Sommaire :

1 Introduction :

L'objectif de ce TP est de découvrir les prédicteurs les plus performants de la présence des volcans. L'ensemble de données comprend 1679 enregistrements de volcans dont 839 ne contiennent pas de volcans stockées dans le fichier « **mars_train.csv** » . Il y a 3600 variables quantitatives dans l'ensemble de données qui contribuent à prédire la présence du volcan. En utilisant ces caractéristiques, le TP vise à identifier les prédicteurs les plus performants pour le problème de présence des volcans.

Les données à prédire sont stockées dans le fichier « **mars_unknown.csv** » qui compte 419 images de volcans.

1 - Types de variables :

Unités d'observation de **mars_train.csv** :

	0	0.29347	0.097823	0.034586	0.034586.1	0.1	0.2	0.3	0.4	0.5	...	0.3207	0.60678	0.20226	0.14302.2	0.14302.3	0.13484	0.40452	0.3208	0.572
0	0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.12274	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0	(
1	0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.00000	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0	(
2	0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.00000	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0	(
3	0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.00000	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0	(
4	0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.00000	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0	(

5 rows × 3601 columns

Unités d'observation de **mars_unknown.csv** :

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	...	0.3096	0.3097	0.3098	0.3099	0.3100	0.3101	0.3102	0.3103	0.3104
0	0	0.0000	0.000000	0.00000	0.00000	0.000000	0.000000	0	0.00000	0	...	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0
1	0	0.0000	0.000000	0.00000	0.00000	0.000000	0.000000	0	0.00000	0	...	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0
2	0	0.0000	0.000000	0.00000	0.00000	0.000000	0.000000	0	0.00000	0	...	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0
3	0	0.0000	0.000000	0.00000	0.00000	0.000000	0.000000	0	0.00000	0	...	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0
4	0	0.1571	0.052366	0.12343	0.12343	0.017455	0.052366	0	0.29623	0	...	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0

5 rows × 3600 columns

2 | *Analyse des données :*

Les données de mars_train.csv contiennent 3600 variables de prédiction et une étiquette binaire à la colonne 3601 qui indique la présence d'un volcan par le chiffre 2 et son absence par le chiffre 1.

Les données sont distribuées de la manière suivante :

	0	0.29347	0.097823	0.034586	0.034586.1	0.1	0.2	0.3	0.4	0.5	...	0.3207	0.60678	0.
count	1679.0	1679.000000	1679.000000	1679.000000	1679.000000	1679.000000	1679.000000	1679.0	1679.000000	1679.0	...	1679.0	1679.000000	1679.0
mean	0.0	0.017981	0.005994	0.010376	0.010376	0.002780	0.008338	0.0	0.016768	0.0	...	0.0	0.025434	0.0
std	0.0	0.079440	0.026480	0.052552	0.052552	0.016165	0.048494	0.0	0.074813	0.0	...	0.0	0.097370	0.0
min	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	...	0.0	0.000000	0.0
25%	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	...	0.0	0.000000	0.0
50%	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	...	0.0	0.000000	0.0
75%	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	...	0.0	0.000000	0.0
max	0.0	0.698540	0.232850	0.577350	0.577350	0.214290	0.642860	0.0	0.727610	0.0	...	0.0	0.792410	0.2

8 rows × 3601 columns

Et pour l'étiquette (colonne « 1 ») :

```
count    1679.000000
mean        1.500298
std         0.500149
min         1.000000
25%         1.000000
50%         2.000000
75%         2.000000
max         2.000000
Name: 1, dtype: float64
```

Avec 839 chiffre « 1 »

1 – Matrice de corrélation :

	0	0.29347	0.097823	0.034586	0.034586.1	0.1	0.2	0.3	0.4	0.5	...	0.3207	0.60678	0.20226	0.14302.2	0.1430
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	Na
0.29347	NaN	1.000000	1.000000	0.636174	0.636174	0.689669	0.689669	NaN	0.693969	NaN	...	NaN	-0.027548	-0.027548	-0.020827	-0.0208
0.097823	NaN	1.000000	1.000000	0.636173	0.636173	0.689671	0.689670	NaN	0.693969	NaN	...	NaN	-0.027549	-0.027548	-0.020827	-0.0208
0.034586	NaN	0.636174	0.636173	1.000000	1.000000	0.556633	0.556633	NaN	0.779746	NaN	...	NaN	-0.016476	-0.016476	-0.015134	-0.0151
0.034586.1	NaN	0.636174	0.636173	1.000000	1.000000	0.556633	0.556633	NaN	0.779746	NaN	...	NaN	-0.016476	-0.016476	-0.015134	-0.0151
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
0.13484	NaN	-0.026830	-0.026830	-0.002139	-0.002139	-0.023563	-0.023563	NaN	-0.019727	NaN	...	NaN	0.677419	0.677418	0.674112	0.6741
0.40452	NaN	-0.026830	-0.026830	-0.002139	-0.002139	-0.023563	-0.023563	NaN	-0.019727	NaN	...	NaN	0.677419	0.677417	0.674110	0.6741
0.3208	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	Na
0.57208	NaN	-0.020838	-0.020838	-0.024112	-0.024112	-0.009851	-0.009851	NaN	-0.006934	NaN	...	NaN	0.365668	0.365668	0.570589	0.5705
1	NaN	-0.067858	-0.067858	-0.054645	-0.054645	-0.059806	-0.059806	NaN	-0.054605	NaN	...	NaN	-0.046094	-0.046094	-0.013567	-0.0135

3601 rows × 3601 columns

Puisque cette matrice est de grande taille et contient plusieurs valeurs de corrélation supérieures à 0,7 en valeur absolue alors l'analyse de composante principale est nécessaire afin d'éliminer la redondance et de conserver.

En Appliquant l'ACP on obtient le tableau de variabilité suivant :

```
array([0.06407191, 0.05199354, 0.04888068, 0.04026899, 0.03640937,
       0.0323487 , 0.03181474, 0.03026314, 0.02823201, 0.02655836,
       0.02504701, 0.02386342, 0.02320577, 0.02199544, 0.02175377,
       0.02090566, 0.02072746, 0.01938721, 0.01907322, 0.01853188,
       0.0180664 , 0.01761855, 0.01705078, 0.01673604, 0.01637185,
       0.01584782, 0.01579362, 0.01541963, 0.0150385 , 0.0143412 ,
       0.01413352, 0.01370712, 0.01337261, 0.01317714, 0.0129106 ,
       0.01286059, 0.01210267, 0.01187953, 0.01173205, 0.01154979,
       0.01136676, 0.0110704 , 0.0109884 , 0.01081878, 0.01076465,
       0.01029006, 0.01026055, 0.01008473, 0.00979151, 0.00952185])
```

- En utilisant le critère du coude on obtient un nombre de variables égal à 50.
- En utilisant le critère 80% - 90% on obtient 33.

3 | Algorithme :

- On utilise l'apprentissage supervisé pour créer le modèle
- 80% des données sera dédiée à l'entrainement du modèle de prédiction, et 20% sera utilisée pour tester ce modèle. On utilise la validation croisée pour éviter le surapprentissage.
- On a testé : DecisionTreeClassification pour créer le modèle.
    - RandomForestClassification.
    - SVC pour le noyau gaussien et polynomial
    - La régression linéaire n'est pas efficace car on a plusieurs variables.
- On a constaté que SVC et DecisionTreeClassification avait la meilleure précision pour un écart type faible. SVC est bon pour la prédiction des 2 (présence) et l'arbre est bon pour la prédiction des 1 (absence).
- En combinant les deux on réduit les faux positifs et les faux négatifs.
- On a effectué un boost sur DecisionTreeClassification et un bagging sur SVC.
- On a effectué un vote sur ces derniers.

Algorithme :

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from math import *
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import VotingClassifier

#infos sur la prediction
def pred(model_object,predictors,compare):
    """1.model_object = modele
       2.predictors = données à predire
       3.compare = y_train"""
    predicted = model_object.predict(predictors)
    # Déterminer les faux positifs et les vrais positifs
    cm = pd.crosstab(compare,predicted)
    TN = cm.iloc[0,0]
    FN = cm.iloc[1,0]
    TP = cm.iloc[1,1]
    FP = cm.iloc[0,1]
    print("<<------- Prediction ------->> ")
    print(cm)
    print()
    ##vérifier la précision
    print('<<------- Classification  ------->>')
    print('Précision : ', round(((TP+TN)*100)/(TP+TN+FP+FN),2))
    print()

    print(classification_report(compare,predicted))
```

```
#-------------------------main-------------------------------------#

#Extraction des données
f = open('mars_train.csv')
Volcans = pd.read_csv(f)
file = open('mars_unknown.csv')
pre =pd.read_csv(file)

Xe = pre
X = Volcans.drop(["1"], axis=1)
y = Volcans["1"]
#ACP
pca = PCA(n_components=50)
Xp = pca.fit_transform(X)
#unknown
Xe = pca.fit_transform(Xe)
#train data
X_train, X_test, y_train, y_test =
train_test_split(Xp,y,test_size=0.2,random_state=0)
#format standard
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
#unknown
Xe = scaler.fit_transform(Xe)

bags = BaggingClassifier(SVC(), random_state=10, n_jobs=6).fit(X_train,
y_train)
ada = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1),
algorithm="SAMME", n_estimators=200)
#------------------best score-----------------------#
vote = VotingClassifier(estimators=[('bagging(svc)', bags), ('ada(tree)',
ada)], voting='soft').fit(X_train, y_train)
pred(vote,X_test,y_test)
#unknown
ye = vote.predict(Xe)
print(ye)
#Generation de Y final
df =  pd.DataFrame(ye)
df.to_csv('prediction.csv', sep=',', header=None, index=None)
```

4 | Résultats :

Résultats sur **mars_train.csv** (y_test) :

<<-------Prediction ------->>
col_0 1 2
1
1 164 6
2 21 145
<<-------Classification ------->>
Precision : 91.96

	precision	recall	f1-score	support
1	0.89	0.96	0.92	170
2	0.96	0.87	0.91	166
Accuracy			0.92	336
macro avg	0.92	0.92	0.92	336
weighted avg	0.92	0.92	0.92	336

Résultats sur **mars_unknown.csv** :

mars_train.csv	Contient les données d'entrainement et de test
mars_unknown.csv	Contient des données brutes sans étiquette
prediction.csv	Contient la prédiction de mars_unknown.csv

5 | Conclusion :

- Le modèle choisi est performant puisque la moyenne de la précision est 92% et du rappel est 92% pour un chiffre significatif de tests égal à 100, ce qui montre la pertinence du modèle.

- Le modèle manque de précision pour les images qui ne contiennent pas de volcans et ceci est dû au bruit d'images non totalement filtrées.