PRÉDICTIONS DE VENTES CROISÉE D'ASSURENCE MALADIE ET ASSURENCE AUTOMOBILE

FOUILLE DE DONNÉES

RÉALISÉ PAR:

WISSEM MERIDJ

RAID GUEDOUDJ

PLAN DE TRAVAIL

- Introduction
- Dataset
- Nettoyage
- Algo / Modèle
- Train / Validation
- Résultats
- Discussion
- Code
- Contributions

INTRODUCTION

Notre projet est à propos des prédictions de ventes croisée d'assurance maladie et assurance automobile, c'est à dire: Prédire les propriétaires d'assurance maladie qui seront intéressés par l'assurance automobile.

OBJECTIF DU SUJET

- Notre client est une compagnie d'assurance qui a fourni une assurance maladie à ses clients.
- Donc notre objectif est de *** créer un modèle permettant de prédire si les assurés (clients) de l'année dernière seront également intéressés par l'assurance véhicule fournie par la société ***

DESCRIPTION DE LA DATASET

Lien de dataset : https://www.kaggle.com/anmolkumar/health-insurance-cross-sell-prediction

Notre dataset est composée de 3 fichiers csv:

- Train.csv: Contient les données sur notre modèle sera entraîné
- Test.csv: Contient des données sur lesquelles on test la performance de notre modèle
- Sample_submission,csv

EXPLORATION DE DONNÉES

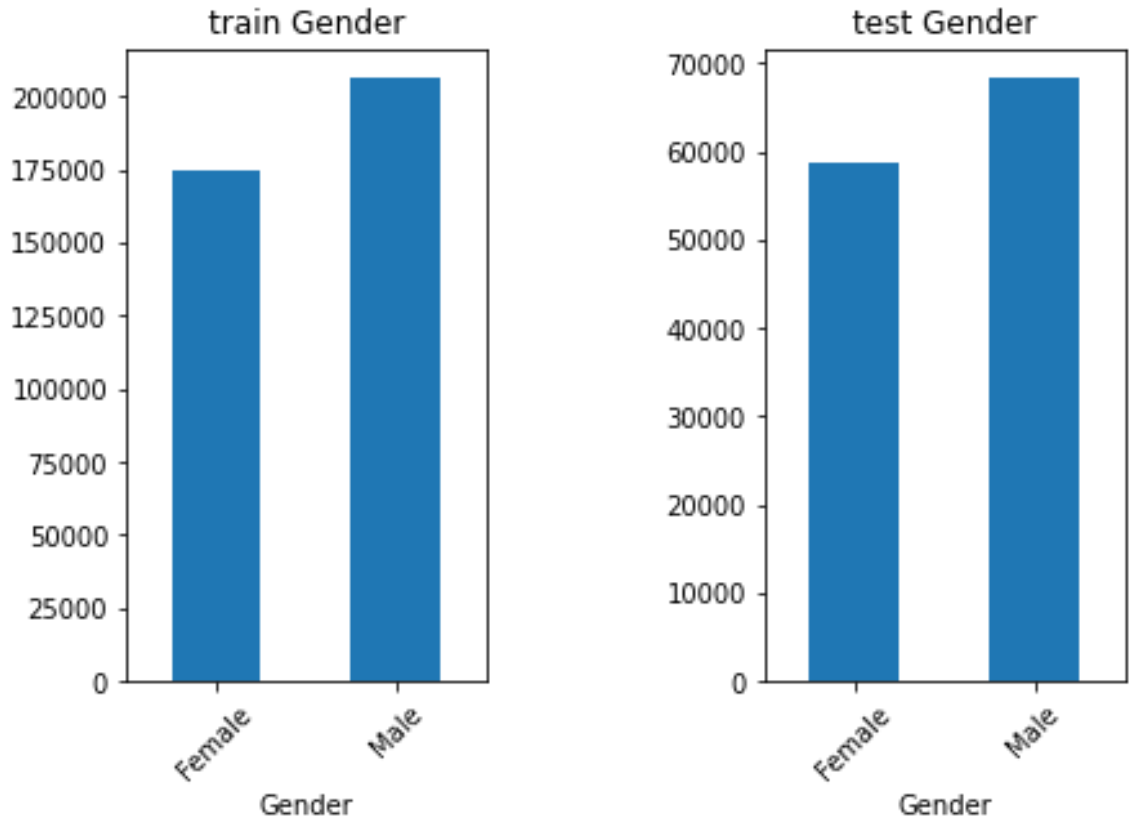
Train.csv (11 colonnes)

	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vehicle_Damage	Annual_Premium	Policy_Sales_Channel	Vintage	Response
0	Male	44	1	28.0	0	> 2 Years	Yes	40454.0	26.0	217	1
1	Male	76	1	3.0	0	1-2 Year	No	33536.0	26.0	183	0
2	Male	47	1	28.0	0	> 2 Years	Yes	38294.0	26.0	27	1
3	Male	21	1	11.0	1	< 1 Year	No	28619.0	152.0	203	0
4	Female	29	1	41.0	1	< 1 Year	No	27496.0	152.0	39	0

Test.csv (10 colonnes)

	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vehicle_Damage	Annual_Premium	Policy_Sales_Channel	Vintage
0	Male	25	1	11.0	1	< 1 Year	No	35786.0	152.0	53
1	Male	40	1	28.0	0	1-2 Year	Yes	33762.0	7.0	111
2	Male	47	1	28.0	0	1-2 Year	Yes	40050.0	124.0	199
3	Male	24	1	27.0	1	< 1 Year	Yes	37356.0	152.0	187
4	Male	27	1	28.0	1	< 1 Year	No	59097.0	152.0	297

EXPLORATION DE DONNÉES



Nombre d’homme et femmes dans les data train et test

PREPROCESSING

1. ÉLIMINATION DES COLONNES INUTILES

Eliminer les colonnes inutiles pour entaîner notre modèle

- Eliminer la colonne **‘id’** dans les deux dataframes `train_original_df` et `test_original_df`

```
# Eliminer les colonnes inutiles
train_original_df = train_original_df.drop(['id'], axis=1) # eliminer la colonne 'id' du train_original_df
test_original_df = test_original_df.drop(['id'], axis=1) # eliminer la colonne 'id' du test_original_df
```

PREPROCESSING

2. NETTOYAGE DES VALEURS NULLES

Vérifier s'il existe des valeurs nulles dans les deux dataframes `train_original_df` et `test_original_df`

```
# Vérifier si on a des valeurs null dans # Vérifier si on a des valeur null dans test_original_df et test_original_df
train_original_df.isnull().sum() # retourne la somme des valeurs nulles dans chaque colonne du dataframe
test_original_df.isnull().sum()

# Ou avec test_original_df.isnull() sans rajouter sum() qui retourne un boolean
```

```
Gender                0
Age                   0
Driving_License       0
Region_Code           0
Previously_Insured    0
Vehicle_Age           0
Vehicle_Damage        0
Annual_Premium        0
Policy_Sales_Channel  0
Vintage               0
Response              0
dtype: int64
```

train_original_df

```
Gender                0
Age                   0
Driving_License       0
Region_Code           0
Previously_Insured    0
Vehicle_Age           0
Vehicle_Damage        0
Annual_Premium        0
Policy_Sales_Channel  0
Vintage               0
dtype: int64
```

test_original_df

PREPROCESSING

3. ENCODING

- Transformer toutes les valeurs de type string en valeurs numériques dans les deux dataframes train_original_df et test_original_df
- Les colonnes à transformer sont : **‘Gender’, ‘Vehicle_Age’, ‘Vehicle_Damage’**

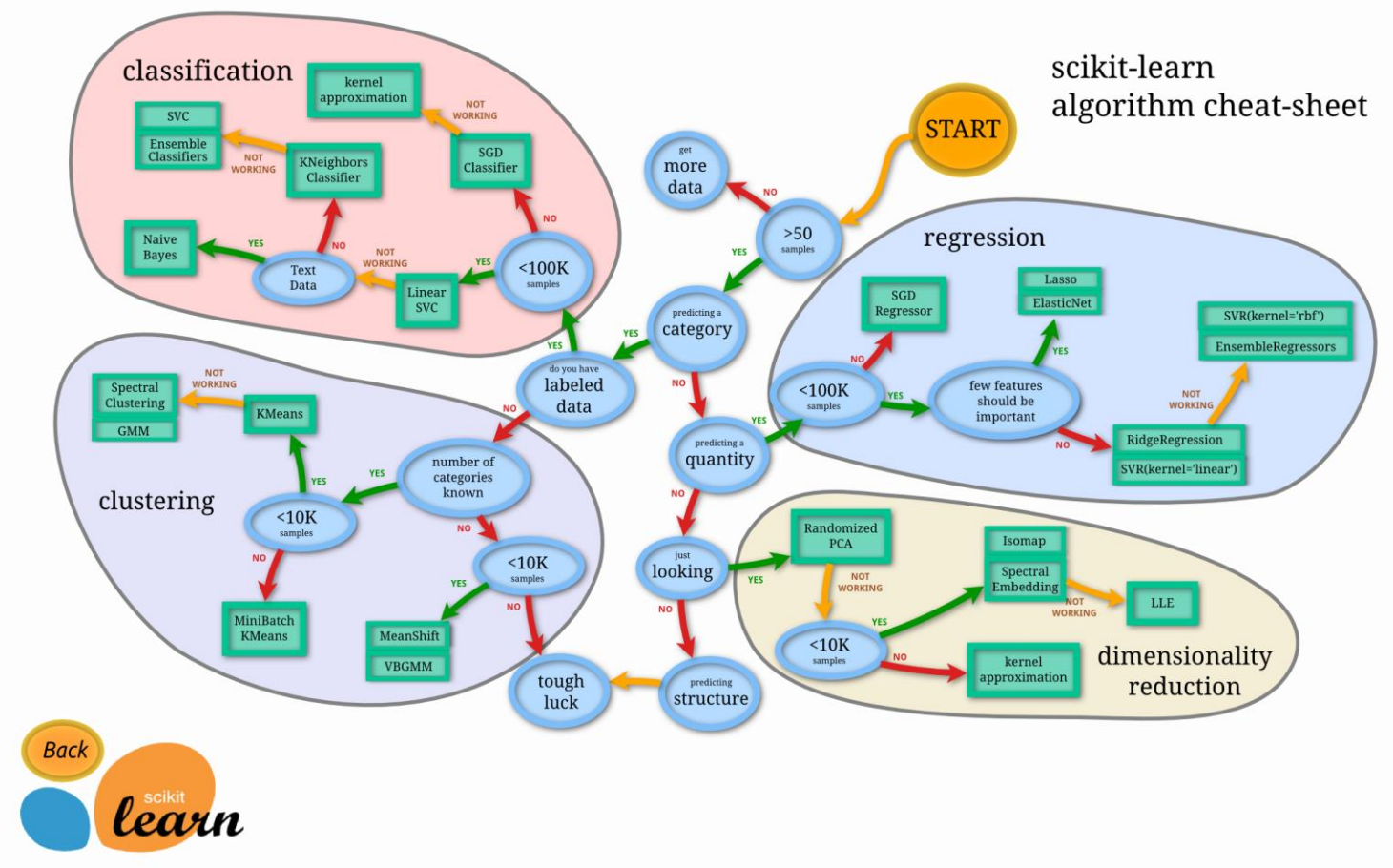
```
from sklearn.preprocessing import LabelEncoder
def toEncodeData(df, colomn):
    le = LabelEncoder()
    le.fit(df[colomn]) # détecter les classes
    print(le.classes_) # afficher les classes sous forme d'une liste
    df[colomn] = le.transform(df[colomn]) # transformer les classes (toutes les valeurs string) en valeurs numériques
```

	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vehicle_Damage	Annual_Premium	Policy_Sales_Channel	Vintage	Response
0	1	44	1	28.0	0	2	1	40454.0	26.0	217	1
1	1	76	1	3.0	0	0	0	33536.0	26.0	183	0
2	1	47	1	28.0	0	2	1	38294.0	26.0	27	1
3	1	21	1	11.0	1	1	0	28619.0	152.0	203	0
4	0	29	1	41.0	1	1	0	27496.0	152.0	39	0

train_encoded_df

ENTRAÎNEMENT DU MODÈLE

1. CHOIX DE L'ALGORITHME



ENTRAÎNEMENT DU MODÈLE

2. PRÉPARATION DE TABLEAU NUMPY X ET Y

```
# préparer les tableau X et Y
y = train_encoded_df['Response']
X = train_encoded_df.drop('Response', axis=1).to_numpy() # eliminer la colonne 'Response'
```

3. ENTRAÎNER NOTRE MODÈLE

```
# entraîner notre modèle
model.fit(X, y)
```

ÉVALUER LA PERFORMANCE DU MODÈLE

```
# évaluer notre loèle
model.score(X, y)
```

Le résultat de l’évaluation

```
0.8852926590555458
```

PRÉDICTION

FONCTION DE PRÉDICTION

```
# préparer les tableau X et Y
def interested(gender, age, driving_license, region_code, previously_insured, vehicle_age, vehicle_damage, annual_premium, policy_sales_channel, vintage):
    x_predict = np.array([
        gender, age, driving_license, region_code, previously_insured, vehicle_age, vehicle_damage, annual_premium, policy_sales_channel, vintage
    ]).reshape(1, 10)
    predictClass = model.predict(x_predict) # pédicri à quelle classe (intérissé ou non)
    probaPredict = model.predict_proba(x_predict) # la probabilité d'être intérissé ou non
    return (predictClass, probaPredict)
```

TESTER LE MODÈLE

```
print(interested(1, 25, 1, 11.0, 1, 1, 1, 35786.0, 152.0, 53))
… (array([0], dtype=int64), array([[1., 0.]]))


print(interested(1, 74, 1, 11.0, 1, 2, 1, 35786.0, 152.0, 53))
… (array([0], dtype=int64), array([[0.8, 0.2]]))
```

CONCLUSION

conclusion