visited

We will go through all of them.

We also have the Revenue feature giving us the information about the purchase in a boolean.

The dataset : Information about the visitor

5 features out of 17 give us data about the visitor himself.

First, we have the Operating System. Represented by a number between 1 and 8, it will allow the model to use the relationship between the OS used by the user and the purchase. We also have the Browser from which the visitor is browsing (1 to 13).

We also have the Region (1 to 9) from which the visitor is, and the TrafficType (how the user entered the website) from 1 to 20.

For those first 4 features. The groups are represented with numbers, it can be for reasons of confidentiality or not to have problems with bad publicity, or just not to be influenced by the value. Having numbers allows to treat the groups anonymously, even if it's a mistake to think the groups are ordered as numbers are.

Finally we have the VisitorType. It can be a 'Returning_visitor', a 'New_visitor' or 'Other'.

The dataset : Information about the time

3 features out of 17 give us data about the moment of the visit.

First, we have Month. Represented with a string, it gives us the 3 first letters of the month of the visit.

Then we have the WeekEnd feature. It is a boolean, 'False' meaning the visit was between Monday and Friday, and 'True' meaning it was on Saturday or Sunday.

Last but not least, we have SpecialDay. This feature is a float between 0 and 1. 0 means no special day were 'close' to the visit, and 1 means a special day was really 'close'. It can take values from 0 to 1 with 0.1 steps.
By 'close' (1) we mean that the visit day + delay estimated for the delivery = Special Day. The further we get (in days), the lower the SpecialDay value.

The dataset : Google Analytics' information

3 features out of 17 are given by Google Analytics.

The first one is ExitRates. It's a float giving us the proportion of visitors leaving through a given page of the website.

BounceRates is similar. It's the proportion of visitors that entered and left through a given page.

Finally, the PageValues is the score of a given page as computed by Google, depending on the percentage of transactions performed having previously consulted the page.

This last feature seems quite interesting because it already represents the relationship between an instance and the Revenue.

The dataset : Content visited information

6 features out of 17 are about the content visited.

There are 3 categories of content: Administrative, Informational, ProductRelated. For each of them there are 2 features. The first one is giving the number of pages consulted for this category of content. The second one is giving the total amount of time spent on those pages.

This gives us the 6 following features: Administrative, Administrative_Duration, Informational, Informational_Duration, ProductRelated, ProductRelated_Duraton.

All those features are numerical.

1) Data Exploration : Cleaning

In order to understand better our data before modeling, and get a good idea of the **relationships between the variables and the target feature**, we went through a first phasis of exploration.
It took place on a Jupyter Notebook.

First we imported and cleaned the dataset a little to improve its intelligibility and consistency. We especially looked at the **datatypes** and the **missing values**.

We took care of Month and VisitorType, both strings that were first imported as object types.

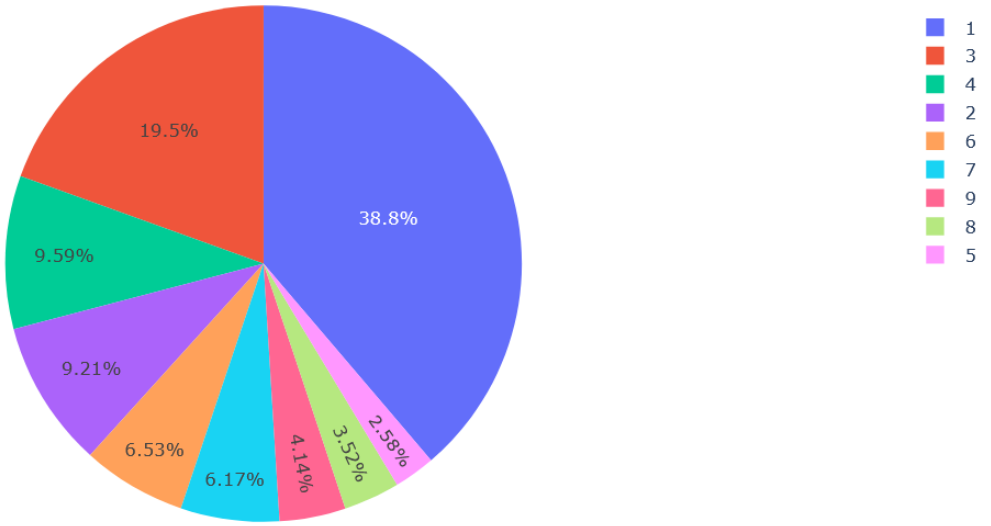
Our dataset did not contain any missing values.

1) Data Exploration : Univariate exploration

We started the exploration by exploring the relationship between each feature and the target independently.

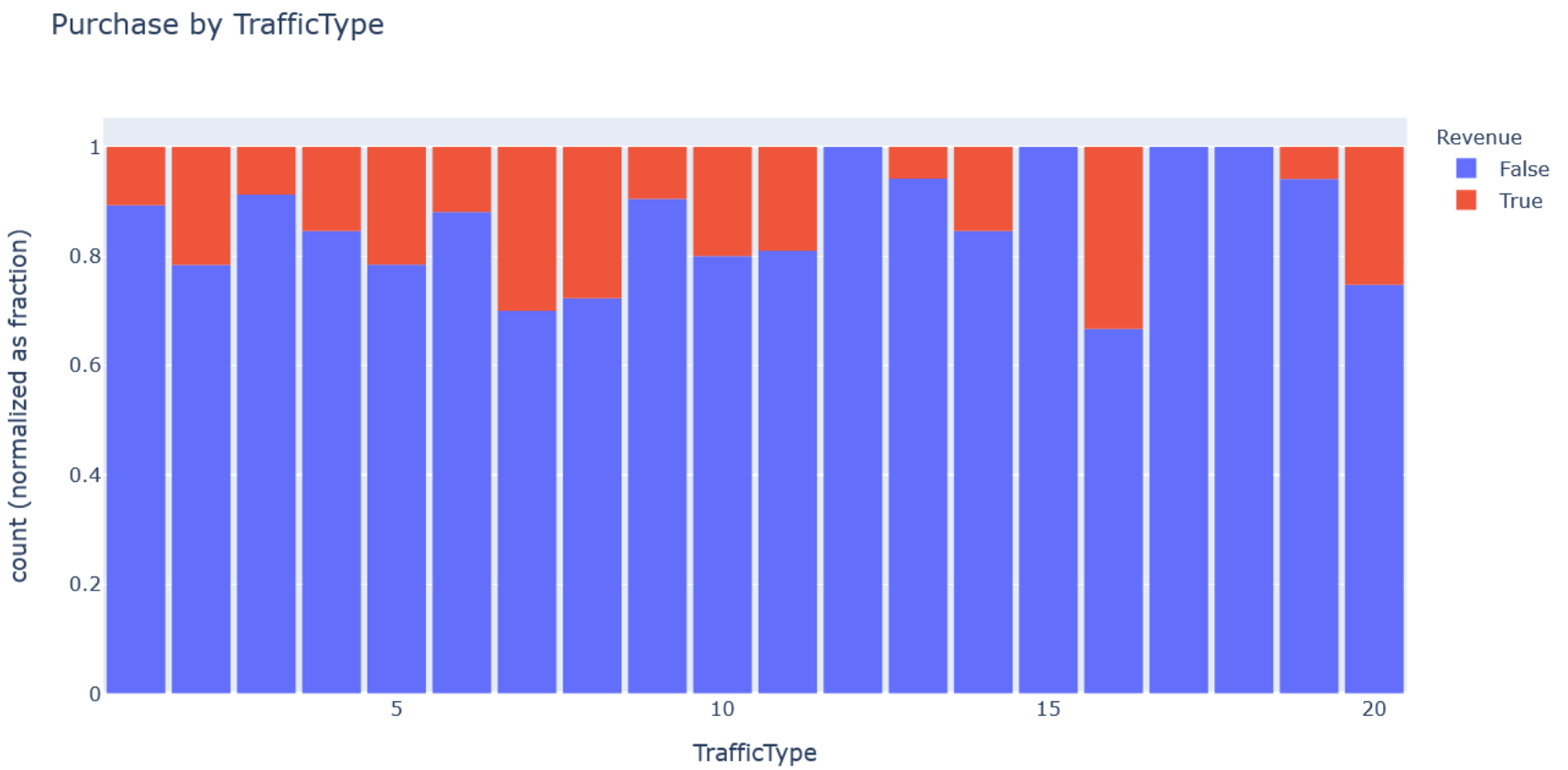
For each feature, we add a look at its **distribution**, then at the **proportion of purchase** for each predominant value of the feature.
For this, we have to remember that on the full dataset, the purchase rate is 15.47%.

Each value might have a positive or negative correlation to the purchase, henca having a higher or lower purchase rate.



Example : Region distribution

1) Data Exploration : Univariate exploration



Example : Relationship between TrafficTypes and Revenue

On this histogram, we can see that the Purchase Rate is very different from one TrafficType to another. Those 2 features have an interesting relationship.

1) Data Exploration : Univariate exploration

To sum up our results on the univariate exploration, the features who seemed to have a clear relation with the target are the TrafficType, the VisitorType, the Month, the content pages (all of them), the Google Analytics features.

The other ones seemed to have less relevance.

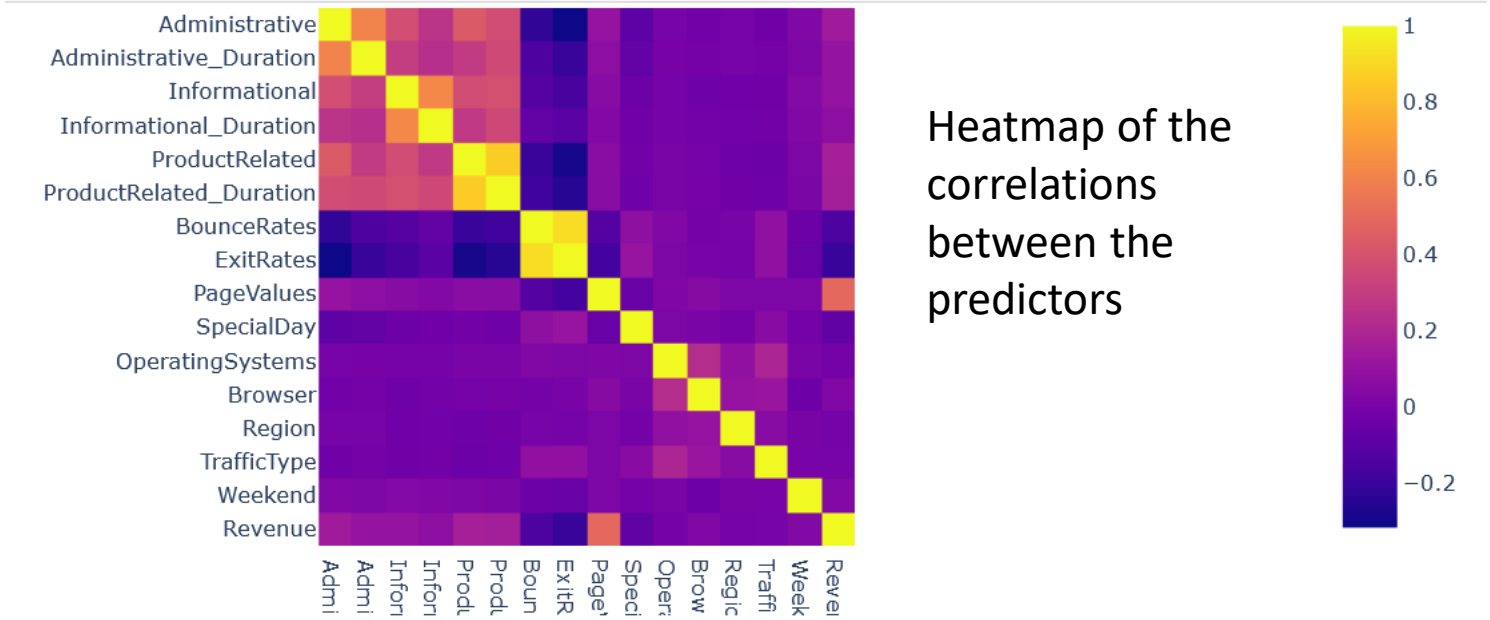
Special mention for the SpecialDay feature, which seems to not have met the hope placed into it. The purchase rate is rather lower on the SpecialDays. But this doesn't mean its a problem, because even if the relationship is negative, it is still a relationship to use for our model.

1) Data Exploration : Multivariate Exploration

After the univariate exploration, we performed a PCA. The Principal Component Analysis is a multivariate tool used for **dimensionality reduction**.

The goal is to fing and keep only the most significant features for the upcoming modeling.

We first had a look at the **correlations** between the predictors through this heatmap.



Heatmap of the correlations between the predictors

1) Data Exploration : Multivariate Exploration

The first heatmap allowed us to identify correlations between the features. We removed those correlations by combining the correlated features into a new one.

```
df['ProductRelated_Combined'] = df['ProductRelated']/(df['ProductRelated_Duration']+0.00001)
df['Administrative_Combined'] = df['Administrative']/(df['Administrative_Duration']+0.00001)
df['Informational_Combined'] = df['Informational']/(df['Informational_Duration']+0.00001)
df['Bounce_by_Exit_Rate']= df['BounceRates']/(df['ExitRates']+0.00001)
del df['ProductRelated']
del df['Administrative']
del df['Informational']
del df['ProductRelated_Duration']
del df['Administrative_Duration']
del df['Informational_Duration']
del df['ExitRates']
del df['BounceRates']
```

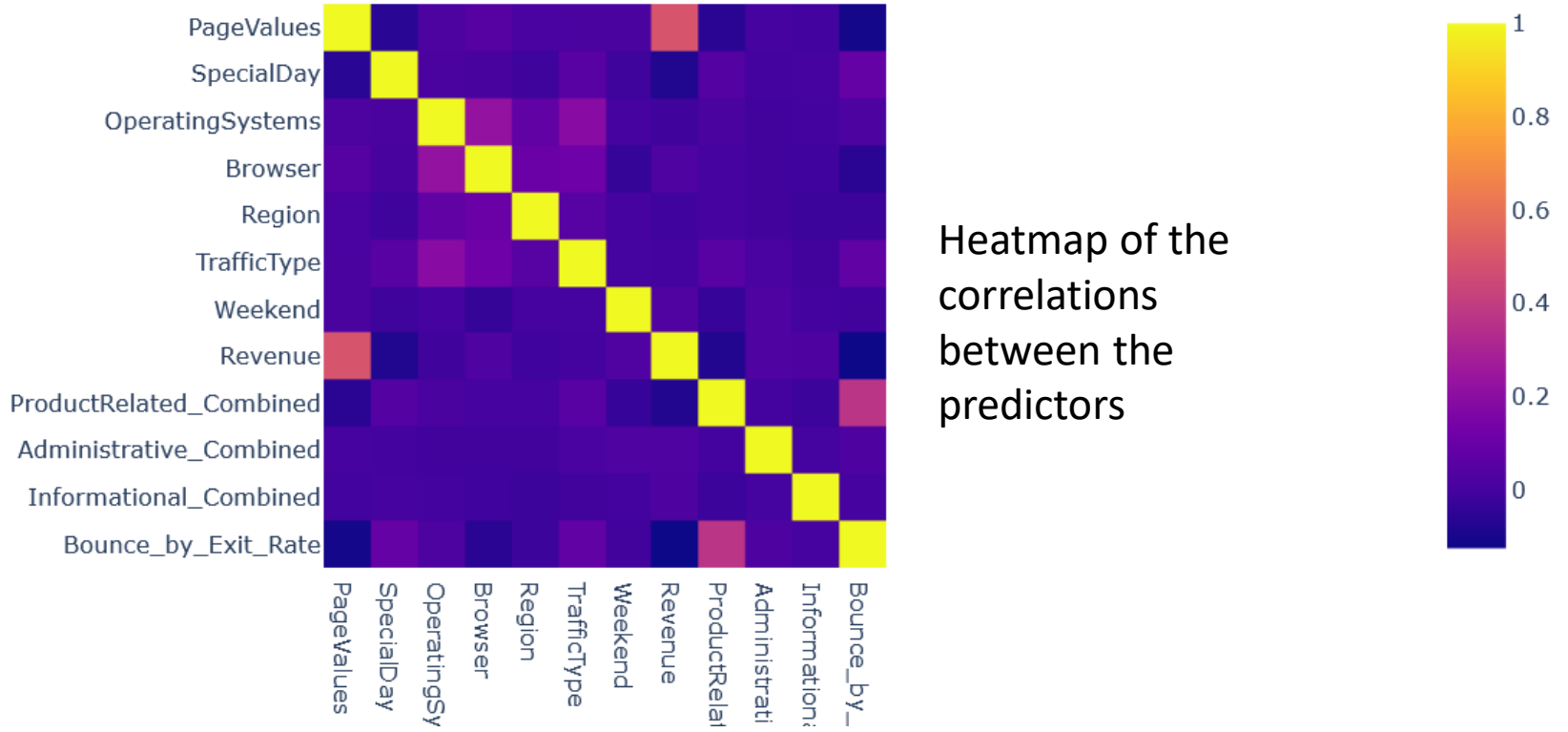
We also could have removed some of the redundant features.

1) Data Exploration : Multivariate Exploration

After modoficying our dataset, we recreated the heatmap with the new variables.

This time, correlations are less important, except for the Revenue variable.

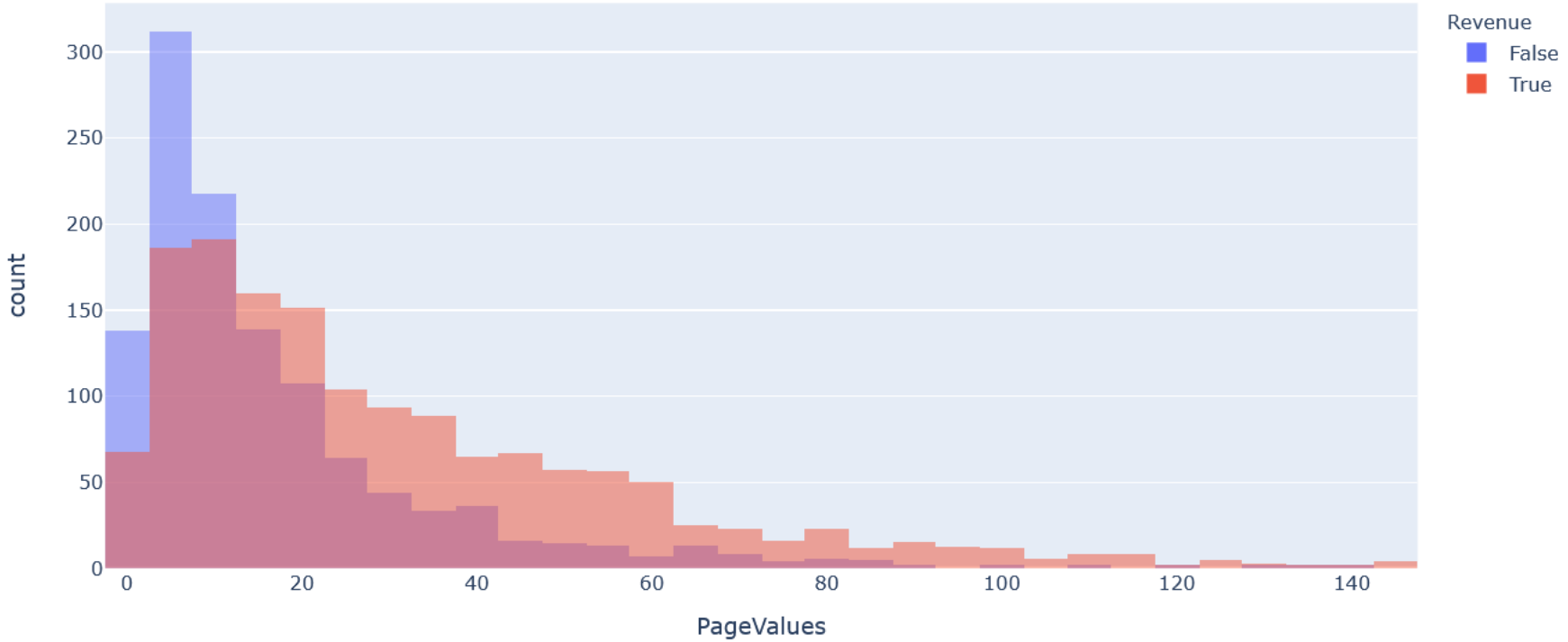
We notice the new correlation between the Bounce_by_Exit_Rate and ProductRelated_Combined features.



Heatmap of the correlations between the predictors

1) Data Exploration : Multivariate Exploration

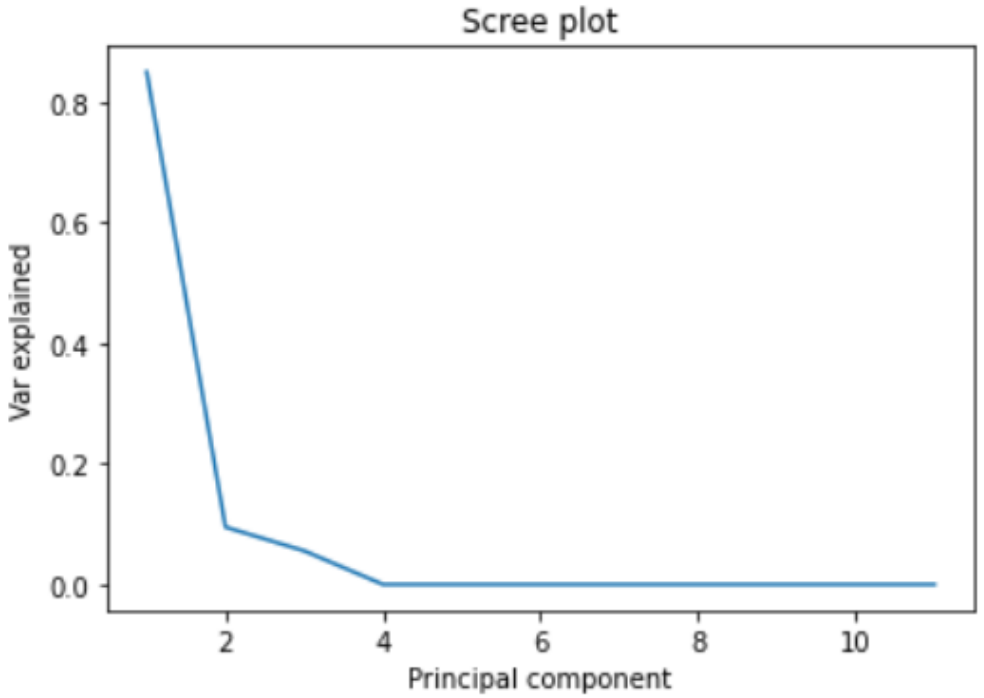
After the new heatmap, we had a look at the newly highlighted relationships.



Non-null PageValues distribution with Revenue

1) Data Exploration : PCA

Then, we performed the PCA. We had a look at the explained variance per feature, plotted the scree plot.



Scree plot

The **scree plot** gives us the proportion of explained variance for each feature in the model.

Here, the first one explains more than 80% of the variance, about 10% for the second and so on.

This graph helps us choose how many features to keep. Here 2 seems to be a good choice, explainging 90% of the variance.

1) Data Exploration : PCA

After the scree plot, we know we neef to only keep 2 components. We create a new dataframe by using the pca.transform() method.

```
Entrée [68]: reduced_df=pd.DataFrame(PCA(2).fit(df_pca).transform(df_pca))
             reduced_df.shape

 Out[68]: (12330, 2)
```

We will also try models on this dataframe later on.

2) Modeling : Preprocessing

Before the modeling, we need to **prepare our data**. That part is required so that our data is ready to be used.

First thing we did was the dummification. The categorical features were until now represented with their group numbers, like from 1 to 8 for the Browser for instance. For the modeling, this could be misleading because it gives the model the impression that the group 2 is between 1 and 3 etcetera. To avoid that, we dumify, meaning for each categorical feature, we create n new features (columns here) where n is the number of different values that the feature can take. Each new column is filled with 0s and 1s, 1s for the instances that belonged in this group and 0s for all the others.

This might seem a little messy, but it isn't meant to be intelligible like it was until now, but only to be fairly treated by the models.

2) Modeling : Preprocessing

We used Pandas' get_dummies method as follows:

```
dummies=pd.get_dummies(df['Month'])
df=df.join(dummies)
del df['Month']
```

```
dummies=pd.get_dummies(df['OperatingSystems'])
dummies.columns=['OS1','OS2','OS3','OS4','OS5','OS6','OS7','OS8']
df=df.join(dummies)
del df['OperatingSystems']
```

```
dummies=pd.get_dummies(df['Browser'])
dummies.columns=['Browser1','Browser2','Browser3','Browser4','Browser5','Browser6','Browser7','Browser8','Browser9','
df.join(dummies)
del df['Browser']
```

```
8]: #df.head()
    len(df.columns)
```

```
58
```

We end up with 58 columns in the end.

2) Modeling : Preprocessing

After the dummification, splitted the dataframe into X, y. We also used the train_test_split method to have the training and test sets:

```
: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
```

```
: reduced_X_train,reduced_X_test,reduced_y_train,reduced_y_test=train_test_split(reduced_df,y, test_size=0.33, random_s
```

The reduced ones are those obtained on the dataset on which we performed the ACP.

Finally, we prepare a dictionary in which we will store the performances of our models:

```
performances={}
```

2) Modeling : Modeling

Once the preprocessing was done, we moved on to the modeling. We tried several **classifications models**, from the basic ones to the more complicated ones.

On each one, we had a look a the **hyperparameters**, changed some combinations and had a look at the **performances** (here accuracies or confusion matrixes).

We also tried the reduced data obtained with the ACP on some, and finally we did some **GridSearchs** on a few of them.

Entrée [105]:
```
LG_model = sklearn.linear_model.LogisticRegression(solver='liblinear',penalty='l2', random_state=1)
LG_model.fit(X_train, y_train)
LG_preds=LG_model.predict(X_test)
LG_acc=sklearn.metrics.accuracy_score(LG_preds,y_test)
performances['Logistic_Regression']=LG_acc
LG_acc
```

Out[105]: 0.8906365200294912

Example: modeling for Logistic Regression, with an accuracy of 89%

2) Modeling : Results

To sum up our results, the data obtained with the **PCA** never gave better results than the full one.

For the **hyperparameters**, some improvements were made by trying some combinations, for example on the KNN, we got better results with a higher number of neighbours (30). But in general, the default parameters were often the best ones.

Indeed, the **gridsearchs** didn't give us significantly good results. We only tried it on the algorithms were a lot of hyperparameters could be change, and still the results were not better.

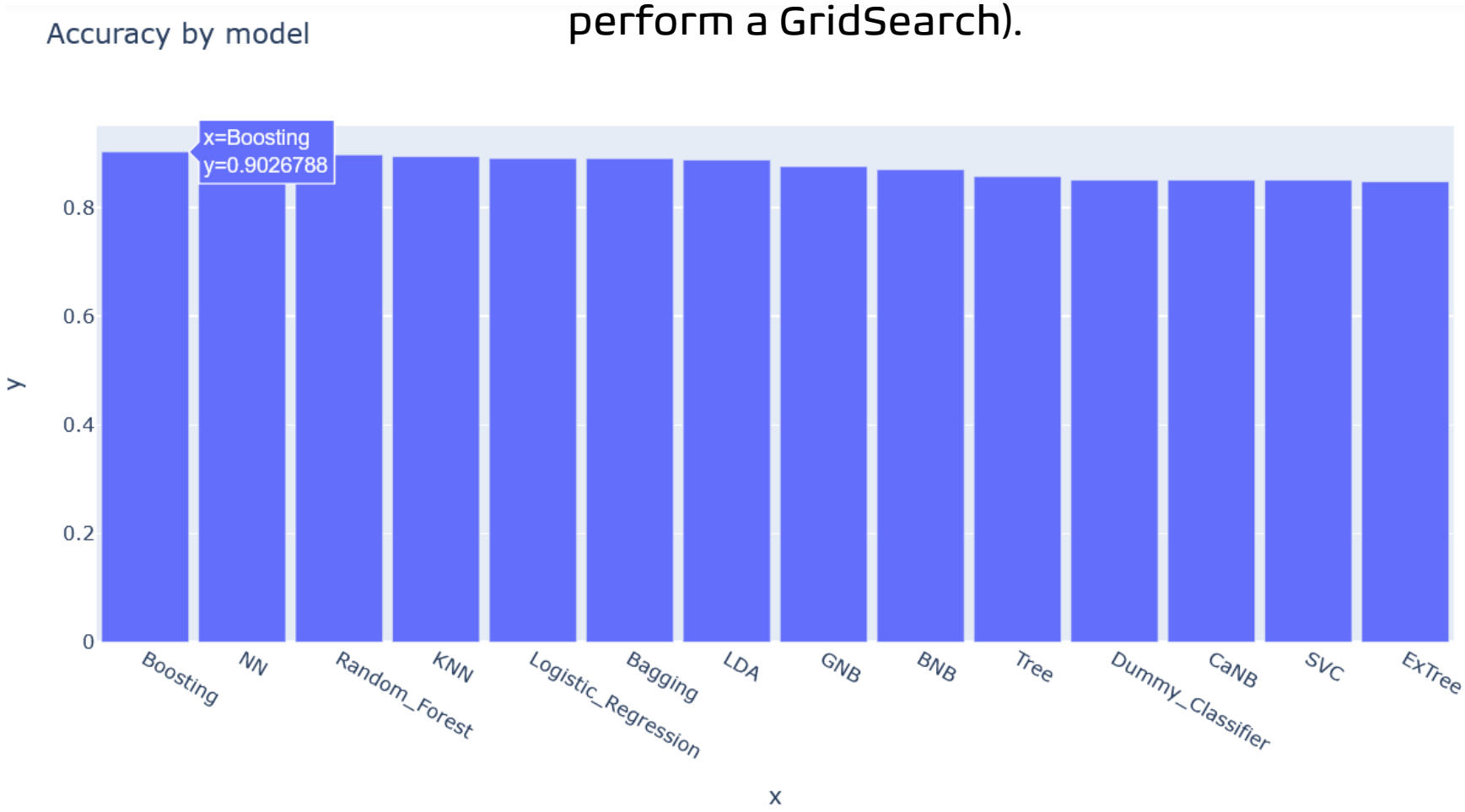
Also, we have to mention that we encountered some issues with the gridsearch : the amount of time taken for it to run was insanely long. We tried one on the boosting model who runed for about 60 hours, until we stopped it manually.

2) Modeling : Results

The best model was the **boosting** with default parameters (we couldn't perform a GridSearch).

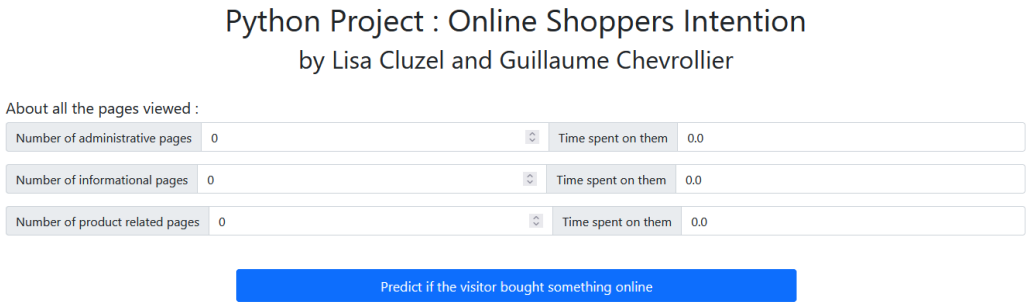
Finally, we took the best accuracy of each algorithm in the performances dictionnary.

We plotted the accuracies on a histogram:



3) API

We decided to go for the **Flask** Framework as it seemed easier while Django is really time consumming and too much for a two pages' API.



We did two html templates to present our solution, the first one is a form composed of inputs and selects to complete the informations of the new csv line to predict, the second one is a simple page that shows the prediction and a button to go back to the first one.

3) API

We used **pickle** to serialize the sk-learn model to do the prediction and use it in our python file with flask.


```
numAd=request.form['numAd']
timeAd=request.form['timeAd']
adRel_Combined = int(numAd)/(float(timeAd)+0.00001)
```

The flask API allows us to request the informations inquired in the form, so we can transform it. On top of that, we stored two lines of the transformed dataset used in the jupyter notebook (two lines because just one is doing a Serie) to modify it and predict easily.

Conclusion

We are quite satisfied by the results of our work. The notebook emphasizes the main process as we went through the project. The API showcases the deployment of our model.

This project helped us develop our skills on the different processes a Data Scientist can set up to handle a project (data cleaning, transformation, visualization, the prediction of new data and the presentation of his results).

We also improved ourselves in exposing our results in a web framework for python, as we already worked with Django, we learnt how to use Flask to make easier API and we used Bootstrap that is a library of predefined css fonts that makes our html code look better.

Conclusion : Thanks

Sakar, C.O., Polat, S.O., Katircioglu, M. et al. Neural Comput & Applic (2018)

https://link.springer.com/article/10.1007%2Fs00521-018-3523-0

https://www.youtube.com/watch?v=4_5OFsrYk6c&t=567s

https://www.youtube.com/watch?v=m2CCki_APxE

https://getbootstrap.com/