PROJET DATA MINING :

Sujet : la santé mentale dans la technologie

Introduction :

L'exploration de données est le processus de classification automatique des cas en fonction des modèles de données obtenus à partir d'un ensemble de données. Un certain nombre d'algorithmes ont été développés et mis en œuvre pour extraire des informations et découvrir des modèles de connaissances qui peuvent être utiles pour l'aide à la décision. Une fois ces modèles extraits, ils peuvent être utilisés pour la classification automatique des mélanges de cas. Bien que la recherche ait été menée à l'aide de divers algorithmes sur de petits ensembles de données (d'une dizaine à quinze attributs).

L'objectif de ce rapport est d'analyser l'attitude des entreprises vis-à-vis de la santé mentale, et d'examiner la fréquence des maladies mentales chez les travailleurs du secteur des technologies.

Certaines des questions qui seront abordées dans l'analyse sont les suivantes :

Les maladies de santé mentale sont-elles plus fréquentes chez les travailleurs du secteur technologique que chez les autres ?

Quel est le rapport entre la taille de l'entreprise et le fait qu'un employeur discute officiellement de la santé mentale ?

Le processus de ce travail était le suivant :

- Bibliothèque et chargement des données
- Nettoyage des données
- Encodage des données
- Matrice de covariance. Comparaison de la variabilité entre les catégories de variables
- Quelques graphiques pour voir les relations entre les données
- Mise à l'échelle et ajustement
- Tuning
- Évaluer les modèles
- Égression logistique
- Classificateur KNeighbors
- Classificateur de l'arbre de décision
- Forêts au hasard
- Mise en sac
- Stimuler
- Empilage
- Prévoir avec le réseau de neurones
- Méthode de réussite
- Créer des prédictions sur le jeu de test
- Soumission
- Conclusions

Source de données :

L'ensemble de données provient d'une enquête réalisée en 2014 par Open Sourcing Mental Illess (OSMI). L'enquête est menée en ligne sur le site Web de l'OSMI et l'équipe d'OSMI a l'intention d'utiliser ces données pour sensibiliser et améliorer les conditions des personnes atteintes de maladie mentale sur le lieu de travail des TI. Il convient de noter que, comme il s'agit d'une enquête en ligne, elle peut être sujette à un biais de réponse volontaire et peut entraîner une surreprésentation des données. L'échantillon de répondants n'a été obtenu par aucune approche d'échantillonnage aléatoire.

De plus, comme il s'agit d'une étude observationnelle avec des biais d'échantillonnage potentiels, la causalité ne peut être inférée. Les résultats de l'enquête peuvent ne pas être généralisables à l'ensemble de la population de techniciens / informaticiens en raison du manque d'échantillonnage aléatoire.

En gardant à l'esprit les limites ci-dessus et en faisant preuve de prudence dans nos interprétations, nous pouvons toujours utiliser les données pour avoir un aperçu de l'état de santé mentale dans le milieu de travail technologique.

Présentation :

Notre objectif est de construire un modèle qui peut prédire si un employé recherche ou non un traitement pour un problème de santé mentale. Plus précisément, nous aimerions nous concentrer sur les facteurs qui influencent le fait qu'une personne recherche ou non une aide professionnelle

Notre analyse est réalisée à l'aide du langage de programmation Python. Nous avons d'abord divisé l'ensemble de données en ensembles d'entraînement et de test. Nous nous sommes ensuite concentrés sur les techniques de classification, qui incluent la comparaison de l'erreur après ajustement des modèles sur l'ensemble de test. Les modèles que nous avons choisi de réaliser sont un arbre de décision, des arbres agrégés bootstrap et une régression logistique. Grâce à la courbe ROC, qui montre la relation entre le taux de faux positifs et le taux de vrais positifs d'un modèle, et l'aire sous ces courbes, nous avons conclu que le modèle de régression logistique était le meilleur modèle pour classer si une personne a choisi ou non chercher un traitement pour un problème de santé mentale

Description des donnees :

L'ensemble de données contient 1259 lignes avec les réponses des personnes qui ont participé à l'enquête. Cette enquête a examiné les personnes en fonction de 27 facteurs (colonnes) pouvant être associés à l'état de santé mentale.

Cet ensemble de données contient les variables suivantes:

Variable	Signification
Horodatage	
Âge	
Le genre	
Pays	
état	Si vous habitez aux États-Unis, dans quel état ou territoire habitez-vous?
indépendant	êtes-vous indépendant?

family_history	Avez-vous des antécédents familiaux de maladie mentale?
traitement	Avez-vous cherché un traitement pour un problème de santé mentale?
work_interfere	Si vous avez un problème de santé mentale, pensez-vous que cela interfère avec votre travail?
no_employees	Combien d'employés votre entreprise ou organisation compte-t-elle?
remote_work	Travaillez -vous à distance (en dehors d'un bureau) au moins 50% du temps?
avantages sociaux	votre employeur offre-t-il des avantages pour la santé mentale?
tech_company	Votre employeur est-il principalement une entreprise / organisation technologique?
care_options	Connaissez-vous les options de soins de santé mentale offertes par votre employeur?
wellness_program	Votre employeur a-t-il déjà discuté de la santé mentale dans le cadre d'un programme de mieux-être des employés?
seek_help	Votre employeur fournit-il des ressources pour en savoir plus sur les problèmes de santé mentale et comment demander de l'aide
anonymat	votre anonymat est-il protégé si vous choisissez de profiter des ressources de traitement de la santé mentale ou de la toxicomanie?
congé	Est-il facile pour vous de prendre un congé de maladie pour un problème de santé mentale?
Conséquences sur la santé mentale	Pensez-vous que discuter d'un problème de santé mentale avec votre employeur aurait des conséquences négatives?
Phys santé conséquence	Pensez - vous que discuter d' un problème de santé physique avec votre employeur aurait des conséquences négatives?
collègues	Seriez-vous prêt à discuter d'un problème de santé mentale avec vos collègues?
superviseur	Seriez-vous prêt à discuter d'un problème de santé mentale avec votre ou vos supérieurs directs?
entretien de santé mentale	Souhaitez-vous évoquer un problème de santé mentale avec un employeur potentiel lors d'une entrevue?

Phys santé entrevue	apporteriez - vous un problème de santé physique avec un employeur potentiel dans une interview?
mental vs physique	Pensez-vous que votre employeur prend la santé mentale aussi au sérieux que la santé physique?
obs_consequence	Avez-vous entendu parler ou observé des conséquences négatives pour des collègues souffrant de problèmes de santé mentale sur votre lieu de travail?
commentaires	toute note ou commentaire supplémentaire

Problématique :

Quels sont les facteurs et les impactent qui poussent un employé à chercher de l'aide professionnel en ce qui Concerne sa santé mentale ?

Bibliothèque utilisées:

Numpy :Elle introduit une gestion facilitée des tableaux de nombres.

Pandas :Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données.

Matplotlib : Matplotlib est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques.

```
import os
import numpy as np
import pandas as pd


import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
```

Visualisation des données de notre dataset :

L'affichage des informations de

Notre dataset nous donne une

Vision plus claire sur les

Variables , leurs types , la

Dimension de notre dataset

Et surtout l'index .

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Timestamp                  1259 non-null   object
 1   Age                        1259 non-null   int64
 2   Gender                     1259 non-null   object
 3   Country                    1259 non-null   object
 4   state                      744 non-null    object
 5   self_employed              1241 non-null   object
 6   family_history             1259 non-null   object
 7   treatment                  1259 non-null   object
 8   work_interfere             995 non-null    object
 9   no_employees               1259 non-null   object
 10  remote_work                1259 non-null   object
 11  tech_company               1259 non-null   object
 12  benefits                   1259 non-null   object
 13  care_options               1259 non-null   object
 14  wellness_program           1259 non-null   object
 15  seek_help                  1259 non-null   object
 16  anonymity                  1259 non-null   object
 17  leave                      1259 non-null   object
 18  mental_health_consequence  1259 non-null   object
 19  phys_health_consequence    1259 non-null   object
 20  coworkers                  1259 non-null   object
 21  supervisor                 1259 non-null   object
 22  mental_health_interview    1259 non-null   object
 23  phys_health_interview      1259 non-null   object
 24  mental_vs_physical         1259 non-null   object
 25  obs_consequence            1259 non-null   object
 26  comments                   164 non-null    object
dtypes: int64(1), object(26)
memory usage: 265.7+ KB
```

On remarque que notre dataset contient 26 colones ,cette affichage nous permet d'identifier les colones qu'on doit nettoyer. à identifier que les colones : Genre / self-employed/Age .

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	mental_health_consequence	phys_health_consequence	coworkers	supervisor	mental_health_in
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes	Often	6-25	...	Somewhat easy	No	No	Some of them	Yes	
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No	Rarely	More than 1000	...	Don't know	Maybe	No	No	No	
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	Rarely	6-25	...	Somewhat difficult	No	No	Yes	Yes	
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	26-100	...	Somewhat difficult	Yes	Yes	Some of them	No	
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No	Never	100-500	...	Don't know	No	No	Some of them	Yes	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
254	2015-09-12 11:17:21	26	male	United Kingdom	NaN	No	No	Yes	NaN	26-100	...	Somewhat easy	No	No	Some of them	Some of them	
255	2015-09-26 01:07:35	32	Male	United States	IL	No	Yes	Yes	Often	26-100	...	Somewhat difficult	No	No	Some of them	Yes	
256	2015-11-07 12:36:58	34	male	United States	CA	No	Yes	Yes	Sometimes	More than 1000	...	Somewhat difficult	Yes	Yes	No	No	
257	2015-11-30 21:25:06	46	f	United States	NC	No	No	No	NaN	100-500	...	Don't know	Yes	No	No	No	
258	2016-02-01 23:04:31	25	Male	United States	IL	No	Yes	Yes	Sometimes	26-100	...	Don't know	Maybe	No	Some of them	No	

259 rows × 27 columns

Nettoyage,Encodage et préparation des données pour le pré-processing :

On sait très bien que Une condition de «valeur manquante» se produit chaque fois qu'une entrée de données est laissée vide. Alors en commence par le nettoyage des données.

- La première partie du nettoyage des données consiste à nettoyer la variable Genre. Sur la base des réponses, il y a 49 entrées uniques de genre. On divise ces valeurs en trois catégories principales :femelle ,male , Trans.

```
gender = df['Gender'].str.lower()
male_str = ["male", "m", "male-ish", "maile", "mal", "male (cis)", "make", "male ", "man","msle", "mail", "malr","cis man", "Cis Male", "cis male"]
trans_str = ["trans-female", "something kinda male?", "queer/she/they", "non-binary","nah", "all", "enby", "fluid", "genderqueer", "androgyne", "agender", "male leaning androgynous", "guy (-ish)
female_str = ["cis female", "f", "female", "woman",  "femake", "female ","cis-female/femme", "female (cis)", "femail"]

for (row, col) in df.iterrows():
    if str.lower(col.Gender) in male_str:
        df['Gender'].replace(to_replace=col.Gender, value='male', inplace=True)

    if str.lower(col.Gender) in female_str:
        df['Gender'].replace(to_replace=col.Gender, value='female', inplace=True)

    if str.lower(col.Gender) in trans_str:
        df['Gender'].replace(to_replace=col.Gender, value='trans', inplace=True)

l = ['A little about you', 'p']
df = df[~df['Gender'].isin(l)]

print(df['Gender'].unique())
```

```
['female' 'male' 'trans']
```

- La deuxième partie consiste à nettoyer les valeurs de la colonnes self_employed,On remplace le valeur «NaN » par No afin d'avoir seulement deux valeur « Yes » et « «No »
- En profite de l'occasion pour vérifier aussi les valeurs de la colonnes work _interface qui contient aussi des valeurs NaN , et en remplace la valeurs NAN par NO .

```
df["self_employed"].unique()
```

```
array(['NaN', 'Yes', 'No'], dtype=object)
```

```
df['self_employed'] = df['self_employed'].replace([defaultString], 'No')
print(df['self_employed'].unique())
```

```
['No' 'Yes']
```

```
df["work_interfere"].unique()
```

```
array(['Often', 'Rarely', 'Never', 'Sometimes', 'NaN'], dtype=object)
```

```
df['work_interfere'] = df['work_interfere'].replace([defaultString], 'No')
print(df['work_interfere'].unique())
```

```
['Often' 'Rarely' 'Never' 'Sometimes' 'No']
```

- Pour la variable Age on calcule le moyen :

```
moy_age=df['Age'].mean()
moy_age
```

```
79554525.6300716
```

```
df["Age"].replace(np.nan, moy_age, inplace=True)
```

Les modèles statistiques sur lesquels on va travailler ne peuvent pas accepter d'objets ou de chaînes en entrée et, pour l'apprentissage du modèle, alors on doit prendre que les nombres comme entrées.

- Tout d'abord, Nous encodons les variables de type de chaîne de caractères en nombres.Scikit-learn fournit la bibliothèque LabelEncoder pour l'encodage des étiquettes avec une valeur comprise entre 0 et 1.

```
#Encoding data
labelDict = {}
for feature in df:
    le = preprocessing.LabelEncoder()
    le.fit(df[feature])
    le_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
    df[feature] = le.transform(df[feature])



for key, value in labelDict.items():
    print(key, value)


df.head()
```

	Age	Gender	self_employed	family_history	treatment	work_interfere	no_employees	remote_work	tech_company	benefits	...	anonymity	leave	mental_health_consequence	phys_health_consequence	coworkers	supervisor
0	23	0	0	0	1	2	4	0	1	2	...	2	2	1	1	1	2
1	30	1	0	0	0	3	5	0	0	0	...	0	0	0	1	0	0
2	18	1	0	0	0	3	4	0	1	1	...	0	1	1	1	2	2
3	17	1	0	1	1	2	2	0	1	1	...	1	1	2	2	1	0
4	17	1	0	0	0	0	1	1	1	2	...	0	0	1	1	1	2

5 rows × 23 columns

On a opté à cet encodage de données afin d'intégrer les données dans un standard d'expression commun qui garantit que les données sont cohérentes et faciles et qui nous va permettre de faire des comparaisons significatives et compréhensible.

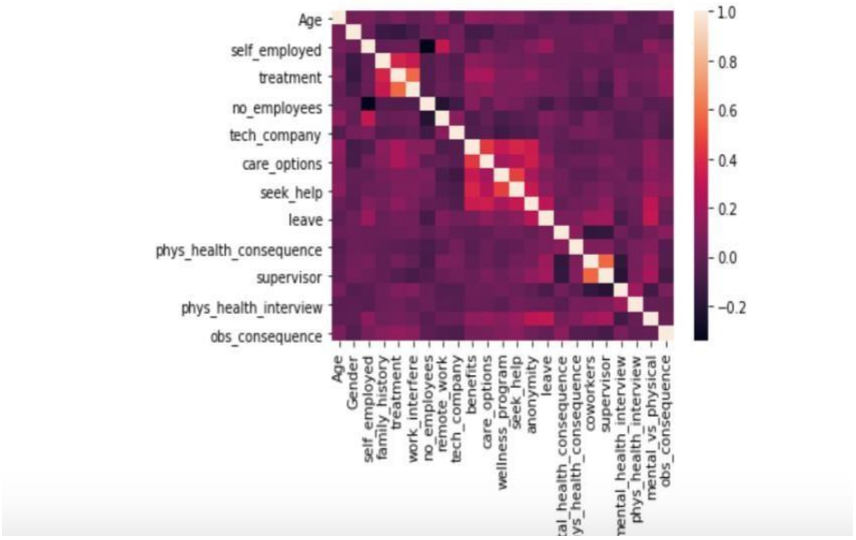
- Extraction des variables importantes :

La question qui se pose à ce niveau, c'est : comment savoir quelle est la valeur qui a plus d'impact sur la santé mental du patient ? pour répondre à cela on visualise tous d'abord la HEATMAP pour voir la corrélation entre les variables de notre base de donnée.

- **Affichage de laHeatMap**

La corrélation est une métrique statistique pour mesurer dans quelle mesure les différentes variables sont interdépendantes.
Pour obtenir la matrice suivante on a utilisé Seaborn pour la visualisation avec.HeatMap et en lui précisant comme argument, le DataFrame(df) avec .corr() qui permet avec Pandas de créer cette matrice de corrélation.

```
Entrée [272]: import seaborn as sns

Entrée [273]: sns.heatmap(df.corr())

Out[273]: <matplotlib.axes._subplots.AxesSubplot at 0x2116bb47848>
```



On peut remarquer une

Corrélation positive

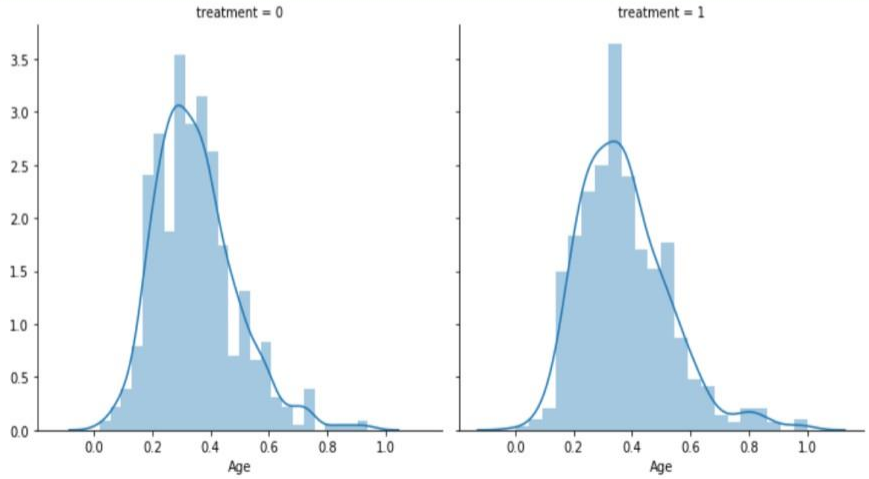
Entre 'age' et 'treatement'

- **FaceGrid**

Certes pour mieux comprendre les données , on sépare la population selon le traitement en utilisant la variable. Age comme référence en utilisant FACEGRID, Cette classe mappe la variable traitement avec l'age.

```
Entrée [275]: # Separate by treatment or not

              g = sns.FacetGrid(df, col='treatment', size=5)
              g = g.map(sns.distplot, "Age")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\axisgrid.py:243: U
serWarning: The `size` parameter has been renamed to `height`; please
update your code.
  warnings.warn(msg, UserWarning)
```

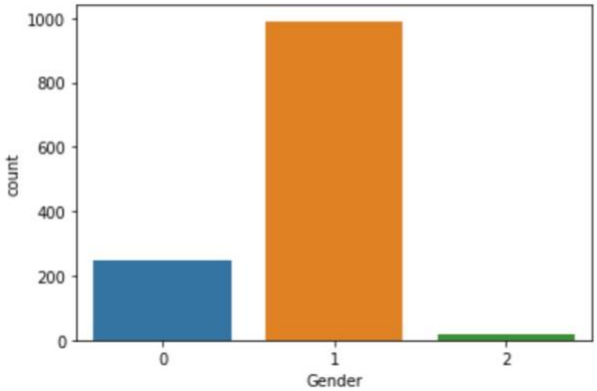



```
Entrée [276]: sns.countplot(df['Gender'])
```

- Countplot

Et puis on filtre selon l'age , grâce à l'encodage on a pu visualiser l'âge le plus dominant dans la population étudiée.

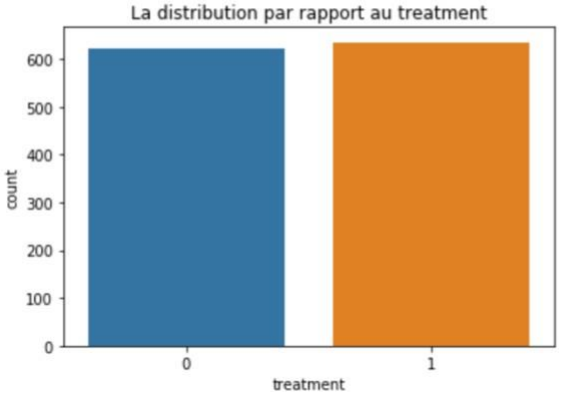
```
Out[276]: <matplotlib.axes._subplots.AxesSubplot at 0x2116c03b548>
```



- Countplot

On peut aussi remarquer que la population suivant le traitement 1 est presque équivalente à la population suivant le traitement 0.

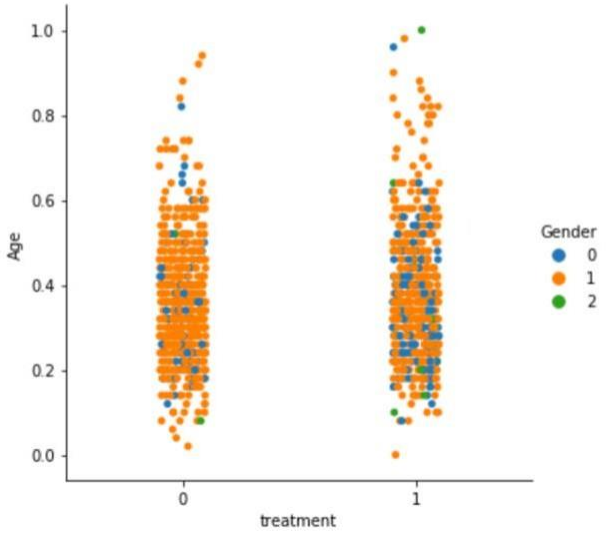
```
Entrée [277]: sns.countplot(x="treatment", data=df)
              plt.title('La distribution par rapport au treatment')
Out[277]: Text(0.5, 1.0, 'La distribution par rapport au treatment')
```



- CatPlot :

Cette fonction permet d'accéder à plusieurs fonctions au niveau des axes qui montrent la relation entre la variable Traitement et Age , tout en distinguant le résultat en basant sur la variable genre.

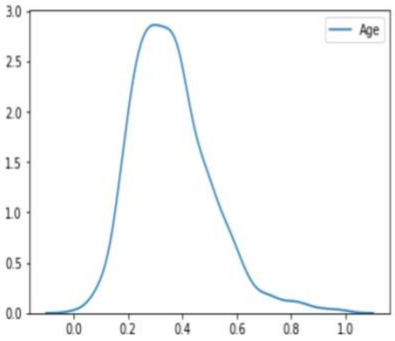
```
Entrée [278]: sns.catplot(x="treatment", y="Age", hue="Gender", data=df)
Out[278]: <seaborn.axisgrid.FacetGrid at 0x2116c0c46c8>
```



- KDEPLOT

Maintenant il est temps d'afficher le graphique d'estimation de densité de noyau (KDE) afin de visualiser la distribution des observations de la variable 'Age '. KDE nous a afficher les données de la variable Age à l'aide d'une courbe de densité de probabilité continue dans une dimension .

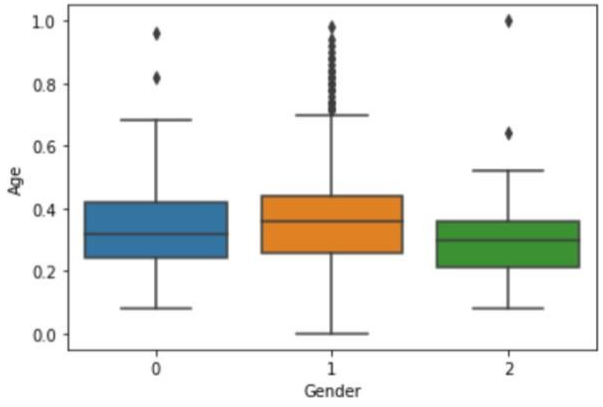
```
Entrée [279]: sns.kdeplot(df['Age'])
Out[279]: <matplotlib.axes._subplots.AxesSubplot at 0x2116c154a08>
```



- Boxplot :

Ce graphique tout simple permet de résumer une variable de manière simple et visuel, d'identifier les valeurs extrêmes et de comprendre la répartition des observations.

```
Entrée [280]: sns.boxplot(x="Gender", y="Age", data=df)
Out[280]: <matplotlib.axes._subplots.AxesSubplot at 0x2116c0e5c08>
```

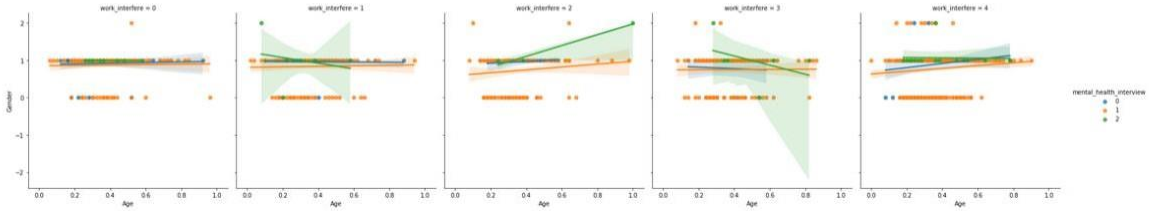


`

- Lmplot :

Cette fonction combine regplot () et FacetGrid. Il est conçu comme une interface pratique pour ajuster les modèles de régression dans les sous-ensembles conditionnels d'un ensemble de données.

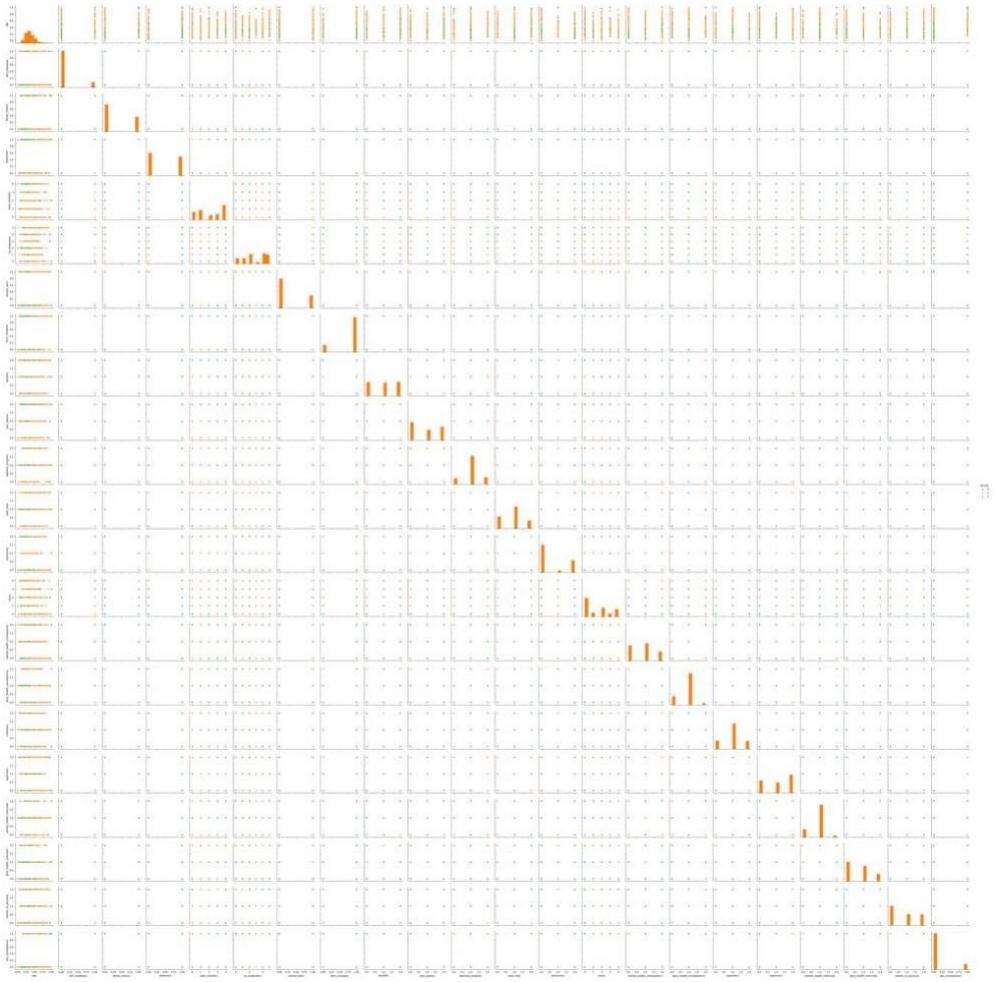
```
Entrée [281]: ax = sns.lmplot(x='Age',y='Gender',col='work_interfere',hue='mental_
```



- Pairplot :

cette fonction créera une grille d'axes de telle sorte que chaque variable numérique dans les données sera partagée sur les axes y sur une seule ligne et les axes x sur une seule colonne.

```
Entrée [282]: sns.pairplot(df, hue="Gender", diag_kind="hist")
Out[282]: <seaborn.axisgrid.PairGrid at 0x2116d714c48>
```



- **Entrainement du modèle :**

On doit tout d'abord faire le compte des valeurs de la variables Treatement , on a remarqué que le nombre des employés avec traitement est de 635 et sans traitement de 622 .

Alors on peut affecter cette variable à une nouvelle variable Y .et on définit X comment étant une nouvelle dataset qui contient les variables suivantes : (age , gender , family_history , benefits , care options , anonimity , leave , work_intefere)

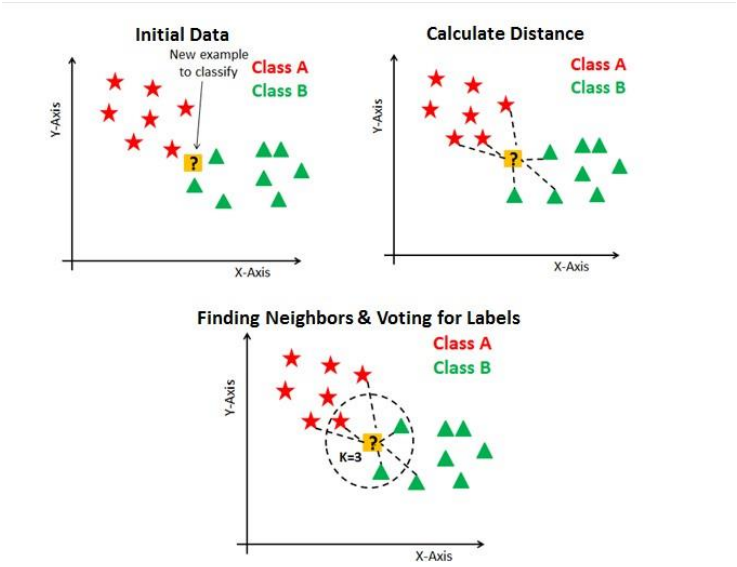
⇨ On définit ensuite les données d'entrainement et les données de test .

```
Entrée [283]: df['treatment'].value_counts()

 Out[283]: 1    635
           0    622
           Name: treatment, dtype: int64

Entrée [284]: X = df[['Age', 'Gender', 'family_history', 'benefits', 'care_options', 'anonymity', 'leave', 'work_interfere']] .val
              X[0:5]

 Out[284]: array([[0.46, 0.  , 0.  , 2.  , 1.  , 2.  , 2.  , 2.  ],
                  [0.6 , 1.  , 0.  , 0.  , 0.  , 0.  , 0.  , 3.  ],
                  [0.36, 1.  , 0.  , 1.  , 0.  , 0.  , 1.  , 3.  ],
                  [0.34, 1.  , 1.  , 1.  , 2.  , 1.  , 1.  , 2.  ],
                  [0.34, 1.  , 0.  , 2.  , 0.  , 0.  , 0.  , 0.  ]])

Entrée [285]: Y = df[['treatment']] .values
              Y[0:5]

 Out[285]: array([[1],
                  [0],
                  [0],
                  [1],
                  [0]])

Entrée [286]: from sklearn.model_selection import train_test_split
              X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size=0.2, random_state=4)
              print ('Train set:', X_train.shape,  y_train.shape)
              print ('Test set:', X_test.shape,  y_test.shape)

              Train set: (1005, 8) (1005, 1)
              Test set: (252, 8) (252, 1)
```

KNN

Comment fonctionne l'algorithme KNN?

Dans KNN, K est le nombre de voisins les plus proches. Le nombre de voisins est le principal facteur décisif. K est généralement un nombre impair si le nombre de classes est de 2. Lorsque K = 1, alors l'algorithme est appelé algorithme du plus proche voisin. C'est le cas le plus simple. Supposons que P1 soit le point pour lequel l'étiquette doit prédire. Tout d'abord, vous trouvez le point le plus proche de P1, puis l'étiquette du point le plus proche affecté à P1.



Supposons que P1 soit le point pour lequel l'étiquette doit prédire. Tout d'abord, vous trouvez le k point le plus proche de P1, puis classez les points par vote majoritaire de ses k voisins. Chaque objet vote pour sa classe et la classe avec le plus de votes est prise comme prédiction. Pour trouver les points similaires les plus proches, vous trouvez la distance entre les points à l'aide de mesures de distance telles que la distance euclidienne, la distance de Hamming, la distance de Manhattan et la distance de Minkowski.

Générer un modèle

Construisons un modèle de classificateur KNN.

Tout d'abord, importez le module KNeighborsClassifier et créez un objet de classification KNN en passant le nombre d'arguments de voisins dans la fonction KNeighborsClassifier ().

Ensuite, ajustez votre modèle sur l'ensemble de trains à l'aide de fit () et effectuez une prédiction sur l'ensemble de test à l'aide de predire ().

Génération du modèle pour K = 15

Construisons un modèle de classificateur KNN pour k = 15.

```
from sklearn.neighbors import KNeighborsClassifier
```

```
k = 15
#Train Model and Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh
```

```
<ipython-input-41-a3b26ba38c89>:3: DataConversionWarning: A column-
xample using ravel().
  neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train
KNeighborsClassifier(n_neighbors=15)
```

Évaluation du modèle pour k = 5

Estimons avec quelle précision le classificateur ou le modèle peut prédire le type de cultivars.

La précision peut être calculée en comparant les valeurs réelles de l'ensemble de test et les valeurs prévues.

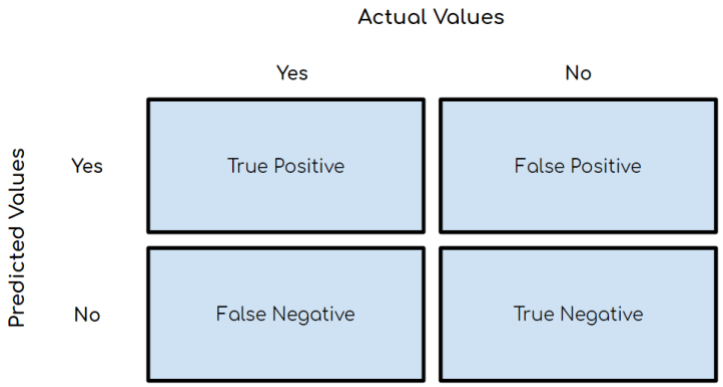
```
from sklearn import metrics
print("Train set Accuracy: ", metrics.accuracy_score(y_train, neigh.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat))
```

```
Train set Accuracy:  0.8308457711442786
Test set Accuracy:  0.8293650793650794
```

Passant par la suite à la matrice de confusion :

Une matrice de confusion, également appelée matrice d'erreur, est un tableau récapitulatif utilisé pour évaluer les performances d'un modèle de classification. Le nombre de prédictions correctes et incorrectes est résumé avec des valeurs de comptage et ventilé par classe.

On résume dans le schéma suivant la matrice de confusion en bref.

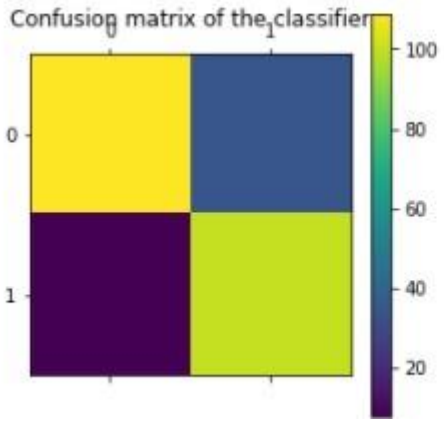


Pour la réalisation de la matrice de confusion on a importé tous d'abord confusion_matrix et pylab .

On a ensuite donné à la matrice les deux variables (y_test et yhat) .

```
[297]: from sklearn.metrics import confusion_matrix
       import pylab as pl
```

```
[298]: cm = confusion_matrix(y_test, yhat)
       pl.matshow(cm)
       pl.title('Confusion matrix of the classifier')
       pl.colorbar()
       pl.show()
```

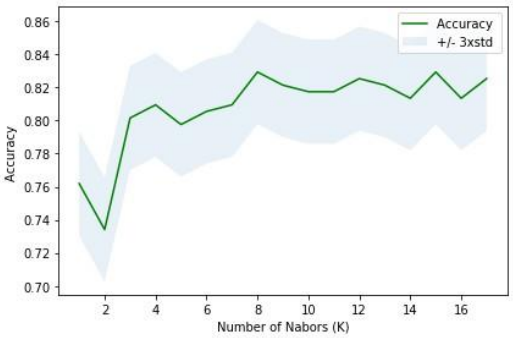


La création de modèle KNN avec un nombre de voisins=18

Pour bien comprendre le modèle KNN et l'effet du nombre voisin , on a opté pour lacréation d'un autre modèle KNN avec un nombre voisin différent du premier : ks=18 .

```
Ks = 18
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))
ConfustionMx = [];
for n in range(1,Ks):

    #Train Model and Predict
    neigh = KNeighborsClassifier(n_neighbors = n).fit(X_train,y_train)
    yhat=neigh.predict(X_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)


    std_acc[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])

mean_acc
```

Entrée [43]:
```
plt.plot(range(1,Ks),mean_acc,'g')
plt.fill_between(range(1,Ks),mean_acc - 1 * std_acc,mean_acc + 1 * std_acc, alpha=0.10)
plt.legend(('Accuracy ', '+/- 3xstd'))
plt.ylabel('Accuracy ')
plt.xlabel('Number of Nabors (K)')
plt.tight_layout()
plt.show()
```



Calcule de la précision à base de K :

D'après les deux modèles KNN étudié, on a décidé de faire la liaison entre K et la précision . alors on a déterminé la précision pour des valeurs de K différentes.

Entrée [50]:
```
print( "The best accuracy was with", mean_acc.max(), "with k=", mean_acc.argmax()+1)
```

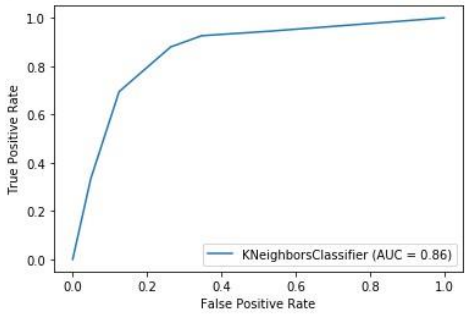
```
The best accuracy was with 0.8293650793650794 with k= 8
```

Entrée [44]:
```
from sklearn.metrics import classification_report
clf = KNeighborsClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print (classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.74      0.81       144
           1       0.71      0.88      0.79       108

    accuracy                           0.80       252
   macro avg       0.80      0.81      0.80       252
weighted avg       0.82      0.80      0.80       252
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was pass
ed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  This is separate from the ipykernel package so we can avoid doing imports until
```

Entrée [45]:
```
metrics.plot_roc_curve(clf, X_test, y_test)
plt.show()
```



L’arbre de décision :

Cet outil d’aide à la décision ou l’exploration de données ce qui permet de représenter un ensemble de choix sous la forme de graphique d’un arbre.

La création d’arbre de décision consiste à importer les bibliothèques nécessaires , et puis faire le premier affichage de l’arbre de décision à base de X et Y .

DecisionTree

Entrée [192]:
```
from sklearn.tree import DecisionTreeClassifier
from scipy import stats
from scipy.stats import randint
```

Entrée [193]:
```
from sklearn.tree import DecisionTreeClassifier
arbreFirst = DecisionTreeClassifier(min_samples_split=10,min_samples_leaf=5)
```

Entrée [194]:
```
arbreFirst.fit(X,Y)
```

Out[194]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=5, min_samples_split=10,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

Comme on sait , il est crucial d’importer plot_tree pour pouvoir faire l’affichage de l’arbre de décision , la définition de X nous permet de contrôler l’affichage .

Entrée [195]:
```
from sklearn.tree import plot_tree
X=df[['Age', 'Gender', 'family_history', 'benefits', 'care_options', 'anonymity', 'leave', 'work_interfere']]
plot_tree(arbreFirst,feature_names = list(X),filled=True)
```

```
Text(2.2814310051107327, 108.72, 'gini = 0.245\nsamples = 7\nvalue = [6, 1]'),
Text(6.844293015332198, 108.72, 'gini = 0.48\nsamples = 5\nvalue = [3, 2]'),
Text(17.110732538330495, 123.216, 'care_options <= 0.5\ngini = 0.116\nsamples = 129\nvalue = [121, 8]'),
Text(11.407155025553664, 108.72, 'Age <= 0.21\ngini = 0.034\nsamples = 57\nvalue = [56, 1]'),
Text(9.12572402044293, 94.22399999999999, 'gini = 0.245\nsamples = 7\nvalue = [6, 1]'),
Text(13.688586030664396, 94.22399999999999, 'gini = 0.0\nsamples = 50\nvalue = [50, 0]'),
Text(22.814310051107327, 108.72, 'benefits <= 0.5\ngini = 0.176\nsamples = 72\nvalue = [65, 7]'),
Text(18.25144804088586, 94.22399999999999, 'Age <= 0.17\ngini = 0.062\nsamples = 31\nvalue = [30, 1]'),
Text(15.970017035775129, 79.72799999999998, 'gini = 0.32\nsamples = 5\nvalue = [4, 1]'),
Text(20.532879045996594, 79.72799999999998, 'gini = 0.0\nsamples = 26\nvalue = [26, 0]'),
Text(27.377172061328793, 94.22399999999999, 'Age <= 0.25\ngini = 0.25\nsamples = 41\nvalue = [35, 6]'),
Text(25.09574105621806, 79.72799999999998, 'gini = 0.0\nsamples = 8\nvalue = [8, 0]'),
Text(29.658603066439525, 79.72799999999998, 'Age <= 0.29\ngini = 0.298\nsamples = 33\nvalue = [27, 6]'),
Text(27.377172061328793, 65.232, 'gini = 0.48\nsamples = 5\nvalue = [3, 2]'),
Text(31.940034071550258, 65.232, 'leave <= 3.5\ngini = 0.245\nsamples = 28\nvalue = [24, 4]'),
Text(29.658603066439525, 50.73599999999999, 'leave <= 0.5\ngini = 0.172\nsamples = 21\nvalue = [19, 2]'),
Text(27.377172061328793, 36.23999999999998, 'gini = 0.0\nsamples = 7\nvalue = [7, 0]'),
Text(31.940034071550258, 36.23999999999998, 'Age <= 0.35\ngini = 0.245\nsamples = 14\nvalue = [12, 2]'),
Text(29.658603066439525, 21.744, 'gini = 0.32\nsamples = 5\nvalue = [4, 1]'),
Text(34.22146507666099, 21.744, 'gini = 0.198\nsamples = 9\nvalue = [8, 1]'),
```

Entrée [ ]:

Entrée [236]:
```
from sklearn.tree import DecisionTreeClassifier
arbreFirst = DecisionTreeClassifier(min_samples_split=190,min_samples_leaf=50)
```

Entrée [237]:
```
arbreFirst.fit(X,Y)
```

Out[237]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=50, min_samples_split=190,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

LogisticRegression

La régression logistique est l'analyse de régression appropriée à effectuer lorsque la variable dépendante est dichotomique (binaire). Comme toutes les analyses de régression, la régression logistique est une analyse prédictive. La régression logistique est utilisée pour décrire les données et pour expliquer la relation entre une variable binaire dépendante et une ou plusieurs variables indépendantes nominales, ordinales, d'intervalle ou de rapport.

Parfois, les régressions logistiques sont difficiles à interpréter; l'outil IntellectusStatistics vous permet facilement de mener l'analyse, puis en anglais clair interprète la sortie.

LogisticRegression

Entrée [300]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=0)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:760: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[300]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Entrée [301]:
```
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
```

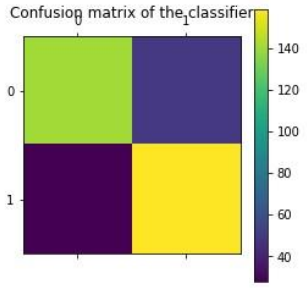
```
Accuracy of logistic regression classifier on test set: 0.79
```

Entrée [302]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[141  50]
 [ 28 159]]
```

Entrée [304]:
```
from sklearn.metrics import confusion_matrix
import pylab as pl
```

Entrée [305]:
```
cm = confusion_matrix(y_test, y_pred)
pl.matshow(cm)
pl.title('Confusion matrix of the classifier')
pl.colorbar()
pl.show()
```



Dans cette partie , on a commencé par l'import des bibliothèques nécessaires pour la réalisation pour la régression linéaire , par la suite on a créer une variable LogReg qu'on a entrainer grâce au données d'entrainement . passant par la suite vers la prédiction , qui consiste à prédire les résultat de y_pred à base de y_test .

Par la suite on a fait la création de la matrice de confusion des deux façon , la façon traditionnel et la façon visuel . on remarque que le résultat est le même dans les deux méthodes .

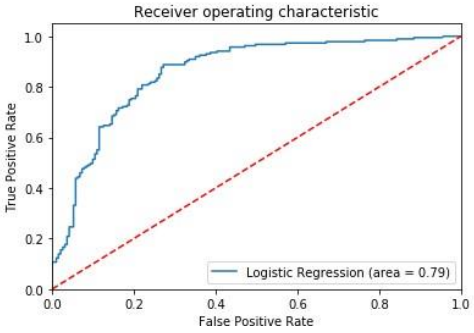
A ce niveau , on a importer Classification_report , pour afficher par la suite la précison , le recall , le f1-score et le support de notre modèle.

Pour visualiser enfin les résultat de notre modèle

Entrée [306]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.74      0.78       191
           1       0.76      0.85      0.80       187

    accuracy                           0.79       378
   macro avg       0.80      0.79      0.79       378
weighted avg       0.80      0.79      0.79       378
```

Entrée [307]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```



Partie Recherche :

Naive-Bayes

Naive Bayes est l'algorithme de classification le plus simple et le plus rapide, qui convient à un grand nombre de données. Le classificateur Naive Bayes est utilisé avec succès dans diverses applications telles que le filtrage anti-spam, la classification de texte, l'analyse des sentiments et les systèmes de recommandation. Il utilise le théorème de Bayes de probabilité pour la prédiction de classe inconnue.

Qu'est-ce que Naive Bayes Classifier?

Le classificateur Naive Bayes suppose que l'effet d'une caractéristique particulière dans une classe est indépendant des autres caractéristiques. Par exemple, un demandeur de prêt est souhaitable ou non selon ses revenus, ses antécédents de prêt et de transaction, son âge et son emplacement. Même si ces caractéristiques sont interdépendantes, ces caractéristiques sont toujours considérées
indépendamment. Cette hypothèse simplifie le calcul, c'est pourquoi elle est considérée comme naïve. Cette hypothèse est appelée indépendance conditionnelle de classe.

Voici le théorème de Bayes: $P(A|B)=P(B|A)*P(A)/P(B)$.

Avantages de Naive-Bayes :

- Ce n'est pas seulement une approche simple, mais aussi une méthode rapide et précise de prévision.
- Naive Bayes a un coût de calcul très faible.
- Il peut fonctionner efficacement sur un grand ensemble de données.
- Il fonctionne bien en cas de variable de réponse discrète par rapport à la variable continue.
- Il peut être utilisé avec plusieurs problèmes de prédiction de classe.
- Il fonctionne également bien dans le cas de problèmes d'analyse de texte.
- Lorsque l'hypothèse d'indépendance se vérifie, un classificateur Naive Bayes fonctionne mieux par rapport à d'autres modèles comme la régression logistique.

Désavantages De Naive Bayes

- L'hypothèse de caractéristiques indépendantes. En pratique, il est presque impossible que le modèle obtienne un ensemble de prédicteurs entièrement indépendants.
- S'il n'y a pas de tuple d'apprentissage d'une classe particulière, cela entraîne une probabilité postérieure nulle. Dans ce cas, le modèle est incapable de faire des prédictions. Ce problème est connu sous le nom de problème de probabilité / fréquence zéro.

Générer un modèleNaive-Bayes

- Générez un modèle à l'aide du classificateur bayésien naïf dans les étapes suivantes:
- Créer un classificateur bayes naïf
- Ajuster l'ensemble de données sur le classificateur
- Effectuer une prédiction

```
from sklearn.naive_bayes import GaussianNB
modele = GaussianNB()
modele.fit(X_train, y_train)
```

```
y_predi = modele.predict(X_test)
```

Modèle d'évaluation

Après la génération du modèle, vérifiez la précision à l'aide des valeurs réelles et prévues.

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8148148148148148
```

81,48% des gens dans le milieu de travail technologique prend un traitement

⇨ Matrice de confusion

La matrice de confusion est une autre métrique souvent utilisée pour mesurer les performances d'un algorithme de classification. Fidèle à son nom, la terminologie liée à la matrice de confusion peut être assez déroutante, mais la matrice elle-même est simple à comprendre

```
from sklearn.metrics import confusion_matrix
confusion=confusion_matrix(y_test,y_predi)
print(confusion)
```

```
[[152  39]
 [ 31 156]]
```

⇨ Précision, rappel et score f1

Outre la précision, il existe plusieurs autres mesures de performance qui peuvent être calculées à partir de la matrice de confusion. Certains des principaux sont obtenus en utilisant la fonction classification_report:

```
from sklearn.metrics import classification_report
clf_NB = GaussianNB()
clf_NB.fit(X_train, y_train)
y_pred = clf_NB.predict(X_test)
print (classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.80      0.81       191
           1       0.80      0.83      0.82       187

    accuracy                           0.81       378
   macro avg       0.82      0.82      0.81       378
weighted avg       0.82      0.81      0.81       378
```

Précision répond à la question:

"Quand il prédit le résultat positif, à quelle fréquence est-il correct?"

Ceci est obtenu en utilisant les formules suivantes:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

La précision est généralement utilisée lorsque l'objectif est de limiter le nombre de faux positifs (FP).

- ⇨ Rappel : il répond à la question:

«Lorsqu'il s'agit en fait d'un résultat positif, à quelle fréquence prédire-t-il correctement?»

Ceci est obtenu en utilisant les formules suivantes:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Le rappel est généralement utilisé lorsque le but est de limiter le nombre de faux négatifs (FN). Le rappel est également appelé «sensibilité» et «taux de vrais positifs» (TPR).

- ⇨ score f1 : ce n'est que la moyenne harmonique de la précision et du rappel:

$$f_1\text{-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Il est utile lorsque vous devez tenir compte à la fois de la précision et du rappel. Si vous essayez d'optimiser uniquement le rappel, votre algorithme prédira que la plupart des exemples appartiennent à la classe positive, mais cela entraînera de nombreux faux positifs et, par conséquent, une faible précision. En revanche, si vous essayez d'optimiser la précision, votre modèle prédira très peu d'exemples comme des résultats positifs (ceux dont la probabilité est la plus élevée), mais le rappel sera très faible.

- ⇨ **Choix des hyperparamètres**

Une première approche consiste à utiliser la recherche par quadrillage(GridSearch). L'idée est plutôt simple en réalité : vous positionnez une liste de possibilités pour chacun des hyper-paramètres et pour chacune des combinaisons vous aller entrainer votre modèle puis calculer son score. A la fin bien sur vous ne conserverez que le meilleur paramétrage.
Stratifieldkfold : Validation croisée des plis K stratifiés. Nous fournit des indices de train / test pour diviser les données dans des ensembles de train / test. Cet objet de validation croisée est une variante

de KFold qui renvoie des plis stratifiés. Les plis sont réalisés en conservant le pourcentage d'échantillons pour chaque classe.
n_splits : Nombre de plis. Doit être au moins 2

GridSearchCV : Recherche exhaustive sur les valeurs de paramètres spécifiées pour un estimateur.

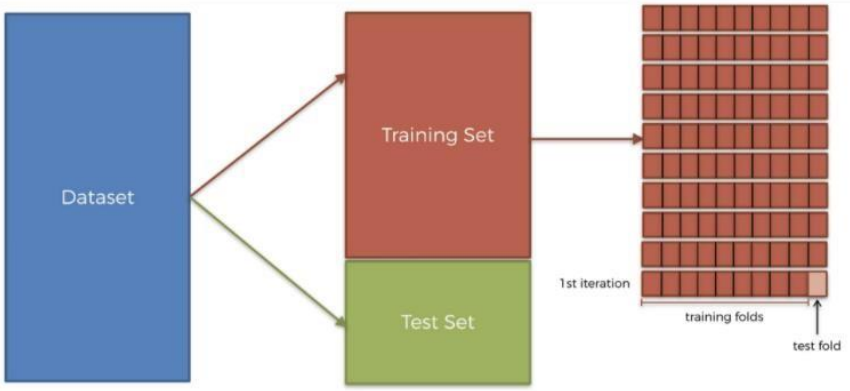
Param_grid= params : dict ou liste de dictionnaires avec des noms de paramètres (str) comme clés et des listes de réglages de paramètres à essayer comme valeurs, ou une liste de ces dictionnaires, auquel cas les grilles couvertes par chaque dictionnaire de la liste sont explorées. Cela permet d'effectuer une recherche sur n'importe quelle séquence de réglages de paramètres.

```
from sklearn.model_selection import StratifiedKFold

params = {}

#gridsearch searches for the best hyperparameters and keeps the classifier with the highest recall score
skf = StratifiedKFold(n_splits=10)

nb2 = GridSearchCV(GaussianNB(), cv=skf, param_grid=params)
%time nb2.fit(X_train, y_train)

# predict values on the test set
y_pred_nb2 = nb2.predict(X_test)

print(y_pred_nb2)

# predicted probabilities on the test set
y_scores_nb2 = nb2.predict_proba(X_test)[:, 1]

print(y_scores_nb2)
```

```
[1 0 0 0 0 1 0 1 1 1 0 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 1 1
 1 1 0 0 0 1 0 1 0 1 1 1 0 1 0 1 1 0 0 0 0 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0
 0 1 1 1 0 0 0 1 0 0 0 1 1 1 0 0 1 0 1 1 0 1 0 0 1 1 1 1 0 0 1 0 0 1 0 0 0
 1 1 0 1 0 1 1 1 0 0 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 1 0 0 1 1 1 1 1 0
 0 0 0 1 1 1 1 1 0 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 0 1 0 1 0 1 1 1 0 0 1
 1 0 0 0 1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 0 1 0 1 0
 0 0 1 0 0 1 0 1 1 1 1 1 0 1 0 1 0 0 1 0 0 1 1 0 1 0 1 1 0 0 0 1 0 1 0 1 1 1
 1 0 1 1 0 0 1 1 1 1 0 0 1 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 1 1 0 1 0 1 0 0 1
 0 1 1 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 1 0 0 1 1 0 1 0 0 1 1 0 1 0 0 1 0 1 0 0 0
 1 0 1 0 1 1 0 1 1 0 0 0 1 0 1 1 1 1 1 1 1 0 1 1 0 0 1 0 1 1 0 1 0 0 0 1 0 0
 0 1 1 1 0 0 0 1]
```

K-Fold

Naive Bayes est l'algorithme le plus simple et le plus puissant. Malgré les avancées significatives du Machine Learning au cours de ces dernières années, il a fait ses preuves. Il a été déployé avec succès dans de nombreuses applications, de l'analyse de texte aux moteurs de recommandation.



B-2 SVM :

Les machines à vecteurs de support sont un ensemble de méthodes d'apprentissage supervisé utilisées pour la classification, la régression et la détection des valeurs aberrantes. Toutes ces tâches sont courantes dans l'apprentissage automatique.

Les SVM sont différents des autres algorithmes de classification en raison de la façon dont ils choisissent la limite de décision qui maximise la distance des points de données les plus proches de toutes les classes. La limite de décision créée par les SVM est appelée le classificateur de marge maximale ou l'hyperplan de marge maximale.

Tous d'abord on fait une validation de modèle en utilisant le K-fold cross validation .cette partie consiste à préparer le Kfold , création de modèle , évaluation de modèle , et calcule de précision et score autrement dit évaluation de performance .

Validation de modèle

Entrée [135]:
```
# evaluate a SVM model using k-fold cross-validation

# prepare the cross-validation procedure
cv = KFold(n_splits=10, random_state=1, shuffle=True)
# create model
model = SVC()
# evaluate model
scores = cross_val_score(model, X_test, y_test, scoring='accuracy', cv=cv, n_jobs=-1)
# report performance
print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
```

```
Accuracy: 0.801 (0.082)
```

Une fois les données sont déjà prêtes , on passe par la suite au choix des hypothèse , ce choix consiste à définir les paramètres , et le ' fit ' du model pour la gridsearch..

Choix des Hyperparametre

Entrée [136]:
```
from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train, y_train)
```

Un simple affichage des paramètres après le tunning , et aussi après les hyper-paramètres ; passant par la suite à la mesure de performance , et le recall et le f1 score et surtout le support ./

```
Entrée [144]: # print best parameter after tuning
              print(grid.best_params_)

              # print how our model looks after hyper-parameter tuning
              print(grid.best_estimator_)

              {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
              SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
                  decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf',
                  max_iter=-1, probability=False, random_state=None, shrinking=True,
                  tol=0.001, verbose=False)

Entrée [145]: grid_predictions = grid.predict(X_test)

              # print classification report
              print(classification_report(y_test, grid_predictions))

                            precision    recall  f1-score   support

                         0       0.90      0.69      0.78       191
                         1       0.74      0.93      0.82       187

                  accuracy                           0.80       378
                 macro avg       0.82      0.81      0.80       378
              weighted avg       0.82      0.80      0.80       378
```

Passant pour une prédiction basique pour les 5 premières lignes.

```
Entrée [150]: from sklearn import svm
              clf = svm.SVC(kernel='rbf', gamma=0.1, C=1)
              clf.fit(X_train, y_train)

              C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:760: DataConversionWarning: A column-vector y
              was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
                y = column_or_1d(y, warn=True)

 Out[150]: SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
               decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf',
               max_iter=-1, probability=False, random_state=None, shrinking=True,
               tol=0.001, verbose=False)

Entrée [151]: y_predic = svc_model.predict(X_test)
              y_predic [0:5]

 Out[151]: array([1, 0, 0, 0, 1])
```

Faisons finalement la matrice de confusion afin de déterminer à quel point on peut se servir de ce modèle pr une prédition finale.

```
Entrée [152]: # Confusion matrix and Accuracy of our model.

              from sklearn.metrics import confusion_matrix
              c_svm=confusion_matrix(y_test,y_predict)
              print(c_svm)
              Accuracy_svm=sum(np.diag(c_svm))/(np.sum(c_svm))
              Accuracy_svm

              [[131  60]
               [ 14 173]]

 Out[152]: 0.8042328042328042
```

On peut évaluer le modèle pour déterminer la précision, la marge d'erreur et le score afin de savoir à quel point ce modèle est performant.

```
Entrée [153]: #Evaluation
              from sklearn.metrics import classification_report
              print(classification_report(y_test,y_predic))
```

	precision	recall	f1-score	support
0	0.90	0.69	0.78	191
1	0.74	0.93	0.82	187
accuracy			0.80	378
macro avg	0.82	0.81	0.80	378
weighted avg	0.82	0.80	0.80	378

Résultats & conclusions :

Après avoir effectué une analyse exploratoire rapide des données sur les données, il y a eu plusieurs résultats intéressants :

La plupart des participants appartiennent à la catégorie d'âge entre 20 et 35 ans et une majorité d'entre eux sont des États-Unis. Il s'agit donc d'un échantillon biaisé. L'âge d'un répondant ne semble pas avoir d'influence sur son niveau d'aisance à discuter des problèmes de santé mentale avec ses superviseurs ou collègues. Cependant, les participants qui avaient des antécédents familiaux de maladie mentale sont plus susceptibles de demander un traitement que ceux qui n'en ont pas.

Nous observons également que les grandes entreprises ont tendance à discuter formellement davantage des problèmes de santé mentale, c'est-à-dire qu'elles ont peut-être mis en place des politiques formelles. De plus, les probabilités de maladie mentale parmi les entreprises de technologie ne sont pas significativement différentes de celles de maladie mentale dans les entreprises non technologiques.