TP statistique

March 10, 2024

1 Bibliothèques et data

[42]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from time import time
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.metrics import confusion_matrix, roc_curve, accuracy_score,␣
 ↪f1_score, roc_auc_score, classification_report
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import cross_val_score
from sklearn.metrics import precision_score, recall_score
import statsmodels.stats.multicomp as multi
df = pd.read_csv('student-mat.csv', sep=';')
```

[4]:
```
df
```

[4]:
```
    school sex  age address famsize Pstatus  Medu  Fedu      Mjob      Fjob  \
0       GP   F   18       U     GT3       A     4     4   at_home   teacher
1       GP   F   17       U     GT3       T     1     1   at_home     other
2       GP   F   15       U     LE3       T     1     1   at_home     other
3       GP   F   15       U     GT3       T     4     2    health  services
4       GP   F   16       U     GT3       T     3     3     other     other
..     …  ..  …     …     …     …   …   …       …       …
390     MS   M   20       U     LE3       A     2     2  services  services
391     MS   M   17       U     LE3       T     3     1  services  services
392     MS   M   21       R     GT3       T     1     1     other     other
393     MS   M   18       R     LE3       T     3     2  services     other
394     MS   M   19       U     LE3       T     1     1     other   at_home

    … famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
0   …      4        3      4     1     1      3        6   5   6   6
```

```
1    …      5        3      3     1     1      3        4   5   5   6
2    …      4        3      2     2     3      3       10   7   8  10
3    …      3        2      2     1     1      5        2  15  14  15
4    …      4        3      2     1     2      5        4   6  10  10
..   …    …      …    …    …   …    …      …  ..  ..  ..
390  …      5        5      4     4     5      4       11   9   9   9
391  …      2        4      5     3     4      2        3  14  16  16
392  …      5        5      3     3     3      3        3  10   8   7
393  …      4        4      1     3     4      5        0  11  12  10
394  …      3        2      3     3     3      5        5   8   9   9

[395 rows x 33 columns]
```

[6]:
```
variables_quantitatives = df.select_dtypes(include=['int', 'float'])
variables_quantitatives
```

[6]:
```
     age  Medu  Fedu  traveltime  studytime  failures  famrel  freetime  \
0     18     4     4           2          2         0       4         3
1     17     1     1           1          2         0       5         3
2     15     1     1           1          2         3       4         3
3     15     4     2           1          3         0       3         2
4     16     3     3           1          2         0       4         3
..   …   …   …         …        …       …     …       …
390   20     2     2           1          2         2       5         5
391   17     3     1           2          1         0       2         4
392   21     1     1           1          1         3       5         5
393   18     3     2           3          1         0       4         4
394   19     1     1           1          1         0       3         2

     goout  Dalc  Walc  health  absences  G1  G2  G3
0        4     1     1       3         6   5   6   6
1        3     1     1       3         4   5   5   6
2        2     2     3       3        10   7   8  10
3        2     1     1       5         2  15  14  15
4        2     1     2       5         4   6  10  10
..     …   …   …       …        …  ..  ..  ..
390      4     4     5       4        11   9   9   9
391      5     3     4       2         3  14  16  16
392      3     3     3       3         3  10   8   7
393      1     3     4       5         0  11  12  10
394      3     3     3       5         5   8   9   9

[395 rows x 16 columns]
```

2 Individus_Attributs

[7]:
```
n=df.shape[0]
p=df.shape[1]
print("nb_individus:",n)
print("nb_attributs:",p)
```

```
nb_individus: 395
nb_attributs: 33
```

3 ATTributs_catégorique

[8]:
```
attribut_names = df.columns.tolist()
print(attribut_names)
```

```
['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu',
'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'failures',
'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet',
'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences',
'G1', 'G2', 'G3']
```

- school : école de l'élève (binaire : "GP" pour "Grande Ville" ou "MS" pour "Milieu Rural")
- sex : sexe de l'élève (binaire : "F" pour féminin ou "M" pour masculin)
- age : âge de l'élève (numérique : de 15 à 22 ans)
- address : type d'adresse du domicile de l'élève (binaire : "U" pour urbain ou "R" pour rural)
- famsize : taille de la famille (binaire : "LE3" pour moins ou égal à 3 ou "GT3" pour plus de 3)
- Pstatus : statut de cohabitation des parents (binaire : "T" pour ensemble ou "A" pour séparés)
- Medu : éducation de la mère (numérique : 0 - aucun, 1 - éducation primaire (4e année), 2 - 5e à 9e année, 3 - éducation secondaire ou 4 - éducation supérieure)
- Fedu : éducation du père (numérique : 0 - aucun, 1 - éducation primaire (4e année), 2 - 5e à 9e année, 3 - éducation secondaire ou 4 - éducation supérieure)
- Mjob : métier de la mère (nominal : "enseignant", "santé" liée aux soins de santé, "services" civils (par exemple, administratif ou police), "à la maison" ou "autre")
- Fjob : métier du père (nominal : "enseignant", "santé" liée aux soins de santé, "services" civils (par exemple, administratif ou police), "à la maison" ou "autre")
- reason : raison de choisir cette école (nominal : "proximité" de la maison, "réputation" de l'école, préférence pour les "cours" ou "autre")
- guardian : tuteur de l'élève (nominal : "mère", "père" ou "autre")
- traveltime : temps de trajet domicile-école (numérique : 1 - <15 min., 2 - 15 à 30 min., 3 - 30 min. à 1 heure, ou 4 - >1 heure)
- studytime : temps d'étude hebdomadaire (numérique : 1 - <2 heures, 2 - 2 à 5 heures, 3 - 5 à 10 heures, ou 4 - >10 heures)
- failures : nombre d'échecs scolaires passés (numérique : n si 1<=n<3, sinon 4)
- schoolsup : soutien éducatif supplémentaire (binaire : oui ou non)
- famsup : soutien éducatif familial (binaire : oui ou non)
- paid : cours supplémentaires payants dans la matière du cours (mathématiques ou portugais)

(binaire : oui ou non)

- activities : activités parascolaires (binaire : oui ou non)
- nursery : fréquentation de la garderie (binaire : oui ou non)
- higher : souhaite poursuivre des études supérieures (binaire : oui ou non)
- internet : accès à Internet à domicile (binaire : oui ou non)
- romantic : en relation amoureuse (binaire : oui ou non)
- famrel : qualité des relations familiales (numérique : de 1 - très mauvaise à 5 - excellente)
- freetime : temps libre après l'école (numérique : de 1 - très bas à 5 - très élevé)
- goout : sorties avec des amis (numérique : de 1 - très bas à 5 - très élevé)
- Dalc : consommation d'alcool en semaine (numérique : de 1 - très faible à 5 - très élevée)
- Walc : consommation d'alcool le week-end (numérique : de 1 - très faible à 5 - très élevée)
- health : état de santé actuel (numérique : de 1 - très mauvais à 5 - très bon)
- absences : nombre d'absences à l'école (numérique : de 0 à 93)
- G1:première période (numérique : de 0 à 20)
- G2:Deuxième période (numérique : de 0 à 20)
- G3:Note finale (numérique : de 0 à 20, objectif de sortie)

4 Classer les données en catégories de Réussite et d'Échec

Le but du jeu de données est d'étudier les facteurs qui différencient les étudiants réussissant de ceux échouant. Ainsi, j'ai choisi de diviser le jeu de données en deux groupes : les étudiants ayant réussi et ceux ayant échoué. Cette division permettra aussi de mieux comprendre les caractéristiques distinctives des deux groupes et me facilite également l'application de modèles de prédiction SVM , Random Forest et KNN .

[9]:
```
df.iloc[:, -3:]
```

[9]:
```
     G1  G2  G3
0     5   6   6
1     5   5   6
2     7   8  10
3    15  14  15
4     6  10  10
..   ..  ..  ..
390   9   9   9
391  14  16  16
392  10   8   7
393  11  12  10
394   8   9   9

[395 rows x 3 columns]
```

[11]:
```
def classifier(df):
    df['G1'] = df['G1'].apply(lambda x: 1 if x >= 10 else 0)
    df['G2'] = df['G2'].apply(lambda x: 1 if x >= 10 else 0)
    df['G3'] = df['G3'].apply(lambda x: 1 if x >= 10 else 0)
```

```
classifier(df)
```

[12]:
```
df.iloc[:, -3:]
```

[12]:

	G1	G2	G3
0	0	0	0
1	0	0	0
2	0	0	1
3	1	1	1
4	0	1	1
..	..	..	..
390	0	0	0
391	1	1	1
392	1	0	0
393	1	1	1
394	0	0	0

[395 rows x 3 columns]

[13]:
```
print(df['G1'].value_counts())
print(df['G2'].value_counts())
print(df['G3'].value_counts())
```

```
1    253
0    142
Name: G1, dtype: int64
1    249
0    146
Name: G2, dtype: int64
1    265
0    130
Name: G3, dtype: int64
```

5 Numérisation & standardisation

Ensuite, j'ai décidé de convertir les variables qualitatives en variables quantitatives afin de les utiliser dans les modèles SVM, KNN et Random Forest. Ensuite, j'ai standardisé les données quantitatives quanen soustrayant la moyenne et en divisant par l'écart type.

[14]:
```
def numeriser_data():
    df['school'] = df['school'].map({'GP': 0, 'MS': 1})
    df['sex'] = df['sex'].map({'M': 0, 'F': 1})
    df['address'] = df['address'].map({'U': 0, 'R': 1})
    df['famsize'] = df['famsize'].map({'LE3': 0, 'GT3': 1})
    df['Pstatus'] = df['Pstatus'].map({'T': 0, 'A': 1})
```

```
    df['Mjob'] = df['Mjob'].map({'teacher': 0, 'health': 1, 'services': 2,␣
 ↪'at_home': 3, 'other': 4})
    df['Fjob'] = df['Fjob'].map({'teacher': 0, 'health': 1, 'services': 2,␣
 ↪'at_home': 3, 'other': 4})
    df['reason'] = df['reason'].map({'home': 0, 'reputation': 1, 'course': 2,␣
 ↪'other': 3})
    df['guardian'] = df['guardian'].map({'mother': 0, 'father': 1, 'other': 2})
    df['schoolsup'] = df['schoolsup'].map({'no': 0, 'yes': 1})
    df['famsup'] = df['famsup'].map({'no': 0, 'yes': 1})
    df['paid'] = df['paid'].map({'no': 0, 'yes': 1})
    df['activities'] = df['activities'].map({'no': 0, 'yes': 1})
    df['nursery'] = df['nursery'].map({'no': 0, 'yes': 1})
    df['higher'] = df['higher'].map({'no': 0, 'yes': 1})
    df['internet'] = df['internet'].map({'no': 0, 'yes': 1})
    df['romantic'] = df['romantic'].map({'no': 0, 'yes' : 1})




def standardiser(df):

    for column in variables_quantitatives.columns[:-3]:
        # Calculer la moyenne et l'écart type de la colonne
        mean = np.mean(df[column])
        std = np.std(df[column])

        # Standardiser la colonne si l'écart type est non nul
        if std != 0:
            df[column] = (df[column] - mean) / std
```

[15]:
```
numeriser_data()
standardiser(df)
```

[16]:
```
df
```

[16]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	\
0	0	1	1.023046	0	1	1	1.143856	1.360371	
1	0	1	0.238380	0	1	0	-1.600009	-1.399970	
2	0	1	-1.330954	0	0	0	-1.600009	-1.399970	
3	0	1	-1.330954	0	1	0	1.143856	-0.479857	
4	0	1	-0.546287	0	1	0	0.229234	0.440257	
..	…	…	…	…	…	…	…	…	
390	1	0	2.592380	0	0	1	-0.685387	-0.479857	
391	1	0	0.238380	0	0	0	0.229234	-1.399970	
392	1	0	3.377047	1	1	0	-1.600009	-1.399970	

```
393       1    0  1.023046        1         0         0  0.229234 -0.479857
394       1    0  1.807713        0         0         0 -1.600009 -1.399970

     Mjob  Fjob  …    famrel  freetime     goout      Dalc      Walc  \
0       3     0  …  0.062194 -0.236010  0.801479 -0.540699 -1.003789
1       3     4  …  1.178860 -0.236010 -0.097908 -0.540699 -1.003789
2       3     4  …  0.062194 -0.236010 -0.997295  0.583385  0.551100
3       1     2  … -1.054472 -1.238419 -0.997295 -0.540699 -1.003789
4       4     4  …  0.062194 -0.236010 -0.997295 -0.540699 -0.226345
..    …   …  …       …        …        …        …       …
390     2     2  …  1.178860  1.768808  0.801479  2.831553  2.105989
391     2     2  … -2.171138  0.766399  1.700867  1.707469  1.328545
392     4     4  …  1.178860  1.768808 -0.097908  1.707469  0.551100
393     2     4  …  0.062194  0.766399 -1.896683  1.707469  1.328545
394     4     3  … -1.054472 -1.238419 -0.097908  1.707469  0.551100

       health  absences  G1  G2  G3
0   -0.399289  0.036424   0   0   0
1   -0.399289 -0.213796   0   0   0
2   -0.399289  0.536865   0   0   1
3    1.041070 -0.464016   1   1   1
4    1.041070 -0.213796   0   1   1
..        …       …  ..  ..  ..
390  0.320890  0.661975   0   0   0
391 -1.119469 -0.338906   1   1   1
392 -0.399289 -0.338906   1   0   0
393  1.041070 -0.714236   1   1   1
394  1.041070 -0.088686   0   0   0

[395 rows x 33 columns]
```

6 Division_data

[17]:
```
y=df.G3
target=["G3"] #les classes reussite et echoue
x = df.drop(target,axis = 1 )
```

[18]:
```
y
```

[18]:
```
0      0
1      0
2      1
3      1
4      1
      ..
390    0
```

```
391    1
392    0
393    1
394    0
Name: G3, Length: 395, dtype: int64
```

[19]:
```
x
```

[19]:
```
     school  sex       age  address  famsize  Pstatus      Medu      Fedu  \
0         0    1  1.023046        0        1        1  1.143856  1.360371
1         0    1  0.238380        0        1        0 -1.600009 -1.399970
2         0    1 -1.330954        0        0        0 -1.600009 -1.399970
3         0    1 -1.330954        0        1        0  1.143856 -0.479857
4         0    1 -0.546287        0        1        0  0.229234  0.440257
..      ...  ...       ...      ...      ...      ...       ...       ...
390       1    0  2.592380        0        0        1 -0.685387 -0.479857
391       1    0  0.238380        0        0        0  0.229234 -1.399970
392       1    0  3.377047        1        1        0 -1.600009 -1.399970
393       1    0  1.023046        1        0        0  0.229234 -0.479857
394       1    0  1.807713        0        0        0 -1.600009 -1.399970

     Mjob  Fjob  …  romantic    famrel  freetime     goout      Dalc  \
0       3     0  …         0  0.062194 -0.236010  0.801479 -0.540699
1       3     4  …         0  1.178860 -0.236010 -0.097908 -0.540699
2       3     4  …         0  0.062194 -0.236010 -0.997295  0.583385
3       1     2  …         1 -1.054472 -1.238419 -0.997295 -0.540699
4       4     4  …         0  0.062194 -0.236010 -0.997295 -0.540699
..    ...   ...  …       ...       ...       ...       ...       ...
390     2     2  …         0  1.178860  1.768808  0.801479  2.831553
391     2     2  …         0 -2.171138  0.766399  1.700867  1.707469
392     4     4  …         0  1.178860  1.768808 -0.097908  1.707469
393     2     4  …         0  0.062194  0.766399 -1.896683  1.707469
394     4     3  …         0 -1.054472 -1.238419 -0.097908  1.707469

         Walc    health  absences  G1  G2
0   -1.003789 -0.399289  0.036424   0   0
1   -1.003789 -0.399289 -0.213796   0   0
2    0.551100 -0.399289  0.536865   0   0
3   -1.003789  1.041070 -0.464016   1   1
4   -0.226345  1.041070 -0.213796   0   1
..        …         …         …  ..  ..
390  2.105989  0.320890  0.661975   0   0
391  1.328545 -1.119469 -0.338906   1   1
392  0.551100 -0.399289 -0.338906   1   0
393  1.328545  1.041070 -0.714236   1   1
394  0.551100  1.041070 -0.088686   0   0
```

```
[395 rows x 32 columns]
```

7 split_data_20%_80%

J'ai ensuite divisé le jeu de données en 20 % pour les tests et 80 % pour l'entraînement.

[20]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
↪random_state=42)
```

8 SVM_poly

[21]:
```
svm_model = SVC(kernel='poly', probability=True, random_state=42)  # Choisis le␣
↪noyau approprié et d'autres paramètres si nécessaire
svm_model.fit(X_train, y_train)
svm_predictions = svm_model.predict(X_test)
```

[22]:
```
accuracy_linear = cross_val_score(svm_model, X_test, y_test, cv=10).mean()
print("Mean cross_val accuracy for Linear Regression =", accuracy_linear)
print("precision",precision_score(y_test,svm_predictions))
print("recall",recall_score(y_test,svm_predictions))
```

```
Mean cross_val accuracy for Linear Regression = 0.7339285714285715
precision 0.8888888888888888
recall 0.9230769230769231
```

9 SVM_rbf

[23]:
```
svm_model_rbf= SVC(kernel='rbf', probability=True, random_state=42)  # Choisis␣
↪le noyau approprié et d'autres paramètres si nécessaire
svm_model_rbf.fit(X_train, y_train)
svm_predictions_rbf = svm_model_rbf.predict(X_test)
```

[24]:
```
accuracy_linear = cross_val_score(svm_model_rbf, X_test, y_test, cv=10).mean()
print("Mean cross_val accuracy for Linear Regression =", accuracy_linear)
print("precision",precision_score(y_test,svm_predictions_rbf))
print("recall",recall_score(y_test,svm_predictions_rbf))
```

```
Mean cross_val accuracy for Linear Regression = 0.7982142857142857
precision 0.9787234042553191
recall 0.8846153846153846
```

10 Random_Forest

[25]:
```
# Création du modèle de forêt aléatoire
rndmForest_model = RandomForestClassifier(n_estimators=100, random_state=42)
rndmForest_model.fit(X_train, y_train)
rndomForest_predictions = rndmForest_model.predict(X_test)
```

[26]:
```
accuracy_linear = cross_val_score(rndmForest_model, X_test, y_test, cv=10).
↪mean()
print("Mean cross_val accuracy for Linear Regression =", accuracy_linear)
print("precision",precision_score(y_test,rndomForest_predictions))
print("recall",recall_score(y_test,rndomForest_predictions))
```

```
Mean cross_val accuracy for Linear Regression = 0.8732142857142857
precision 0.9787234042553191
recall 0.8846153846153846
```

11 Knn_N

[27]:
```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train.values, y_train.values)
knn_predictions = knn_model.predict(X_test.values)
```

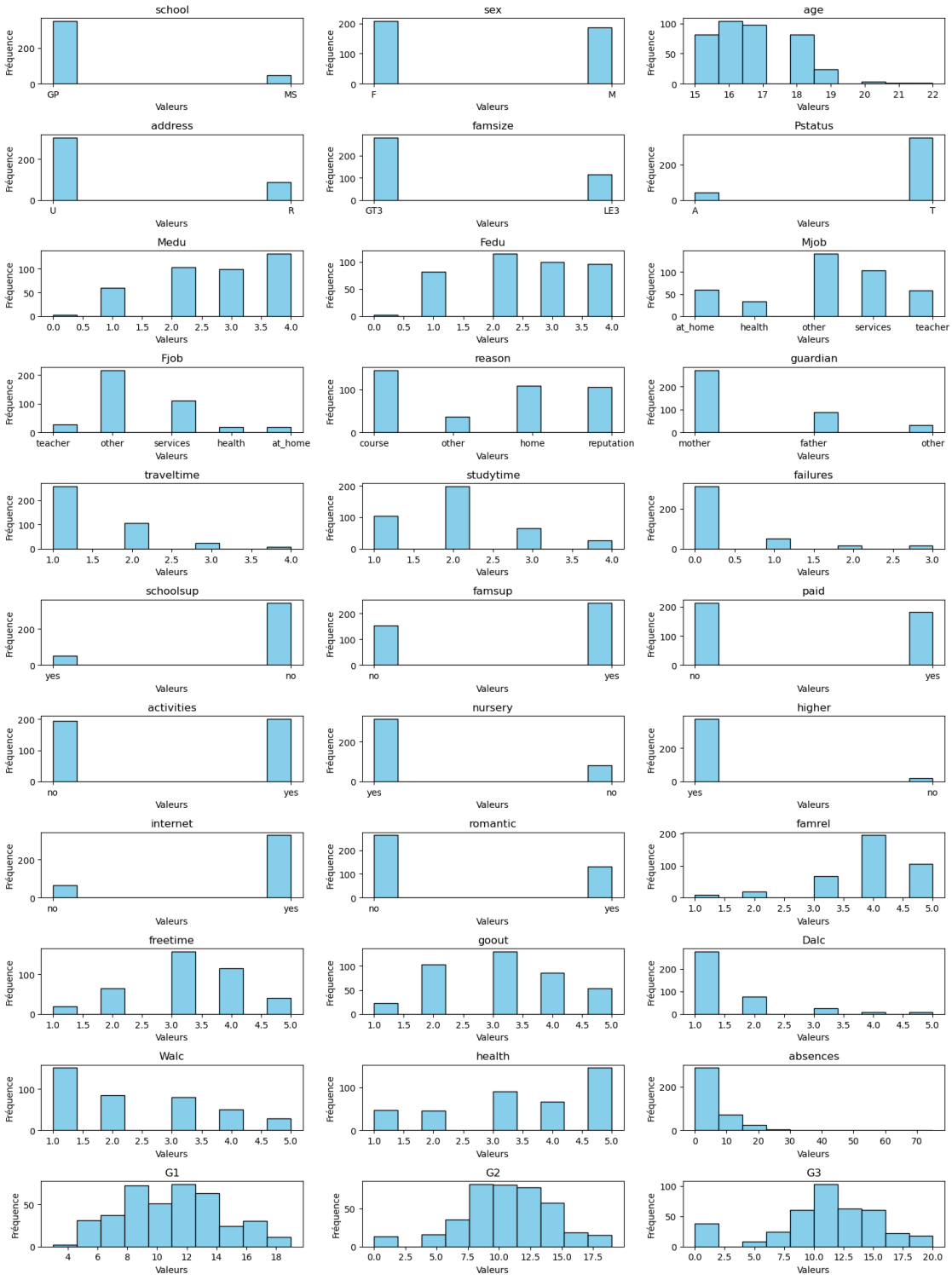
[28]:
```
accuracy_linear = cross_val_score(knn_model, X_test.values, y_test.values,␣
↪cv=10).mean()
print("Mean cross_val accuracy for Linear Regression =", accuracy_linear)
print("precision",precision_score(y_test.values,knn_predictions))
print("recall",recall_score(y_test.values,knn_predictions))
```

```
Mean cross_val accuracy for Linear Regression = 0.7232142857142857
precision 0.746031746031746
recall 0.9038461538461539
```

12 Histogramme des varaibles

[29]:
```
import matplotlib.pyplot as plt

# Définir le nombre de sous-graphiques par ligne et par colonne
data = pd.read_csv('student-mat.csv', sep=';')
n_cols = 3


n_rows = (len(data.columns) - 1) // n_cols + 1

fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, 20))

for i, col in enumerate(data.columns):
```

```
        ax = axes[i // n_cols, i % n_cols]  # Sélectionner l'axe approprié
        ax.hist(data[col], bins=10, color='skyblue', edgecolor='black')  #␣
 ↪Tracer l'histogramme
        ax.set_title(col)  # Définir le titre de l'histogramme
        ax.set_xlabel('Valeurs')  # Définir le label de l'axe x
        ax.set_ylabel('Fréquence')  # Définir le label de l'axe y

# Ajuster automatiquement l'espacement entre les sous-graphiques
plt.tight_layout()

# Afficher l'ensemble des histogrammes
plt.show()
```



Puisque les notes en G1, G2 et G3 semblent suivre une distribution normale, j'ai testé la distribution des notes en utilisant les tests d'adéquation Shapiro-Wilk, de lilliefors et Kolmogorov-Smirnov.

13 Test d'adéquation

[30]:
```
from scipy import stats
```

14 Test de Shapiro-Wilk

[31]:
```
#test de Normalite des notes en math  pour la groupe G3  et G2 et G1
X1= data["G1"]
X2= data["G1"]
X3= data["G1"]
print("Test de normalité  de shapiro wilk des notes en mathématiques :")

def Normality(X):
    p_value=stats.shapiro(X)[1]
    print("on a :p_value=" ,stats.shapiro(X)[1])
    if p_value<0.05:
        print("les notes ne suivent pas la distribution normale")
    else:
        print('les notes suivent la distribution normale')
print("G1 :")
Normality(X1)
print("G2:")
Normality(X2)
print("G3:")
Normality(X3)
```

```
Test de normalité  de shapiro wilk des notes en mathématiques :
G1 :
on a :p_value= 2.4554813080612803e-06
les notes ne suivent pas la distribution normale
G2:
on a :p_value= 2.4554813080612803e-06
les notes ne suivent pas la distribution normale
G3:
on a :p_value= 2.4554813080612803e-06
les notes ne suivent pas la distribution normale
```

15 Test de Kolmogorov-Smirnov

[32]:
```
from scipy.stats import kstest

#test de Normalite des notes en math  pour la groupe G3  et G2 et G1
X1= data["G1"]
X2= data["G2"]
X3= data["G3"]
print("Test de normalité de Kolmogorov des notes en mathématiques :")
```

```
def Normality_kol(X):
    p_value=kstest(X,'norm')[1]
    print("on a :p_value=" ,p_value)
    if p_value<0.05:
        print("les notes ne suivent pas la distribution normale")
    else:
        print('les notes suivent la distribution normale')

print("G1 :")
Normality_kol(X1)
print("G2:")
Normality_kol(X2)
print("G3:")
Normality_kol(X3)
```

```
Test de normalité de Kolmogorov des notes en mathématiques :
G1 :
on a :p_value= 0.0
les notes ne suivent pas la distribution normale
G2:
on a :p_value= 0.0
les notes ne suivent pas la distribution normale
G3:
on a :p_value= 0.0
les notes ne suivent pas la distribution normale
```

16 Test de lilliefors

[33]:
```
from statsmodels.stats.diagnostic import lilliefors
#test de Normalite des notes en math  pour la groupe G3  et G2 et G1
X1= data["G1"]
X2= data["G2"]
X3= data["G3"]
print("Test de normalité de lilliefors des notes en mathématiques :")

def Normality_lill(X):
    p_value=lilliefors(X)[1]
    print("on a :p_value=" ,p_value)
    if p_value<0.05:
        print("les notes ne suivent pas la distribution normale")
    else:
        print('les notes suivent la distribution normale')

print("G1 :")
Normality_lill(X1)
print("G2:")
```

```
Normality_lill(X2)
print("G3:")
Normality_lill(X3)
```

```
Test de normalité de lilliefors des notes en mathématiques :
G1 :
on a :p_value= 0.0009999999999998899
les notes ne suivent pas la distribution normale
G2:
on a :p_value= 0.0009999999999998899
les notes ne suivent pas la distribution normale
G3:
on a :p_value= 0.0009999999999998899
les notes ne suivent pas la distribution normale
```

Comme les résultats montrent que les notes ne suivent pas une distribution normale, comme montré sur le diagramme, je vais néanmoins essayer d'appliquer le test t de Student et le test de khi deux pour évaluer l'impact d'un facteur sur les notes. J'ai choisi les variables 'romantic', 'address' et 'sex' comme facteurs à tester pour le test t test et la variable "Internet" pour le test de khi deux .

17 Test paramétrique

18 Test de t-test(student)

[34]:
```
# Comparer les notes des élèves entre garçons et filles, entre les urbains et␣
 ↪ruraux, entre les élèves amoureux et non amoureux
Note = data["G1"]

def t_test(df_comp):
    if df_comp.name == "sex":
        tstat, pval = stats.ttest_ind(Note[df_comp == "M"], Note[df_comp ==␣
 ↪"F"])
        print("On a, p-value =", pval)
        if pval < 0.05:
            print("Il existe une différence significative entre les notes en␣
 ↪math des garçons et des filles.")
        else:
            print("Il n'existe pas de différence significative entre les notes␣
 ↪en math des garçons et des filles.")
    elif df_comp.name == "address":
        tstat, pval = stats.ttest_ind(Note[df_comp == "U"], Note[df_comp ==␣
 ↪"R"])
        print("On a, p-value =", pval)
        if pval < 0.05:
            print("Il existe une différence significative entre les notes en␣
 ↪math pour les élèves qui vivent en ville et en campagne.")
        else:
```

```
            print("Il n'existe pas de différence significative entre les notes␣
 ↪en math pour les élèves qui vivent en ville et en campagne.")
    elif df_comp.name == "romantic":
        tstat, pval = stats.ttest_ind(Note[df_comp == "yes"], Note[df_comp ==␣
 ↪"no"])
        print("On a, p-value =", pval)
        if pval < 0.05:
            print("Il existe une différence significative entre les notes en␣
 ↪math pour les élèves en relation amoureuse ou non.")
        else:
            print("Il n'existe pas de différence significative entre les notes␣
 ↪en math pour les élèves en relation amoureuse ou non.")

gender = data["sex"]
t_test(gender)
address = data["address"]
t_test(address)
romantic = data["romantic"]
t_test(romantic)
```

```
On a, p-value = 0.06825227168840965
Il n'existe pas de différence significative entre les notes en math des garçons
et des filles.
On a, p-value = 0.16677639907625813
Il n'existe pas de différence significative entre les notes en math pour les
élèves qui vivent en ville et en campagne.
On a, p-value = 0.46111256674789636
Il n'existe pas de différence significative entre les notes en math pour les
élèves en relation amoureuse ou non.
```

19 Test de khi deux

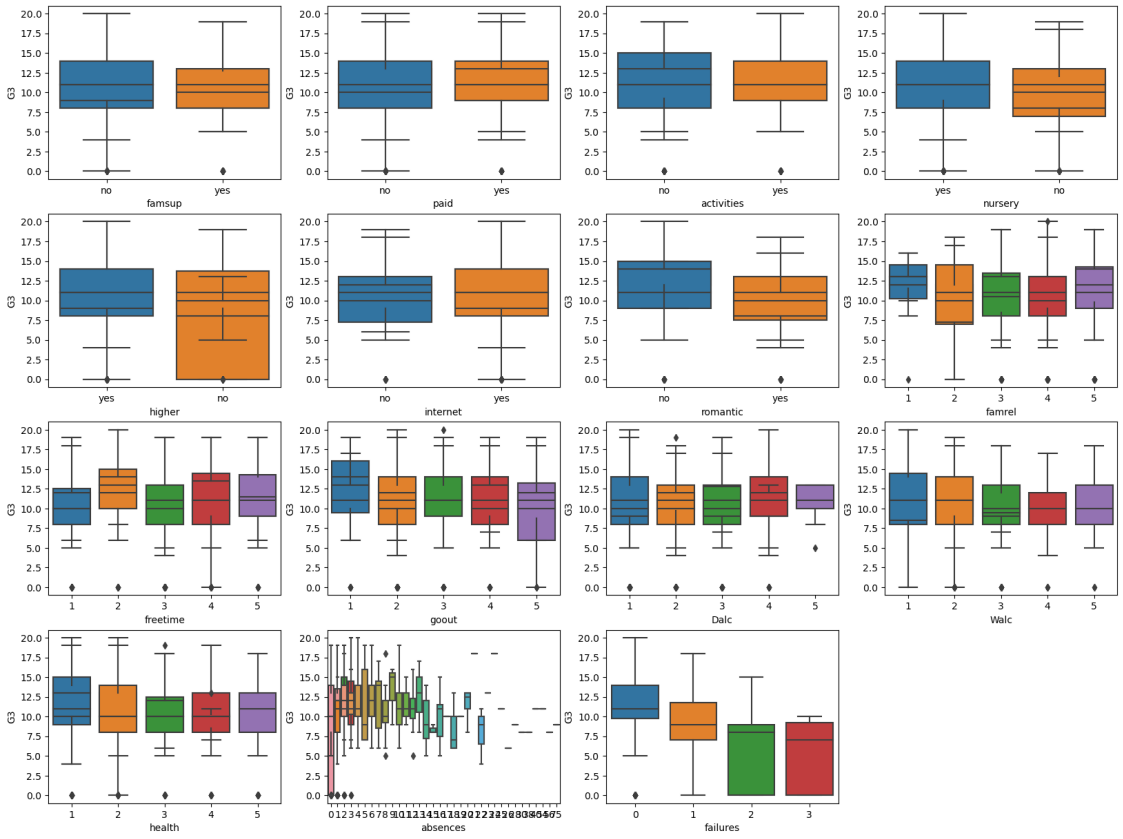
[35]:
```
#savoir si l'accées à l'internet à la maison a un impact sigificatif sur les␣
 ↪performances des étudiants :
print("Tester l'impact de l'accées à l'internet domicile ")
Note=data["G1"]
internet=data["internet"]
# Création du tableau de contingence
contingency_table = pd.crosstab(Note, internet)
# Calcul de la statistique de test du khi-deux
chi2, pval, dof, expected = stats.chi2_contingency(contingency_table)
print("pval=",pval)
if pval < 0.05:
  print("l'accées à l'internet domicile n'a pas d'impact sur les performances␣
 ↪des élèves ")
else:
```

```
 print("l'accées à l'internet a un impact sigificatif sur les perfomances des␣
↪élèves ")
```

```
Tester l'impact de l'accées à l'internet domicile
pval= 0.985258041416009
l'accées à l'internet a un impact sigificatif sur les perfomances des élèves
```

20 ANOVA en utilisant le tableau comparatif des performances

[36]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Supposons que 'data' est votre DataFrame et il a déjà été défini
num_col = data.drop('G3', axis=1).columns.tolist()  # Exclure G3 pour ne pas se␣
↪comparer à soi-même

plt.figure(figsize=(20, 15))
for index, col in enumerate(num_col[:15]):
    if (col!='G1' and col!='G2'):
        plt.subplot(4,4,index+1)
        #plt.subplot((len(num_col) + 1) // 2, 2, index + 1)  # Arrange les␣
↪plots en 2 colonnes
        sns.boxplot(x=col, y='G3', data=data)

for index, col in enumerate(num_col[16:]):
    if (col!='G1' and col!='G2'):
        plt.subplot(4,4,index+1)
        #plt.subplot((len(num_col) + 1) // 2, 2, index + 1)  # Arrange les␣
↪plots en 2 colonnes
        sns.boxplot(x=col, y='G3', data=data)

plt.show()

data.describe()
```



[36]:

	age	Medu	Fedu	traveltime	studytime	failures	\
count	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	
mean	16.696203	2.749367	2.521519	1.448101	2.035443	0.334177	
std	1.276043	1.094735	1.088201	0.697505	0.839240	0.743651	
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	
25%	16.000000	2.000000	2.000000	1.000000	1.000000	0.000000	
50%	17.000000	3.000000	2.000000	1.000000	2.000000	0.000000	
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	

	famrel	freetime	goout	Dalc	Walc	health	\
count	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	
mean	3.944304	3.235443	3.108861	1.481013	2.291139	3.554430	
std	0.896659	0.998862	1.113278	0.890741	1.287897	1.390303	
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	
25%	4.000000	3.000000	2.000000	1.000000	1.000000	3.000000	
50%	4.000000	3.000000	3.000000	1.000000	2.000000	4.000000	
75%	5.000000	4.000000	4.000000	2.000000	3.000000	5.000000	
max	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	

```
          absences          G1          G2          G3
count   395.000000  395.000000  395.000000  395.000000
mean      5.708861   10.908861   10.713924   10.415190
std       8.003096    3.319195    3.761505    4.581443
min       0.000000    3.000000    0.000000    0.000000
25%       0.000000    8.000000    9.000000    8.000000
50%       4.000000   11.000000   11.000000   11.000000
75%       8.000000   13.000000   13.000000   14.000000
max      75.000000   19.000000   19.000000   20.000000
```

Donc d'apres les boites de moustaches et le tableau de description de la mediane ,on peut choisur health,absences et failures comme des facteurs pour appliquer l'anova

[37]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols
model = ols('G3~ C(failures)', data=data).fit()
# Réalisation de l'ANOVA
anova_results = sm.stats.anova_lm(model, typ=2)
print(anova_results)
```

```
                sum_sq     df          F        PR(>F)
C(failures)  1137.135742    3.0  20.778271  1.642166e-12
Residual     7132.773118  391.0        NaN           NaN
```

le nombre d'échecs qu'un étudiant a subi a un impact significatif sur ses notes G3.

[236]:
```
# Formule incluant une interaction entre le statut amoureux et l'accès à␣
 ↪Internet
model = ols('G3~ C(absences)', data=data).fit()

# Réalisation de l'ANOVA
anova_results = sm.stats.anova_lm(model, typ=2)
print(anova_results)
```

```
                sum_sq     df         F    PR(>F)
C(absences)  1034.831402   33.0  1.564659  0.027447
Residual     7235.077459  361.0       NaN       NaN
```

le nombre d'absences a un impact sigificatifs sur les notes en G3

[238]:
```
# Formule incluant une interaction entre le statut amoureux et l'accès à␣
 ↪Internet
model = ols('G3~ C(romantic)', data=data).fit()

# Réalisation de l'ANOVA
anova_results = sm.stats.anova_lm(model, typ=2)
print(anova_results)
```

```
                sum_sq     df         F    PR(>F)
C(romantic)   139.696855    1.0  6.752698  0.009713
```

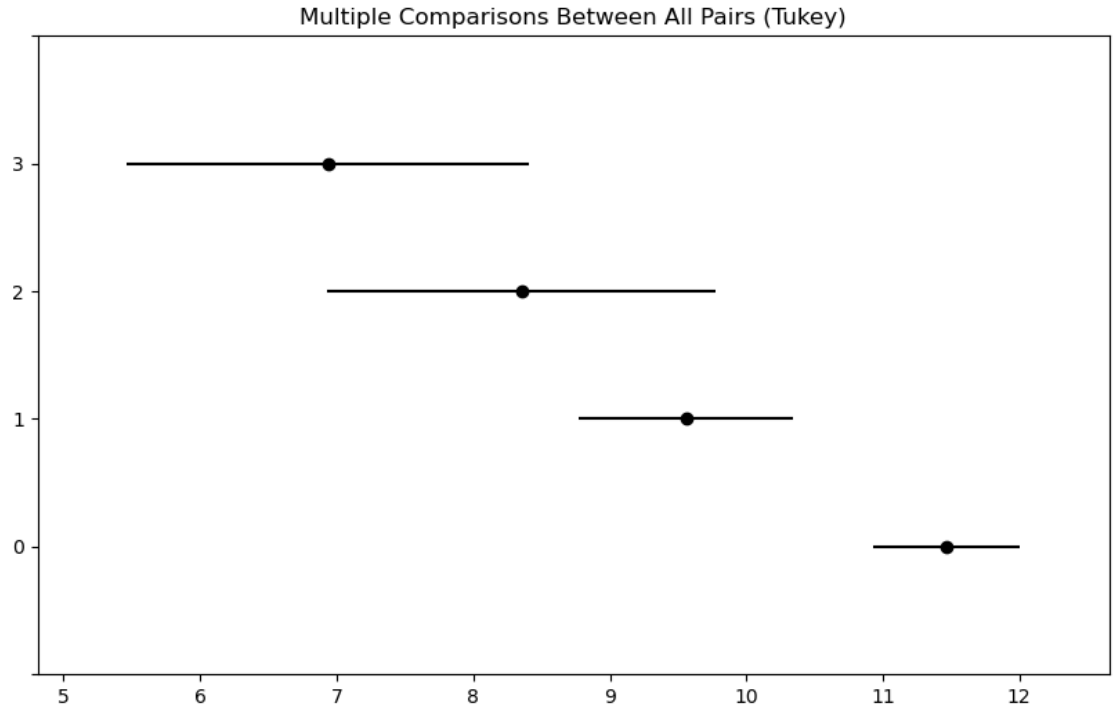
```
Residual      8130.212006  393.0        NaN        NaN
```

Être dans une relation amoureuse a un impact significatif sur les notes en G3.

21 Le diagramme des différences critiques

[38]:
```
# Réalisation du test post hoc de Tukey
tukey = multi.pairwise_tukeyhsd(endog=data['G1'],      # Données
                                groups=data['failures'],    # Groupes
                                alpha=0.05)                 # Niveau de␣
 ↪signification

# Afficher les résultats du test de Tukey
print(tukey)

# Créer le diagramme des différences critiques
tukey.plot_simultaneous()
plt.show()
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================
group1 group2 meandiff p-adj   lower   upper  reject
----------------------------------------------------
     0      1  -1.9079 0.0004 -3.1314 -0.6845   True
     0      2   -3.115 0.0004 -5.1153 -1.1147   True
     0      3  -4.5304    0.0 -6.5892 -2.4717   True
     1      2  -1.2071 0.5119  -3.462  1.0479  False
     1      3  -2.6225 0.0186 -4.9294 -0.3156   True
     2      3  -1.4154   0.56  -4.213  1.3821  False
----------------------------------------------------
```

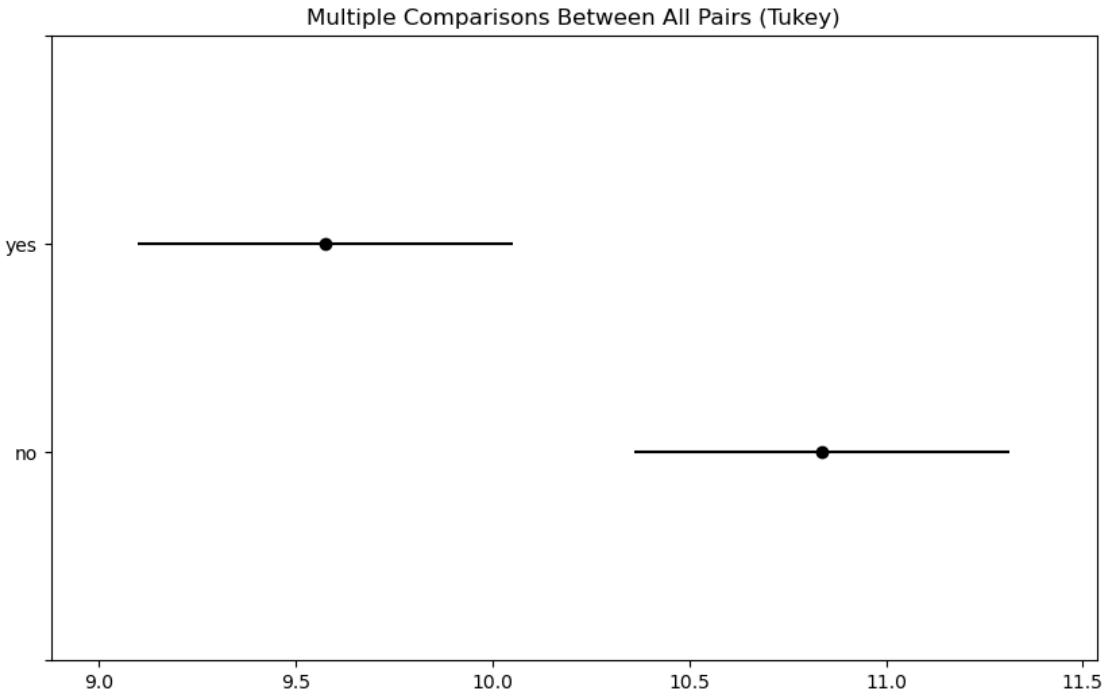


Sur le graphique, on peut voir que les intervalles de confiance pour les comparaisons des groupes 0-1, 0-2, et 0-3 ne chevauchent pas zéro, ce qui indique des différences significatives. L'intervalle pour le groupe 1-3 semble chevaucher zéro, ce qui suggère qu'il n'y a pas de différence significative entre ces groupes, ce qui concorde avec le tableau.

[40]:
```
import statsmodels.stats.multicomp as multi

# Réalisation du test post hoc de Tukey
tukey = multi.pairwise_tukeyhsd(endog=data['G3'],     # Données
                                groups=data['romantic'],    # Groupes
                                alpha=0.05)               # Niveau de␣
 ↪signification

# Afficher les résultats du test de Tukey
print(tukey)

# Créer le diagramme des différences critiques
tukey.plot_simultaneous()
plt.show()
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================
group1 group2 meandiff p-adj   lower   upper  reject
---------------------------------------------------
    no    yes  -1.2607 0.0097 -2.2146 -0.3069   True
```

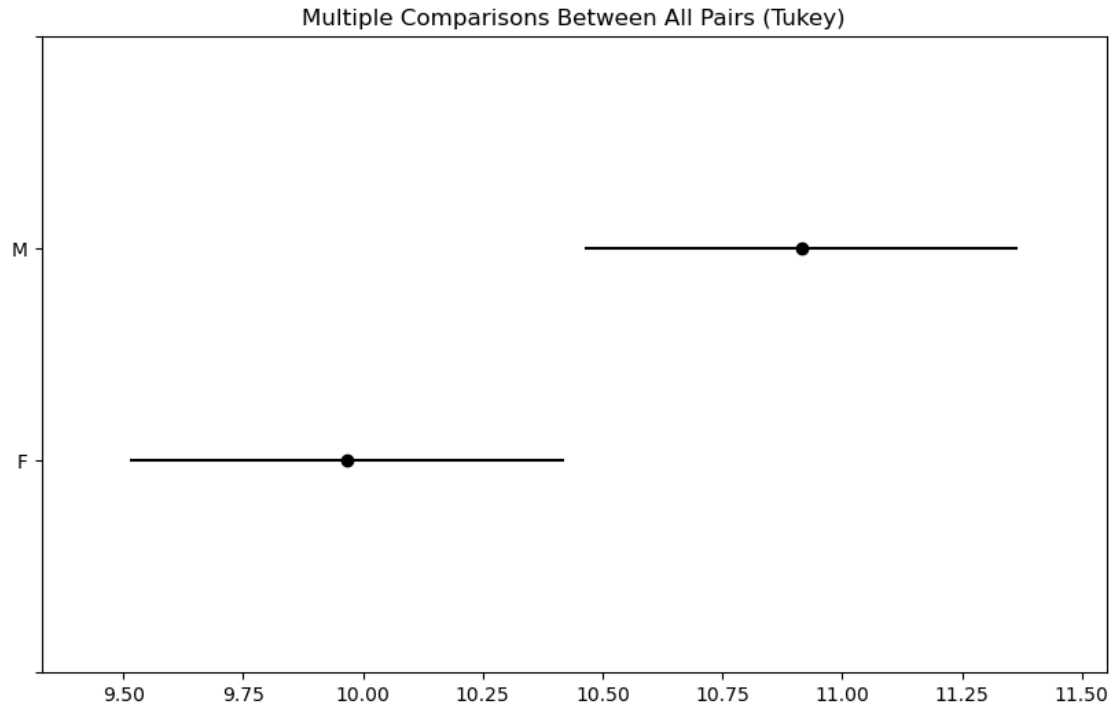
```
-------------------------------------------------
```



ce graphique et le tableau indiquent une différence statistiquement significative entre les élèves qui sont dans une relation romantique et ceux qui ne le sont pas en ce qui concerne la note finale G3 car l'intervalle de confiance pour la différence entre "no" et "yes" ne chevauche pas la ligne verticale .

[41]:
```
# Réalisation du test post hoc de Tukey
tukey = multi.pairwise_tukeyhsd(endog=data['G3'],      # Données
                                groups=data['sex'],    # Groupes
                                alpha=0.05)               # Niveau de␣
 ↪signification

# Afficher les résultats du test de Tukey
print(tukey)

# Créer le diagramme des différences critiques
tukey.plot_simultaneous()
plt.show()
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================
group1 group2 meandiff p-adj  lower  upper  reject
-------------------------------------------------
     F      M   0.9481 0.0399 0.0441 1.8521   True
```



on constate qu'il existe une différence significative dans la moyenne de la variable 'G3' entre les sexes, avec une moyenne plus élevée pour le groupe masculin par rapport au groupe féminin .