

Analyse et prédiction des décisions de crédit

- ESSOMBA LENJI Chris Manuel
- 29/05/2023



SOMMAIRE

- INTRODUCTION
- METHODOLOGIE
- RESULTATS
- CONCLUSION

INTRODUCTION

- La gestion des décisions de crédit est un enjeu majeur pour de nombreuses institutions financières. Dans ce contexte, l'analyse et la prédiction des décisions de crédit sont des tâches cruciales pour évaluer la solvabilité des demandeurs et prendre des décisions éclairées.
- Dans le cadre de ce projet, nous avons entrepris une étude approfondie sur l'analyse et la prédiction des décisions de crédit. Notre objectif était de comprendre les facteurs qui influencent ces décisions et de développer un modèle prédictif précis pour aider à automatiser ce processus.

MÉTHODOLOGIE

Etape 0 : Importation des données sur Kaggle

- Téléchargement du jeu de données sur les décisions de crédit depuis Kaggle.

Etape 1 : Nettoyage des données

- Traitement des valeurs manquantes, des doublons et des incohérences.

Etape 2 : Analyse exploratoire des données

- Visualisation des variables clés et des relations entre elles.

Etape 3 : Réalisation du modèle

- Préparation des données et division en ensembles d'entraînement et de test.
- Sélection des algorithmes de classification (régression logistique, arbre de décision, k plus proches voisins).
- Entraînement et évaluation des modèles.

Etape 4 : Déploiement du modèle

- Sauvegarde du modèle entraîné.
- Création d'une interface utilisateur conviviale avec Flask.
- Déploiement de l'application sur une plateforme en ligne.

A hand holding a spray bottle, spraying a fine mist against a dark background. The mist is visible as a light, hazy cloud emanating from the nozzle of the bottle. The hand is positioned on the left side of the frame, and the spray extends towards the right. The overall scene is dimly lit, with the primary light source being the spray itself, creating a soft glow around the mist.

ETAPE 1 : NETTOYAGE DES DONNÉES

SÉPARATION DES DONNÉES QUALITATIVES ET NUMÉRIQUES

```
: #creons des listes
cat_data = []
num_data = []
for i,c in enumerate(df.dtypes) :
    if c==object :
        cat_data.append(df.iloc[:, i]) #iloc[:,i] recupere tout les lignes de la colonne d'ordre i
    else :
        num_data.append(df.iloc[:, i])
```

- Afin de traiter efficacement les valeurs manquantes de notre base de données, nous avons opté pour une approche consistant à diviser les données en deux ensembles distincts : l'un contenant les variables qualitatives et l'autre les variables numériques. Cette division nous permet d'éliminer les valeurs nulles et de préparer les données pour les étapes ultérieures de notre analyse et de notre modélisation.

TRAITEMENT DES DONNÉES MANQUANTES : REMPLACEMENT PAR LE MODE DE LA COLONNE

```
: #renseignons les données manquantes pour les variables catégoriques
cat_data = cat_data.apply(lambda x:x.fillna(x.value_counts().index[0]))
cat_data.isnull().sum().any()

: False
```

- Afin de traiter les données manquantes présentes dans notre base de données, nous allons mettre en place un algorithme spécifique. Cet algorithme parcourra notre base de données et remplacera chaque valeur manquante par la valeur la plus fréquente (mode) de la colonne correspondante.

TRAITEMENT DES DONNÉES MANQUANTES : REMPLACEMENT PAR LA VALEUR PRÉCÉDENTE NON NULLE

```
: # Pour les variables numeriques
num_data.fillna(method='bfill', inplace=True)
num_data.isnull().sum().any()

: False
```

- Pour accomplir cette tâche, nous avons mis en place un algorithme qui parcourt notre base de données et remplace chaque valeur nulle par la valeur non nulle qui la précède. Cette approche nous permet de combler les données manquantes en utilisant les valeurs existantes précédentes dans la même colonne.

ENCODAGE DES VALEURS QUALITATIVES

- Cette opération consiste à transformer des variables qualitatives en des valeurs numérique afin de permettre à des algorithmes de Machine Learning de les traiter

```
: le = LabelEncoder()  
for i in cat_data:  
    cat_data[i] = le.fit_transform(cat_data[i])  
cat_data
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Property_Area
0	0	1	0	0	1	0	2
1	1	1	1	1	1	0	0
2	2	1	1	0	1	1	2
3	3	1	1	0	0	0	2
4	4	1	0	0	1	0	2
...
609	609	0	0	0	1	0	0
610	610	1	1	3	1	0	0
611	611	1	1	1	1	0	2
612	612	1	1	2	1	0	2
613	613	0	0	0	1	1	1

614 rows × 7 columns

Rapport réalisé par Chris ESSOMBA.

CONCATÉNATION DES TABLES ET DIVISIONS DES DONNÉES EN VARIABLE CIBLES ET PRÉDICTIVES

- Après le nettoyage nos deux tables nous allons les concaténer en une seule table et ensuite depuis cette table extraire notre variable cible c'est à dire la variable que nous voulons prédire.

```
: #Concatenons nos deux data en une seule pour la suite du traitement  
#Notre target value sera à par et les autres aussi  
X= pd.concat([cat_data, num_data], axis=1)  
y= pd.DataFrame(target)
```

```
: y
```

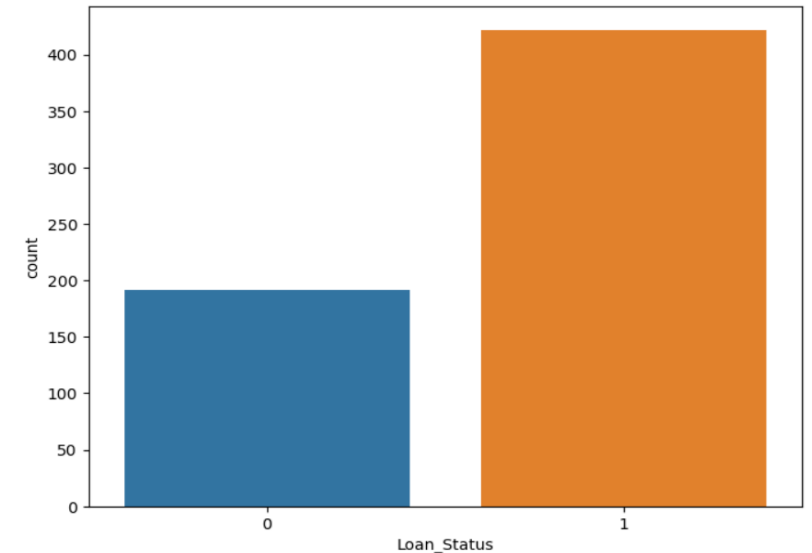
```
:  
      Loan_Status  
0                1  
1                0  
2                1  
3                1
```

The background is a dark, blurred image featuring financial data visualizations. It includes a line graph with white circular markers and a yellow bar chart. Faint numerical values are visible, such as '153.102', '154.178', and '2455'.

ETAPE 2 : ANALYSE EXPLORATOIRE

ANALYSE DE LA PROPORTION DES CRÉDITS ACCORDÉS ET REFUSÉS

- À la lumière de notre analyse, nous constatons que près de 69% des demandes de crédit sont accordées, tandis que 31% sont refusées. Ces résultats nous amènent à nous interroger sur les facteurs qui influencent les décisions d'accord ou de refus de crédit.

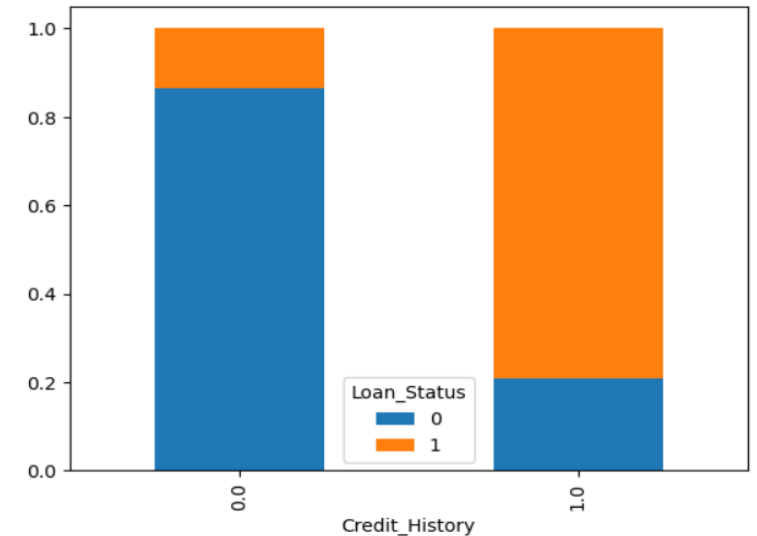


```
: #Quel sont concretement le pourcentage de chacun d'eux?
plt.figure(figsize=(8,6))
sns.countplot(target)
yes=target.value_counts()[1]/len(target)
no=target.value_counts()[0]/len(target)
print(f'Le pourcentage des crédits accordé est : {yes}')
print(f'Le pourcentage des crédits non accordé est : {no}')
```

Le pourcentage des crédits accordé est : 0.6872964169381107
Le pourcentage des crédits non accordé est : 0.3127035830618892

ANALYSE DES ANTÉCÉDENTS DE CRÉDIT

- En examinant le diagramme des clients qui ont obtenu un crédit précédemment (à droite), nous observons que la majorité de ces clients ont déjà bénéficié d'un crédit antérieurement (représentés en orange). En revanche, parmi les clients dont la demande de crédit a été refusée, très peu d'entre eux ont déjà reçu un crédit par le passé.
- Ces observations nous permettent de conclure que le fait d'avoir déjà obtenu un crédit précédemment est un facteur important dans la décision d'accorder ou de refuser un crédit.



```
#Quel est l'influence de la variable Credit_History?
ct = pd.crosstab(df['Credit_History'], df['Loan_Status'])

# Calculer le pourcentage de chaque catégorie dans chaque groupe de revenu
ct_percent = ct.div(ct.sum(axis=1), axis=0)

# Créer le graphique à barres
ax = ct_percent.plot(kind='bar', stacked=True)

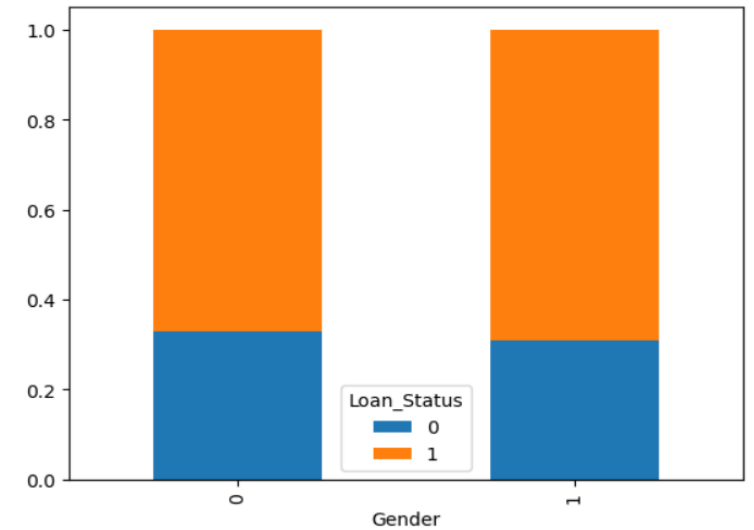
# Ajouter les valeurs de pourcentage aux barres
for i in ax.containers:
    ax.bar_label(i, label_type='edge', labels=i.get_height().round(2))

# Ajouter les titres et les échelles
plt.xlabel('Credit_History')
plt.ylabel('Count')
plt.title('Relationship between Categorical Feature and Target Variable')
plt.legend(title='Loan_Status', loc='upper right')

# Afficher le graphe
plt.show()
```

ANALYSE DE L'IMPACT DU GENRE DES DEMANDEURS

- Nous souhaitons déterminer si les hommes ont plus de chances d'obtenir un crédit que les femmes, ou si c'est l'inverse, ou encore si la décision d'accord ou de refus de crédit n'est pas influencée par le genre du demandeur.
- L'analyse des diagrammes d'accord et de refus de crédit révèle que la proportion d'hommes et de femmes est répartie de manière équitable de part et d'autre. Nous constatons ainsi qu'il n'y a pas de disparité significative entre les deux groupes en termes de décisions d'octroi ou de refus de crédit basées sur le genre.
- Nous pouvons donc conclure que le genre du demandeur n'influence pas la décision de crédit



```
ct = pd.crosstab(df['Gender'], df['Loan_Status'])

# Calculer le pourcentage de chaque catégorie dans chaque groupe de revenu
ct_percent = ct.div(ct.sum(axis=1), axis=0)

# Créer le graphique à barres
ax = ct_percent.plot(kind='bar', stacked=True)

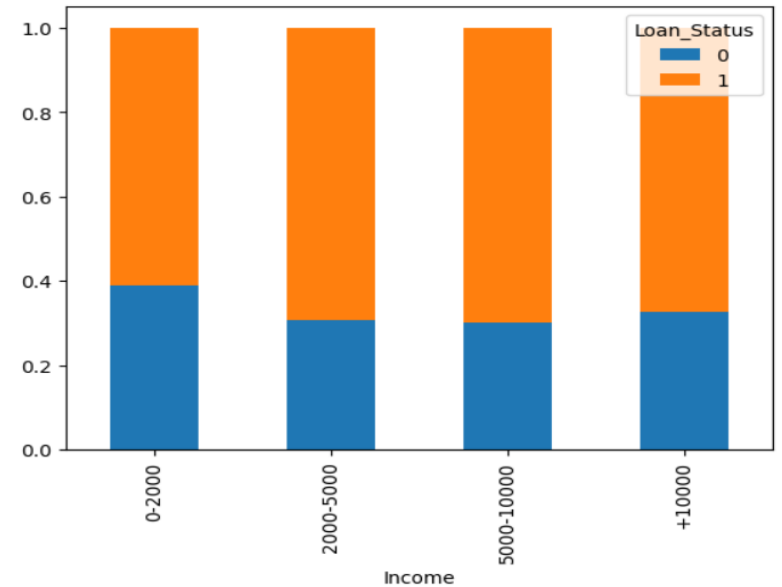
# Ajouter les valeurs de pourcentage à la barre
for i in ax.containers:
    ax.bar_label(i, label_type='edge', labels=i.get_height().round(2))

# Ajoutons les titres et les échelles
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Relationship between Categorical Feature and Target Variable')
plt.legend(title='Loan_Status', loc='upper right')

# Show the plot
plt.show()
```

ANALYSE DE L'IMPACT DU REVENU DU DEMANDEUR

- En examinant le pourcentage de crédits acceptés (représentés en orange) en fonction des différentes tranches de salaire, nous observons que celui-ci est d'environ 60% pour les revenus jusqu'à 2000 euros par mois, tandis qu'il atteint près de 70% au-delà de ce montant.
- Cette analyse nous conduit à conclure que le montant du revenu est un facteur important dans la décision d'octroi de crédit.



```
#Quel est l'influence du revenu du demandeur?
#Comme le minimum est 150 et le maximum 81.000 on va separer en 10 par de 10.000
df['Income'] = pd.cut(df['ApplicantIncome'], bins=[ 0, 2000, 5000, 10000, 100000000 ], 1

# Create a cross-tabulation table of "age_group" and a binary target variable called "y"
ct = pd.crosstab(df['Income'], df['Loan_Status'])

# Calculate the percentage of each category within each income group
ct_percent = ct.div(ct.sum(axis=1), axis=0)

# Create the bar plot
ax = ct_percent.plot(kind='bar', stacked=True)

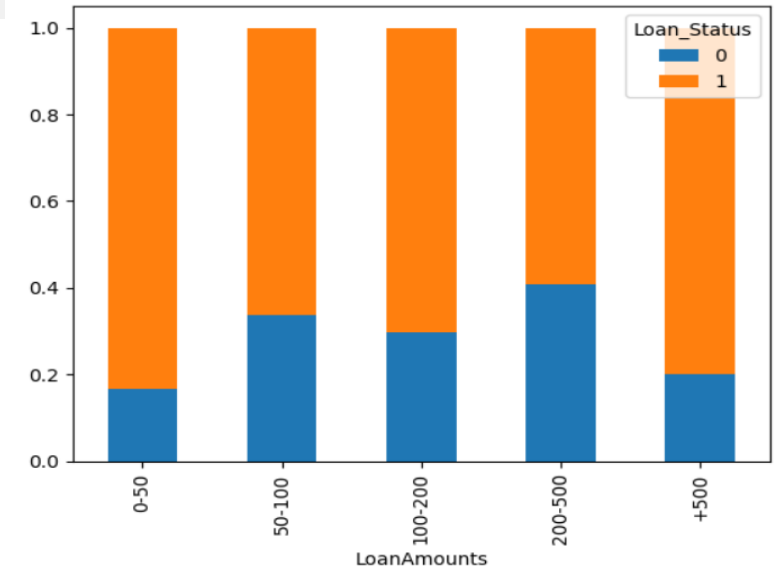
# Add the percentage values to the bars
for i in ax.containers:
    ax.bar_label(i, label_type='edge', labels=i.get_height().round(2))

# Add a legend and axis labels
plt.legend(title='Loan_Status')
plt.xlabel('ApplicantIncome')
plt.ylabel('Percentage')

# Show the plot
plt.show()
```


ANALYSE DE L'IMPACT DU MONTANT DU CRÉDIT DEMANDÉ

- Lorsque nous examinons les pourcentages de crédits acceptés en fonction des différentes tranches de montants de crédit et de revenus, nous constatons des tendances significatives. Les crédits de montants très faibles (entre 0 et 50) ainsi que les revenus très élevés (supérieurs à 500) ont une probabilité plus élevée d'être acceptés, avec des taux d'acceptation de plus de 80%. En revanche, pour les revenus situés dans la plage de 50 à 500, les taux d'acceptation oscillent entre 60% et 70%.
- Ainsi, nous pouvons conclure que les montants de crédit très faibles et les revenus très élevés ont plus de chances d'être acceptés, tandis que les revenus situés entre 50 et 500 présentent une probabilité d'acceptation légèrement inférieure



```
#Quel est l'influence du montant du credit?
#Comme le minimum eu c'est 9 et le maximum 700 et la moyenne c'est 100
df['LoanAmounts'] = pd.cut(df['LoanAmount'], bins=[ 0, 50, 100, 200, 500, 1000000000 ], labels=['0-50', '50-100', '100-200', '200-500', '+500'])

# Créer un tableau croisé de "age_group" et une variable cible binaire appelée "y"
ct = pd.crosstab(df['LoanAmounts'], df['Loan_Status'])

#Calculer le pourcentage de chaque catégorie dans chaque groupe de revenu
ct_percent = ct.div(ct.sum(axis=1), axis=0)

# Créer Le graphique à barres
ax = ct_percent.plot(kind='bar', stacked=True)

# Ajouter Les valeurs de pourcentage aux barres
for i in ax.containers:
    ax.bar_label(i, label_type='edge', labels=i.get_height().round(2))

# Ajouter une légende et des étiquettes d'axe
plt.legend(title='Loan_Status')
plt.xlabel('LoanAmount')
plt.ylabel('count')

# Afficher Le graphe
plt.show()
```

ETAPE 3 : RÉALISATION DU MODEL

SÉPARATION DES DONNÉES EN ENSEMBLES D'ENTRAÎNEMENT ET DE TEST

- Nous allons équitablement diviser notre base de données en deux ensembles :
- L'ensemble de données d'entraînement est utilisé pour ajuster les paramètres du modèle, c'est-à-dire pour lui permettre d'apprendre à partir des exemples fournis.
- L'ensemble de données de test est utilisé pour évaluer les performances du modèle sur des données inconnues.

```
sss = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train, test in sss.split(X, y):
    X_train, X_test = X.iloc[train], X.iloc[test]
    y_train, y_test = y.iloc[train], y.iloc[test]

print('X_train taille' , X_train.shape)
print('X_test taille' , X_test.shape)
print('y_train taille' , y_train.shape)
print('y_test taille' , y_test.shape)
```

```
X_train taille (491, 11)
X_test taille (123, 11)
y_train taille (491, 1)
y_test taille (123, 1)
```

DÉFINITION DE LA FONCTION D'ENTRAÎNEMENT

- Pour définir notre fonction d'entraînement, nous allons entraîner nos données avec trois algorithmes de classification performants qui sont : la régression logistique, l'arbre de décision et le Classifieur des k plus proches voisins
- Comme on peut l'observer, cette fonction retourne le pourcentage de précision de chacun de ces modèles, qui est l'information la plus importante pour le choix du modèle à utiliser pour la prédiction des décisions de crédit.

```
: #Définissons nos model
models={
    'LogisticRegression' : LogisticRegression(random_state=42),
    'DecisionTreeClassifier' : DecisionTreeClassifier(max_depth=1, random_st
    'KNeighborsClassifier': KNeighborsClassifier()
}
#Définissons la fonction de precision (metrique)
def accu(y_true, y_pred, retu=False):
    acc = accuracy_score(y_true, y_pred)
    if retu :
        return acc
    else :
        print(f'La precision du model est : {acc}')

#définissons la fonction d'entraînement
def train_test_eval(models, X_train, y_train, X_test, y_test):
    for name, model in models.items():
        print(name, ':')
        model.fit(X_train, y_train)
        accu(y_test, model.predict(X_test))
        print('-'*30)

train_test_eval(models, X_train, y_train, X_test, y_test)

LogisticRegression :
La precision du model est : 0.8536585365853658
-----
DecisionTreeClassifier :
La precision du model est : 0.8455284552845529
-----
KNeighborsClassifier :
La precision du model est : 0.6504065040650406
-----
```

CHOIX DU MODEL

- Dans notre processus de prédiction des décisions de crédit, nous avons sélectionné uniquement les champs pertinents de notre base de données. Ces champs correspondent aux informations nécessaires que l'utilisateur doit fournir pour obtenir sa décision de crédit.
- Ensuite nous appliquons à ces données la fonction d'entraînement définie précédemment
- Sur la base du pourcentage de précision choisissons le model de l'arbre de décision pour la prédiction des décisions de crédit.

```
: X_2= X[['LoanAmount', 'ApplicantIncome', 'Credit_History', 'Married', 'CoapplicantIncome']]

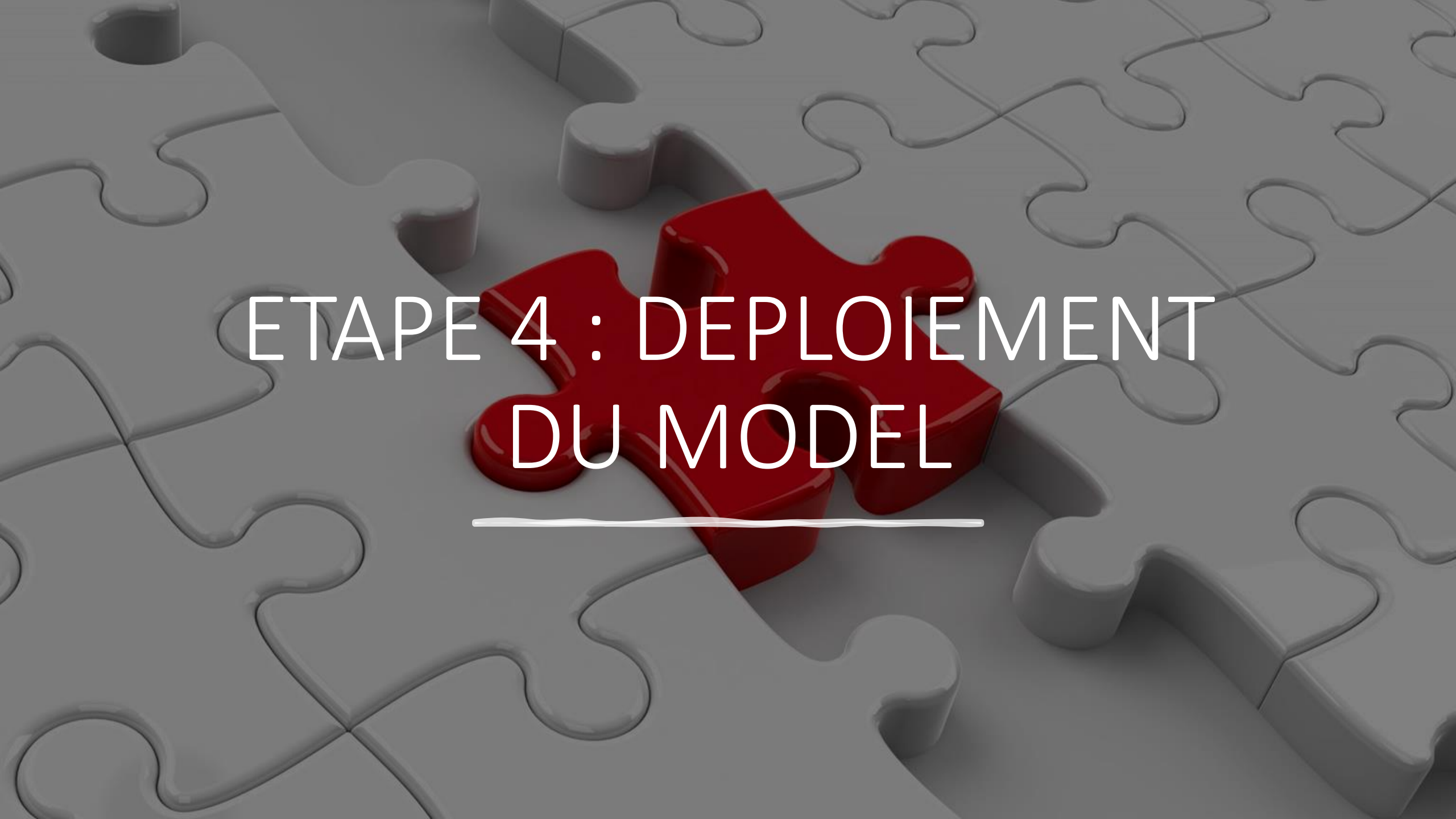
: #Entrainons
sss = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train, test in sss.split(X_2, y):
    X_train, X_test = X_2.iloc[train], X_2.iloc[test]
    y_train, y_test = y.iloc[train], y.iloc[test]

print('X_train taille' , X_train.shape)
print('X_test taille' , X_test.shape)
print('y_train taille' , y_train.shape)
print('y_test taille' , y_test.shape)

X_train taille (491, 5)
X_test taille (123, 5)
y_train taille (491, 1)
y_test taille (123, 1)

: train_test_eval(models, X_train, y_train, X_test, y_test)

LogisticRegression :
La precision du model est : 0.7804878048780488
-----
DecisionTreeClassifier :
La precision du model est : 0.8455284552845529
-----
KNeighborsClassifier :
La precision du model est : 0.6422764227642277
-----
```



ETAPE 4 : DEPLOIEMENT DU MODEL

ENTRAÎNEMENT DU MODÈLE ET SAUVEGARDE DE LA SÉRIALISATION DU MODEL

- Après avoir préparé nos données, nous procédons à l'entraînement du modèle en utilisant les informations fournies par l'utilisateur. Une fois l'entraînement terminé, nous stockons le modèle entraîné dans une variable.
- Pour assurer la persistance du modèle, nous le sérialisons en le convertissant en une séquence d'octets. Cette sérialisation nous permet de stocker le modèle dans un fichier afin de pouvoir le récupérer ultérieurement. Dans notre cas, nous stockons la sérialisation du modèle dans le fichier "model.pkl".

```
: #Appliquer l'arbre de decision sur notre base de données  
Classifier = DecisionTreeClassifier(max_depth=1)  
Classifier.fit(X_2, y)  
  
: DecisionTreeClassifier(max_depth=1)  
  
: #Enregistrer le modele  
pickle.dump(Classifier, open('model.pkl', 'wb'))  
  
:
```


RÉALISATION DE L'INTERFACE WEB

- Pour développer l'interface web que les utilisateurs utiliseront pour saisir leurs informations, nous avons choisi d'utiliser le Framework Flask de Python. Flask offre les fonctionnalités nécessaires pour créer une application web facilement.
- Nous adaptons Flask pour intégrer notre modèle et permettre à l'application d'importer le modèle entraîné. De plus, nous configurons l'application pour accéder à l'interface HTML, qui sera utilisée pour l'affichage des formulaires.
- Lorsque l'utilisateur saisit ses informations dans le formulaire, l'application récupère ces données et les utilise comme entrées pour effectuer une prédiction en utilisant le modèle entraîné. En fonction du résultat de la prédiction, l'application affiche un message indiquant si le crédit est accordé ou refusé.

```
1  import numpy as np
2  from flask import Flask, request, jsonify, render_template
3  import pickle
4
5
6  #Importation du model
7  app = Flask(__name__)
8  model = pickle.load(open('model.pkl', 'rb'))
9
10 #Connexion avec la page web d'accueil de l'application
11 @app.route('/')
12 def home():
13     return render_template('index.html')
14
15 #Connexion avec les données entrées par l'utilisateur
16 @app.route('/predict', methods=['POST'])
17 def predict():
18     '''
19     For rendering results on HTML GUI
20     '''
21     int_features = [int(x) for x in request.form.values()]
22     final_features = [np.array(int_features)]
23     prediction = model.predict(final_features)
24
25     output = round(prediction[0], 2)
26
27     #Choix du message retourné
28     f = output
29     if(f==1):
30         return render_template('index.html', prediction_text='Le crédit est accordé')
31     else :
32         return render_template('index.html', prediction_text='Le crédit est refusé')
33
```




CONCLUSION

- Notre projet d'analyse et de prédiction des décisions de crédit a été réalisé avec succès en suivant une méthodologie rigoureuse.
- Nous avons importé les données de crédit depuis Kaggle et les avons préparées en effectuant un nettoyage approfondi.
- L'analyse exploratoire des données nous a permis de comprendre les principaux facteurs influençant les décisions de crédit.
- En utilisant des techniques de modélisation telles que la régression logistique, l'arbre de décision et le Classifieur des k plus proches voisins, nous avons obtenu des modèles précis pour prédire les décisions de crédit.
- L'application a été déployée avec succès, offrant aux utilisateurs la possibilité de saisir leurs informations et de recevoir instantanément une décision de crédit.
- Nos résultats démontrent l'importance de variables telles que le montant du crédit, l'historique des crédits précédents et le revenu dans la prise de décision de crédit.
- Ce projet ouvre des perspectives intéressantes pour améliorer les systèmes de prédiction des décisions de crédit et contribuer à une prise de décision plus équitable et précise.
- Nous recommandons d'approfondir l'analyse en explorant davantage de variables et en utilisant des techniques avancées de modélisation pour améliorer les performances du modèle.



RÉFÉRENCE

- La base de données utilisée dans notre projet a été téléchargée à partir de Kaggle, une plateforme en ligne populaire pour les scientifiques des données et les praticiens du Machine Learning. Le jeu de données spécifique que nous avons utilisé est intitulé *Loan Prediction Problem Dataset* et il a été fourni par DEBDATTA CHATTERJEE sur Kaggle. Ce jeu de données offre une vaste collection d'informations sur les demandes de crédit, ce qui en fait une ressource précieuse pour notre analyse et nos prédictions.



Je vous remercie
pour votre attention !