université PARIS DIDEROT PARIS 7

CLASSIFICATION DES CATÉGORIES DE PRODUITS LORS D'UN BLACK-FRIDAY

Par HAMANI Khalil

Travail réalisé

- Classification de la catégorie d'un produit
  - *Models utilisés : SVM, Arbre de décision, forêt aléatoire*
- Régression sur le coût de la transaction
  - *Models utilisés : Régression linéaire, par Arbre de décision, par forêt aléatoire*

Présentation du dataset

- 1 ligne = 1 transaction

Présentation du dataset

- 1 ligne = 1 transaction
- Que pourrait-on faire avec ?

Présentation du dataset

- 1 ligne = 1 transaction
- Que pourrait-on faire avec ?
  - *Classifications utiles : Age, Catégorie du produit*

Présentation du dataset

- 1 ligne = 1 transaction
- Que pourrait-on faire avec ?
  - *Classifications utiles : Age, Catégorie du produit*
  - *Classifications inutiles : Sexe, Occupation (signification inconnue), État civil*

Présentation du dataset

- 1 ligne = 1 transaction
- Que pourrait-on faire avec ?
  - *Classifications utiles : Age, Catégorie du produit*
  - *Classifications inutiles : Sexe, Occupation (signification inconnue), État civil*
  - *Régression : Coût de la transaction*

Présentation du dataset

- 1 ligne = 1 transaction
- Que pourrait-on faire avec ?
  - *Classifications utiles : Age, Catégorie du produit*
  - *Classifications inutiles : Sexe, Occupation (signification inconnue), État civil*
  - *Régression : Coût de la transaction*
  - *Clusturing : Faire de la recommandation sur les produits*

Classification des catégories des produits

- Nettoyage du dataset (pré-traitements)
- Encodage
- Entrainement
- Mesures de performances

Outils utilisés

Classification de la catégorie

Pré-traitements (Catégorie)

```
In [3]: bf_df = pd.read_csv("./BlackFriday.csv")
        bf_df.head()
```

Out[3]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	

- Remplissage des vides par des 0

```
bf_df.fillna(0).head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_
0	1000001	P00069042	F	0-17	10	A	2	0	3	0.0	
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	
2	1000001	P00087842	F	0-17	10	A	2	0	12	0.0	
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	
4	1000002	P00285442	M	55+	16	C	4+	0	8	0.0	

Pré-traitements (Catégorie)

- Encodage des catégories
  - *Catégorie 1 = {1, …, 18}* => *Catégorie 1 = {A, …, R}*
  - *Catégorie 2 = {2, …, 18}* => *Catégorie 1 = {B, …, R, 0}*
  - *Catégorie 3 = {3, …, 18}* => *Catégorie 1 = {C, …, R, 0}*
- Concaténation des catégories
  - *Product_Category : Catégorie 1 + Catégorie 2 + Catégorie 3*
    - Exemple :
      - *Catégorie 1 : 1 => A*
      - *Catégorie 2 : 2 => B* => *Product_Category : ABC*
      - *Catégorie 3 : 3 => C*

*On a jusqu'ici 235 catégories

Pré-traitements (Catégorie)

- Suppression des doublons par combinaison
  - *Exemple : Les catégories suivantes sont considérées comme étant la même catégorie*
    - DEF, EDF, EFD, DFE, FDE, FED
- Après ce pré-traitement le nombre de catégories est resté le même i.e. 235

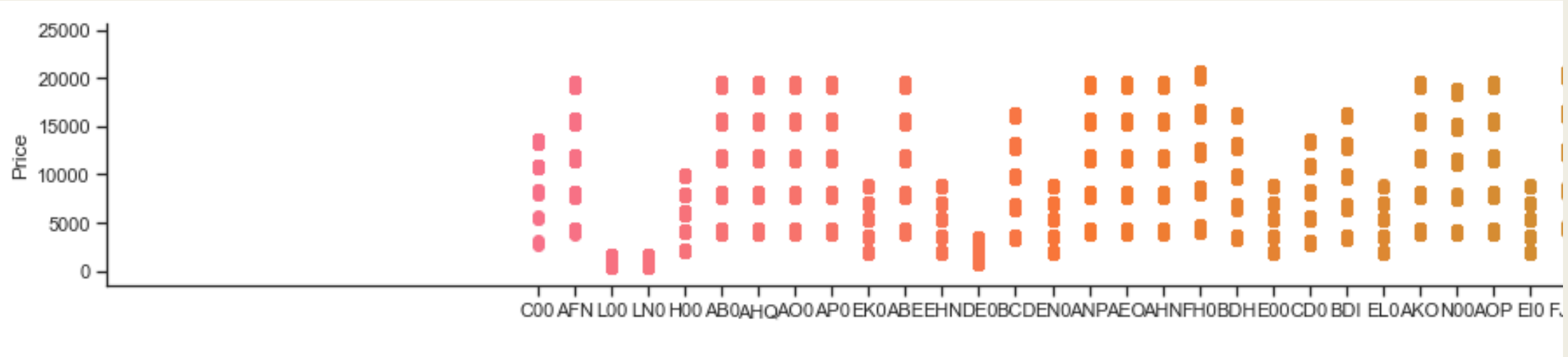
Pré-traitements (User_ID)

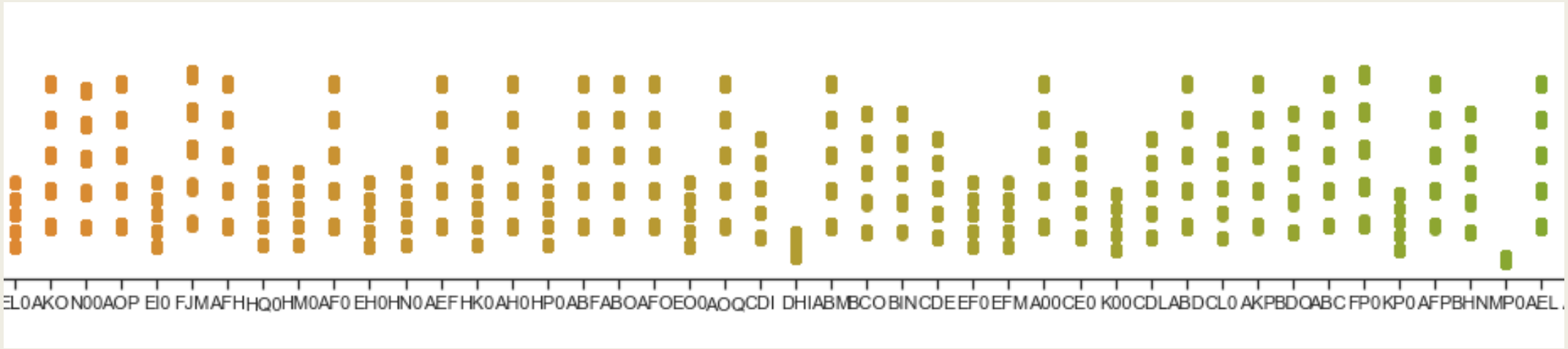
- Tout simplement supprimé pour cause d’impretinence

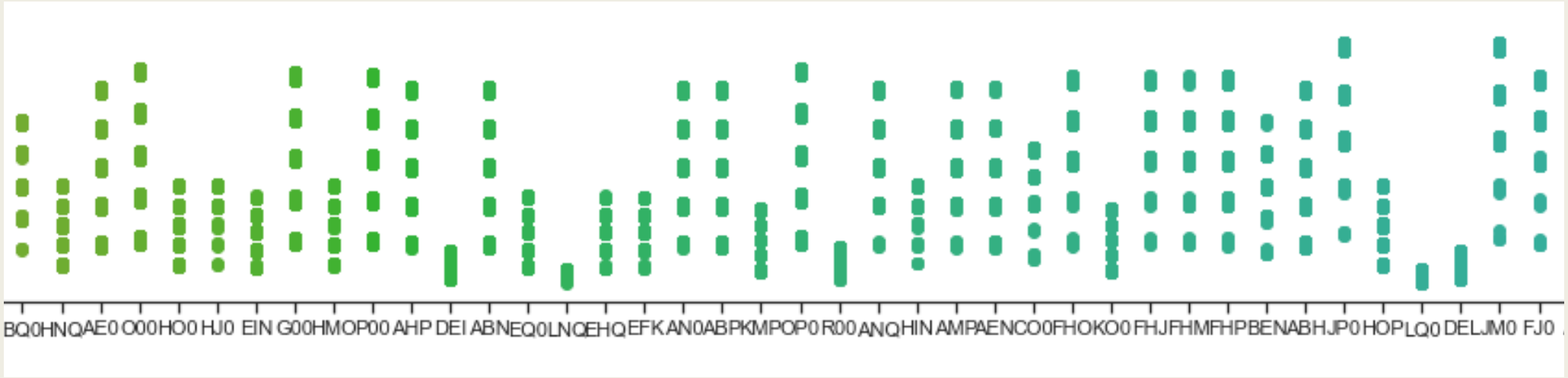
Encodage

- Séquenciel et par ordre d'apparition : 0, 1, 2, 3… N pour toutes les colonnes à l'exception du coût de la transaction

- Le coût de la transaction a été encodé en utilisant un normalisation par Minimum (=0) et Maximum (=1)




EL0 AKO N00 AOP EI0 FJM AFH HQ0 HM0 AF0 EH0 HN0 AEF HK0 AH0 HP0 ABF ABO AFO EO0 AOQ CDI DHI ABM BCO BIN CDE EF0 EFM A00 CE0 K00 CDL ABD CL0 AKP BDQ ABC FP0 KP0 AFP BHN MP0 AEL


BQ0 HNQ AE0 O00 HO0 HJ0 EIN G00 HMOP00 AHP DEI ABN EQ0 LNQ EHQ EFK AN0 ABPK MPOP0 R00 ANQ HIN AMPA ENCO0 FHOKO0 FHJF HMFHP BENA BHJP0 HOP LQ0 DELJM0 FJ0

Division du dataset

- Division du dataset en 70% train et 30% test avec l'option shuffle pour prendre au hazard et non de manière séquentielle

Classification (SVM)

- Précision moyenne : 0.2%
- Recall moyen : 0.6%

Classification (SVM)

- Précision moyenne : 0.2%
- Recall moyen : 0.6%

Classification (Arbre de décision)

- Accuracy : 97% (Is this overfiting ?!!!)
- Précision moyenne : 93.7%
- Recall moyen : 93.5%

Classification (Arbre de décision)

Profondeur plafonnée à 27

- Accuracy : 93%
- Précision moyenne : 92.3%
- Recall moyen : 87.5%

Classification (Forêt aléatoire)

- Accuracy : 42%
- Précision moyenne : 25.4%
- Recall moyen : 19.7%

```
1. in precision :  0
0. in precision :  31
1. in recall :  0
0. in recall :  31
```

Classification (Forêt aléatoire)

- Accuracy : 42%
- Précision moyenne : 25.4%
- Recall moyen : 19.7%

Regression sur le coût d’une transaction

- Pré-traitements
- Encodage
- Entrainement
- Mesures de performances

Pré-traitements

Lecture du DataSet

In [30]:
```
bf_df = pd.read_csv("./BlackFriday.csv")
bf_df = bf_df.fillna(0)
bf_df.head()
```

Out[30]:

_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
)42	F	0-17	10	A	2	0	3	0.0	0.0	8370
)42	F	0-17	10	A	2	0	1	6.0	14.0	15200
)42	F	0-17	10	A	2	0	12	0.0	0.0	1422
)42	F	0-17	10	A	2	0	12	14.0	0.0	1057
)42	M	55+	16	C	4+	0	8	0.0	0.0	7969

In [31]:
```
bf_df["Product_Category_2"] = bf_df["Product_Category_2"].astype(int)
bf_df["Product_Category_3"] = bf_df["Product_Category_3"].astype(int)
bf_df = bf_df.drop(columns=["User_ID"])
bf_df.head()
```

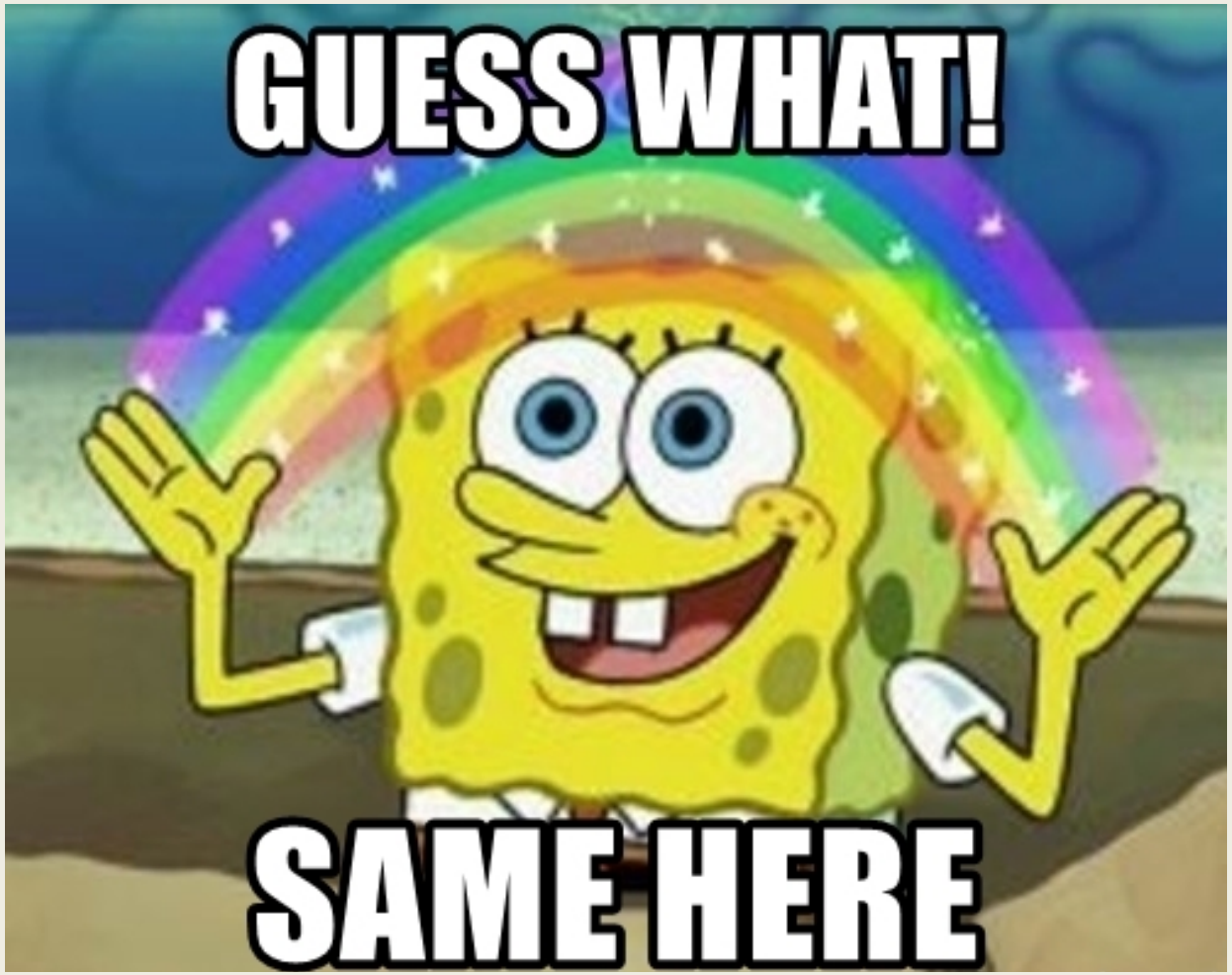
Out[31]:

_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
)42	F	0-17	10	A	2	0	3	0	0	8370
)42	F	0-17	10	A	2	0	1	6	14	15200
)42	F	0-17	10	A	2	0	12	0	0	1422
)42	F	0-17	10	A	2	0	12	14	0	1057
)42	M	55+	16	C	4+	0	8	0	0	7969

In [32]:
```
bf_df.to_csv("./BlackFriday_Price_Preprocessed.csv", index=False)
```

Encodage

Encodage



Régression linéaire

- Score : 0.6%

Linear Regression

Entrainement

```
In [16]: reg = LinearRegression()

         reg = train_model(reg, X_train, y_train)
```

```
Model entrainé en : 00:00:00
```

Mesures de performances

```
In [17]: reg.score(X_test, y_test)
```

```
Out[17]: 0.059940413938787 46
```

Régression par arbre de décision

- Pas de limitation sur la profondeur de l'arbre :
  - *Score : 43%*

- Après limitation de la profondeur à 13 :
  - *Score : 68%*

Régression par forêt aléatoire

- Pas de limitation sur la profondeur de l'arbre :
  - *Score : 43%*
- Après limitation de la profondeur à 13 :
  - *Score : 70%*

***Nombre d'arbres = 200**