PRÉDICTION DE DIFFÉRENTES CONDITIONS D'UTILISATIONS DES PNEUMATIQUES

Groupe :
CANCEL Alexandre
FRADIER Corentin
JAN Alexandre
JHUGROO Ganesh

23 janvier 2023

Superviseurs : Clément Linares
Emeline Queguiner

SOMMAIRE

01 - INTRODUCTION

Constat du marché:

Quel peut être la solution pour améliorer le confort et la sécurité de la conduite des utilisateurs?

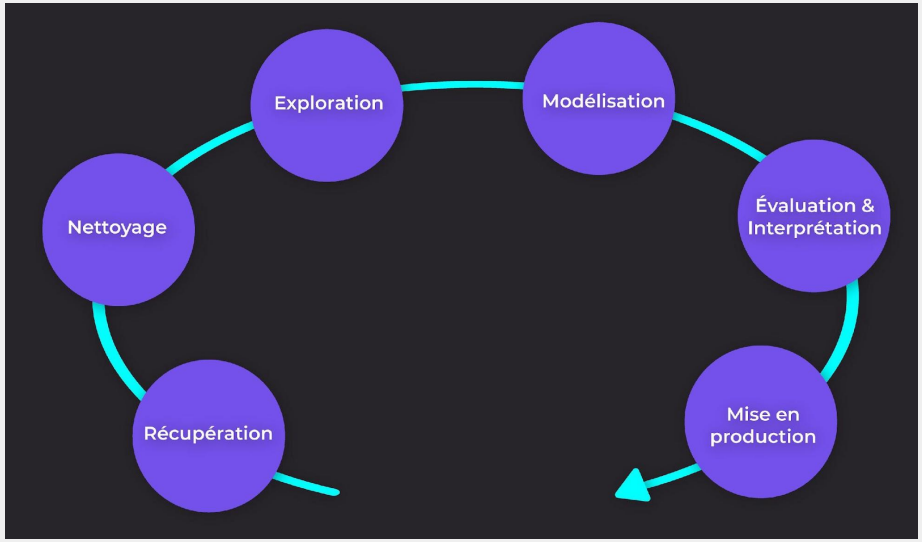
Les données:

'DateTime', 'Imbibition', 'Vehicle', 'TireBrand', 'TireName', 'TireSizeFront', 'TireSizeRear', 'TirePressure', 'Load', 'Track', 'FrontRear', 'channel', 'latitudeT', 'longitudeT', 'speed', 'accelX','accelY', 'speedClass', 'ambiantTemperature', 'weatherCheck', 'textureCheck', 'TireWear'

02 - SOLUTION

- Techniques de data sciences
- Aide à la décision
- Modèles statistiques à partir des données

02 - SOLUTION



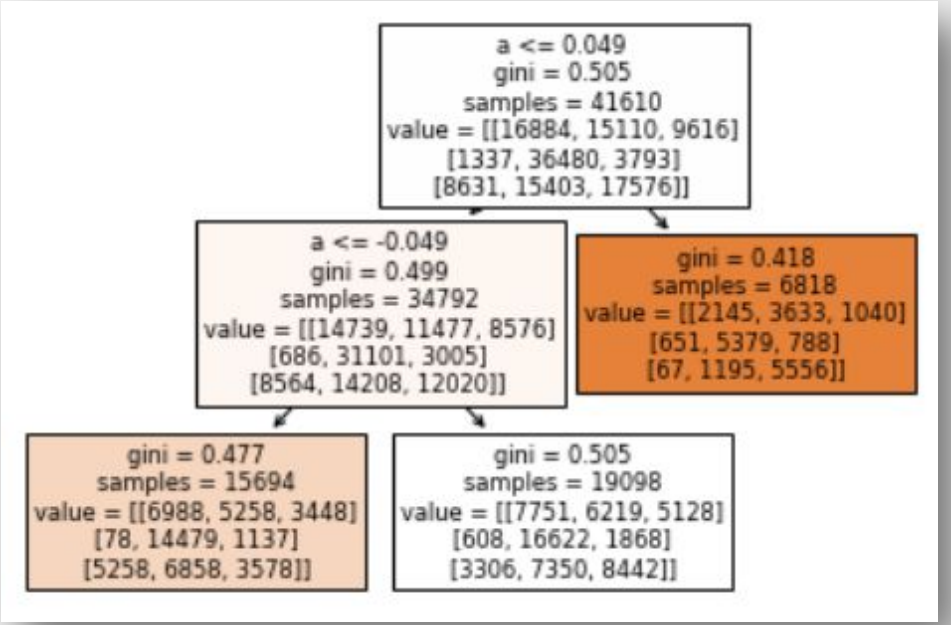
Cycle de traitement et exploitation des données

- Machine Learning supervisé

03 - PRODUIT

Arbre de decision:

un ensemble de règles simples qui permettent de réaliser des prévisions ou de segmenter une population



Avantages:

- Simple à comprendre et à interpréter avec la visualisation
- Peu de préparation de données nécessaire

04 – AVANTAGES COMMERCIAUX

- Amélioration de la sécurité routière
- Eviter les coûts de réparation
- Renforcer la fidélité à long terme

CONCLUSION

Présentations des résultats

$$\text{Accuracy} = \frac{Vrai\ positif + Vrai\ négatif}{Total}$$

```
print(metrics.accuracy_score(Y_valid['weatherCheck'],predvalid[:,0]))
print(metrics.accuracy_score(Y_valid['textureCheck'],predvalid[:,1]))
print(metrics.accuracy_score(Y_valid['TireWear'],predvalid[:,2]))

print(metrics.confusion_matrix(Y_valid['TireWear'],predvalid[:,2]))
```

```
0.9944324712643678
0.7948994252873564
0.7776580459770115
[[ 687    0  225]
 [ 241 2094  381]
 [ 311   80 1549]]
```

06 – PARTIE TECHNIQUE

- Nettoyage des données
  - Suppression de variables explicatives « inutiles »
  - Séparation des données
  - Discrétisation des données
- Arbre de décision
  - Pré-traitements
  - Mise en place de l'arbre de décision

SUPPRESSION DES COLONNES

Suppression des variables explicatives « inutile » (suppression des variables avec une seule valeurs)

```
#Supression des colonnes inutiles
supr=['Vehicle','TireBrand','TireName','TireSizeFront','TireSizeRear','TirePressure','Track','latitudeT','longitudeT']
DF = DF.drop(supr,axis=1)
```

```
#Convertir le DataFrame en excel
DF.to_excel('clean_data2.xlsx')
```

Création d'une nouvelle base de données

SEPARATION DES DONNEES PAR DATE

```
ListDate=DF['DateTime'].unique()
ListeIndexTrain=[]
ListeIndexTest=[]
ListeIndexValid=[]

for i in range(DF.shape[0]):
  if DF.loc[i]['DateTime'] in ListDate[permut[:132]] :
    ListeIndexTrain.append(i)
  elif DF.loc[i]['DateTime'] in ListDate[permut[132:169]] :
    ListeIndexTest.append(i)
  elif DF.loc[i]['DateTime'] in ListDate[permut[169:]] :
    ListeIndexValid.append(i)
```

```
train_set=DF.loc[ListeIndexTrain][:].copy()
test_set=DF.loc[ListeIndexTest][:].copy()
valid_set=DF.loc[ListeIndexValid][:].copy()
```

Création des trois DataFrame

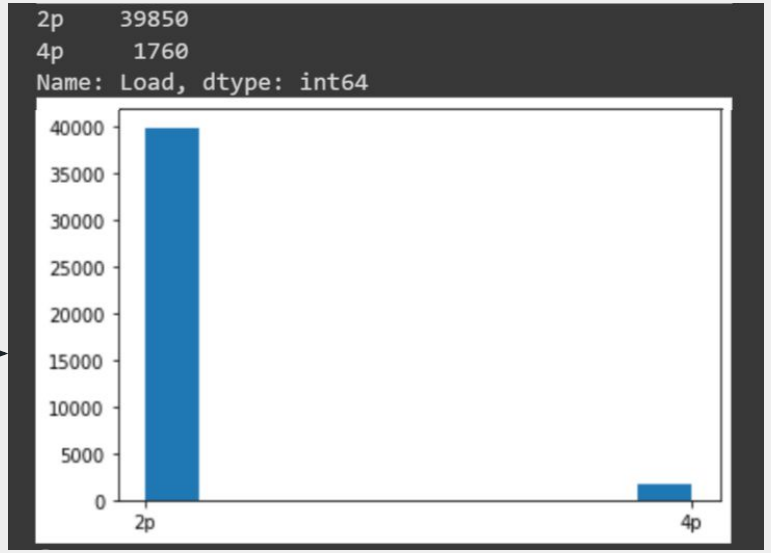
- 1 individu = 1 date
- 10% données de validation
- 20% données de test
- 70% données d'apprentissage

DISCRETISATION DES DONNEES

Histogrammes des variables explicatives

```
for i in range(0, df.shape[1]):
    plt.hist(df[:][df.columns[i]])
    print(df[:][df.columns[i]].value_counts())
    plt.show()
```

```
2p    39850
4p     1760
Name: Load, dtype: int64
```



Histogrammes de « Load » à supprimer

```
Liste_supp = ['Load']
Liste_gard = ['Imbibition', 'FrontRear', 'channel', 'speedClass']
Liste_modi = ['speed', 'accelX', 'accelY', 'ambiantTemperature']
```

Liste des variables explicatives à supprimer, à modifier et à garder

DISCRETISATION DES DONNEES

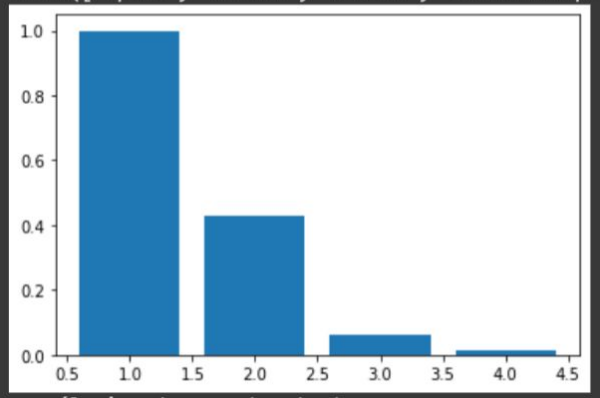
```
Liste_quanti = ['speed', 'accelX', 'accelY', 'ambiantTemperature']
Liste_quali = ['Imbibition', 'FrontRear', 'channel', 'speedClass', 'weatherCheck','textureCheck', 'TireWear']
```

Listes des variables quantitatives et qualitatives

Corrélation entre les variables quantitatives et qualitatives

```
#On commence par les quanli par rapports aux quanti
ListeCouple=[]#liste des couples de forte corrélation
for i in Liste_quali:
    print("Corrélation de ",i)
    rapport=rapport_corr(DF[:][i],DF[:][Liste_quanti])
    if rapport[0]>0.75:# correlation > 0.75
        ListeCouple.append([i,rapport.index[0]])
```

```
Corrélation de  speedClass
speed                0.999187
accelY               0.430749
accelX               0.061609
ambiantTemperature   0.011640
dtype: float64
Index(['speed', 'accelY', 'accelX', 'ambiantTemperature']
```



Couple avec un corrélation > 0.75 : (« speedClass »,<speed »)

DISCRETISATION DES DONNEES

Corrélation entre les variables quantitatives et quantitatives

```
#Meme principe que précédemment mais entre les quanti
i=0
while corr["abscorr"][corr.index[i]]>0.75:
    ListeCouple.append([corr["nom1"][corr.index[i]],corr["nom2"][corr.index[i]]])
    i+=1
print(ListeCouple)
```

Aucun couple de variables quantitatives avec une corrélation > 0.75

DISCRETISATION DES DONNEES

Corrélation entre les variables qualitatives et qualitatives avec la fonction du V de cramer

```
# quali-quali
for i in range (0, len(Liste_quali)) :
  for j in range (i+1, len(Liste_quali)):
    print(Liste_quali[i], Liste_quali[j], cramers_v(DF[:][Liste_quali[i]], DF[:][Liste_quali[j]]))
    if(cramers_v(DF[:][Liste_quali[i]], DF[:][Liste_quali[j]])>0.75):
        ListeCouple.append([Liste_quali[i],Liste_quali[j]])
```

- Aucun couple avec une corrélation > 0.75
- (« Imbibition » , « weatherCheck ») avec une corrélation de 0.63 < 0.75

DISCRETISATION DES DONNEES

- Supprimer une des variables du couple corrélé (« speedClass », »speed »)

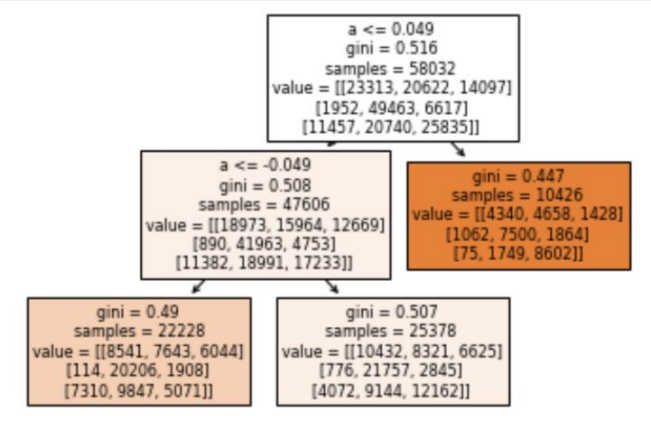
- Suppression de « speed » car :
  - « speedClass » appartient à la liste des variables à garder
  - « speed » appartient à la liste des variables à modifier

```
#On change donc les listes avec les nouvelles modifications
Liste_supp = ['Load', 'speed']
Liste_gard = ['Imbibition', 'FrontRear', 'channel', 'speedClass']
Liste_modi = ['accelX', 'accelY', 'ambiantTemperature']
```

Nouvelle liste des variables explicatives à supprimer, à modifier et à garder

DISCRETISATION DES DONNEES

```
#'accelX'
#Arbre de décision
varapred = ['textureCheck', 'weatherCheck', 'TireWear']
arbre_cla=DecisionTreeClassifier(max_depth=3,min_samples_leaf=5000)
arbre_cla.fit(DF['accelX'].to_numpy().reshape(DF['accelX'].to_numpy().shape[0],1),DF[varapred])
plot_tree(arbre_cla, filled=True,feature_names="accelX")
plt.show()
```



Arbre de decision pour créer les groupe pour les variables à modifier (accelX, accelY, ambiantTemperature)

FONCTION DE DISCRETISATION

```
def modif(donnees):
    #on supprime les variables à supprimer
    varSupp=['Load', 'speed']
    for i in varSupp:
        del donnees[i]
    #'accelX'
    donnees["accelX2"]=pd.Series(index=range(donnees.shape[0]),dtype='str')
    donnees.loc[donnees["accelX"]<=(-0.049),"accelX2"]="-0.4905-(-0.049)"
    donnees.loc[(donnees["accelX"]>(-0.049)) & (donnees["accelX"]<=0.049),"accelX2"]="<0.049 - 0.049"
    donnees.loc[donnees["accelX"]>0.049,"accelX2"]=">0.049"

    #'accelY'
    # ...

    #'ambiantTemperature'
    # ...

    #on supprimer les colonnes qu on à modifiés
    varMod=['accelX', 'accelY', 'ambiantTemperature']
    for i in varMod:
        del donnees[i]
    #on transforme les variables garder en str
    varGard=['Imbibition', 'FrontRear', 'channel', 'speedClass']
    for i in varGard:
        donnees[i]=donnees[i].astype("str")
    return donnees
```

ARBRE DE DECISION

- Application de la fonction de discrétisation sur les données d'apprentissage, de tests et de validations

```
DF_A = modif(DF_A)
DF_T = modif(DF_T)
DF_V = modif(DF_V)
```

- Création d'une liste des variables qualitatives.

```
lstQuali = [var for var in DF_A.columns[:-3] if DF_A[var].dtype == np.object_]
```

- Convertir les variables qualitatives avec des variables indicatrices pour les données d'apprentissage, de tests et de validations

```
dfQualiEncoded = pd.get_dummies(DF_A[lstQuali])
for var in lstQuali :
    del DF_A[var]
DF_A= pd.concat([dfQualiEncoded, DF_A],axis=1)
```

- On sépare les variables à prédire des autres pour l'ensemble des jeux données d'apprentissages, de tests et de validations

```
X_app = DF_A[DF_A.columns[:-3]]
Y_app = DF_A[DF_A.columns[-3:]]
```

ARBRE DE DECISION

- Arbre de décision sur les données tests avec Sklearn.
- On test plusieurs valeurs pour les paramètres.

```
depth=[25,50]
leaf=[10,25,50]
for dep in depth:
    for lea in leaf:
        arbre1=DecisionTreeClassifier(max_depth=dep,min_samples_leaf=lea)
        arbre1.fit(X_app,Y_app)
        plot_tree(arbre1, filled=True)
        plt.show()
        print("depth=",dep,"leaf=",lea)

        predfirst=arbre1.predict(X=X_test)

        print(confusion_matrix(Y_test['weatherCheck'], predfirst[:,0]))
        print(metrics.accuracy_score(Y_test['weatherCheck'],predfirst[:,0]))

        print(confusion_matrix(Y_test['textureCheck'], predfirst[:,1]))
        print(metrics.accuracy_score(Y_test['textureCheck'],predfirst[:,1]))

        print(confusion_matrix(Y_test['TireWear'], predfirst[:,2]))
        print(metrics.accuracy_score(Y_test['TireWear'],predfirst[:,2]))
```

ARBRE DE DECISION

```
arbrefinale=DecisionTreeClassifier(max_depth=50,min_samples_leaf=50)
arbrefinale.fit(X_app,Y_app)
predvalid=arbrefinale.predict(X=X_valid)
print(metrics.accuracy_score(Y_valid['weatherCheck'],predvalid[:,0]))
print(metrics.accuracy_score(Y_valid['textureCheck'],predvalid[:,1]))
print(metrics.accuracy_score(Y_valid['TireWear'],predvalid[:,2]))
```

- On applique sur notre meilleur arbre de décision (depth = 50, leaf = 50), les données de validation.

- Affichage de l'accuracy pour chaque variables à prédire
  « weatherCheck » : 0.9944324712643678
  « textureCheck » : 0.7948994252873564
  « TireWear » : 0.7776580459770115

CE QUE NOUS A APPORTER CE PROJET

AVANTAGES

- Développement de la capacité à travailler en équipe et à gérer un projet de grande envergure, y compris la gestion des difficultés et des échecs.
- Amélioration des compétences en programmation en Python
- Possibilité d'apprendre de ses erreurs et de les corriger pour des projets futurs.

INCONVÉNIENT

- Manque de données pour les conditions météorologiques