1.Data_Exploration

May 11, 2025

Myopia Study: Comprehensive Analysis, Modeling and Reporting

---

1 Table of Contents

- Next step : Predicting modeling (see 2.Predicting_modeling.ipynb)

---

2 1. Initialisation

2.0.1 Importing libraries and loading the myopia dataset for analysis

[1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
from scipy import stats
import plotly.graph_objects as go
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import TSNE
import seaborn as sns
import shap
import plotly.express as px
from sklearn.metrics import (accuracy_score, roc_auc_score,␣
 ↪classification_report,
                             confusion_matrix, roc_curve, f1_score)

from sklearn.ensemble import RandomForestClassifier,␣
 ↪HistGradientBoostingClassifier
from sklearn.inspection import permutation_importance
from sklearn.cluster import KMeans
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-
packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update
jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

[2]:
```
df = pd.read_csv('myopia.csv', sep=';')
df
```

[2]:
```
      ID  STUDYYEAR  MYOPIC  AGE  GENDER  SPHEQ     AL    ACD     LT    VCD  \
0      1       1992       1    6       1 -0.052  21.89  3.690  3.498  14.70
1      2       1995       0    6       1  0.608  22.38  3.702  3.392  15.29
2      3       1991       0    6       1  1.179  22.49  3.462  3.514  15.52
3      4       1990       1    6       1  0.525  22.20  3.862  3.612  14.73
4      5       1995       0    5       0  0.697  23.29  3.676  3.454  16.16
..   …         …       …   …       …      …      …      …      …      …
613  614       1995       1    6       0  0.678  22.40  3.663  3.803  14.93
614  615       1993       0    6       1  0.665  22.50  3.570  3.378  15.56
```

```
615  616       1995       0    6       0  1.834  22.94  3.624  3.424  15.89
616  617       1991       0    6       1  0.665  21.92  3.688  3.598  14.64
617  618       1994       0    6       0  0.802  22.26  3.530  3.484  15.25

     SPORTHR  READHR  COMPHR  STUDYHR  TVHR  DIOPTERHR  MOMMY  DADMY
0         45       8       0        0    10         34      1      1
1          4       0       1        1     7         12      1      1
2         14       0       2        0    10         14      0      0
3         18      11       0        0     4         37      0      1
4         14       0       0        0     4          4      1      0
..       …       …       …       …    …          …      …     …
613        2       0       7        3    14         37      1      0
614        6       0       1        0     8         10      1      1
615        8       0       0        0     4          4      1      1
616       12       2       1        0    15         23      0      0
617       25       0       2        0    10         14      1      1

[618 rows x 18 columns]
```

Columns : - **ID** : Incremental ID - **Study Year** : Year subject entered the study - **Myopic** : Myopia within the first five years of follow up - **Age** : Age at the first visit - **Gender** : Genre - **SPHEQ** : Spherical equivalent refraction - **AL** : Axial Length (mm) - **ACD** : Lens Thickness (mm) - **SPORTHR** : Time spent engaging in sports/outdoor activities (hour/week) - **READHR** : Time spend for pleasure (hours/week) - **COMPHR** : Time spend playing video/computer games or working on the computer (hours/week) - **STUDYHR** : Time spend reading or study for school assignments (hours/week) - **TVHR** : Time spend watching television (hours/week) - **DIOPTERHR** : Composite of near-work activities (hours/week) - **MOMMY** : Was the subject's mother myopic ? - **DADMY** : Was the subject's father myopic ?

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 618 entries, 0 to 617
Data columns (total 18 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   ID         618 non-null    int64
 1   STUDYYEAR  618 non-null    int64
 2   MYOPIC     618 non-null    int64
 3   AGE        618 non-null    int64
 4   GENDER     618 non-null    int64
 5   SPHEQ      618 non-null    float64
 6   AL         618 non-null    float64
 7   ACD        618 non-null    float64
 8   LT         618 non-null    float64
 9   VCD        618 non-null    float64
 10  SPORTHR    618 non-null    int64
 11  READHR     618 non-null    int64
 12  COMPHR     618 non-null    int64
 13  STUDYHR    618 non-null    int64
 14  TVHR       618 non-null    int64
 15  DIOPTERHR  618 non-null    int64
 16  MOMMY      618 non-null    int64
 17  DADMY      618 non-null    int64
dtypes: float64(5), int64(13)
memory usage: 87.0 KB
```

[4]:
```
df.describe(include='all')
```

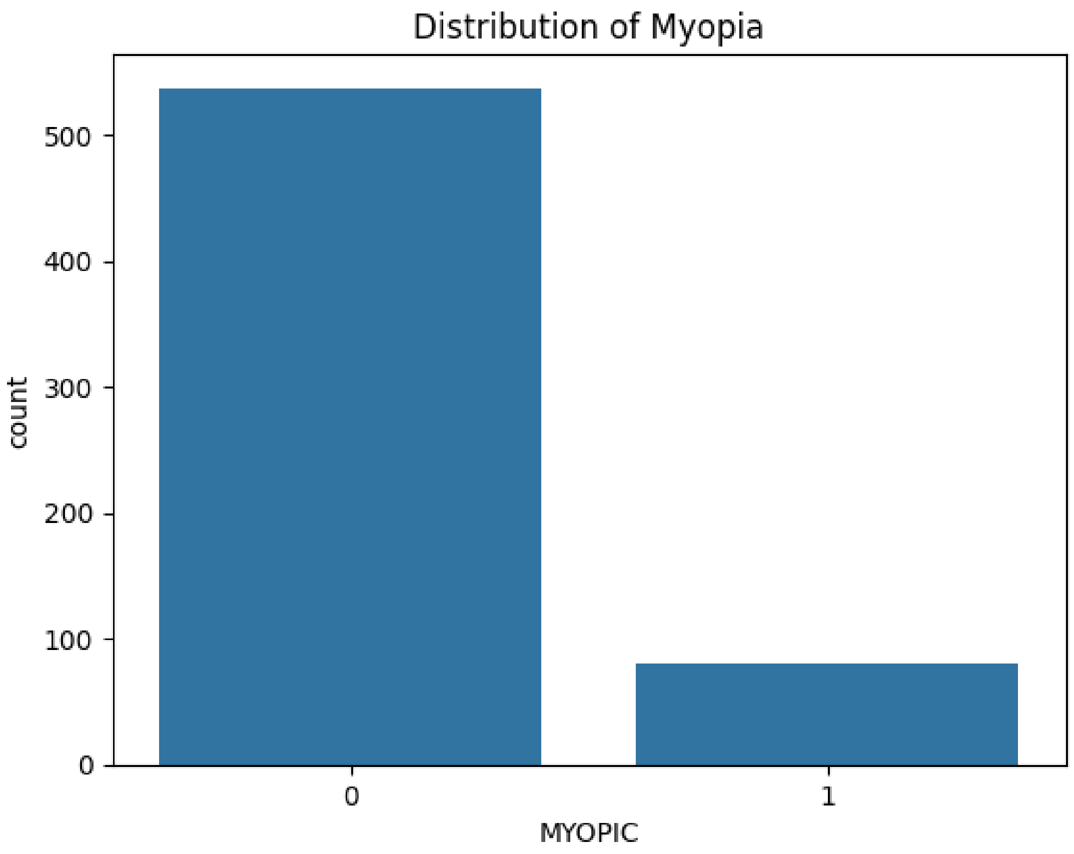
[4]:
```
               ID    STUDYYEAR      MYOPIC         AGE      GENDER  \
count  618.000000   618.000000  618.000000  618.000000  618.000000
mean   309.500000  1992.359223    0.131068    6.299353    0.488673
std    178.545512     1.734507    0.337748    0.712950    0.500277
min      1.000000  1990.000000    0.000000    5.000000    0.000000
25%    155.250000  1991.000000    0.000000    6.000000    0.000000
50%    309.500000  1992.000000    0.000000    6.000000    0.000000
75%    463.750000  1994.000000    0.000000    6.000000    1.000000
max    618.000000  1995.000000    1.000000    9.000000    1.000000

            SPHEQ          AL         ACD          LT         VCD     SPORTHR  \
count  618.000000  618.000000  618.000000  618.000000  618.000000  618.000000
mean     0.801010   22.496780    3.578629    3.541453   15.376780   11.953074
std      0.625918    0.680141    0.230394    0.154519    0.664183    7.968296
min     -0.699000   19.900000    2.772000    2.960000   13.380000    0.000000
25%      0.456250   22.040000    3.424000    3.436000   14.930000    6.000000
50%      0.729000   22.465000    3.585000    3.542000   15.360000   10.000000
75%      1.034000   22.970000    3.730000    3.640000   15.840000   16.000000
max      4.372000   24.560000    4.250000    4.112000   17.300000   45.000000

           READHR      COMPHR     STUDYHR        TVHR   DIOPTERHR       MOMMY  \
count  618.000000  618.000000  618.000000  618.000000  618.000000  618.000000
mean     2.796117    2.105178    1.490291    8.948220   26.017799    0.506472
std      3.068191    3.056508    2.216207    5.719021   16.031715    0.500363
min      0.000000    0.000000    0.000000    0.000000    2.000000    0.000000
25%      0.000000    0.000000    0.000000    4.250000   15.000000    0.000000
50%      2.000000    1.000000    1.000000    8.000000   23.000000    1.000000
75%      4.000000    3.000000    2.000000   12.000000   34.000000    1.000000
max     20.000000   30.000000   15.000000   31.000000  101.000000    1.000000

            DADMY
count  618.000000
mean     0.498382
std      0.500402
min      0.000000
25%      0.000000
```

```
50%      0.000000
75%      1.000000
max      1.000000
```

[5]:
```python
print("Shape:", df.shape)
print(df.isnull().sum())
```

```
Shape: (618, 18)
ID           0
STUDYYEAR    0
MYOPIC       0
AGE          0
GENDER       0
SPHEQ        0
AL           0
ACD          0
LT           0
VCD          0
SPORTHR      0
READHR       0
COMPHR       0
STUDYHR      0
TVHR         0
DIOPTERHR    0
MOMMY        0
DADMY        0
dtype: int64
```

[6]:
```python
sns.countplot(x='MYOPIC', data=df)
plt.title('Distribution of Myopia')
plt.show()
print('Class distribution (%):')
print((df['MYOPIC'].value_counts(normalize=True) * 100).round(2))
```



```
Class distribution (%):
MYOPIC
0    86.89
1    13.11
Name: proportion, dtype: float64
```

⇒ Dataset imbalanced

2.0.2 *Data Engineering*

```
[7]: df['PARENTSMY'] = ((df['DADMY']==1) | (df['MOMMY']==1)).astype(int)
     df = df.drop(['MOMMY', 'DADMY', 'ID', 'STUDYYEAR'], axis=1)
```

```
[8]: df['SCREENHR'] = df['COMPHR'] + df['TVHR'] # Screens hour
     df['CLOSEHR'] = df['READHR'] + df['STUDYHR'] + df['DIOPTERHR'] # Activities␣
      ↪with static eyes or close
     df = df.drop(['COMPHR', 'TVHR', 'READHR', 'STUDYHR', 'DIOPTERHR'], axis=1)
```

```
[9]: df
```

```
[9]:      MYOPIC  AGE  GENDER  SPHEQ     AL    ACD     LT    VCD  SPORTHR  \
     0         1    6       1 -0.052  21.89  3.690  3.498  14.70       45
     1         0    6       1  0.608  22.38  3.702  3.392  15.29        4
     2         0    6       1  1.179  22.49  3.462  3.514  15.52       14
     3         1    6       1  0.525  22.20  3.862  3.612  14.73       18
     4         0    5       0  0.697  23.29  3.676  3.454  16.16       14
     ..      …   …     …     …      …     …      …      …     …
     613       1    6       0  0.678  22.40  3.663  3.803  14.93        2
     614       0    6       1  0.665  22.50  3.570  3.378  15.56        6
     615       0    6       0  1.834  22.94  3.624  3.424  15.89        8
     616       0    6       1  0.665  21.92  3.688  3.598  14.64       12
     617       0    6       0  0.802  22.26  3.530  3.484  15.25       25

          PARENTSMY  SCREENHR  CLOSEHR
     0            1        10       42
     1            1         8       13
     2            0        12       14
     3            1         4       48
     4            1         4        4
     ..         …         …      …
     613          1        21       40
     614          1         9       10
     615          1         4        4
     616          0        16       25
     617          1        12       14

     [618 rows x 12 columns]
```

Summary Key features were engineered to synthesize parental myopia risk and consolidate hours spent on screens or in close-up activities. Irrelevant or redundant variables were removed, resulting in a cleaner and more interpretable dataset. This step both streamlines later modeling and enhances overall scientific clarity.

3 2. Data Exploration

The dataset is separated into numerical and categorical components to enable targeted exploratory analysis. This approach allows for tailored statistical summaries and visualizations, enhancing our understanding of both continuous variables and key risk subgroups before further modeling.

```
[10]: cat = ['MYOPIC', 'GENDER', 'PARENTSMY']
      myopianum = df.drop(cat, axis=1)
      myopiafact = df[cat]
```

```
[11]: myopianum.head(5)
```

[11]:
```
   AGE  SPHEQ     AL    ACD     LT    VCD  SPORTHR  SCREENHR  CLOSEHR
0    6 -0.052  21.89  3.690  3.498  14.70       45        10       42
1    6  0.608  22.38  3.702  3.392  15.29        4         8       13
2    6  1.179  22.49  3.462  3.514  15.52       14        12       14
3    6  0.525  22.20  3.862  3.612  14.73       18         4       48
4    5  0.697  23.29  3.676  3.454  16.16       14         4        4
```

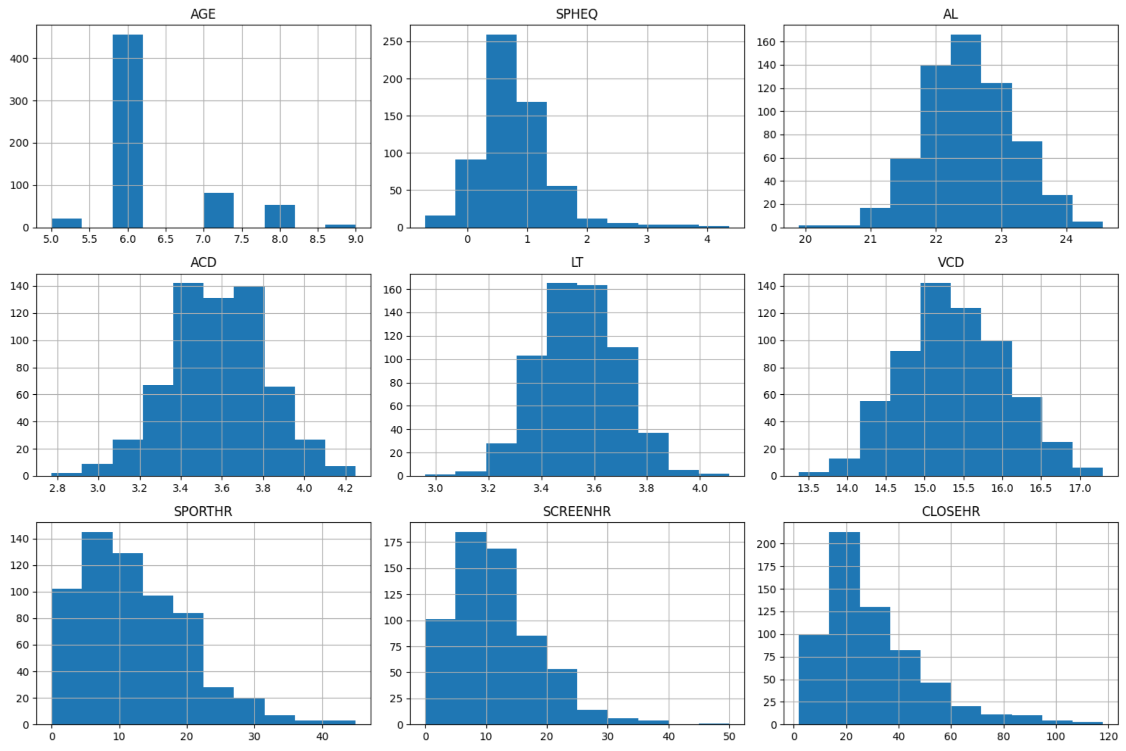
[12]:
```python
myopiafact.head(5)
```

[12]:
```
   MYOPIC  GENDER  PARENTSMY
0       1       1          1
1       0       1          1
2       0       1          0
3       1       1          1
4       0       0          1
```

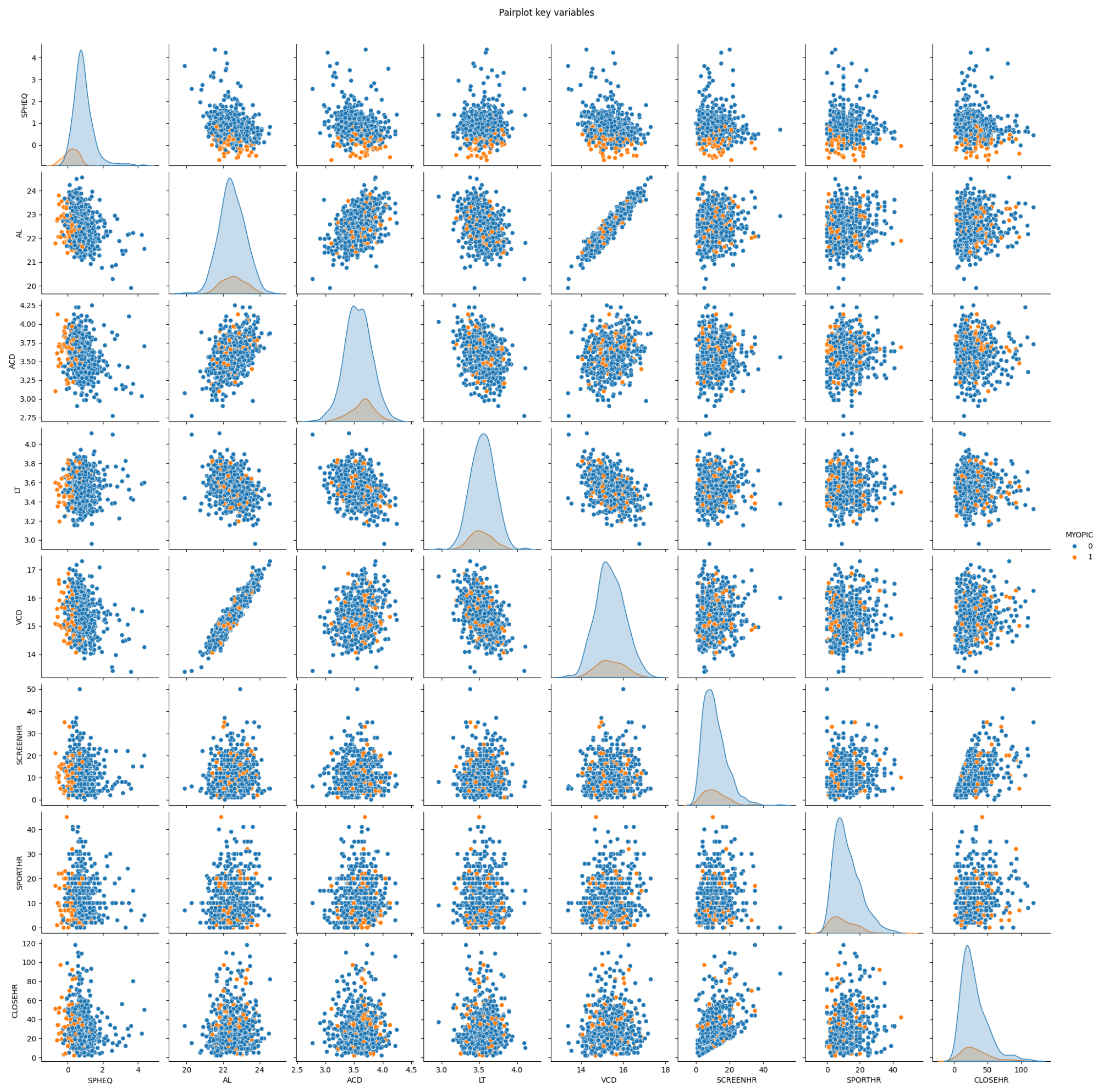
3.0.1 Univariate Analysis and Multivariate Visualization

- **Distribution of Continuous Features**: Visualized using boxplots and histograms.
- **Categorical Features Breakdown**: Analyzed to understand their distribution and impact.

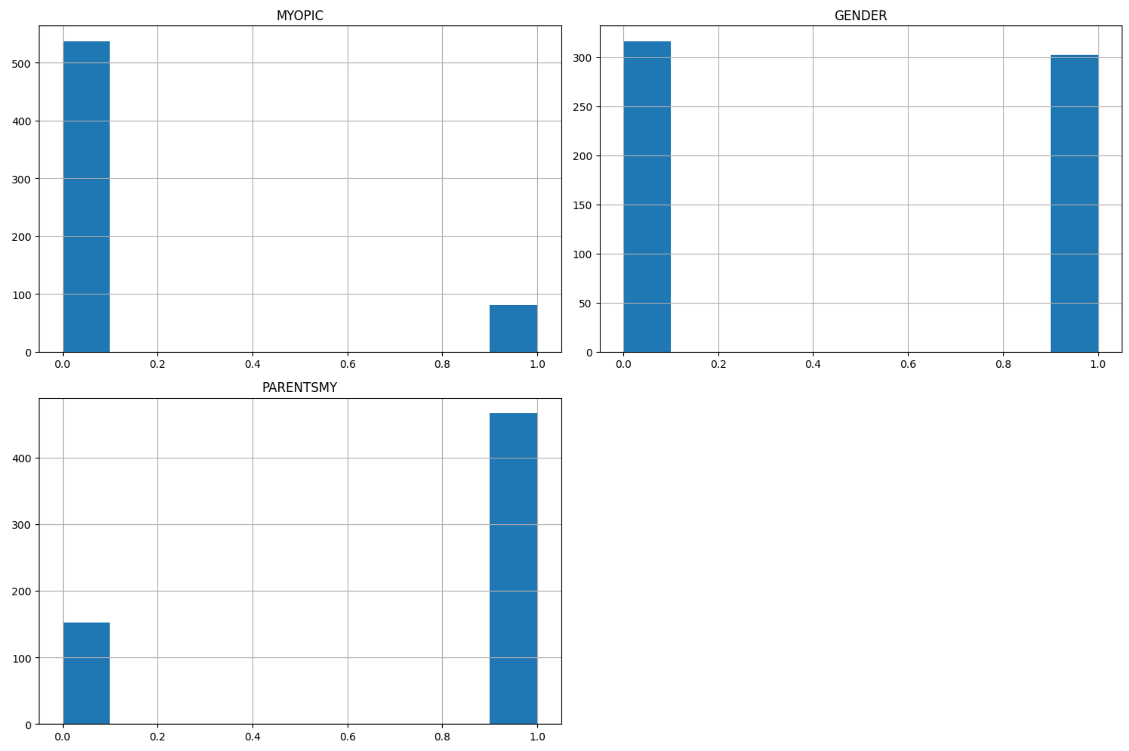
[13]:
```python
myopianum.hist(figsize=(15,10))
plt.tight_layout()
plt.show()
```



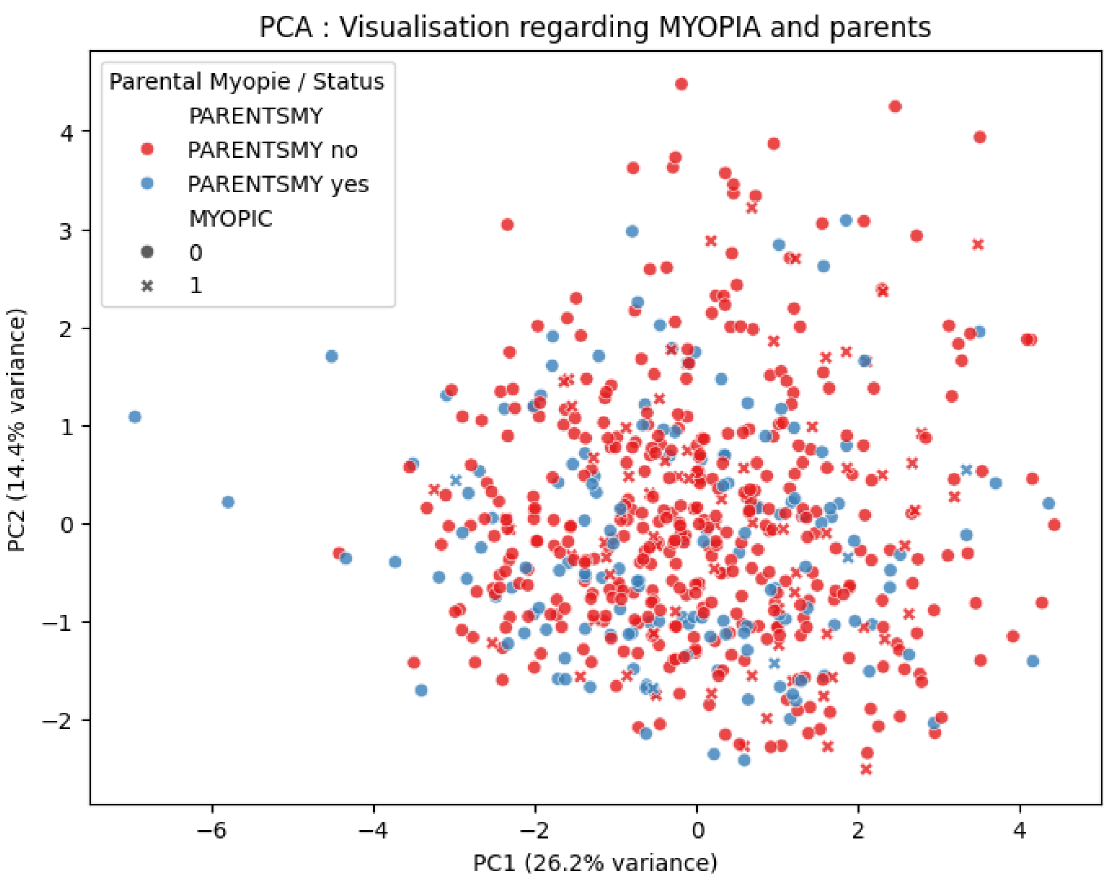
```
[14]: sns.pairplot(df, hue='MYOPIC', vars=['SPHEQ','AL', 'ACD', 'LT','VCD',␣
       ↪'SCREENHR' ,'SPORTHR','CLOSEHR'])
      plt.suptitle("Pairplot key variables", y=1.02)
      plt.show()
```



```
[15]: myopiafact.hist(figsize=(15,10))
      plt.tight_layout()
      plt.show()
```



[16]:
```python
X = df.drop('MYOPIC', axis=1)
y = df['MYOPIC']

group = df['PARENTSMY'].map({0: 'PARENTSMY yes', 1: 'PARENTSMY no'})

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(8,6))
sns.scatterplot(x=X_pca[:,0], y=X_pca[:,1], hue=group, style=y, palette='Set1',␣
 ↪alpha=0.8)
plt.xlabel(f'PC1 ({pca.explained_variance_ratio_[0]*100:.1f}% variance)')
plt.ylabel(f'PC2 ({pca.explained_variance_ratio_[1]*100:.1f}% variance)')
plt.title("PCA : Visualisation regarding MYOPIA and parents")
plt.legend(title='Parental Myopie / Status')
plt.show()
```



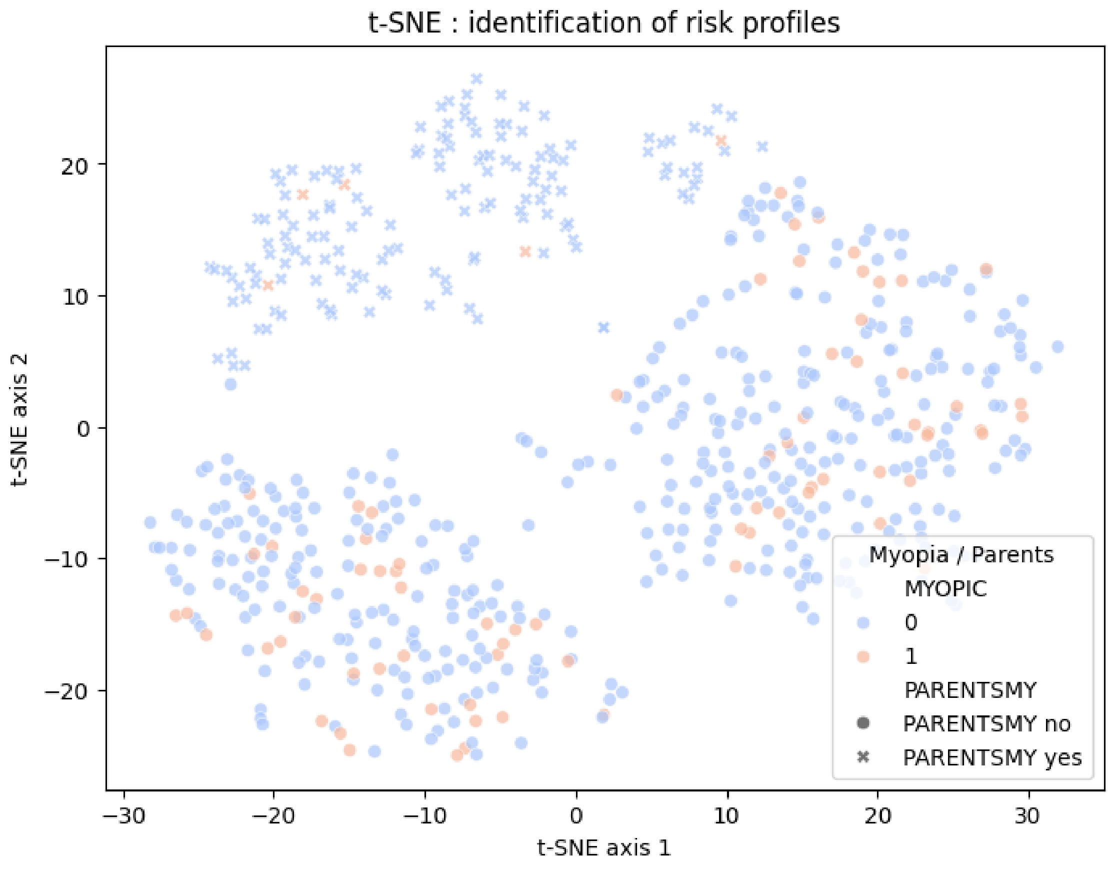
PCA Plot Interpretation

- **Dimensionality Reduction for Interpretation**
  The PCA plot summarizes the variation across all features into 2 principal axes. PC1 (26.2% variance) and PC2 (14.4%) together capture about 40% of the overall data structure.
  This reduction allows us to view complex data as a 2D scatterplot, highlighting potential clusters and patterns otherwise hidden in high dimensions.

- **Risk Profile Overlay**
  Data points are annotated both by **myopic status (MYOPIC: 0/1)** and **parental myopia (PARENTSMY: yes/no)**, providing a clinical context.
  This dual-label strategy visually explores the hypothesis that parental history (a known risk factor) may stratify children into more or less vulnerable clusters.

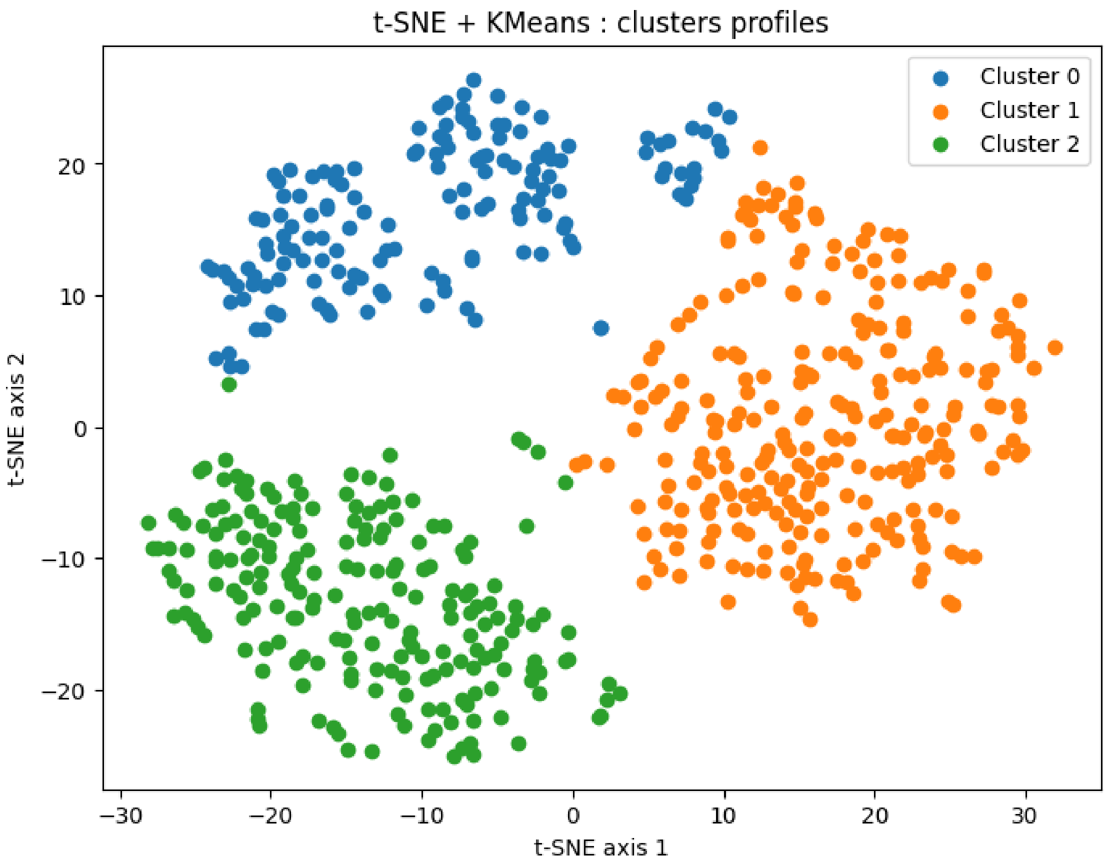
- **Interpreting Distributions**
  While myopic cases (red) and controls (blue) are not clearly separable (indicating complex or non-linear feature relations), there may be subtle tendencies for points with parental myopia to congregate in selected regions, hinting at multi-factorial risk.

[17]:
```
data = df.copy()

tsne = TSNE(n_components=2, perplexity=30, random_state=42)
X_tsne = tsne.fit_transform(X_scaled)

plt.figure(figsize=(8,6))
sns.scatterplot(x=X_tsne[:,0], y=X_tsne[:,1], hue=y, style=group,␣
 ↪palette='coolwarm', alpha=0.7)
plt.title('t-SNE : identification of risk profiles')
plt.xlabel('t-SNE axis 1')
plt.ylabel('t-SNE axis 2')
plt.legend(title='Myopia / Parents')
plt.show()
```



[18]:
```
data['TSNE-1'] = X_tsne[:,0]
data['TSNE-2'] = X_tsne[:,1]

kmeans = KMeans(n_clusters=3, random_state=42)
data['cluster'] = kmeans.fit_predict(X_tsne)
```

```
plt.figure(figsize=(8,6))
for c in range(3):
    plt.scatter(
        data.loc[data['cluster']==c, 'TSNE-1'],
        data.loc[data['cluster']==c, 'TSNE-2'],
        label=f'Cluster {c}'
    )
plt.legend()
plt.title('t-SNE + KMeans : clusters profiles')
plt.xlabel('t-SNE axis 1')
plt.ylabel('t-SNE axis 2')
plt.show()
```



[19]:
```
# Statistiques descriptives par cluster
print(data.groupby('cluster')[['MYOPIC', 'PARENTSMY']].mean())

# Nombre d'individus par cluster
```

```
print(data['cluster'].value_counts())
```

```
           MYOPIC  PARENTSMY
cluster
0        0.033113   0.000000
1        0.145038   0.996183
2        0.185366   1.000000
cluster
1    262
2    205
0    151
Name: count, dtype: int64
```

t-SNE Plot Interpretation

- **Nonlinear Embedding for Cluster Discovery**
  t-SNE further condenses complex feature interactions into a 2D manifold, preserving local similarities. This can reveal groupings and structure not visible via linear methods like PCA.
- **Enhanced Profiling of At-Risk Groups**
  Overlaying both **myopic outcome** and **parental myopia** again, t-SNE sometimes reveals local "clouds" or concentrations. If visible, these can be interpreted as multi-featured risk profiles—potential targets for clinical intervention or finer stratification.
- **Cluster Identification with t-SNE**
  The t-SNE visualization reveals three clearly separated clusters, indicating distinct profiles in the data. Each individual is represented in a two-dimensional space based on their original features (e.g. myopia status, parental myopia, and other numeric variables). Individuals that appear close together in the t-SNE plot are more similar to each other in the original data space.

  By examining the cluster composition, we can identify groups with differing risk factors—for example, clusters where most children are myopic and have myopic parents, versus clusters with low genetic risk. This segmentation enables deeper analysis of risk trajectories and can guide targeted interventions or the development of subgroup-specific predictive models.

3.1 2.1. Statistic analysis

3.1.1 2.1.1. Univariate Analysis

- Distribution of continuous features (boxplot and histogram).
- Categorical features breakdown.

[20]:
```
num_cols = df.drop(cat, axis=1).columns.tolist()
for col in num_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')

df['GENDER'] = df['GENDER'].astype('category')
df['PARENTSMY'] = df['PARENTSMY'].astype('category')
df['MYOPIC'] = df['MYOPIC'].astype('category')
```

```
[21]: # crosstab "MYOPIC" x "GENDER"
      print("MYOPIC x GENDER")
      print('Nb Boys : ', (df['GENDER']==0).sum())
      print('Nb Girls : ', (df['GENDER']==1).sum())
      # Chi-2 test on this crosstab
      table1 = pd.crosstab(df['MYOPIC'], df['GENDER'])
      chi2, p, dof, ex = stats.chi2_contingency(table1)
      print('p-value chi-2:', p)
      print('chi-value chi-2:', chi2)
      print('dof-value chi-2:', dof)
      print('ex-value chi-2:', ex)
      # crosstab
      table_genre = pd.crosstab(df['MYOPIC'], df['GENDER'], normalize='columns').
       ↪round(2)
      table_genre.index = ['No-Myopic','Myopic']
      table_genre.columns = ['Man', 'Woman']
      display(table_genre)

      # crosstab "MYOPIC" x "PARENTMY"
      print("\n", "-" * 30, "\nMYOPIC x PARENTSMY")
      print('Nb Myopic Parents : ', (df['PARENTSMY']==1).sum())
      print('Nb Non Myopic Parents : ', (df['PARENTSMY']==0).sum())
      # Chi-2 test on this crosstab
      table2 = pd.crosstab(df['MYOPIC'], df['PARENTSMY'])
      chi2, p, dof, ex = stats.chi2_contingency(table2)
      print('p-value chi-2:', p)
      print('chi-value chi-2:', chi2)
      print('dof-value chi-2:', dof)
      print('ex-value chi-2:', ex)

      # Crosstab
      table_Parents = pd.crosstab(df['MYOPIC'], df['PARENTSMY'], normalize='index').
       ↪round(2)
      table_Parents.index = ['No-Myopic','Myopic']
      table_Parents.columns = ['No','Yes']
      display(table_Parents)
```

```
MYOPIC x GENDER
Nb Boys :  316
Nb Girls :  302
p-value chi-2: 0.15822974722920058
chi-value chi-2: 1.9910632919718099
dof-value chi-2: 1
ex-value chi-2: [[274.58252427 262.41747573]
 [ 41.41747573  39.58252427]]

            Man  Woman
No-Myopic  0.89   0.85
```

```
Myopic     0.11   0.15


 ------------------------------
MYOPIC x PARENTSMY
Nb Myopic Parents :  466
Nb Non Myopic Parents :  152
p-value chi-2: 6.556104369308498e-05
chi-value chi-2: 15.934831626844861
dof-value chi-2: 1
ex-value chi-2: [[132.0776699 404.9223301]
 [ 19.9223301  61.0776699]]

             No   Yes
No-Myopic  0.27  0.73
Myopic     0.06  0.94
```

[22]:
```
fig = go.Figure(data=[
    go.Bar(name='GENDER',
           x=['Males','Females'],
           y=table_genre.loc['Myopic'].values*100),
    go.Bar(name='PARENTMY',
           x=['No Myopic Parents','At least one Myopic Parent'],
           y=table_Parents.loc['Myopic'].values*100)
])
fig.update_layout(barmode='group', yaxis_title="Percentage of Myopic(%)")
fig.show()
```

crosstab	p value	chi2	Proportion Analysis
Myopic-Gender	0.158	1.99	p-value above 0.05. It implies that there is no statistically significant association between gender and myopia.
Myopic-Parentsmy	6.5 E-5	15.9	A highly significant p-value shows a strong association between parental myopia and child myopia. Children with at least one myopic parent are much more likely to be myopic themselves.

Analysis of Gender and Parental Myopia Association with Myopia Status

Gender and Myopia:

No statistically significant association was found between gender and myopia incidence ($p > 0.05$).

Boys and girls have similar rates of myopia in this population.

Parental Myopia:

A strong and highly significant association was found between having at least one myopic parent and being myopic ($p < 0.001$).

Children with myopic parents are much more likely to develop myopia themselves.

Implications:

Gender does not appear to be a risk factor for myopia in this dataset.

Parental (hereditary) myopia should be prioritized when assessing a child's risk for developing myopia.

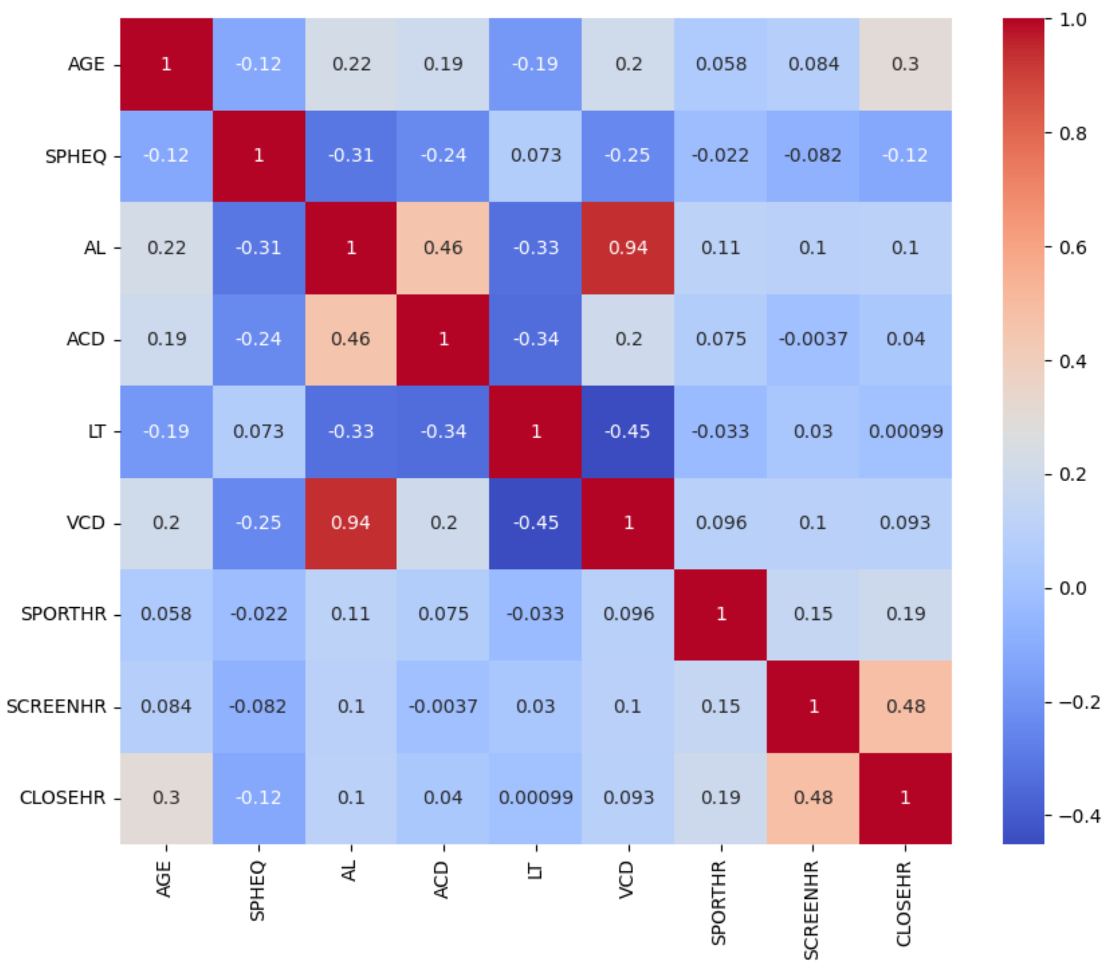
3.1.2 2.1.2. Quant values

[34]:
```
# Bad output display
#for col in myopianum.columns:
#    group0 = df.loc[df['MYOPIC'] == 0, col]
#    group1 = df.loc[df['MYOPIC'] == 1, col]
#    stat, p = stats.ttest_ind(group0, group1, nan_policy='omit')
#    fig = px.box(df, x='MYOPIC', y=col, color='MYOPIC', points="all",␣
↪title=col)
#    fig.show()
#    print(f"T-test {col}: statistic={stat:.2f}, p-value={p:.4f} \n-----")
```

Col	NonMyopic BoxPlot	Myopic Boxplot	Test-T	Conclusion
SPORTHR	q1 6, median 10, q3 16	q1 3, median 8, q3 15	p-value = 0.0145	Myopia ⇒ less sport
SPHEQ	q1 0.545, median 0.791, q3 1.097	q1 -0.0735, median 0.234, q3 0.507	p-value = 0.0000	Myopia ⇒ lower SPHEQ

Statistical Comparison of Myopic and Non-Myopic Groups

[24]:
```
corr = myopianum.corr()
plt.figure(figsize=(10,8))
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.show()
```



Col	Correlation
VCD - AL	Postive correlation
VCD - LT	Negative correlation
AL - LT	Negative correlation
ACD - LT	Negative correlation
SPHEQ-AL	Negative correlation
SPHEQ-ACD	Negative correlation
SPHEQ-VCD	Negative correlation

Correlation between all features of the dataset

3.2 Conclusion and Synthesis

- **Strongest Associations:**

  - The variable **SPHEQ** (spherical equivalent) shows a highly significant difference between myopic and non-myopic groups, making it the best discriminator for myopia in this dataset.
  - **ACD** (anterior chamber depth) also shows a significant difference between the groups.
- **Physical Activity:**
  - Myopic individuals tend to spend slightly less time practicing sports (**SPORTHR**). This difference is statistically significant but the effect size is modest.
- **Screen Time and Near-Work:**
  - No statistically significant differences between myopic and non-myopic individuals for screen time (**SCREENHR**) or near-work (**CLOSEHR**).
- **Correlations:**
  - **SPHEQ** shows strong correlation with biometric eye measures, particularly **AL** (axial length) and **VCD** (vitreous chamber depth).
  - Other activity-related or demographic variables show only weak or no correlation with myopia status.
- **Practical Implications:**
  - Promoting physical activity may provide some protective effect against myopia, but the impact is relatively small according to this dataset.
  - Screen time and near-work do not appear to be major factors here, though findings may vary with different populations and study designs.
  - Ocular biometric measures remain the strongest predictors or indicators for myopia diagnosis in the data.

4 3. Predicting Model - Simplest models

[25]:
```
df['MYOPIC'] = df['MYOPIC'].astype(int)
```

[26]:
```
x = df.drop(['MYOPIC'], axis=1)
x['GENDER'] = x['GENDER'].astype(int)
x['PARENTSMY'] = x['PARENTSMY'].astype(int)
y = df['MYOPIC'].astype(int)
```

[27]:
```
def eval_model(model, X_train, y_train, X_test, y_test, seuil=0.27,␣
↪name='model', cv=5):
    model.fit(X_train, y_train)
    y_pred_proba = model.predict_proba(X_test)[:, 1]
    y_pred_label = (y_pred_proba > seuil).astype(int)
    print(f"\n===== {name} =====")
    print("Accuracy:", accuracy_score(y_test, y_pred_label))
    print("AUC:", roc_auc_score(y_test, y_pred_proba))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_label))
    print(classification_report(y_test, y_pred_label))
    # ROC
    fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
    plt.plot(fpr, tpr, label=f"{name} (AUC={roc_auc_score(y_test, y_pred_proba):
↪.2f})")
```

```
    plt.plot([0, 1], [0, 1], 'k--', alpha=0.4)
    plt.xlabel('FPR')
    plt.ylabel('TPR')
    plt.title('ROC Curve')
    plt.legend()
    plt.show()
    # Cross-validated ROC-AUC
    cv_scores = cross_val_score(model, X_train, y_train, cv=cv,␣
 ↪scoring='roc_auc')
    print(f"Mean ROC-AUC (cross-validation): {np.mean(cv_scores):.3f}")

    ## Analysis
    test_results = X_test.copy()
    test_results['y_true'] = y_test
    test_results['y_pred'] = y_pred_label
    test_results['proba_pred'] = y_pred_proba
    return test_results, model

def analyse_erreurs(test_results):
    fn = test_results[(test_results['y_true'] == 1) & (test_results['y_pred']␣
 ↪== 0)]
    fp = test_results[(test_results['y_true'] == 0) & (test_results['y_pred']␣
 ↪== 1)]
    print("FALSE NEGATIVES (should have been detected!):")
    display(fn.head())
    print("FALSE POSITIVES (true non-myopics, false alarm):")
    display(fp.head())
    return fn, fp

def eval_by_group(X, y_true, y_pred, group_col):
    groups = X[group_col].unique()
    for grp in groups:
        idx = X[group_col] == grp
        print(f"\n--- {group_col} = {grp} ---")
        print(classification_report(y_true[idx], y_pred[idx]))
```

[28]:
```
def stats_descriptives(comparaison, features):
    stats = {}
    for nom, sous_groupe in comparaison.items():
        stats[nom] = sous_groupe[features].describe().T[["mean", "std", "min",␣
 ↪"25%", "50%", "75%", "max"]]
    return stats

def plot_boxplots(comparaison, features, nb_cols=3):
    n_features = len(features)
    n_rows = int(np.ceil(n_features / nb_cols))
    fig, axes = plt.subplots(n_rows, nb_cols, figsize=(nb_cols*5, n_rows*4))
    axes = axes.flatten()
    for i, feature in enumerate(features):
        sns.boxplot(
            data=pd.concat([df.assign(Groupe=nom) for nom, df in comparaison.
 ↪items()]),
            x="Groupe", y=feature, ax=axes[i], showmeans=True)
        axes[i].set_title(feature)
    for j in range(i + 1, len(axes)):
        fig.delaxes(axes[j])
    plt.tight_layout()
    plt.show()

def plot_parallel_coordinates(selected, features, y_col="true_label", title=""):
    temp = selected[features + [y_col]].copy()
    temp[y_col] = temp[y_col].astype(str)
    plt.figure(figsize=(12, 5))
    pd.plotting.parallel_coordinates(temp, y_col, colormap=plt.cm.Set1)
    plt.title(title)
    plt.show()

def analyse_false (tn, tp):
    comparaison_FN = {
        'False Negatives': fn_lr,
        'True Positives': tp
    }
    comparaison_FP = {
        'False Positives': fp_lr,
        'True Negatives': tn
    }
    print("\nStatistiques descriptives - FN vs TP")
    display(stats_descriptives(comparaison_FN, features)['False Negatives'])
    display(stats_descriptives(comparaison_FN, features)['True Positives'])

    print("\nStatistiques descriptives - FP vs TN")
    display(stats_descriptives(comparaison_FP, features)['False Positives'])
    display(stats_descriptives(comparaison_FP, features)['True Negatives'])

    print("\nComparaison visuelle FN/TP")
    plot_boxplots(comparaison_FN, features)
    print("\nComparaison visuelle FP/TN")
    plot_boxplots(comparaison_FP, features)

    fn_lr['true_label'] = 'False Negative'
    tp['true_label'] = 'True Positive'
    fp_lr['true_label'] = 'False Positive'
    tn['true_label'] = 'True Negative'
    print("FN/TP Parallèles")
    plot_parallel_coordinates(pd.concat([fn_lr, tp]), features,␣
 ↪y_col="true_label", title="FN vs TP")
    print("FP/TN Parallèles")
    plot_parallel_coordinates(pd.concat([fp_lr, tn]), features,␣
 ↪y_col="true_label", title="FP vs TN")
```

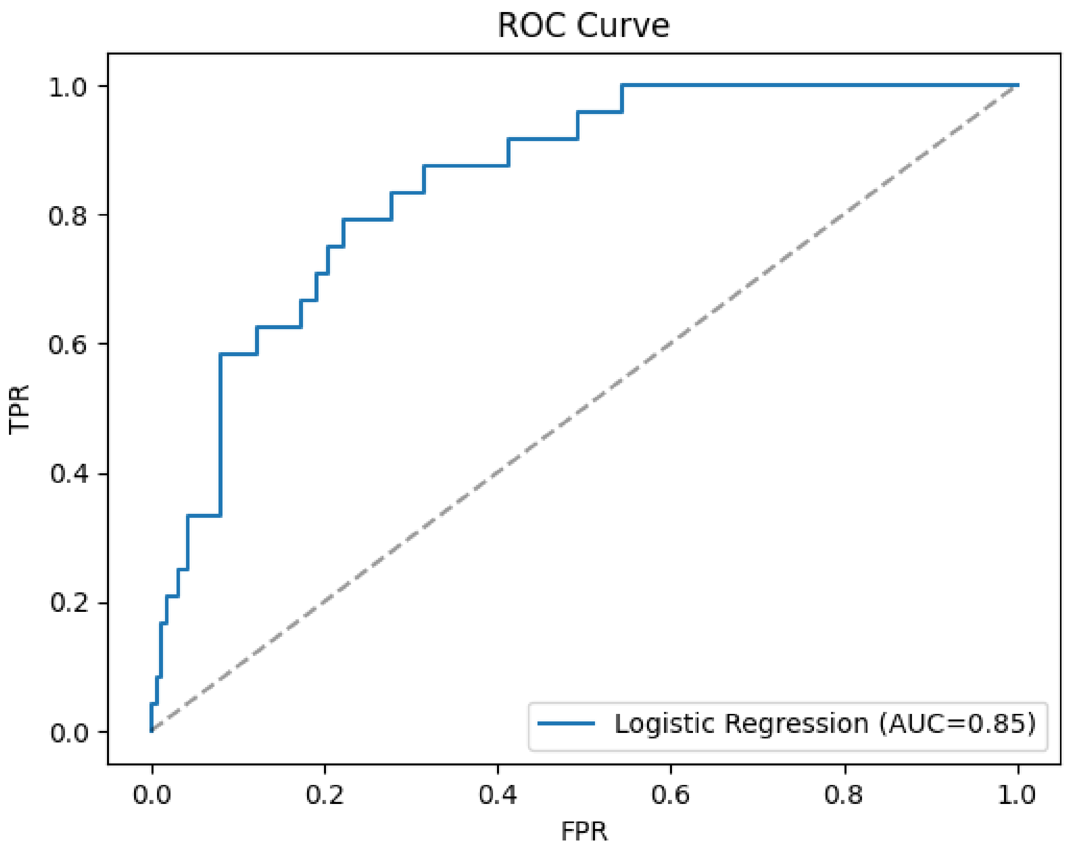
[29]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3,␣
 ↪random_state=42, stratify=y)
```

[30]:
```
features = ["SPHEQ", "SPORTHR", "PARENTSMY", "ACD", "AL"]
```

4.0.1 3.0.1. Logistic Regression

[31]:
```
# Logistic Regression
print("="*30, 'Logistic Regression', "="*30)

lr = LogisticRegression(solver='liblinear', max_iter=10000,␣
 ↪class_weight='balanced')
test_results_lr, model_lr = eval_model(
    lr, X_train, y_train, X_test, y_test, name='Logistic Regression', seuil=0.31
)

fn_lr, fp_lr = analyse_erreurs(test_results_lr)
eval_by_group(X_test, test_results_lr['y_true'], test_results_lr['y_pred'],␣
 ↪group_col='PARENTSMY')
tn = test_results_lr[(test_results_lr['y_true']==0) &␣
 ↪(test_results_lr['y_pred']==0)]
tp = test_results_lr[(test_results_lr['y_true']==1) &␣
 ↪(test_results_lr['y_pred']==1)]
analyse_false(tn, tp)

# Feature importance
plt.figure(figsize=(8, 5))
coefs = pd.Series(model_lr.coef_[0], index=X_train.columns)

coefs_sorted = coefs.abs().sort_values()

explainer = shap.Explainer(model_lr, X_train)
shap_values = explainer(X_test)

# Affichage (beeswarm ou bar : importance feature, etc.)
shap.summary_plot(shap_values, X_test, plot_type="bar")
shap.summary_plot(shap_values, X_test)  # beeswarm
```

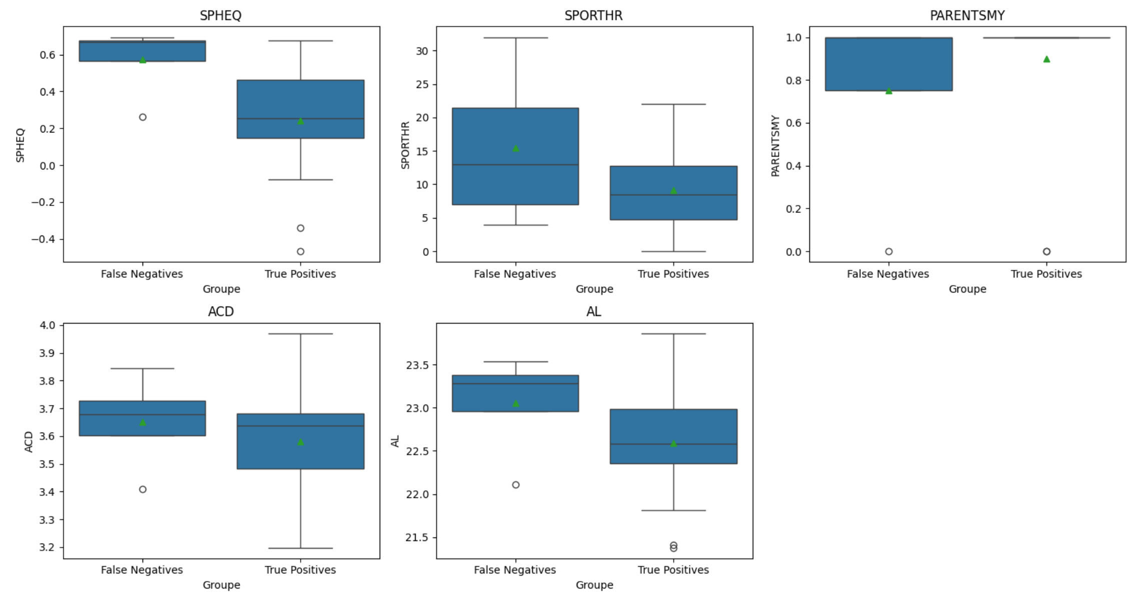
```
============================== Logistic Regression
==============================
===== Logistic Regression =====
Accuracy: 0.7311827956989247
AUC: 0.8497942386831275
Confusion Matrix:
 [[116  46]
 [  4  20]]
              precision    recall  f1-score   support

           0       0.97      0.72      0.82       162
           1       0.30      0.83      0.44        24

    accuracy                           0.73       186
   macro avg       0.63      0.77      0.63       186
weighted avg       0.88      0.73      0.77       186
```



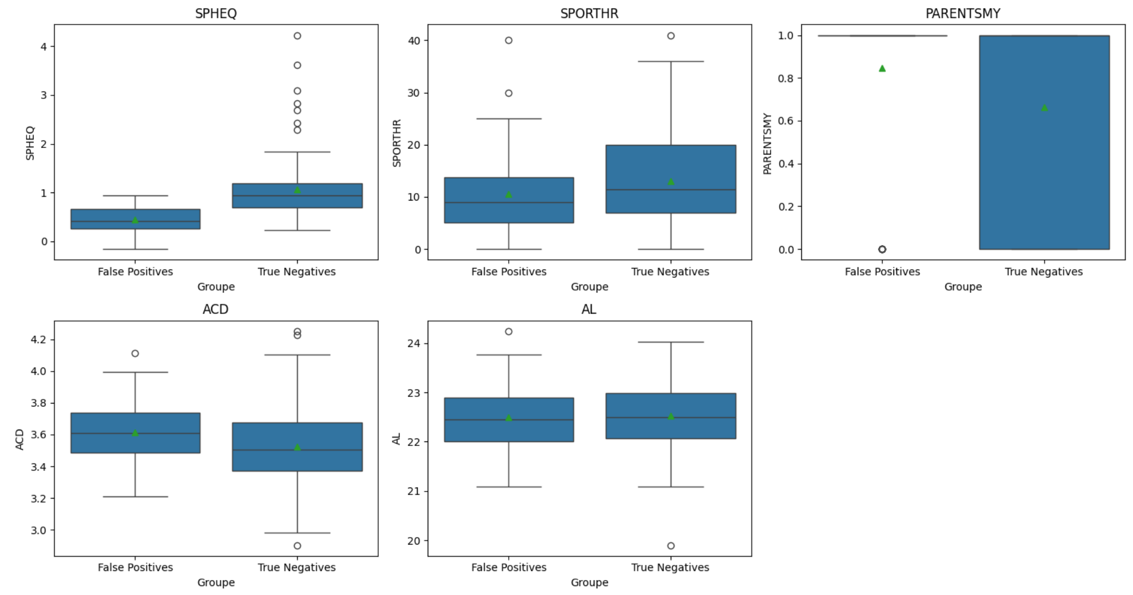
```
Mean ROC-AUC (cross-validation): 0.884
FALSE NEGATIVES (should have been detected!):

     AGE  GENDER  SPHEQ     AL    ACD     LT    VCD  SPORTHR  PARENTSMY  \
77     6       0  0.665  23.24  3.690  3.498  16.05        8          1
```

```
215    6       1  0.695  23.54  3.845  3.403  16.30        4          0
281    7       0  0.261  23.32  3.665  3.388  16.26       32          1
369    6       1  0.668  22.11  3.410  3.570  15.13       18          1

     SCREENHR  CLOSEHR  y_true  y_pred  proba_pred
77         20       78       1       0    0.249340
215        22       31       1       0    0.198744
281        18       92       1       0    0.180016
369        16       32       1       0    0.299161
FALSE POSITIVES (true non-myopics, false alarm):
     AGE  GENDER  SPHEQ     AL    ACD     LT    VCD  SPORTHR  PARENTSMY  \
98     6       1  0.290  23.50  3.786  3.584  16.13        7          1
59     6       1  0.596  22.45  3.488  3.710  15.25        5          1
375    6       0  0.519  22.35  3.902  3.468  14.98       21          1
355    6       0  0.500  22.64  3.532  3.498  15.61        9          1
535    6       1  0.378  21.83  3.464  3.896  14.47        8          1

     SCREENHR  CLOSEHR  y_true  y_pred  proba_pred
98         16       34       0       1    0.796691
59         10       22       0       1    0.596455
375        10       41       0       1    0.418594
355        14       34       0       1    0.431677
535         8       48       0       1    0.728417


--- PARENTSMY = 1 ---
              precision    recall  f1-score   support

           0       0.96      0.66      0.79       116
           1       0.32      0.86      0.46        21

    accuracy                           0.69       137
   macro avg       0.64      0.76      0.62       137
weighted avg       0.86      0.69      0.74       137


--- PARENTSMY = 0 ---
              precision    recall  f1-score   support

           0       0.97      0.85      0.91        46
           1       0.22      0.67      0.33         3

    accuracy                           0.84        49
   macro avg       0.60      0.76      0.62        49
weighted avg       0.93      0.84      0.87        49
```

```
Statistiques descriptives - FN vs TP

              mean        std     min       25%      50%       75%     max
SPHEQ      0.57225   0.207938   0.261   0.56400   0.6665   0.67475   0.695
SPORTHR   15.50000  12.476645   4.000   7.00000  13.0000  21.50000  32.000
PARENTSMY  0.75000   0.500000   0.000   0.75000   1.0000   1.00000   1.000
ACD        3.65250   0.180208   3.410   3.60125   3.6775   3.72875   3.845
AL        23.05250   0.641008  22.110  22.95750  23.2800  23.37500  23.540

              mean       std     min       25%      50%      75%     max
SPHEQ      0.24080  0.300165  -0.467   0.14775   0.2535   0.4650   0.677
SPORTHR    9.15000  6.343459   0.000   4.75000   8.5000  12.7500  22.000
PARENTSMY  0.90000  0.307794   0.000   1.00000   1.0000   1.0000   1.000
ACD        3.58185  0.190826   3.198   3.48150   3.6370   3.6825   3.970
AL        22.58900  0.634996  21.380  22.36000  22.5850  22.9825  23.860


Statistiques descriptives - FP vs TN

               mean       std     min       25%      50%       75%     max
SPHEQ      0.440435  0.266337  -0.158   0.26525   0.4175   0.66725   0.944
SPORTHR   10.565217  8.344398   0.000   5.00000   9.0000  13.75000  40.000
PARENTSMY  0.847826  0.363158   0.000   1.00000   1.0000   1.00000   1.000
ACD        3.612739  0.210382   3.210   3.48800   3.6110   3.73600   4.114
AL        22.495217  0.684835  21.080  22.01000  22.4450  22.89750  24.240

               mean       std     min       25%     50%     75%     max
SPHEQ      1.061845  0.621170   0.231   0.69450   0.934   1.190   4.228
SPORTHR   13.051724  7.957326   0.000   7.00000  11.500  20.000  41.000
PARENTSMY  0.663793  0.474460   0.000   0.00000   1.000   1.000   1.000
ACD        3.524931  0.248699   2.904   3.37225   3.504   3.676   4.250
AL        22.520431  0.705830  19.900  22.06000  22.495  22.990  24.030


Comparaison visuelle FN/TP
```



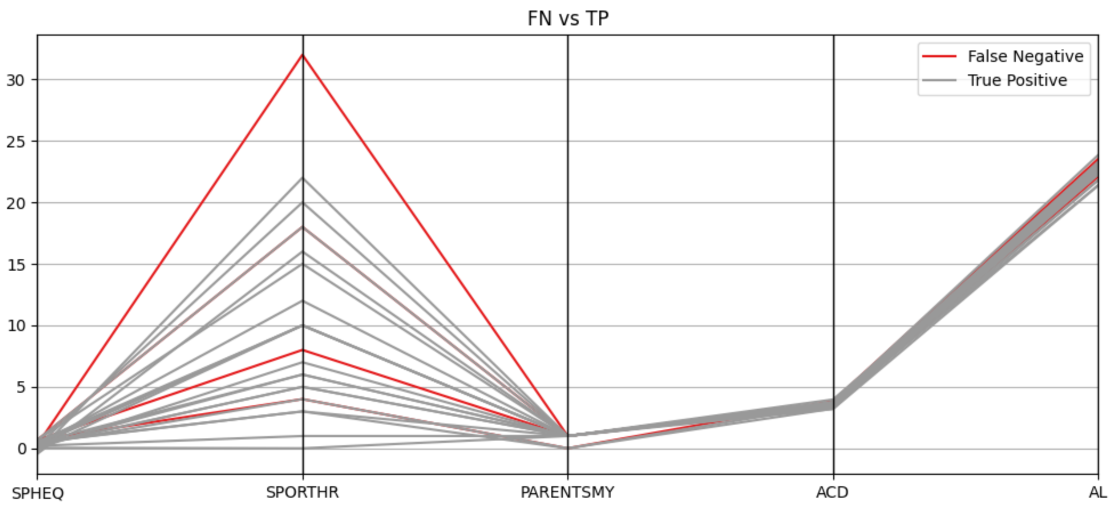
Comparaison visuelle FP/TN



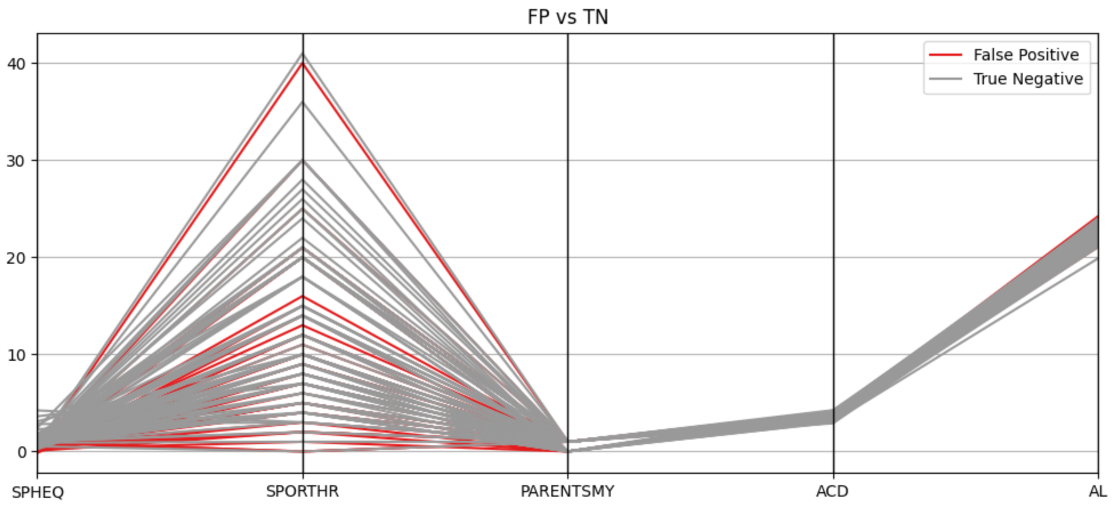
FN/TP Parallèles

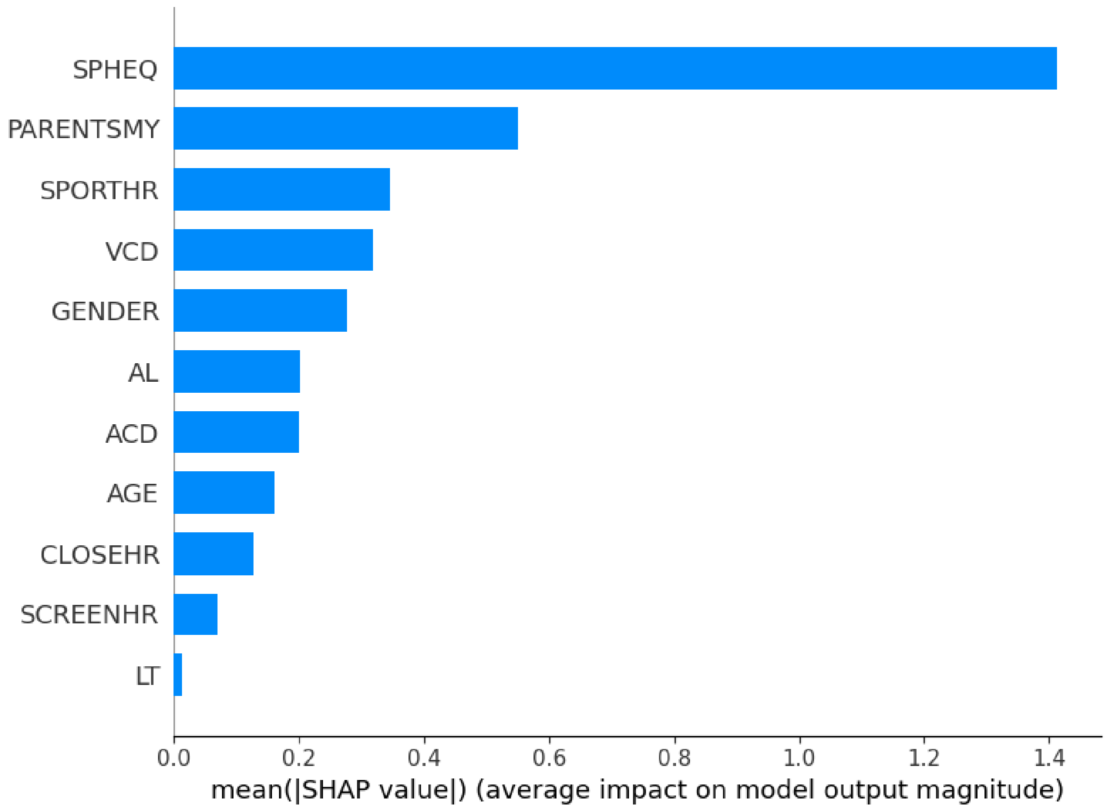
```
/var/folders/13/07j4wbfd4613yv4ymtvk0_b00000gn/T/ipykernel_50456/3293880108.py:5
2: SettingWithCopyWarning:
```

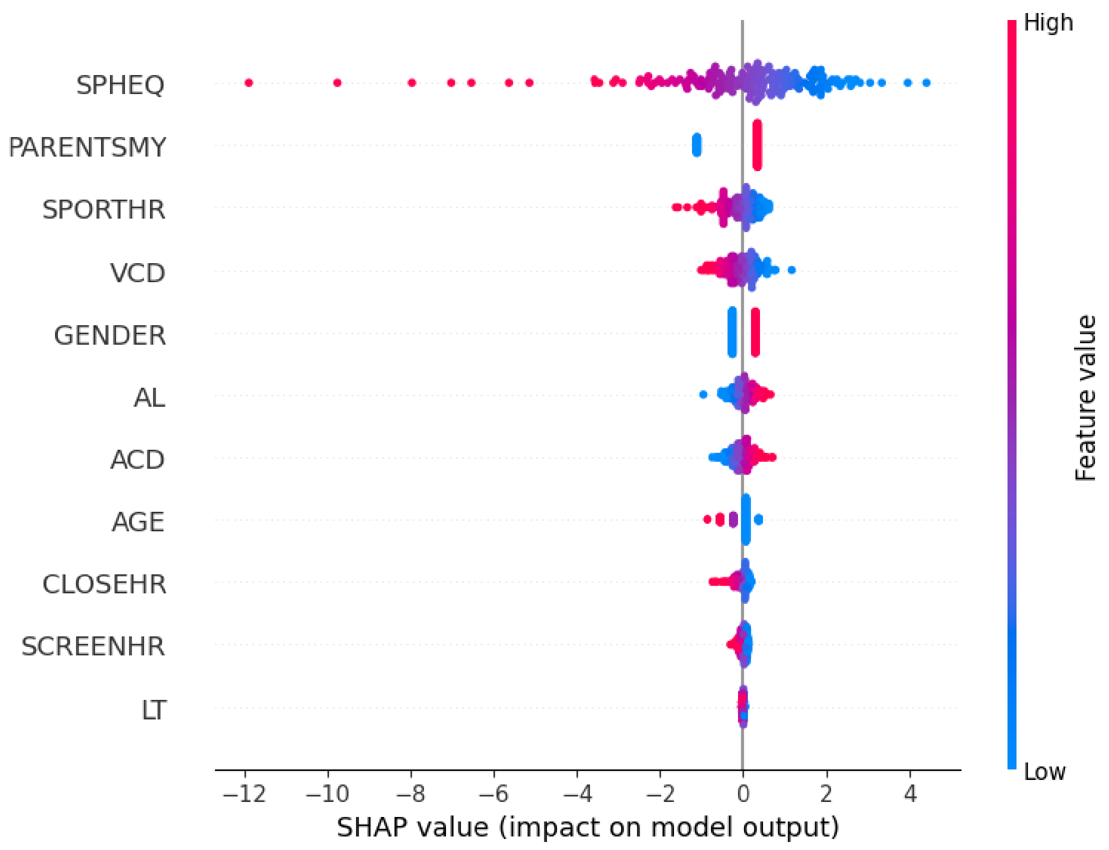
```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/13/07j4wbfd4613yv4ymtvk0_b00000gn/T/ipykernel_50456/3293880108.py:5
3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/13/07j4wbfd4613yv4ymtvk0_b00000gn/T/ipykernel_50456/3293880108.py:5
4: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/13/07j4wbfd4613yv4ymtvk0_b00000gn/T/ipykernel_50456/3293880108.py:5
5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```



FP/TN Parallèles




SPHEQ
PARENTSMY
SPORTHR
VCD
GENDER
AL
ACD
AGE
CLOSEHR
SCREENHR
LT
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
mean(|SHAP value|) (average impact on model output magnitude)

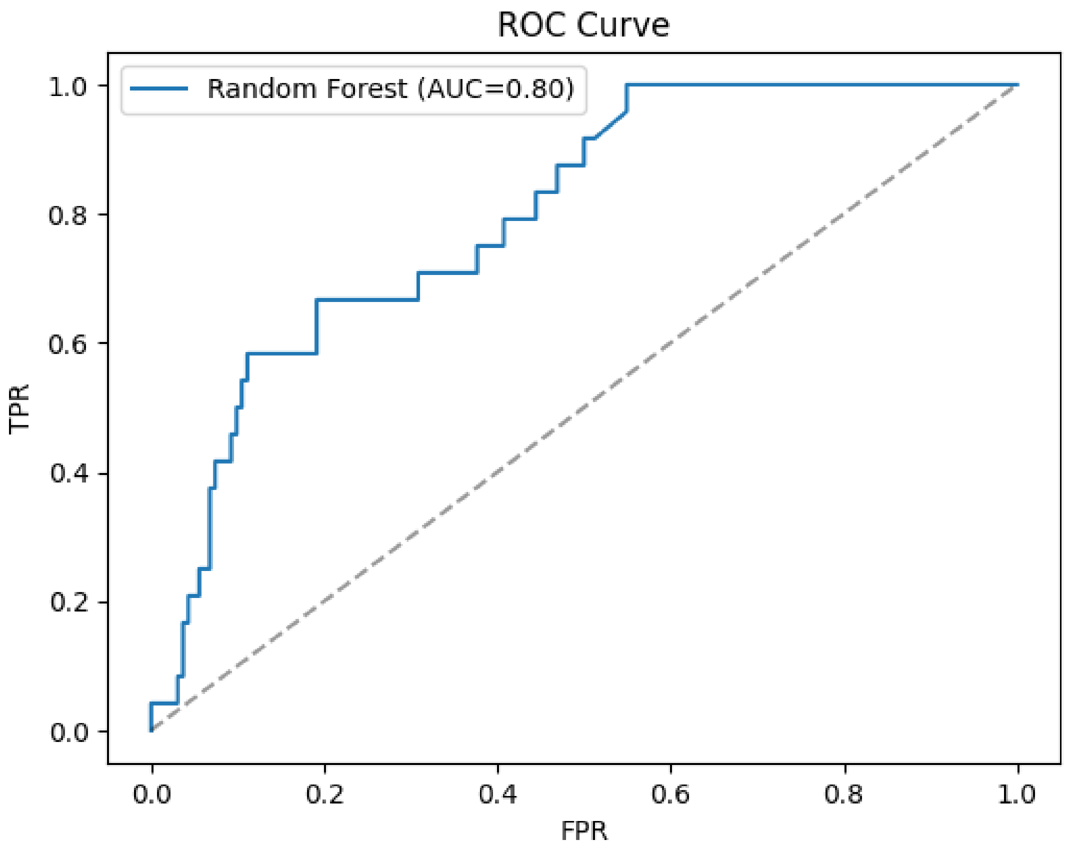


4.0.2 3.0.2. Random Forest

[32]:
```
# Random Forest
print("="*30, 'Random Forest', "="*30)
rf = RandomForestClassifier(n_estimators=100, class_weight='balanced',␣
 ↪max_depth=10)
test_results_rf, model_rf = eval_model(
    rf, X_train, y_train, X_test, y_test, name='Random Forest', seuil=0.5
)

fn_rf, fp_rf = analyse_erreurs(test_results_rf)
eval_by_group(X_test, test_results_rf['y_true'], test_results_rf['y_pred'],␣
 ↪group_col='PARENTSMY')
tn = test_results_rf[(test_results_rf['y_true']==0) &␣
 ↪(test_results_rf['y_pred']==0)]
tp = test_results_rf[(test_results_rf['y_true']==1) &␣
 ↪(test_results_rf['y_pred']==1)]
analyse_false(tn, tp)

# Feature importance
```

```python
plt.figure(figsize=(8, 5))
feat_imp = pd.Series(model_rf.feature_importances_, index=X_train.columns)
feat_imp.sort_values(ascending=True).plot(kind='barh')
plt.title("Var importances (Random Forest)")
plt.show()

# SHAP VALUES
import shap
explainer = shap.TreeExplainer(model_rf)
shap_values = explainer.shap_values(X_test)

print("shap_values type:", type(shap_values))
if isinstance(shap_values, list):
    print("shape[0]:", np.array(shap_values[0]).shape)
    if len(shap_values) > 1:
        print("shape[1]:", np.array(shap_values[1]).shape)
else:
    print("shap_values:", np.array(shap_values).shape)
print("X_test:", X_test.shape)

if isinstance(shap_values, list) and len(shap_values) == 2 and np.
 ↪array(shap_values[1]).shape == X_test.shape:
    shap.summary_plot(shap_values[1], X_test, plot_type="bar")
else:
    shap.summary_plot(shap_values, X_test, plot_type="bar")
```

```
============================== Random Forest ==============================

===== Random Forest =====
Accuracy: 0.8548387096774194
AUC: 0.7975823045267489
Confusion Matrix:
 [[158   4]
 [ 23   1]]
              precision    recall  f1-score   support

           0       0.87      0.98      0.92       162
           1       0.20      0.04      0.07        24

    accuracy                           0.85       186
   macro avg       0.54      0.51      0.50       186
weighted avg       0.79      0.85      0.81       186
```

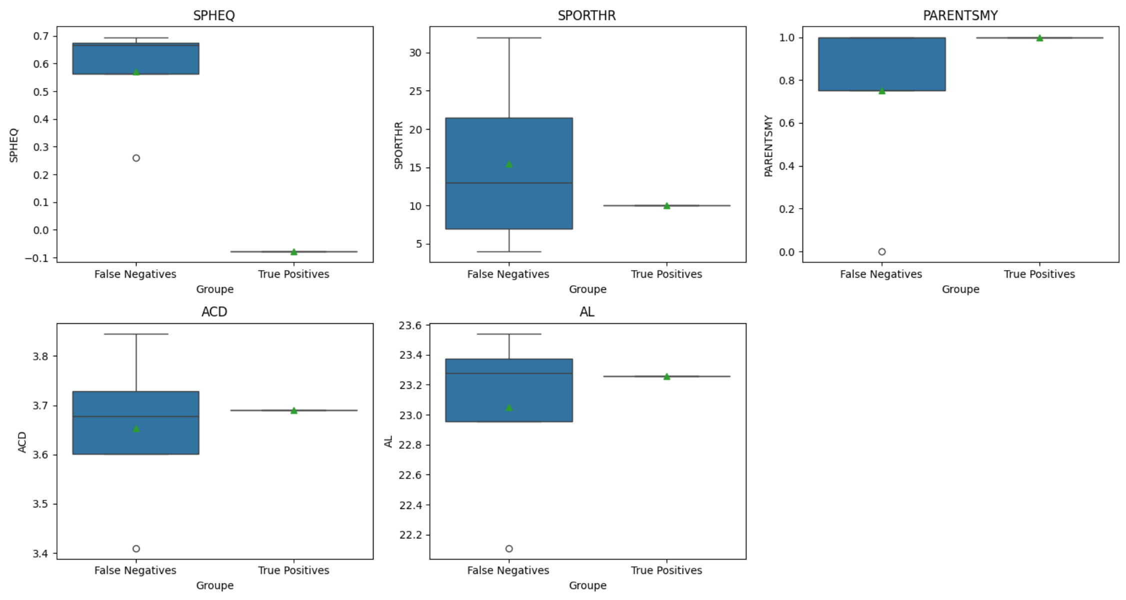


```
Mean ROC-AUC (cross-validation): 0.851
FALSE NEGATIVES (should have been detected!):

     AGE  GENDER  SPHEQ     AL    ACD     LT    VCD  SPORTHR  PARENTSMY  \
493    6       1  0.477  21.41  3.530  3.822  14.06        3          0
172    8       1  0.461  23.24  3.636  3.598  16.01        6          1
579    6       0  0.246  22.56  3.970  3.452  15.14        5          1
188    6       1  0.183  22.53  3.638  3.498  15.40       20          1
77     6       0  0.665  23.24  3.690  3.498  16.05        8          1

     SCREENHR  CLOSEHR  y_true  y_pred  proba_pred
493        12       24       1       0    0.069294
172        11       83       1       0    0.087966
579         8       56       1       0    0.485459
188         6       19       1       0    0.377968
77         20       78       1       0    0.079385

FALSE POSITIVES (true non-myopics, false alarm):

     AGE  GENDER  SPHEQ     AL    ACD     LT    VCD  SPORTHR  PARENTSMY  \
447    6       1  0.058  21.86  3.476  3.378  15.01       12          1
485    6       1 -0.158  22.55  3.434  3.654  15.46       12          1
```

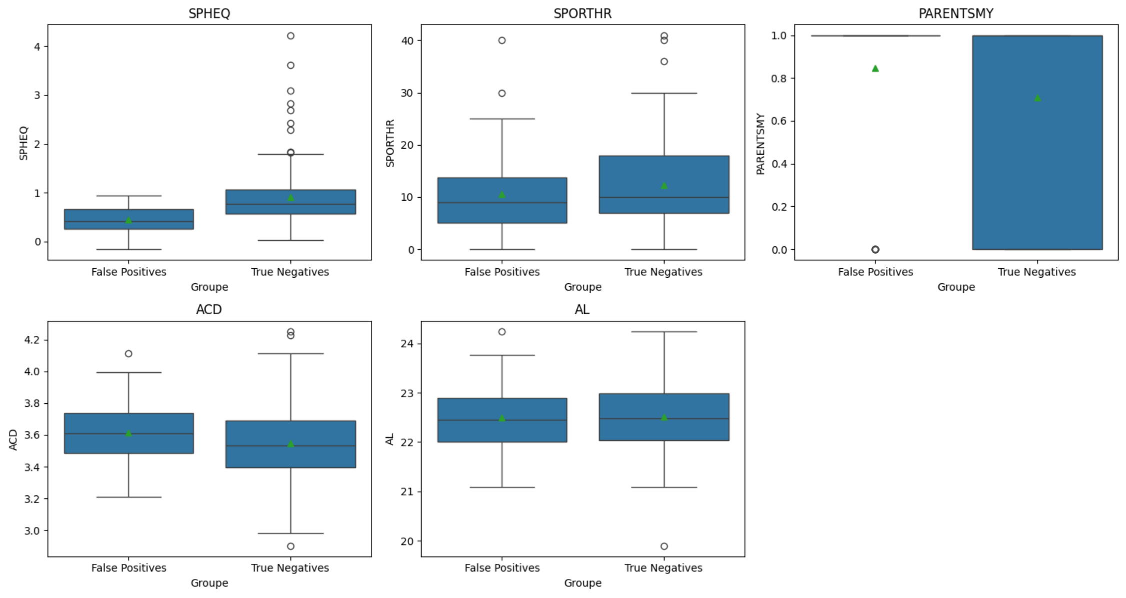
```
22     6       0  0.329  22.16  3.704  3.696  14.76       10          1
394    5       0  0.153  22.82  3.848  3.598  15.37       15          1

     SCREENHR  CLOSEHR  y_true  y_pred  proba_pred
447        10       30       0       1    0.583451
485        13       18       0       1    0.510401
22          6       22       0       1    0.532694
394         9       23       0       1    0.510916


--- PARENTSMY = 1 ---
              precision    recall  f1-score   support

           0       0.85      0.97      0.90       116
           1       0.20      0.05      0.08        21

    accuracy                           0.82       137
   macro avg       0.52      0.51      0.49       137
weighted avg       0.75      0.82      0.78       137


--- PARENTSMY = 0 ---
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        46
           1       0.00      0.00      0.00         3

    accuracy                           0.94        49
   macro avg       0.47      0.50      0.48        49
weighted avg       0.88      0.94      0.91        49


Statistiques descriptives - FN vs TP
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:

Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:

Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
```

```
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.


               mean        std     min        25%       50%        75%     max
SPHEQ       0.57225   0.207938   0.261    0.56400    0.6665    0.67475   0.695
SPORTHR    15.50000  12.476645   4.000    7.00000   13.0000   21.50000  32.000
PARENTSMY   0.75000   0.500000   0.000    0.75000    1.0000    1.00000   1.000
ACD         3.65250   0.180208   3.410    3.60125    3.6775    3.72875   3.845
AL         23.05250   0.641008  22.110   22.95750   23.2800   23.37500  23.540
             mean  std     min     25%     50%     75%     max
SPHEQ      -0.078  NaN  -0.078  -0.078  -0.078  -0.078  -0.078
SPORTHR    10.000  NaN  10.000  10.000  10.000  10.000  10.000
PARENTSMY   1.000  NaN   1.000   1.000   1.000   1.000   1.000
ACD         3.690  NaN   3.690   3.690   3.690   3.690   3.690
AL         23.260  NaN  23.260  23.260  23.260  23.260  23.260


Statistiques descriptives - FP vs TN
                mean       std     min        25%       50%        75%     max
SPHEQ       0.440435  0.266337  -0.158    0.26525    0.4175    0.66725   0.944
SPORTHR    10.565217  8.344398   0.000    5.00000    9.0000   13.75000  40.000
PARENTSMY   0.847826  0.363158   0.000    1.00000    1.0000    1.00000   1.000
ACD         3.612739  0.210382   3.210    3.48800    3.6110    3.73600   4.114
AL         22.495217  0.684835  21.080   22.01000   22.4450   22.89750  24.240
                mean       std     min     25%      50%       75%     max
SPHEQ       0.905392  0.605715   0.024   0.571   0.7745   1.06525   4.228
SPORTHR    12.348101  8.219076   0.000   7.000  10.0000  18.00000  41.000
PARENTSMY   0.708861  0.455732   0.000   0.000   1.0000   1.00000   1.000
ACD         3.548203  0.242422   2.904   3.396   3.5310   3.68800   4.250
AL         22.517468  0.703805  19.900  22.040  22.4750  22.98000  24.240


Comparaison visuelle FN/TP
```

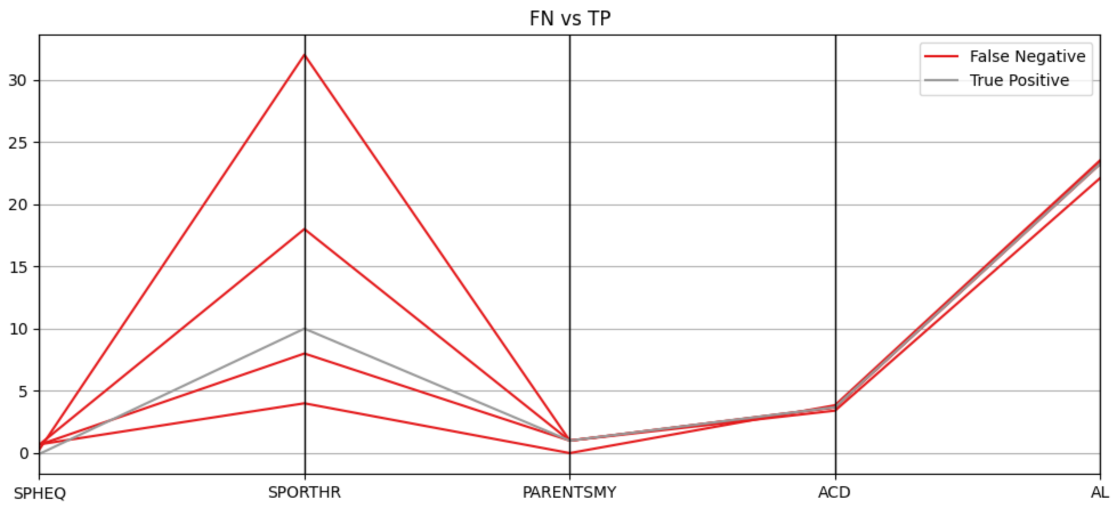


```
Comparaison visuelle FP/TN
```

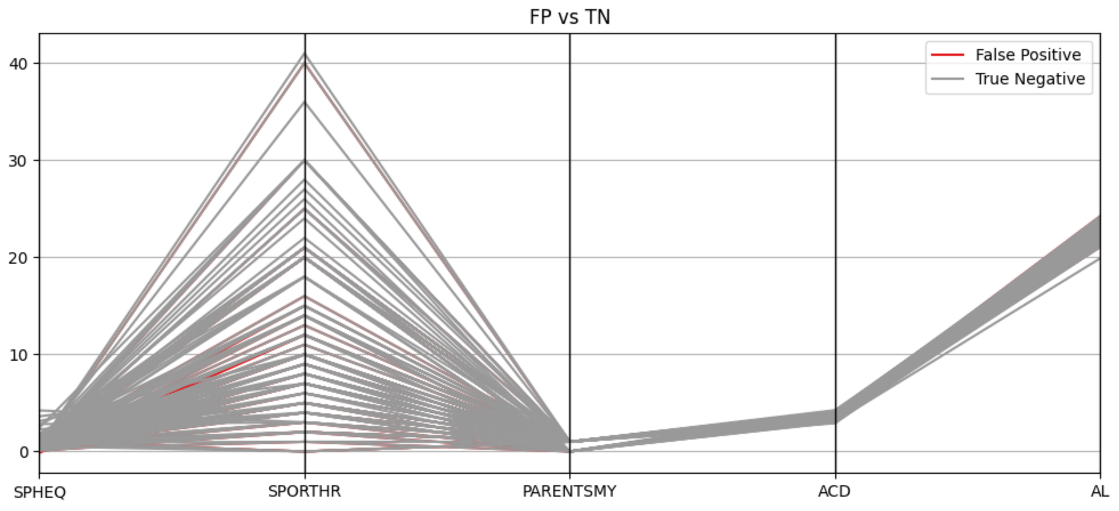


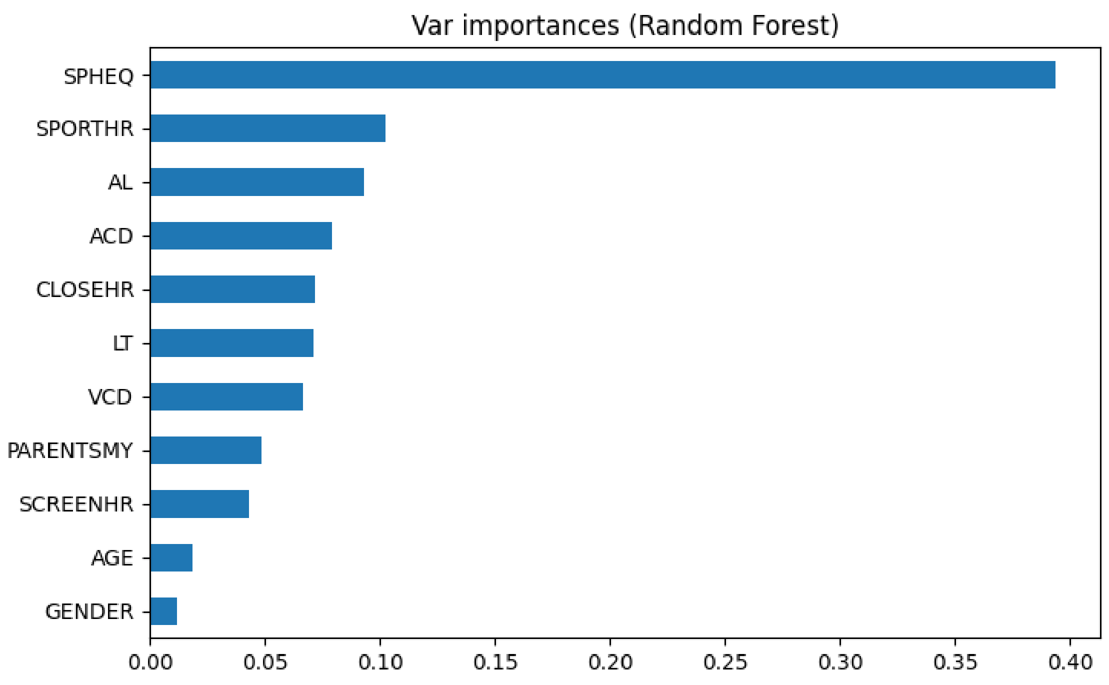
```
FN/TP Parallèles

/var/folders/13/07j4wbfd4613yv4ymtvk0_b00000gn/T/ipykernel_50456/3293880108.py:5
2: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/13/07j4wbfd4613yv4ymtvk0_b00000gn/T/ipykernel_50456/3293880108.py:5
3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/13/07j4wbfd4613yv4ymtvk0_b00000gn/T/ipykernel_50456/3293880108.py:5
4: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/13/07j4wbfd4613yv4ymtvk0_b00000gn/T/ipykernel_50456/3293880108.py:5
5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```



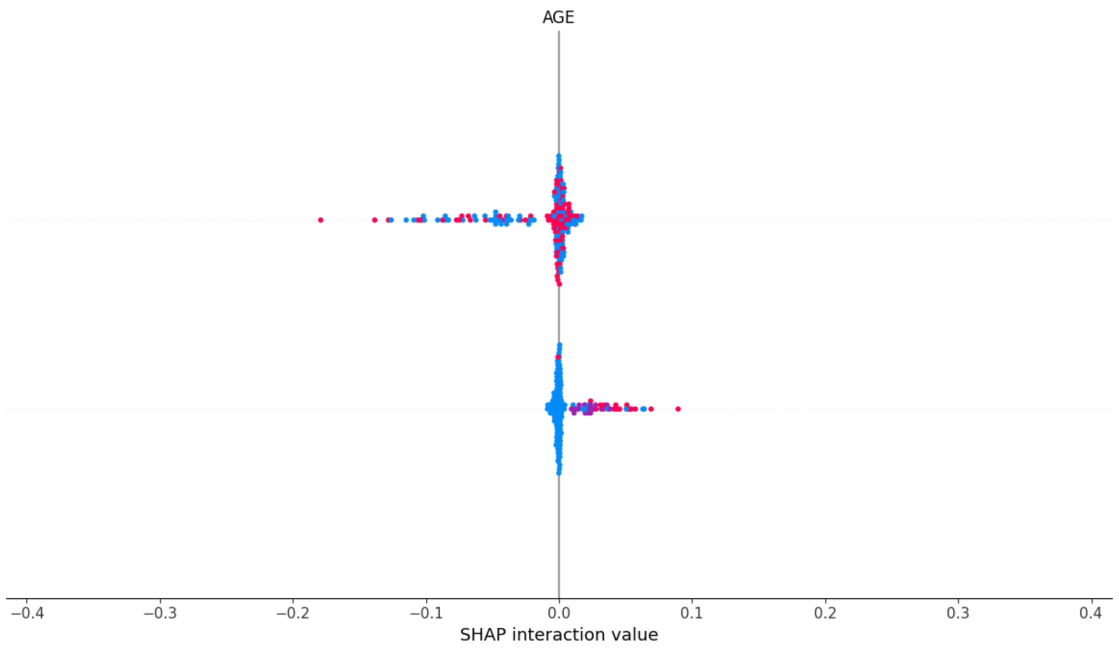
FP/TN Parallèles





```
shap_values type: <class 'numpy.ndarray'>
shap_values: (186, 11, 2)
X_test: (186, 11)
```

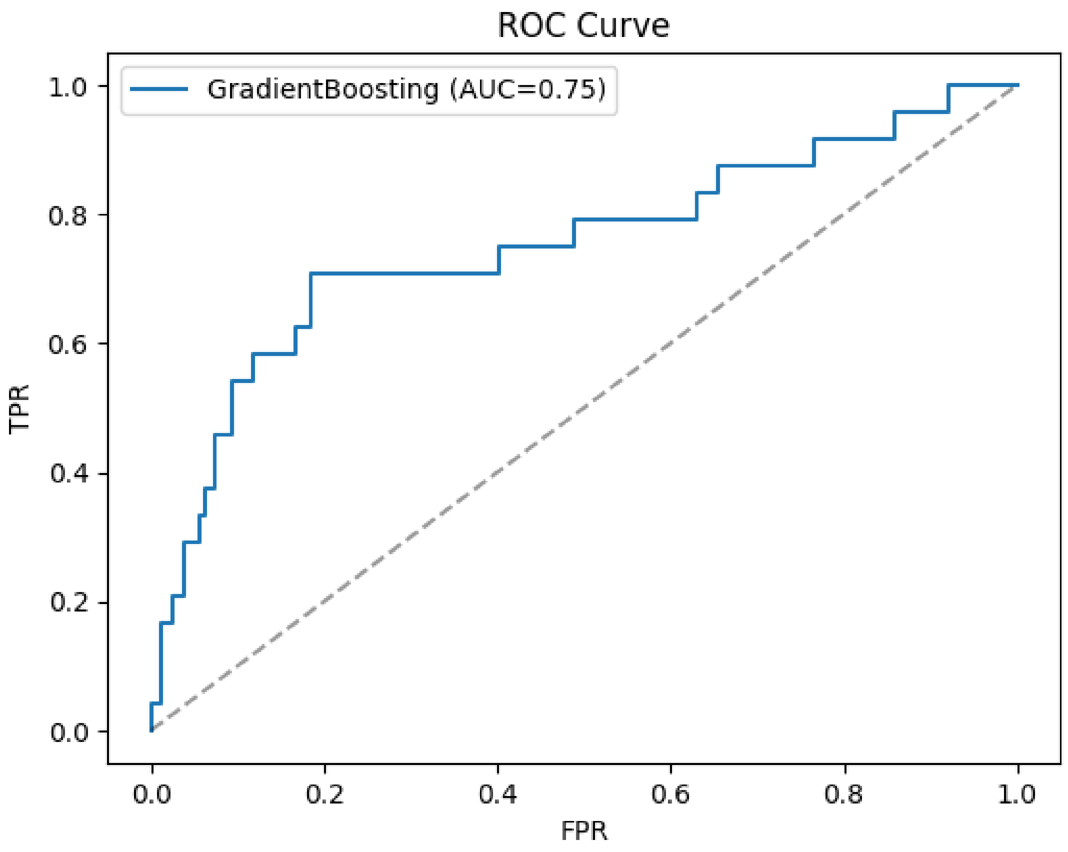
```
<Figure size 640x480 with 0 Axes>
```



4.0.3 3.0.3. Gradient Boost

[33]:
```
# GradientBoosting
print("="*30, 'GradientBoosting', "="*30)
hgb = HistGradientBoostingClassifier(class_weight='balanced', max_iter=100)
test_results_hgb, model_hgb = eval_model(
    hgb, X_train, y_train, X_test, y_test, name='GradientBoosting', seuil=0.5
)

fn_hgb, fp_hgb = analyse_erreurs(test_results_hgb)
eval_by_group(X_test, test_results_hgb['y_true'], test_results_hgb['y_pred'],␣
 ↪group_col='PARENTSMY')
tn = test_results_hgb[(test_results_hgb['y_true']==0) &␣
 ↪(test_results_hgb['y_pred']==0)]
tp = test_results_hgb[(test_results_hgb['y_true']==1) &␣
 ↪(test_results_hgb['y_pred']==1)]
analyse_false(tn, tp)

# Feature importance
plt.figure(figsize=(8, 5))
result = permutation_importance(model_hgb, X_train, y_train, n_repeats=10,␣
 ↪random_state=42, n_jobs=-1)

# SHAP VALUES
explainer = shap.TreeExplainer(model_hgb)
shap_values = explainer.shap_values(X_test)

print("shap_values type:", type(shap_values))
if isinstance(shap_values, list):
    print("shape[0]:", np.array(shap_values[0]).shape)
    if len(shap_values) > 1:
        print("shape[1]:", np.array(shap_values[1]).shape)
else:
    print("shap_values:", np.array(shap_values).shape)
print("X_test:", X_test.shape)

if isinstance(shap_values, list) and len(shap_values) == 2 and np.
 ↪array(shap_values[1]).shape == X_test.shape:
    shap.summary_plot(shap_values[1], X_test, plot_type="bar")
    shap.summary_plot(shap_values[1], X_test)
else:
    # Certains cas (classification One-vs-Rest, régression, etc.)
    shap.summary_plot(shap_values, X_test, plot_type="bar")
    shap.summary_plot(shap_values, X_test)
```

```
============================== GradientBoosting ==============================
===== GradientBoosting =====
Accuracy: 0.8655913978494624
AUC: 0.7518004115226338
Confusion Matrix:
 [[154   8]
 [ 17   7]]
              precision    recall  f1-score   support

           0       0.90      0.95      0.92       162
           1       0.47      0.29      0.36        24

    accuracy                           0.87       186
   macro avg       0.68      0.62      0.64       186
weighted avg       0.84      0.87      0.85       186
```

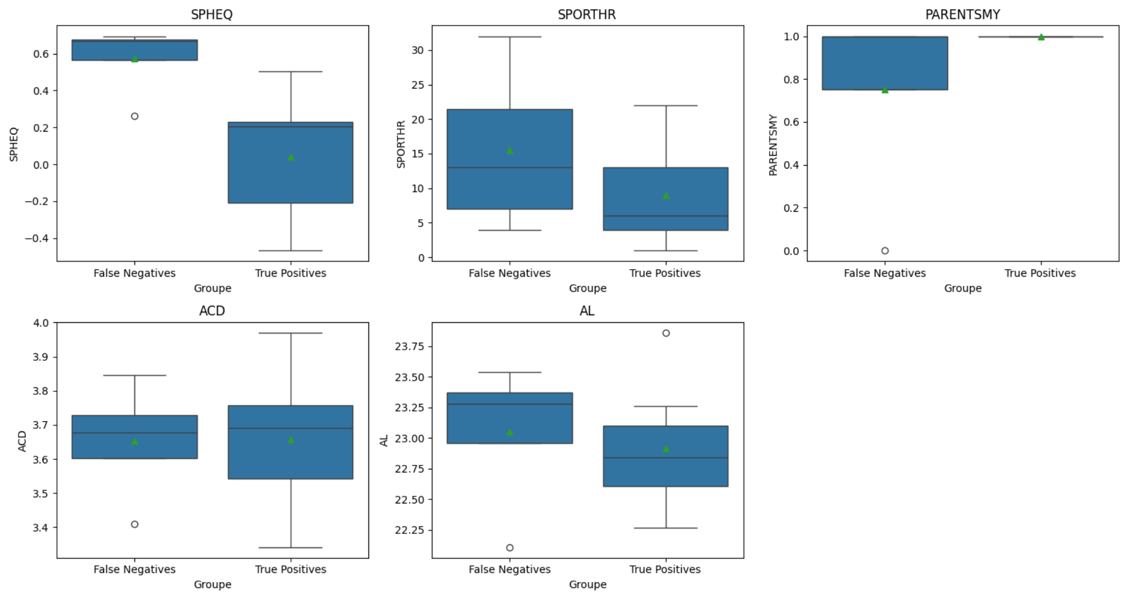


```
Mean ROC-AUC (cross-validation): 0.819
FALSE NEGATIVES (should have been detected!):

     AGE  GENDER  SPHEQ     AL    ACD     LT    VCD  SPORTHR  PARENTSMY  \
493    6       1  0.477  21.41  3.530  3.822  14.06        3          0
172    8       1  0.461  23.24  3.636  3.598  16.01        6          1
188    6       1  0.183  22.53  3.638  3.498  15.40       20          1
77     6       0  0.665  23.24  3.690  3.498  16.05        8          1
558    7       0  0.248  22.39  3.665  3.333  15.40       10          1

     SCREENHR  CLOSEHR  y_true  y_pred  proba_pred
493        12       24       1       0    0.000257
172        11       83       1       0    0.001101
188         6       19       1       0    0.483317
77         20       78       1       0    0.005794
558         8       17       1       0    0.062198
FALSE POSITIVES (true non-myopics, false alarm):
     AGE  GENDER  SPHEQ     AL    ACD     LT    VCD  SPORTHR  PARENTSMY  \
355    6       0  0.500  22.64  3.532  3.498  15.61        9          1
50     5       0  0.265  21.98  3.532  3.466  14.98        6          1
246    6       1  0.569  22.91  3.662  3.478  15.77       16          1
331    6       1  0.308  22.86  3.612  3.468  15.78       10          1
321    6       1  0.503  22.40  3.676  3.726  15.00        5          1

     SCREENHR  CLOSEHR  y_true  y_pred  proba_pred
355        14       34       0       1    0.748273
50          3       12       0       1    0.522698
246         6       22       0       1    0.745671
331         6       21       0       1    0.866190
321         5       45       0       1    0.606872


--- PARENTSMY = 1 ---
              precision    recall  f1-score   support

           0       0.89      0.93      0.91       116
           1       0.47      0.33      0.39        21

    accuracy                           0.84       137
   macro avg       0.68      0.63      0.65       137
weighted avg       0.82      0.84      0.83       137


--- PARENTSMY = 0 ---
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        46
           1       0.00      0.00      0.00         3

    accuracy                           0.94        49
   macro avg       0.47      0.50      0.48        49
```

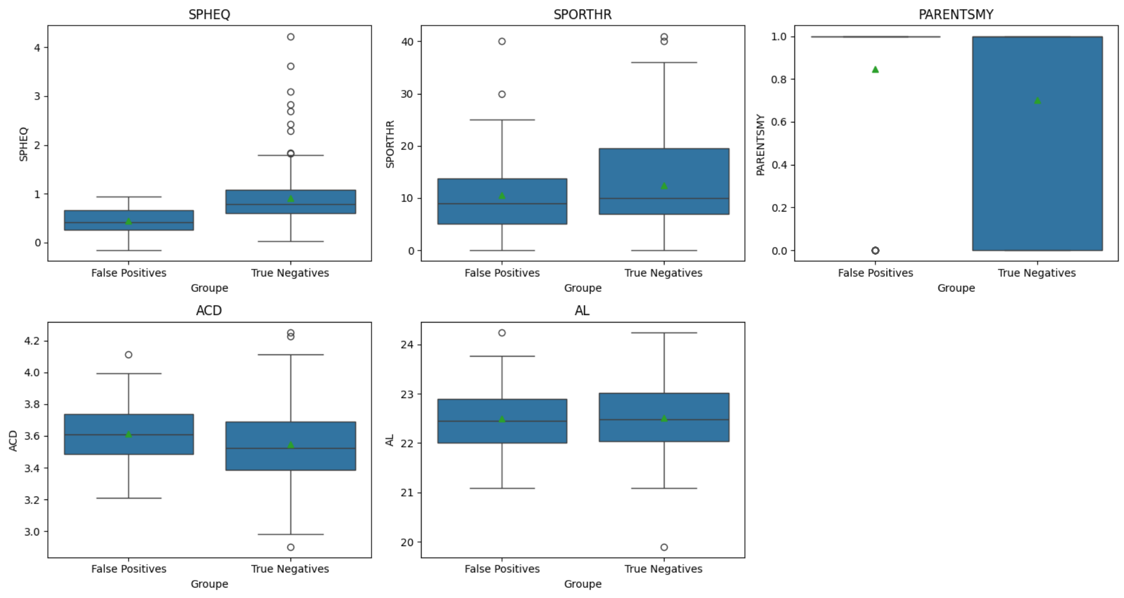
```
weighted avg       0.88      0.94      0.91        49


Statistiques descriptives - FN vs TP
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:

Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:

Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:

Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.

               mean        std     min       25%      50%       75%     max
SPHEQ       0.57225   0.207938   0.261   0.56400   0.6665   0.67475   0.695
SPORTHR    15.50000  12.476645   4.000   7.00000  13.0000  21.50000  32.000
PARENTSMY   0.75000   0.500000   0.000   0.75000   1.0000   1.00000   1.000
ACD         3.65250   0.180208   3.410   3.60125   3.6775   3.72875   3.845
AL         23.05250   0.641008  22.110  22.95750  23.2800  23.37500  23.540

                mean       std     min      25%     50%     75%     max
SPHEQ       0.040429  0.348470  -0.467  -0.2085   0.204   0.230   0.503
SPORTHR     9.000000  7.571878   1.000   4.0000   6.000  13.000  22.000
PARENTSMY   1.000000  0.000000   1.000   1.0000   1.000   1.000   1.000
ACD         3.657143  0.209411   3.342   3.5420   3.690   3.757   3.970
AL         22.912857  0.520471  22.270  22.6100  22.840  23.100  23.860


Statistiques descriptives - FP vs TN
                mean       std     min       25%      50%       75%     max
SPHEQ       0.440435  0.266337  -0.158   0.26525   0.4175   0.66725   0.944
SPORTHR    10.565217  8.344398   0.000   5.00000   9.0000  13.75000  40.000
PARENTSMY   0.847826  0.363158   0.000   1.00000   1.0000   1.00000   1.000
ACD         3.612739  0.210382   3.210   3.48800   3.6110   3.73600   4.114
AL         22.495217  0.684835  21.080  22.01000  22.4450  22.89750  24.240

                mean       std     min       25%      50%       75%     max
SPHEQ       0.915974  0.610069   0.024   0.59425   0.7835   1.08475   4.228
SPORTHR    12.467532  8.278777   0.000   7.00000  10.0000  19.50000  41.000
```

```
PARENTSMY    0.701299  0.459182   0.000   0.00000   1.0000   1.00000   1.000
ACD          3.548377  0.246314   2.904   3.38700   3.5240   3.69000   4.250
AL          22.518117  0.710661  19.900  22.04000  22.4750  23.01000  24.240
```

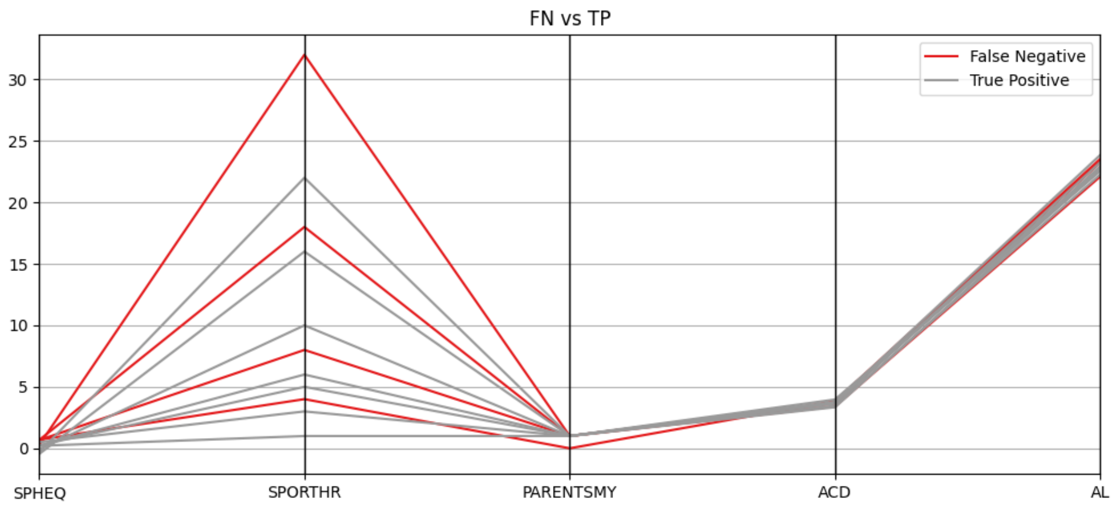
Comparaison visuelle FN/TP



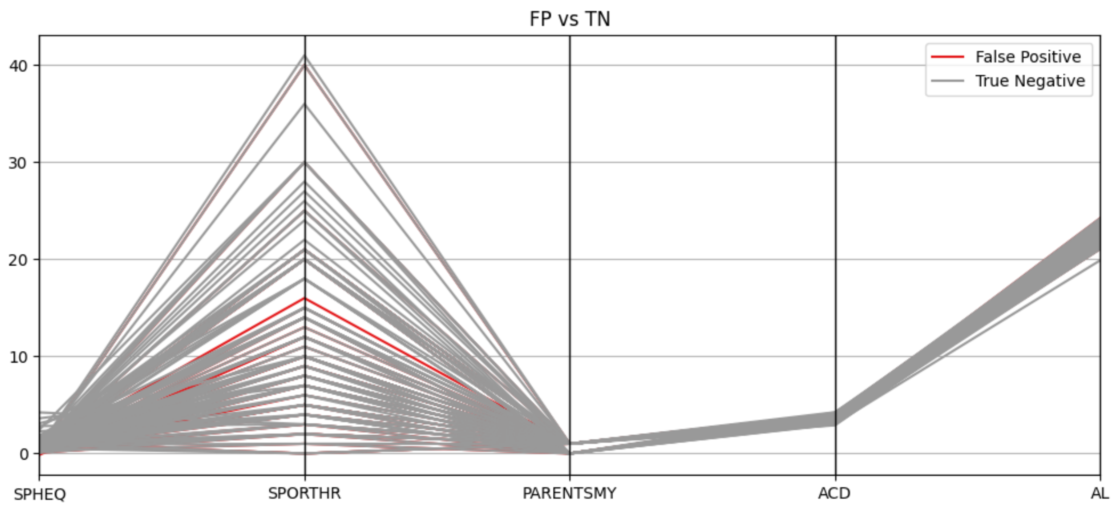
Comparaison visuelle FP/TN



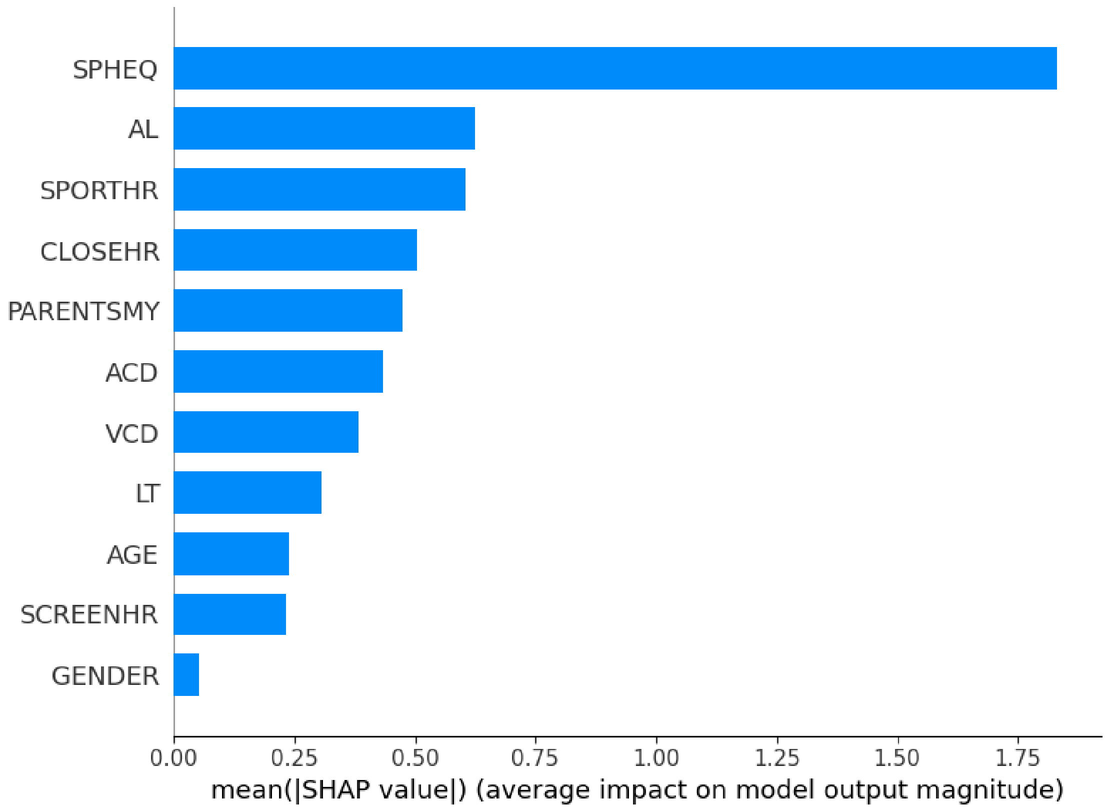
```
FN/TP Parallèles

/var/folders/13/07j4wbfd4613yv4ymtvk0_b00000gn/T/ipykernel_50456/3293880108.py:5
2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/13/07j4wbfd4613yv4ymtvk0_b00000gn/T/ipykernel_50456/3293880108.py:5
3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/13/07j4wbfd4613yv4ymtvk0_b00000gn/T/ipykernel_50456/3293880108.py:5
4: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/13/07j4wbfd4613yv4ymtvk0_b00000gn/T/ipykernel_50456/3293880108.py:5
5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

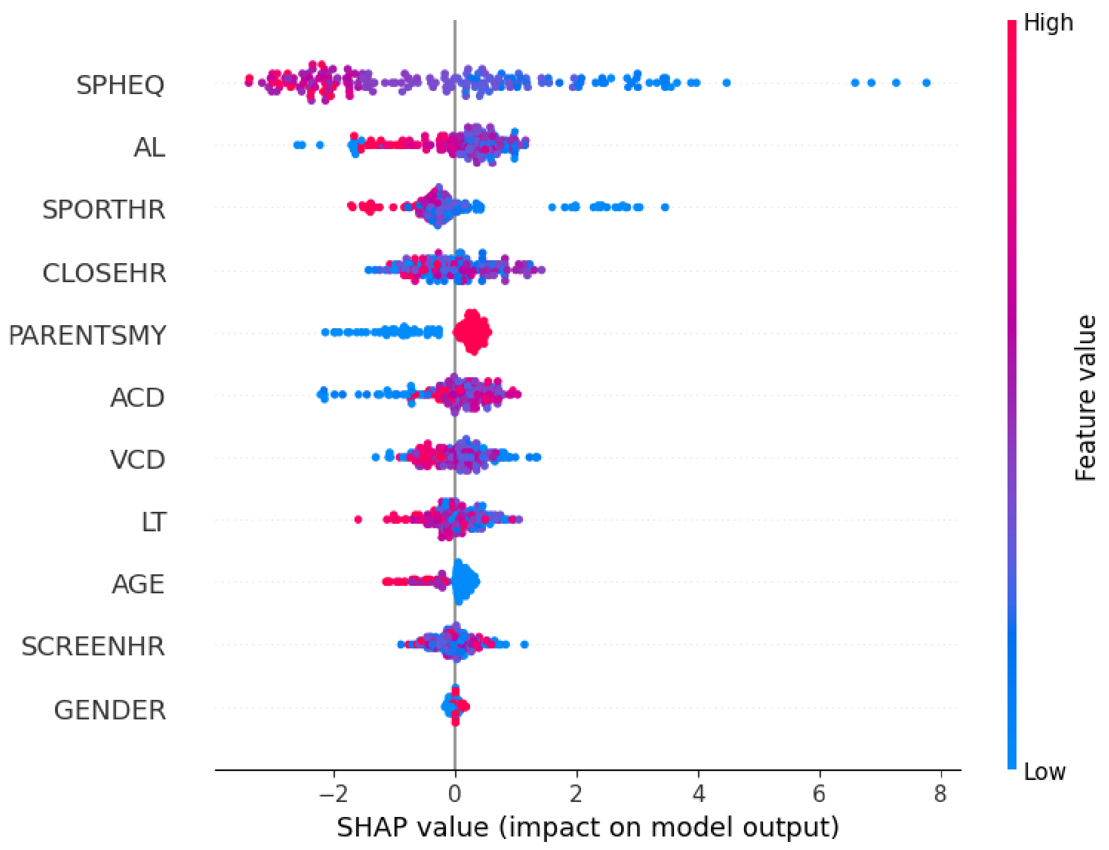


```
FP/TN Parallèles
```



```
shap_values type: <class 'numpy.ndarray'>
shap_values: (186, 11)
X_test: (186, 11)
```


SPHEQ
AL
SPORTHR
CLOSEHR
PARENTSMY
ACD
VCD
LT
AGE
SCREENHR
GENDER
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
mean(|SHAP value|) (average impact on model output magnitude)



4.0.4 3.0.4. Model Performance

- The best model performance in terms of AUC is achieved with **Logistic Regression** (Mean ROC-AUC 0.88) compared to **Random Forest** (0.83) and **Gradient Boosting** (0.81).
- All models suffer from lower recall on the minority class (label=1), indicating difficulty in detecting positive cases (likely those at higher risk). For example, recall for `y=1` is 0.83 (Logistic Regression), 0.04 (Random Forest), and 0.29 (Gradient Boosting).

4.1 3.1. Error Analysis: Global Findings and Recommendations

4.1.1 Overview of Error Patterns

Class ImbalanceandDistribution

- **The dataset is imbalanced**, with the non-myopic class (~85%) much larger than the myopic class (~15%). This imbalance contributes to more frequent **false negatives** (missed myopia cases) and poor recall for myopia detection.
- Metrics such as recall for the “myopic” class are notably low, especially in Random Forest and Gradient Boost models.

False Negatives (FN)

- **Who are the false negatives?**
  False negatives are often individuals with feature values and profiles similar to non-myopes, especially for SPHEQ, SPORTHR, and PARENTSMY.
- **Key Patterns:**
  - Many FNs do not have myopic parents, or their SPHEQ values are not at the extremes, making them harder to distinguish from non-myopes.
  - Lower levels of sport (SPORTHR) are associated with myopes, but this feature alone is not always discriminative, contributing to missed cases.

False Positives (FP)

- **Who are the false positives?**
  FPs are often individuals who have risk factor profiles resembling myopes (e.g., low SPHEQ or myopic parents) but are ultimately diagnosed as non-myopic.
- **Key Patterns:**
  - FPs are more frequent among those who have at least one myopic parent, showing that parental myopia is a strong but not exclusive determinant.
  - Some border or intermediate SPHEQ/ACD/AL values are not fully captured by the model logic, leading to confusion.

Feature InteractionsandModel Confusion

- **SPHEQ dominates:**
  Errors often arise when SPHEQ is near the decision boundary, especially if other variables (like PARENTSMY or SPORTHR) also take ambiguous/intermediate values.
- **Multicollinearity/correlation:**
  High correlations among ocular biometrics (SPHEQ, AL, ACD, VCD, LT) may make it harder for models to separate overlapping cases, especially with moderate or typical values.

Subgroup Sensitivity

- **Gender:**
  No significant effect was found with gender, but recall remains lower for the minority class when stratified by gender, suggesting possible small sample bias or noise.
- **Parental myopia:**
  Strongly increases risk, but not all children of myopic parents are myopic, leading to FPs in high-risk groups.

4.1.2 Recommendations for Model Improvement

Issue	Observations & Impact	Recommendations/Plan
Class imbalance	Low recall for myopia, many FNs	Use class weights, resampling (SMOTE), and focus on recall as a target metric

Issue	Observations & Impact	Recommendations/Plan
Feature dominance & ambiguity	Errors when SPHEQ is in intermediate range; weak additional cues	Engineer new interaction features (e.g., SPHEQ x SPORTHR), use nonlinear transformations (e.g., SPHEQ^3, SPORTHR^2)
Correlated features	Multicollinearity between metrics	Principal Component Analysis (PCA) or feature selection to reduce redundancy
Risk factor overlap	FP in high parent-myopia → not all children are myopic	Consider interaction terms; possible separate models for high-risk subgroups (Parentsmy == 1)
Subgroup underperformance	Some gender-class underperformance, dataset noise	Review model fairness, possibly oversample or stratify lower-represented groups for training

---

5 4. Global SynthesisandRecommendations

5.1 Key Analytical Findings

- **SPHEQ and Ocular Features Drive Prediction:**
  The spherical equivalent (SPHEQ) is the primary variable distinguishing myopia, but other biometrics (AL, ACD, VCD) show strong correlations and may overlap in predictive power.
- **Physical Activity Shows Small Effect:**
  Myopic individuals have slightly less physical activity (SPORTHR), though benefit from intervention may be limited.
- **Family Risk Important, but Not Absolute:**
  Parental myopia elevates risk, but risk overlap means not all at-risk children develop myopia.
- **No Robust Gender or Screen Time Effect:**
  Neither gender nor self-reported screen/near-work time showed significant effects in this dataset.

5.2 Major Error PatternsandModel Challenges

Error Type	Diagnostic Insight	Recommendations
False Negatives	Missed myopia cases are often borderline or “low-risk” by standard metrics.	Focus on recall, tune thresholds, craft new feature interactions.
False Positives	Mainly among “high-risk” (e.g., parental myopia) but actually non-myopic.	Stratify high-risk, adjust for profile overlap.

Error Type	Diagnostic Insight	Recommendations
Feature Overlap	SPHEQ dominates but is ambiguous near clinical cutoffs with weak secondary cues.	Add nonlinearity, test flexible boundaries, engineer new features.
Redundancy	Ocular biometrics highly correlated, which may blur discrimination.	Use PCA or select key features to simplify model.
Class Imbalance	Myopes underrepresented, hurting minority-class performance.	Resample, reweight, and use recall as a main metric.
Subgroup Variability	Some fairness concerns across gender and risk subgroups.	Test for bias; consider stratified sampling or models.

5.3 Summary Table: IssuesandActions

Issue	Next Steps
Low recall for myopia	Weigh/oversample positives, optimize recall threshold, engineer new features
Frequent FPs in "high-risk"	Profile and adjust for subgroups with overlapping features
Feature ambiguity	Nonlinear models, new interactions between risk factors
Multicollinearity	Feature reduction, aggregation, or PCA
Class imbalance	Resample, reweight, select models by recall
Subgroup/model fairness	Continue fairness audits and mitigate bias if detected

5.4 Strategic Recommendations

1. **Maximize Sensitivity for Myopia:**
   Adopt class weighting, SMOTE, and threshold tuning to raise recall for minority class.
2. **Advance Feature Engineering:**
   Create and test interaction and nonlinear features (e.g., SPHEQ $\times$ SPORTHR, PARENTSMY $\times$ SPHEQ).
3. **Reduce Predictor Redundancy:**
   Apply PCA or careful selection to focus on key, independent variables.
4. **Balance Interpretability & Performance:**
   Prefer regularized logistic or shallow ensemble methods for transparent yet strong results.
5. **Monitor Model Fairness:**
   Regularly evaluate performance across subgroups; address any notable gaps.