Data_Exploration

May 9, 2025

Myopia Study: Comprehensive Analysis, Modeling and Reporting

---

1 Table of Contents

- 1. Initialisation
  - Importing libraries and loading the myopia dataset for analysis
    - Data Engineering
    - Summary
- 2. Data Exploration
  - Univariate Analysis and Multivariate Visualization
  - 2.1. Statistic analysis
    - 2.1.1. Univariate Analysis
    - Analysis of Gender and Parental Myopia Association with Myopia Status
    - 4.1.2. Quant values
  - Conclusion et Synthesis
- 5. Predicting Model - Simplest models
  - 5.0.1. Logistic Regression
  - 5.0.2. Random Forest
  - 5.0.3. Gradient Boost
  - 5.0.4. Model Performance
- 5.1. Error Analysis: Global Findings and Recommendations
  - Overview of Error Patterns
    - Class Imbalance et Distribution
    - False Negatives (FN)
    - False Positives (FP)
    - Feature Interactions et Model Confusion
    - Subgroup Sensitivity
  - Recommendations for Model Improvement
- 6. Global Synthesis et Recommendations
  - Key Analytical Findings
  - Major Error Patterns et Model Challenges
  - Summary Table: Issues et Actions
  - Strategic Recommendations

- Future In-depth Predictive Modeling et Advanced Comparative Analysis (see Predict-

ing_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
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
```

```
/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

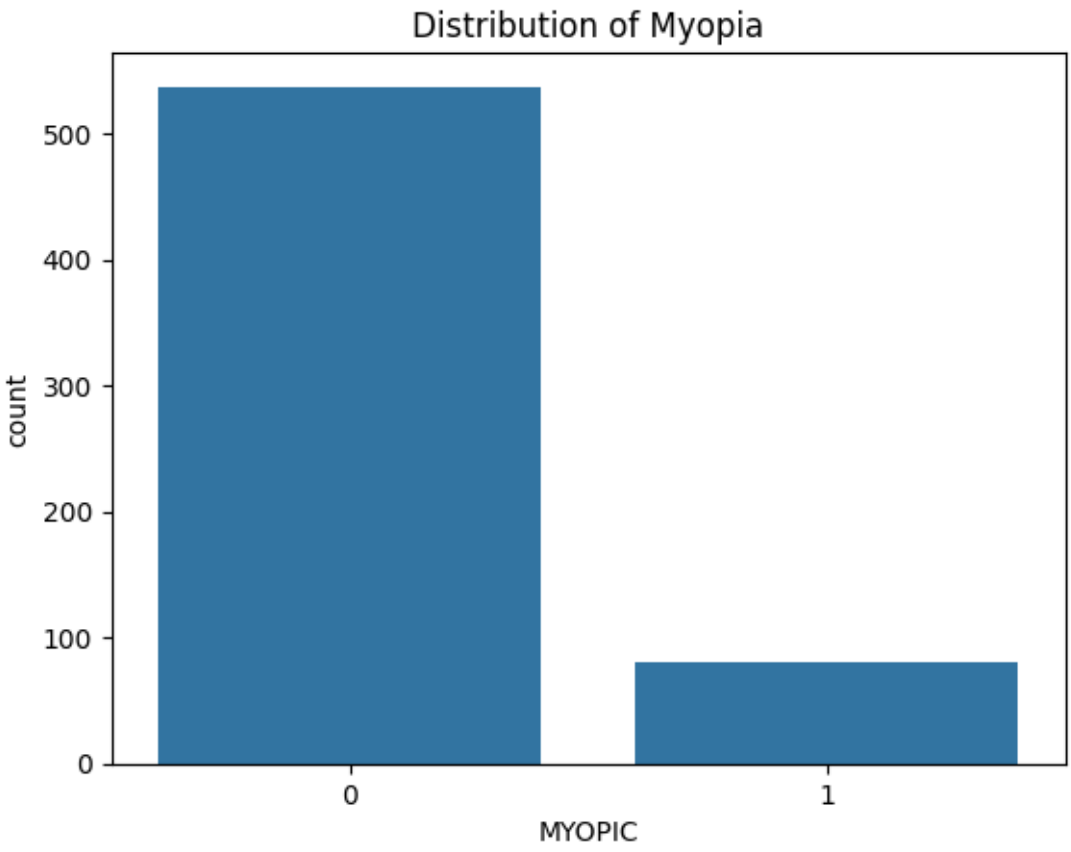
	READHR	COMPHR	STUDYHR	TVHR	DIOPTERHR	MOMMY
count	618.000000	618.000000	618.000000	618.000000	618.000000	618.000000
mean	2.796117	2.105178	1.490291	8.948220	26.017799	0.506472
std	3.068191	3.056508	2.216207	5.719021	16.031715	0.500363
min	0.000000	0.000000	0.000000	0.000000	2.000000	0.000000
25%	0.000000	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]:
```
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]:
```
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	PARENTSMY	SCREENHR	CLOSEHR
0	1	6	1	-0.052	21.89	3.690	3.498	14.70	45	1	10	42
1	0	6	1	0.608	22.38	3.702	3.392	15.29	4	1	8	13
2	0	6	1	1.179	22.49	3.462	3.514	15.52	14	0	12	14
3	1	6	1	0.525	22.20	3.862	3.612	14.73	18	1	4	48
4	0	5	0	0.697	23.29	3.676	3.454	16.16	14	1	4	4
..	…	…	…	…	…	…	…	…	…	…	…	…
613	1	6	0	0.678	22.40	3.663	3.803	14.93	2	1	21	40
614	0	6	1	0.665	22.50	3.570	3.378	15.56	6	1	9	10
615	0	6	0	1.834	22.94	3.624	3.424	15.89	8	1	4	4
616	0	6	1	0.665	21.92	3.688	3.598	14.64	12	0	16	25
617	0	6	0	0.802	22.26	3.530	3.484	15.25	25	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]:
```
myopiafact.head(5)
```

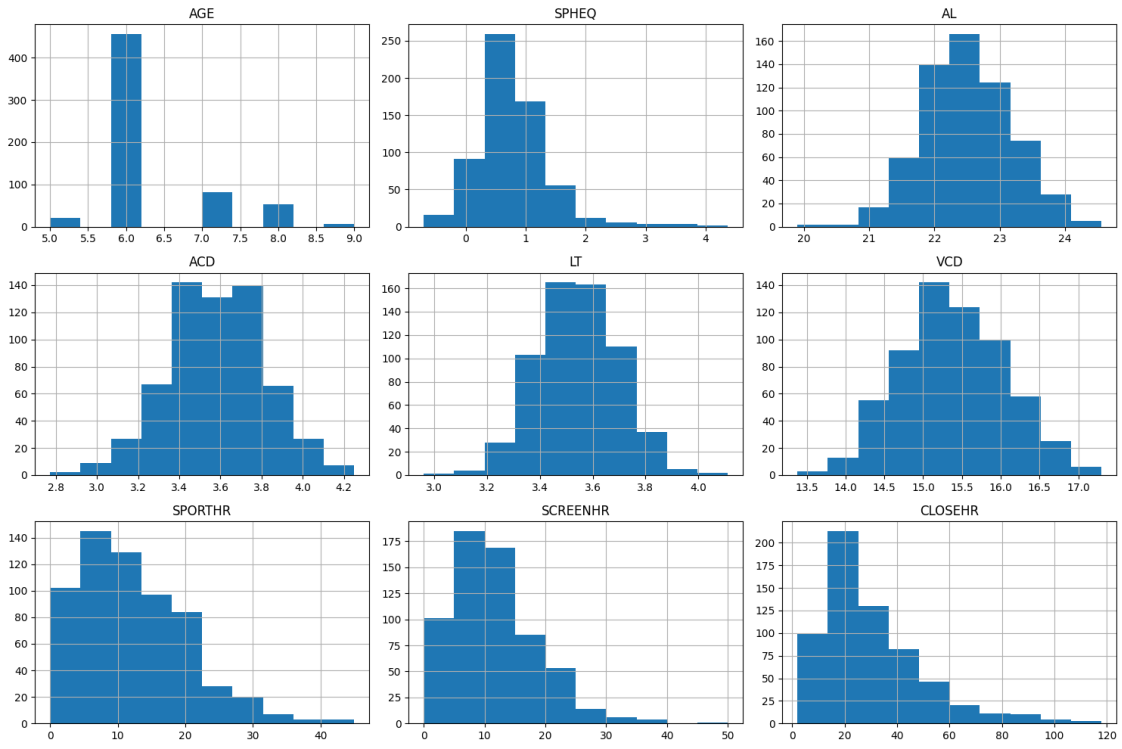
[12]:

	MYOPIC	GENDER	PARENTSMY
0	1	1	1
1	0	1	1
2	0	1	0
3	1	1	1
4	0	0	1

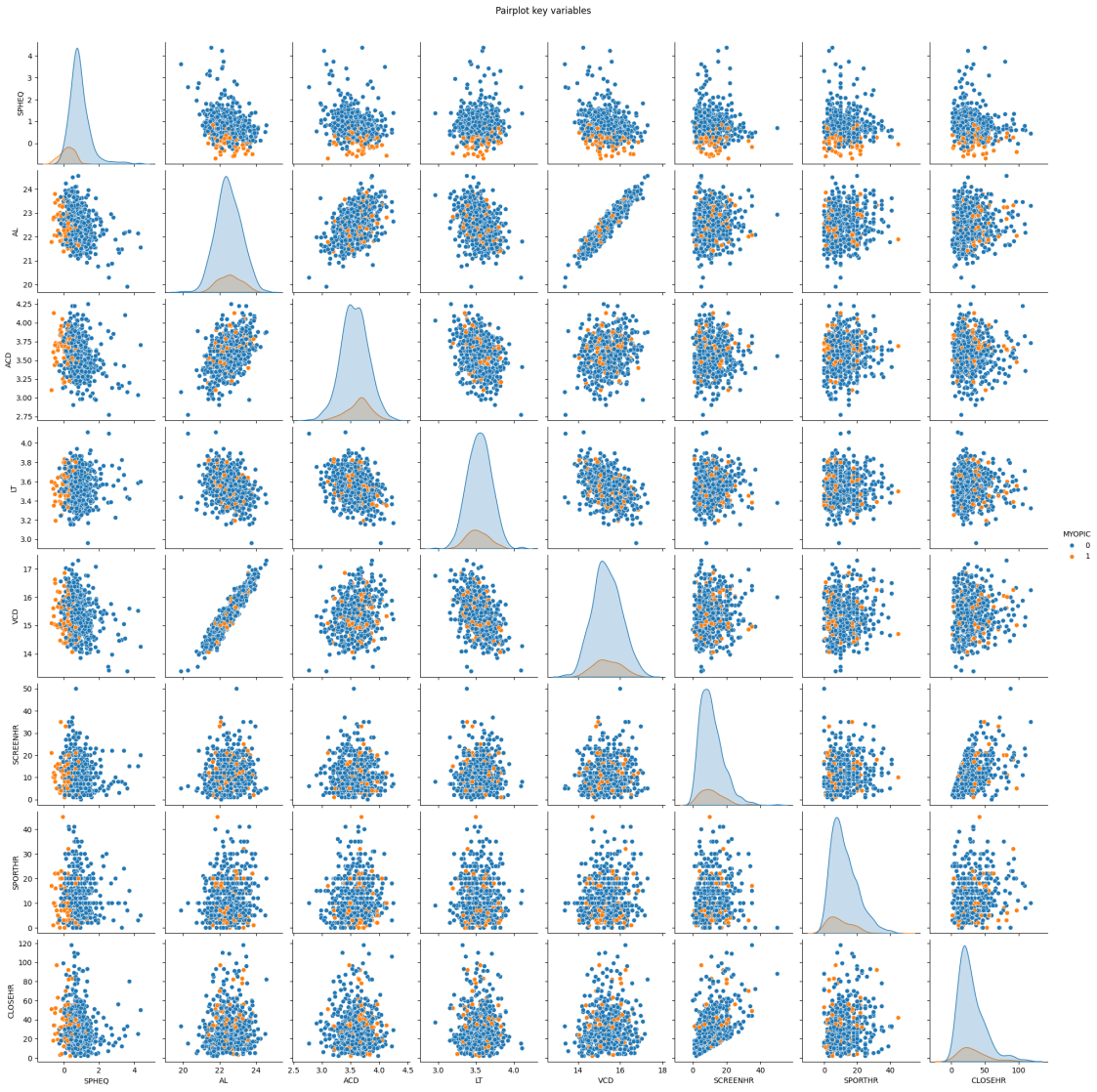
3.0.1 Univariate Analysis and Multivariate Visualization

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

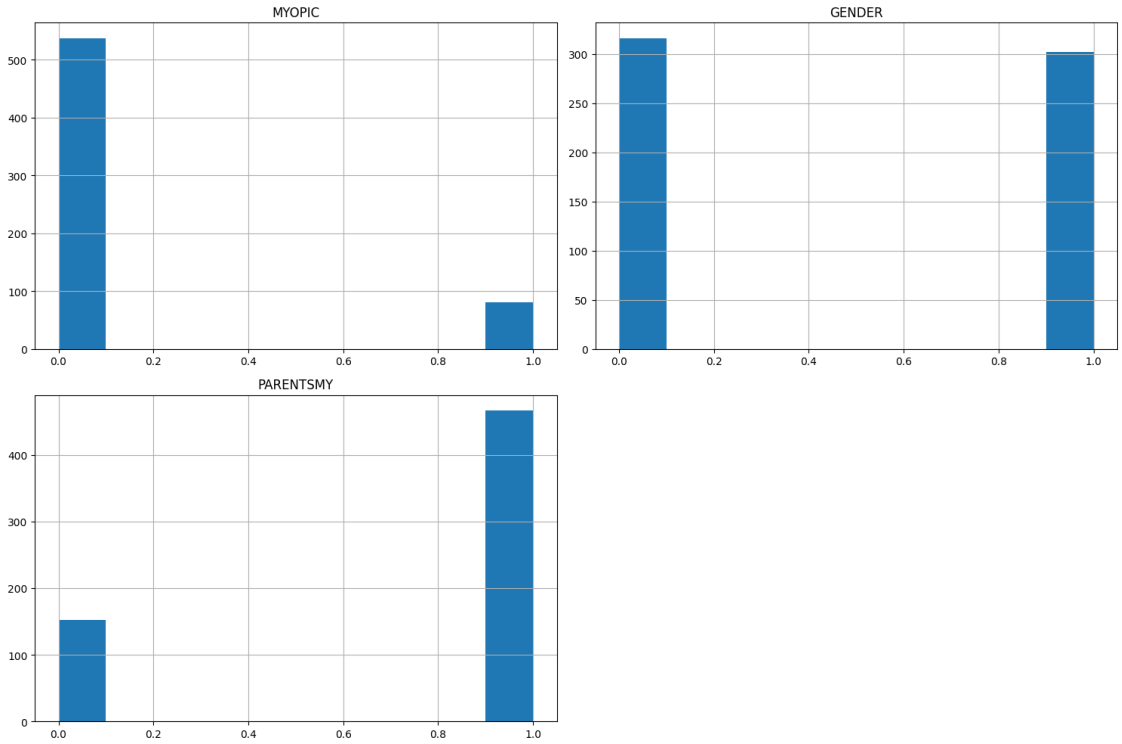
[13]:
```
myopianum.hist(figsize=(15,10))
plt.tight_layout()
plt.show()
```



[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()
```



3.1 2.1. Statistic analysis

3.1.1 2.1.1. Univariate Analysis

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

[16]:
```
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')
```

[17]:
```
# 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
```

[18]:
```
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 4.1.2. Quant values

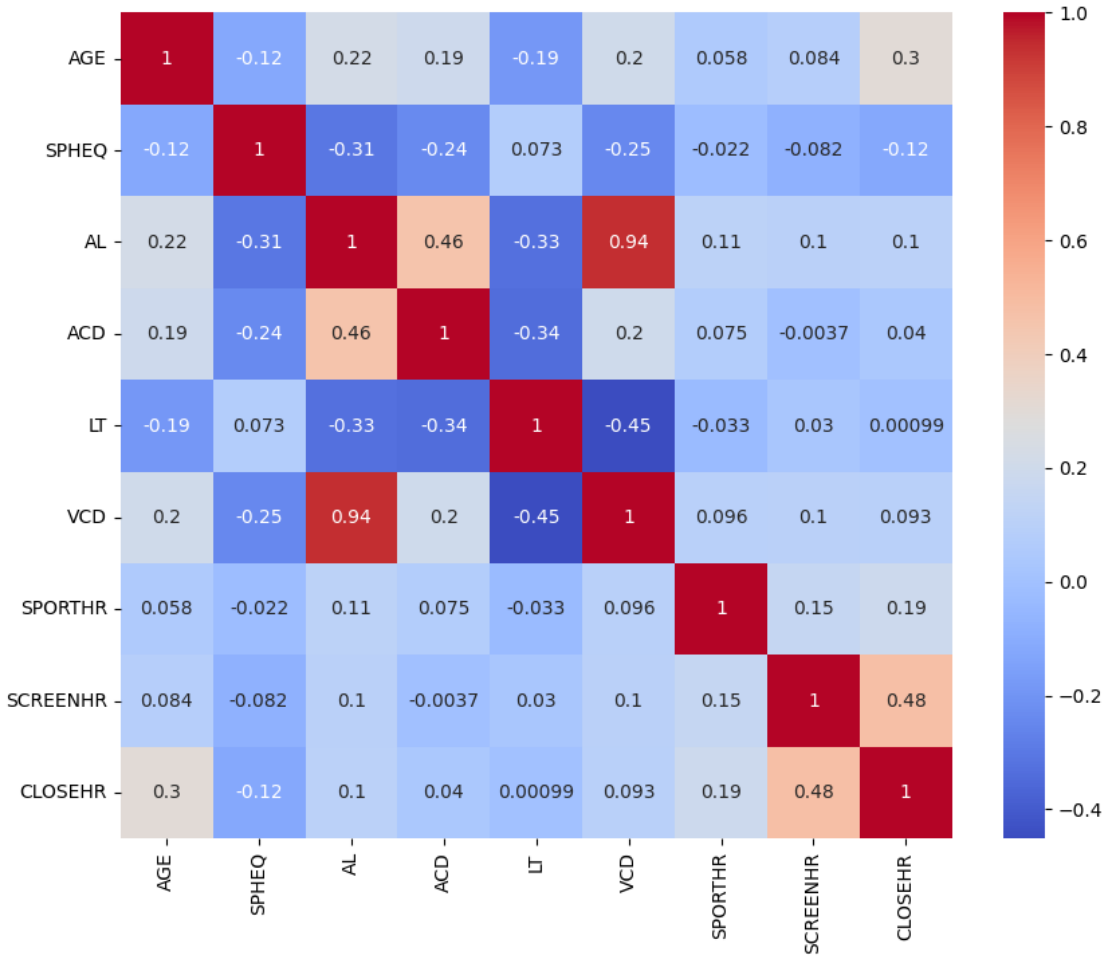
[19]:
```
for col in myopianum.columns:
    group0 = df.loc[df['MYOPIC'] == 0, col]
    group1 = df.loc[df['MYOPIC'] == 1, col]
    stat, p = stats.ttest_ind(group0, group1, nan_policy='omit')
    fig = px.box(df, x='MYOPIC', y=col, color='MYOPIC', points="all", title=col)
    fig.show()
    print(f"T-test {col}: statistic={stat:.2f}, p-value={p:.4f} \n-----")
```

```
T-test AGE: statistic=-0.46, p-value=0.6458
-----

T-test SPHEQ: statistic=10.00, p-value=0.0000
-----

T-test AL: statistic=-0.94, p-value=0.3488
-----

T-test ACD: statistic=-2.70, p-value=0.0072
-----

T-test LT: statistic=1.14, p-value=0.2566
-----

T-test VCD: statistic=-0.29, p-value=0.7687
-----

T-test SPORTHR: statistic=2.45, p-value=0.0145
-----

T-test SCREENHR: statistic=-0.20, p-value=0.8386
-----

T-test CLOSEHR: statistic=-0.94, p-value=0.3475
-----
```

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

[20]:
```
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

Conclusion et 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 5. Predicting Model - Simplest models

[21]:
```
df['MYOPIC'] = df['MYOPIC'].astype(int)
```

[22]:
```
x = df.drop(['MYOPIC'], axis=1)
x['GENDER'] = x['GENDER'].astype(int)
x['PARENTSMY'] = x['PARENTSMY'].astype(int)
y = df['MYOPIC'].astype(int)
```

[23]:
```
def eval_model(model, X_train, y_train, X_test, y_test, seuil=0.27,␣
↪name='modèle', cv=5):
    """Entraine, prédit, affiche tout, renvoie prédictions pour analyse."""
    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 (sur le train !)
    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}")

    ## Analyse: on crée un DataFrame résultat
    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("FAUX NEGATIFS (devraient être détectés!):")
    display(fn.head())
    print("FAUX POSITIFS (vrais non-myopiques, fausse alerte):")
    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]))
```

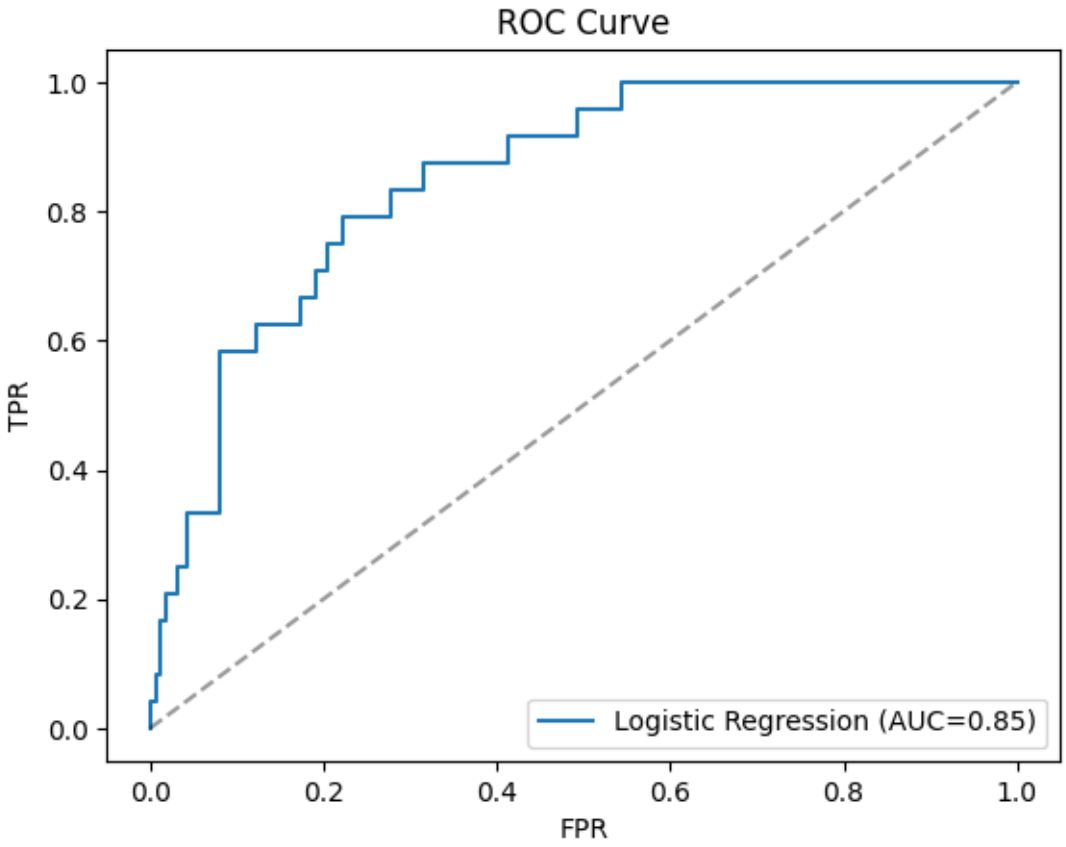
[24]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3,␣
↪random_state=42, stratify=y)
```

4.0.1 5.0.1. Logistic Regression

[25]:
```
# 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=0.5
)

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')

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

# Optionnel : valeur absolue pour trier par importance pure
coefs_sorted = coefs.abs().sort_values()


# SHAP VALUES (optionnel: si besoin explicabilité)

explainer = shap.Explainer(model_lr, X_train)  # Pour sklearn >= 0.39,␣
↪Expliquer auto-détecte le type !
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: 0.8333333333333334
AUC: 0.8497942386831275
Confusion Matrix:
 [[140  22]
 [  9  15]]
              precision    recall  f1-score   support
```

```
              0       0.94      0.86      0.90       162
              1       0.41      0.62      0.49        24

    accuracy                            0.83       186
   macro avg       0.67      0.74      0.70       186
weighted avg       0.87      0.83      0.85       186
```



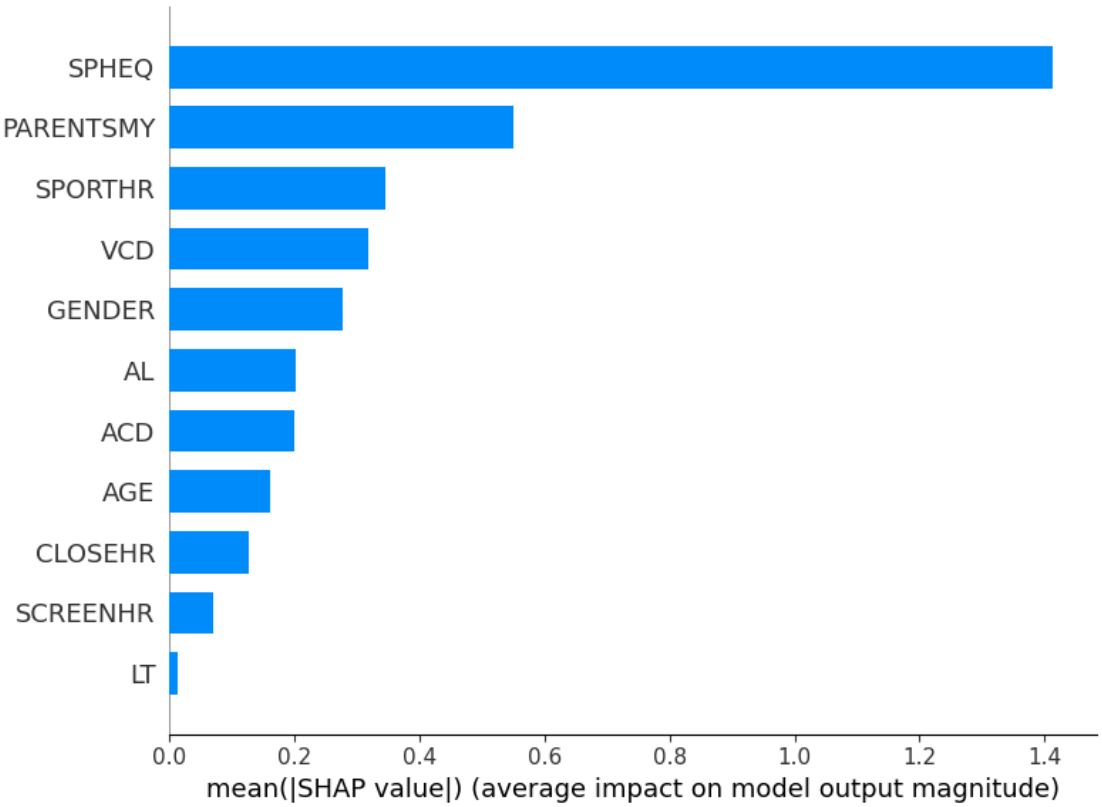

```
Mean ROC-AUC (cross-validation): 0.884
FAUX NEGATIFS (devraient être détectés!):
     AGE  GENDER  SPHEQ     AL    ACD     LT    VCD  SPORTHR  PARENTSMY  \
493    6       1  0.477  21.41  3.530  3.822  14.06        3          0
172    8       1  0.461  23.24  3.636  3.598  16.01        6          1
77     6       0  0.665  23.24  3.690  3.498  16.05        8          1
570    6       1  0.677  21.81  3.650  3.738  14.42       18          1
215    6       1  0.695  23.54  3.845  3.403  16.30        4          0

     SCREENHR  CLOSEHR  y_true  y_pred  proba_pred
493        12       24       1       0    0.442197
172        11       83       1       0    0.421407
```

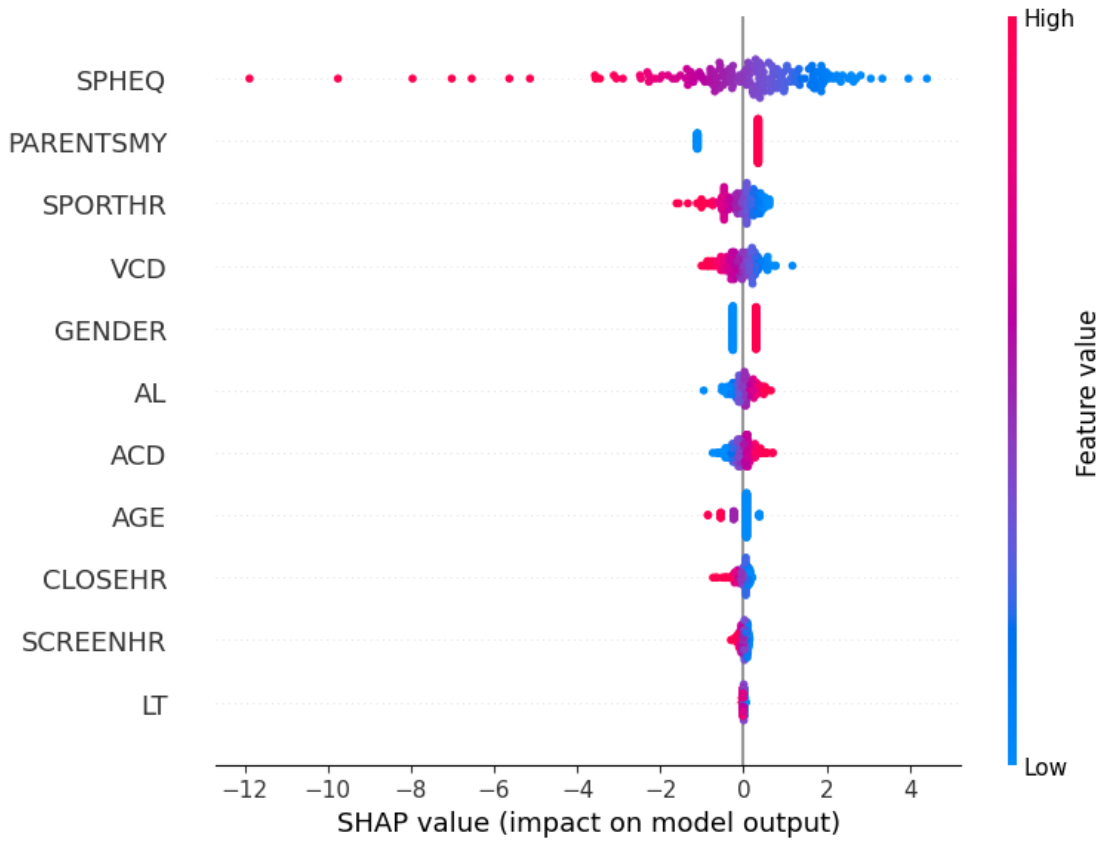
```
77          20        78        1        0    0.249340
570         14        42        1        0    0.405046
215         22        31        1        0    0.198744
```

FAUX POSITIFS (vrais non-myopiques, fausse alerte):

```
     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
535    6       1  0.378  21.83  3.464  3.896  14.47        8          1
458    6       1  0.526  23.40  3.690  3.482  16.22        7          1
50     5       0  0.265  21.98  3.532  3.466  14.98        6          1

     SCREENHR  CLOSEHR  y_true  y_pred  proba_pred
98         16       34       0       1    0.796691
59         10       22       0       1    0.596455
535         8       48       0       1    0.728417
458         7       39       0       1    0.608126
50          3       12       0       1    0.813282
```

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

           0       0.94      0.84      0.89       116
           1       0.44      0.71      0.55        21

    accuracy                           0.82       137
   macro avg       0.69      0.78      0.72       137
weighted avg       0.87      0.82      0.83       137


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

           0       0.93      0.93      0.93        46
           1       0.00      0.00      0.00         3

    accuracy                           0.88        49
   macro avg       0.47      0.47      0.47        49
weighted avg       0.88      0.88      0.88        49
```


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)

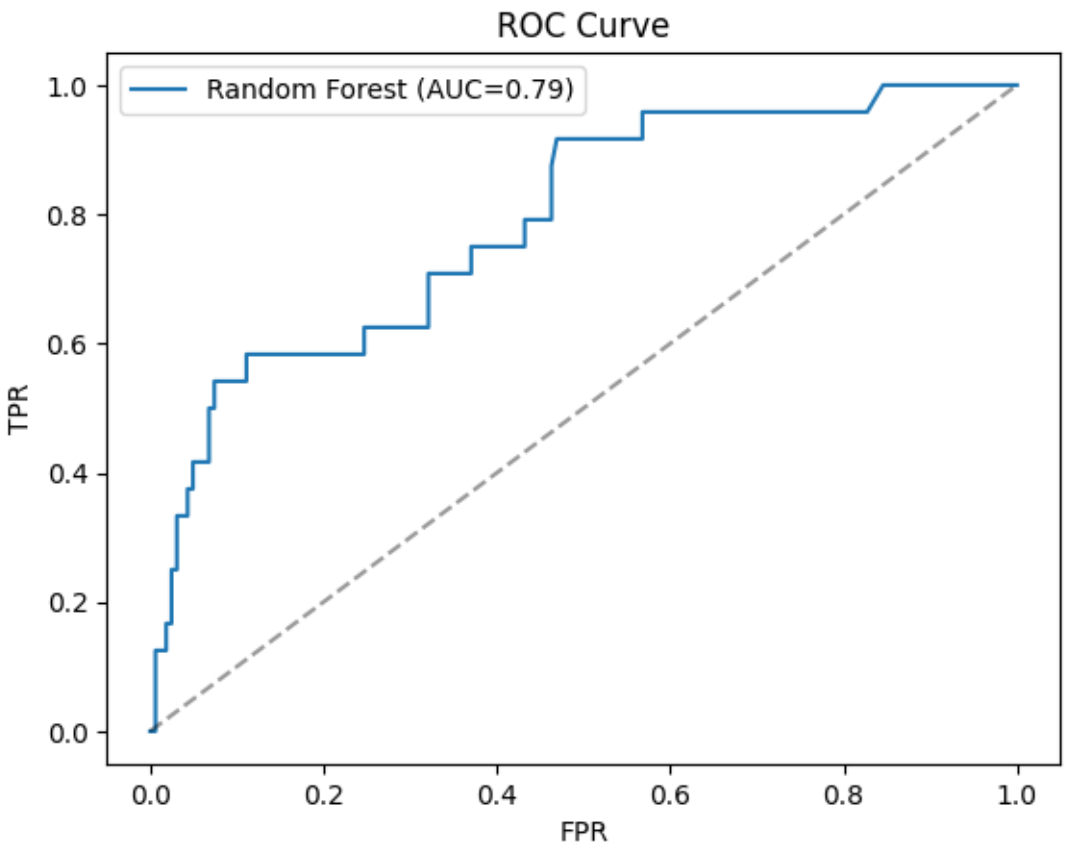


4.0.2 5.0.2. Random Forest

[26]:
```
# 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')

# Feature importance
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("Importances des variables (Random Forest)")
plt.show()
```

```python
# SHAP VALUES (optionnel: si besoin explicabilité)
import shap
explainer = shap.TreeExplainer(model_rf)
shap_values = explainer.shap_values(X_test)

# Affichons la forme pour déboguer :
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)

# Pour un cas binaire (2 classes), chaque sous-tableau aura (n_samples,␣
 ↪n_features)
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:
    # Certains cas (classification One-vs-Rest, régression, etc.)
    shap.summary_plot(shap_values, X_test, plot_type="bar")
```

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

===== Random Forest =====
Accuracy: 0.8709677419354839
AUC: 0.7896090534979424
Confusion Matrix:
 [[161   1]
 [ 23   1]]
              precision    recall  f1-score   support

           0       0.88      0.99      0.93       162
           1       0.50      0.04      0.08        24

    accuracy                           0.87       186
   macro avg       0.69      0.52      0.50       186
weighted avg       0.83      0.87      0.82       186
```



Mean ROC-AUC (cross-validation): 0.849
FAUX NEGATIFS (devraient être détectés!):

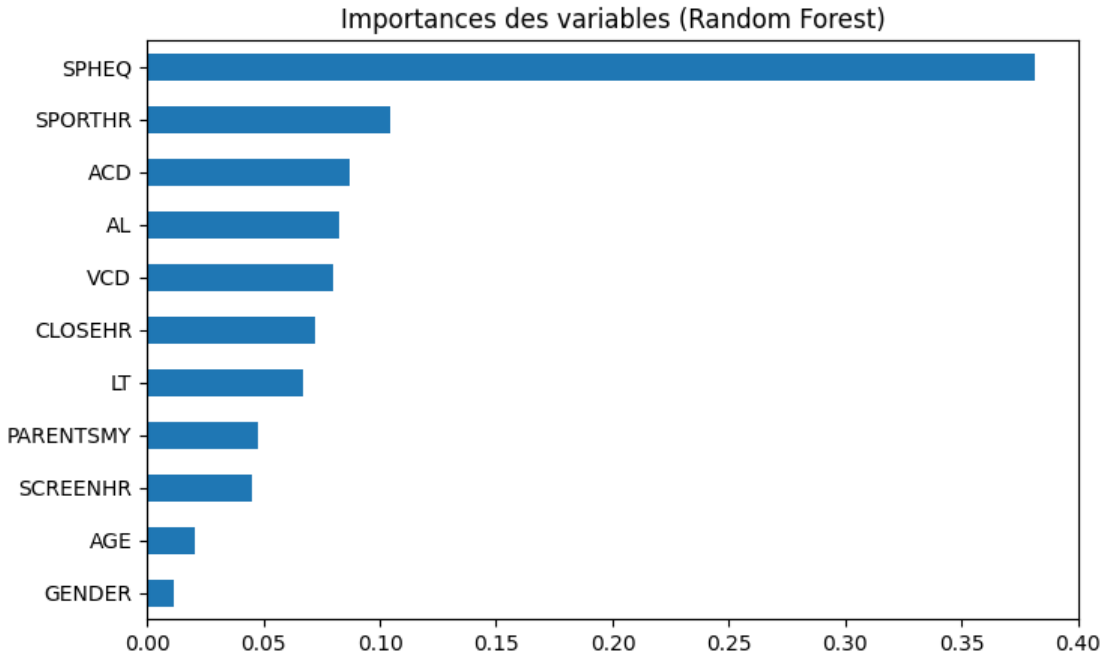
	AGE	GENDER	SPHEQ	AL	ACD	LT	VCD	SPORTHR	PARENTSMY	\
493	6	1	0.477	21.41	3.530	3.822	14.06	3	0	
172	8	1	0.461	23.24	3.636	3.598	16.01	6	1	
579	6	0	0.246	22.56	3.970	3.452	15.14	5	1	
188	6	1	0.183	22.53	3.638	3.498	15.40	20	1	
77	6	0	0.665	23.24	3.690	3.498	16.05	8	1	

	SCREENHR	CLOSEHR	y_true	y_pred	proba_pred
493	12	24	1	0	0.060000
172	11	83	1	0	0.068965
579	8	56	1	0	0.485700
188	6	19	1	0	0.358792
77	20	78	1	0	0.061422

FAUX POSITIFS (vrais non-myopiques, fausse alerte):

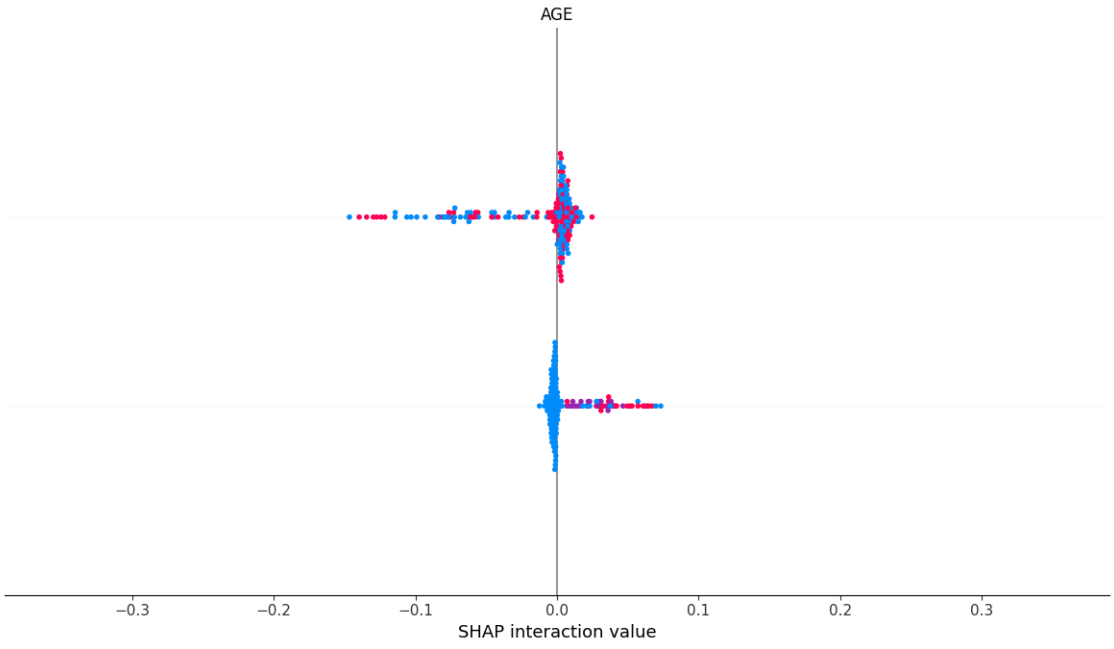
	AGE	GENDER	SPHEQ	AL	ACD	LT	VCD	SPORTHR	PARENTSMY	\
447	6	1	0.058	21.86	3.476	3.378	15.01	12	1	

```
     SCREENHR  CLOSEHR  y_true  y_pred  proba_pred
447        10       30       0       1    0.633001


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

           0       0.85      0.99      0.92       116
           1       0.50      0.05      0.09        21

    accuracy                           0.85       137
   macro avg       0.68      0.52      0.50       137
weighted avg       0.80      0.85      0.79       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


/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.
```




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

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

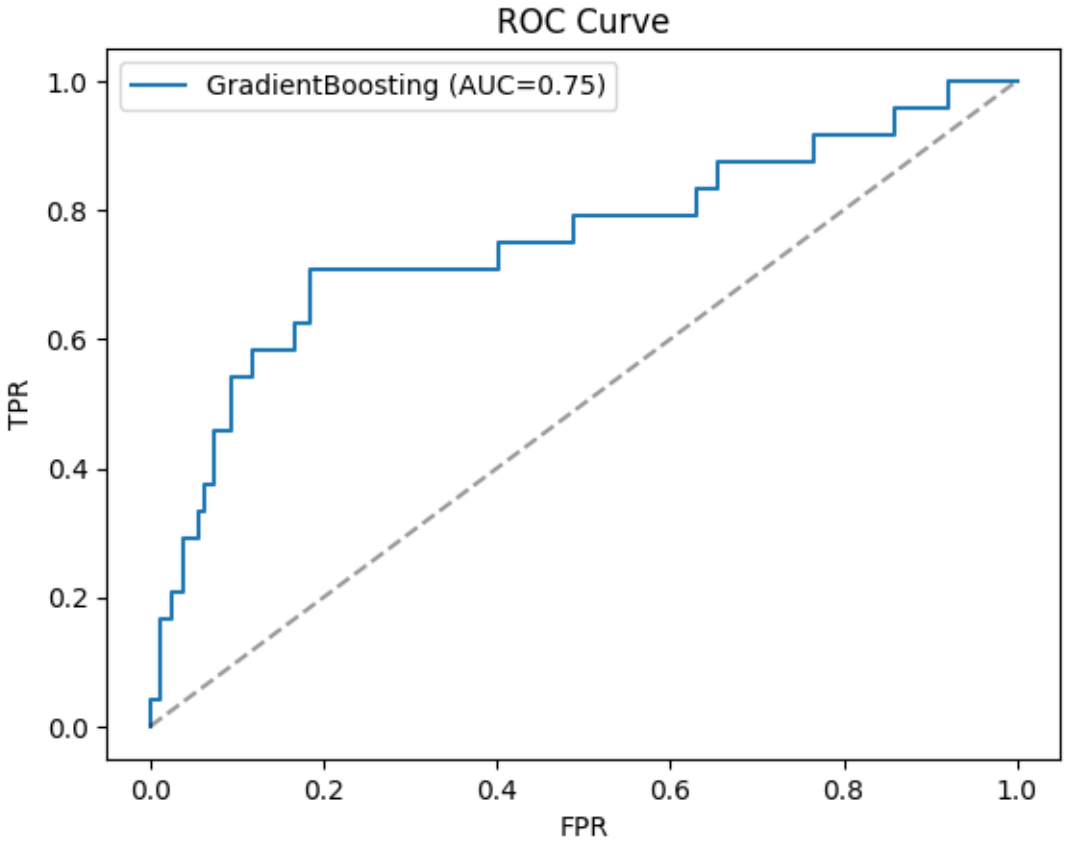


4.0.3 5.0.3. Gradient Boost

[27]:
```
# 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')

# 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 (optionnel: si besoin explicabilité)
explainer = shap.TreeExplainer(model_hgb)
shap_values = explainer.shap_values(X_test)

# Affichons la forme pour déboguer :
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)

# Pour un cas binaire (2 classes), chaque sous-tableau aura (n_samples,␣
 ↪n_features)
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
```



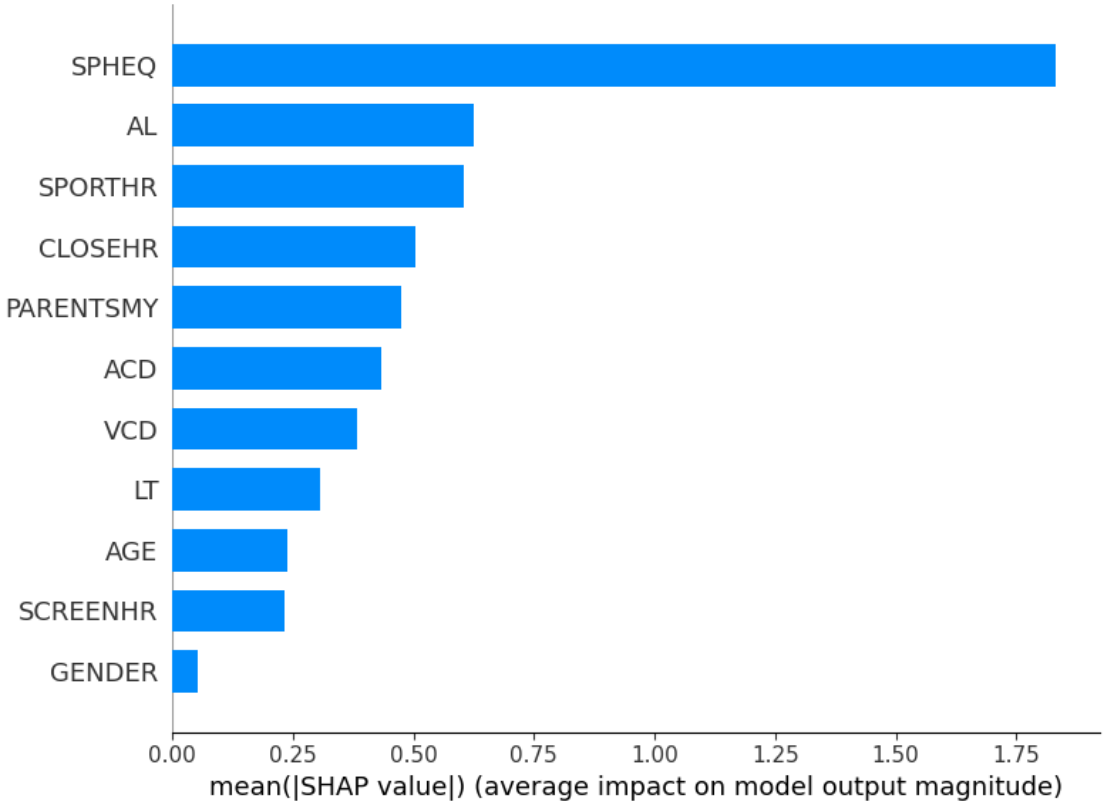

```
Mean ROC-AUC (cross-validation): 0.819
FAUX NEGATIFS (devraient être détectés!):
     AGE  GENDER  SPHEQ     AL    ACD     LT    VCD  SPORTHR  PARENTSMY  \
493    6       1  0.477  21.41  3.530  3.822  14.06        3          0
172    8       1  0.461  23.24  3.636  3.598  16.01        6          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
```

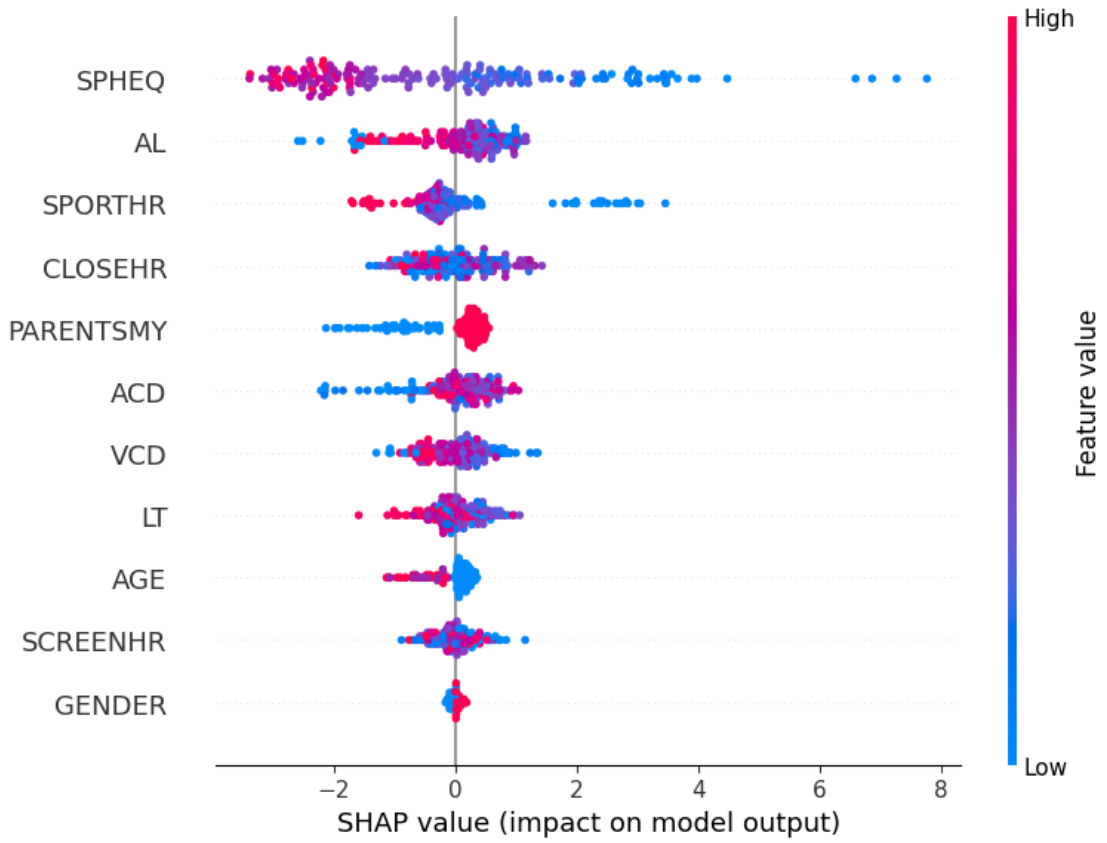
```
558    7       0  0.248  22.39  3.665  3.333  15.40       10          1

     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
```

FAUX POSITIFS (vrais non-myopiques, fausse alerte):

```
     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


/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.


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





4.0.4 5.0.4. 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** (0.84).
- 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 0.62 (Logistic Regression), 0.04 (Random Forest), and 0.33 (Gradient Boosting).

4.1 5.1. Error Analysis: Global Findings and Recommendations

4.1.1 Overview of Error Patterns

Class Imbalance et Distribution

- **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 Interactions et Model 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 6. Global Synthesis et Recommendations

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 Patterns et Model 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: Issues et Actions

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 × SPORTHR, PARENTSMY × 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.

[ ]: