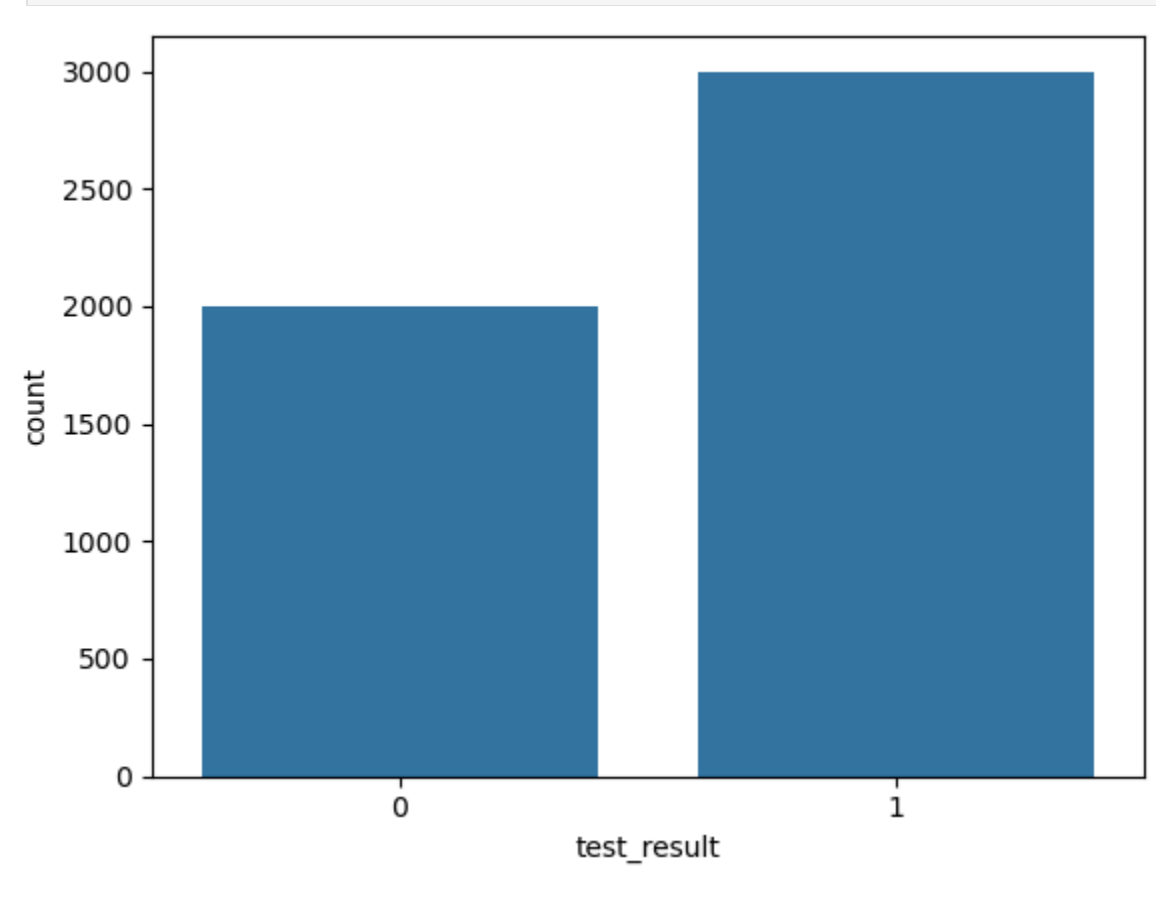
Project: Logistic Regression analysis on "hearing test dataset" downloaded from the Kaggle. This project presents my skills in data science and Logistic regression modeling using Python and scikit-learn. All analysis with results are for my educational purposes.

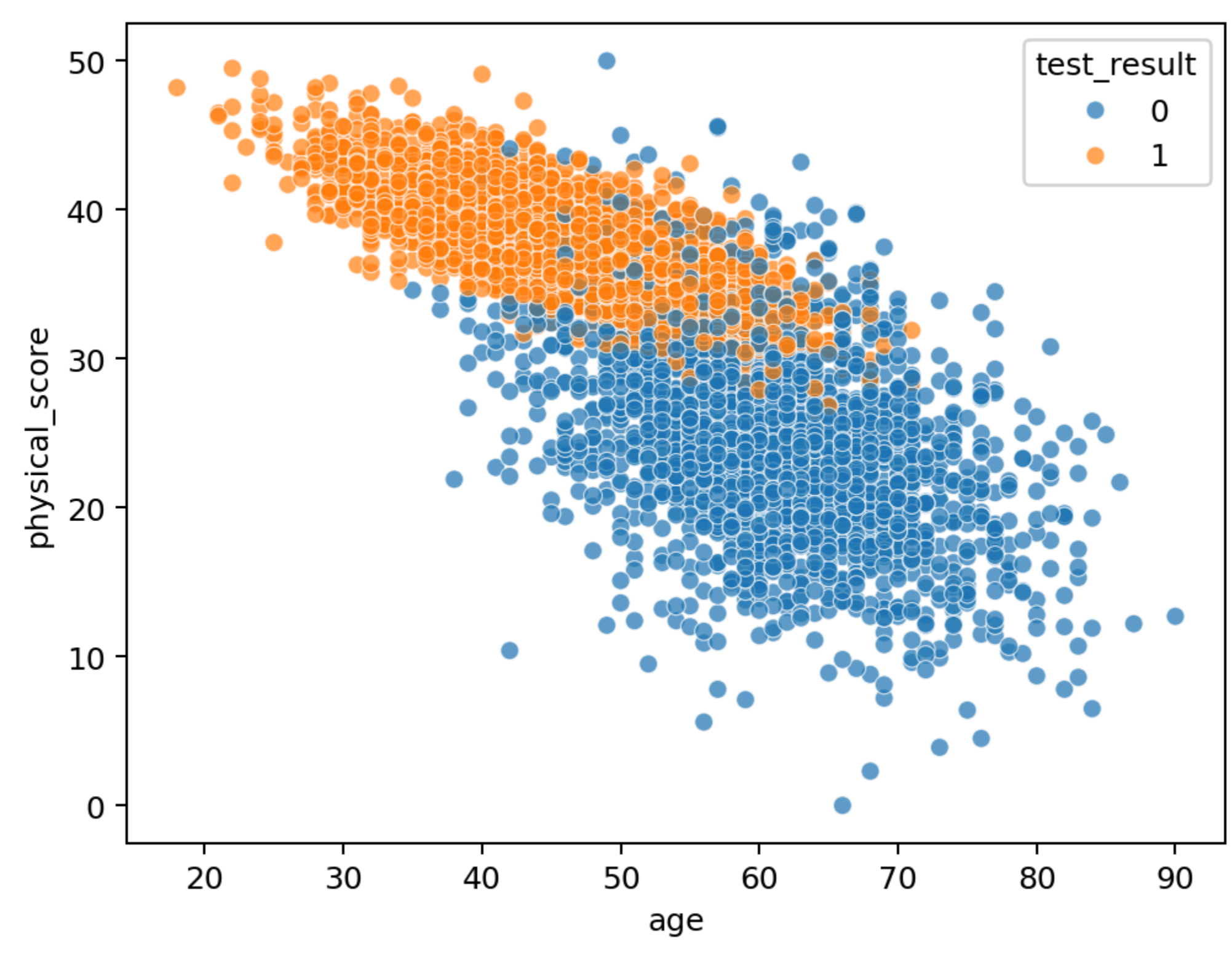
```
# data (https://www.kaggle.com/datasets/vinodbaste123/hearing-test-dataset) used from "Kaggle"
```

Logistic Regression and Scikit-Learn

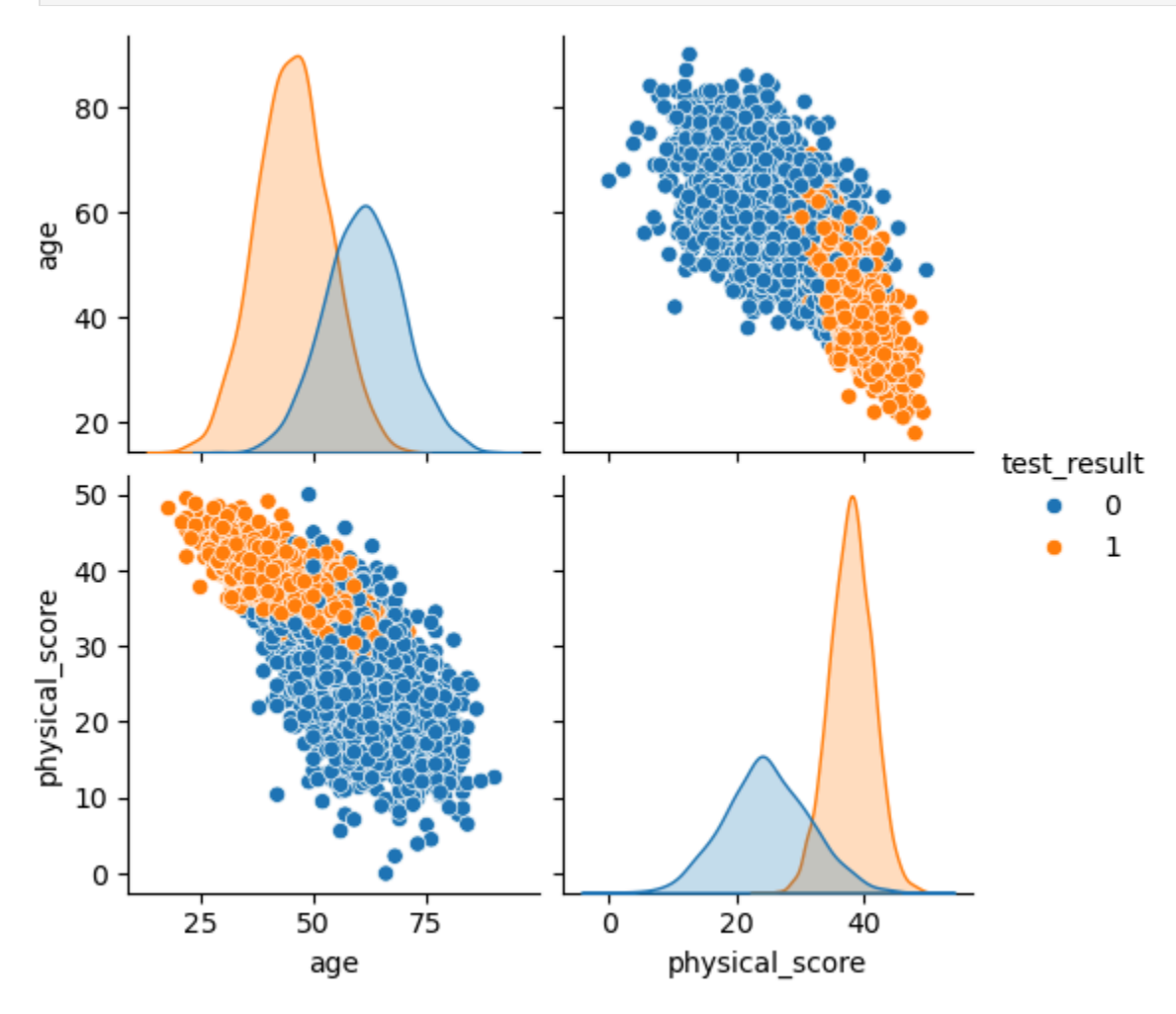
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("hearing_test.csv")

import seaborn as sns
import matplotlib.pyplot as plt

# plot
sns.countplot(x=df["test_result"])

plt.show()
```



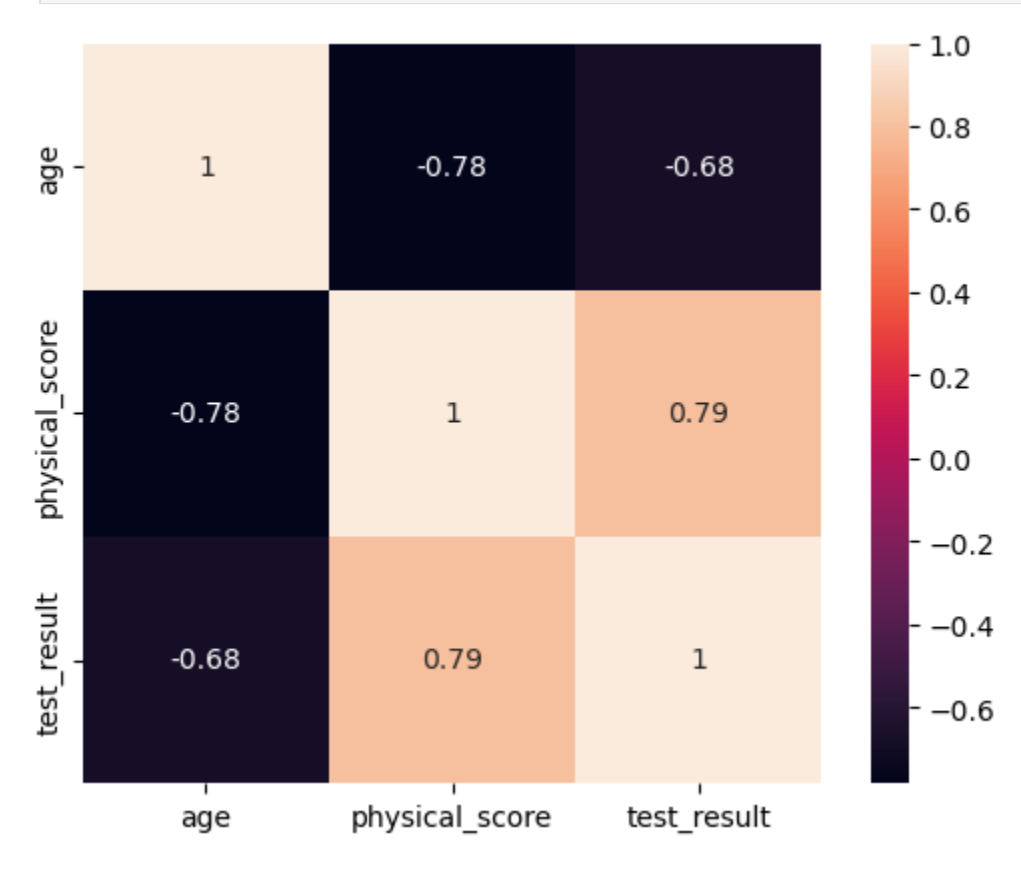

```
plt.figure(dpi = 180)

# scatter plot with hue
sns.scatterplot(data =df, x = "age", y = "physical_score", hue = "test_result", alpha = 0.7)

plt.show()
```



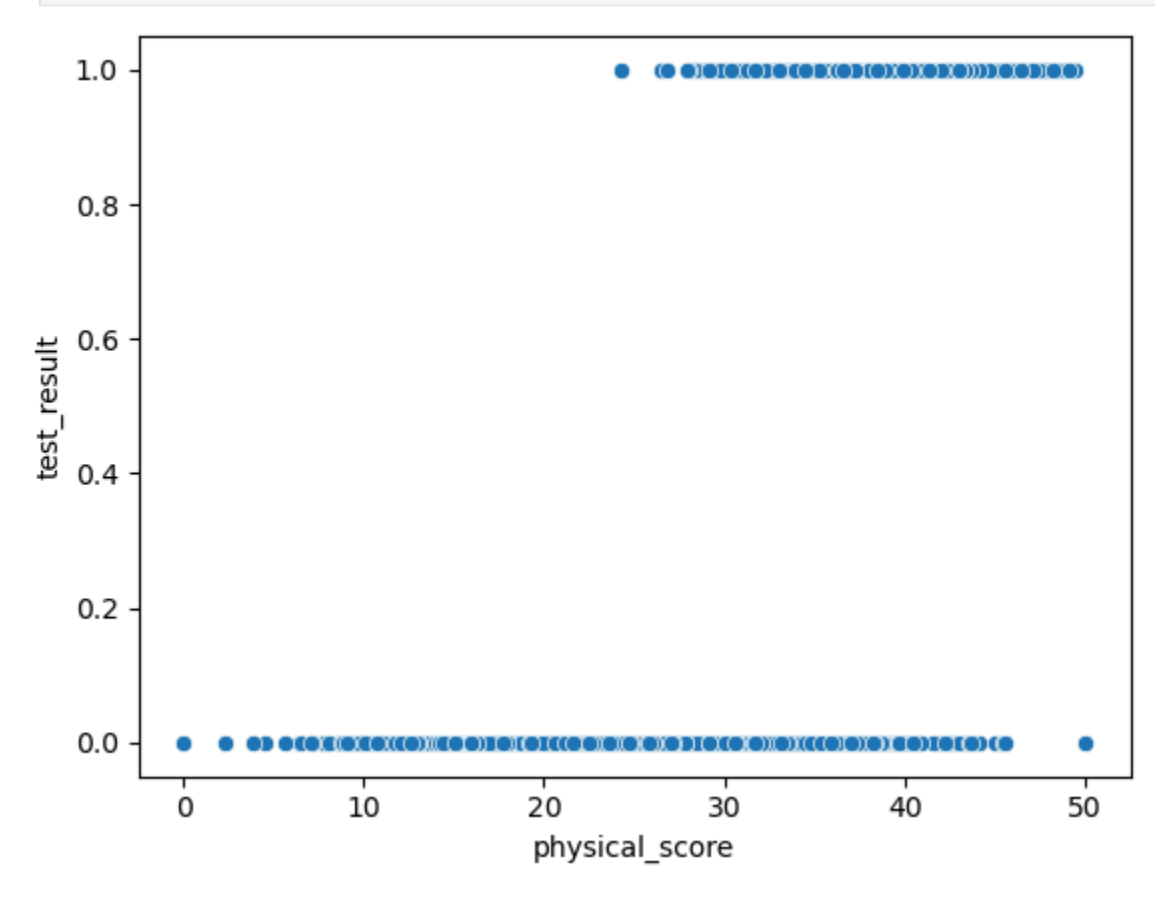

```
# Pair plot
sns.pairplot(df, hue = "test_result")

plt.show()
```



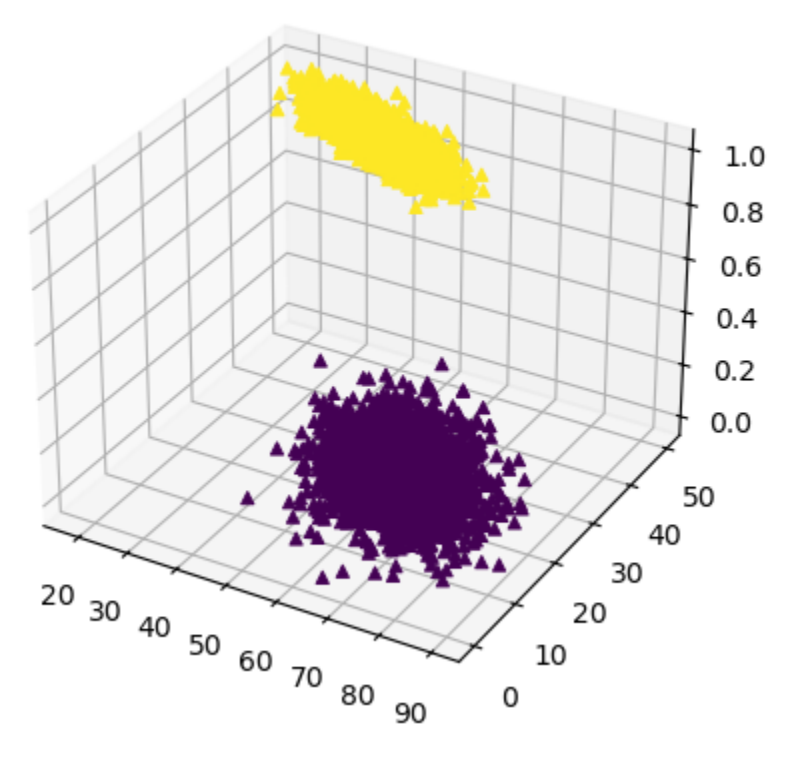

```
# heatmap plot
sns.heatmap(df.corr(), annot = True, linecolor = "white", square = "False")
plt.show()
```




```
# scatter plot
sns.scatterplot(x = "physical_score", data =df, y = "test_result", alpha = 1)
plt.show()
```

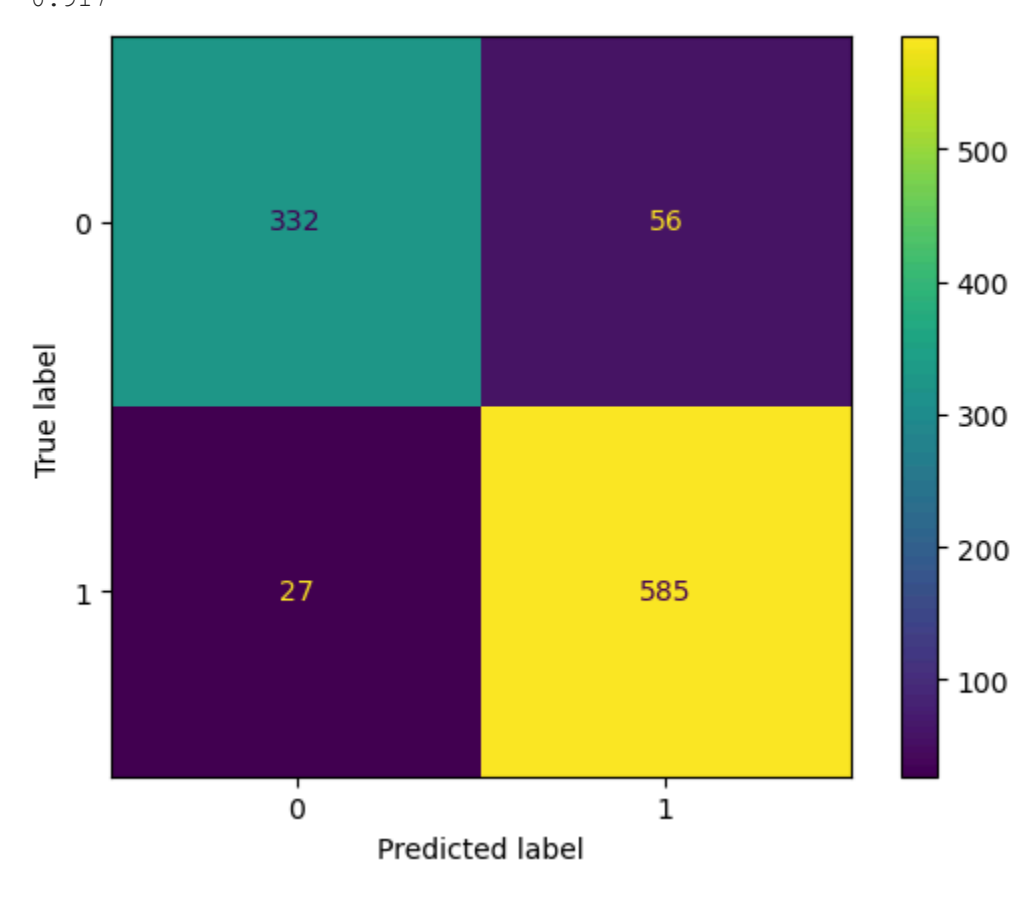



```
# three dimensional scatter plot

#Import Axes3D library
from mpl_toolkits.mplot3d import Axes3D

figure1 = plt.figure()

ax = figure1.add_subplot(111, projection = '3d')

ax.scatter( df["age"], df["physical_score"], df["test_result"], marker = "^", c = df["test_result"], alpha=1, linewidths = 0.7)
plt.show()
```

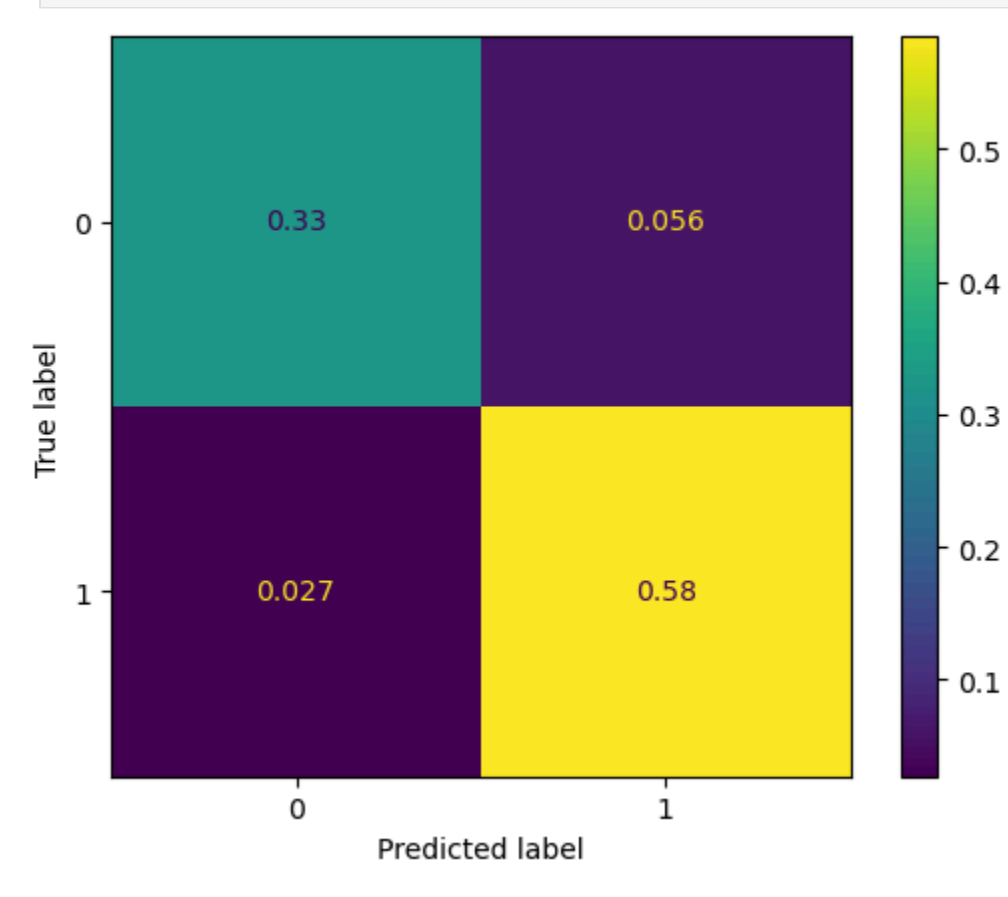


```
# Logistic Regression
# Use Scikit-Learn


X = df.drop('test_result', axis = 1)

y  = df['test_result']


from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 101)

# Use StandardScaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# fitting
X_train_scal = scaler.fit_transform(X_train)
X_test_scal = scaler.transform(X_test)

# Import Logistic Regression model
from sklearn.linear_model import LogisticRegression

# Initialize the model
logis_model = LogisticRegression()

# fit model
logis_model.fit(X_train_scal, y_train)

# Find coefficient
logis_model.coef_

print(log_model.coef_)

# prediction
prediction = logis_model.predict(X_test_scal)
print(prediction)

# prediction probability
prediction_probability = logis_model.predict_proba(X_test_scal)
print(prediction_probability)



# Import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# accuracy score
from sklearn.metrics import accuracy_score

print(accuracy_score(y_test, prediction))

# Calculate confusion matrix
confusion_matrix_1 = confusion_matrix(y_test, prediction)


# plot
ConfusionMatrix_Display = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix_1)
ConfusionMatrix_Display.plot()

plt.show()
```

```
[[-0.9039212   3.48681952]]
[1 1 0 1 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 0 1 1 0 1 1 0 1 1 1 1 1 1 0 1
 1 0 1 1 1 1 0 0 1 1 1 1 1 1 0 0 0 1 0 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0
 1 1 1 0 0 1 1 1 1 1 0 0 1 0 0 1 1 1 0 1 0 0 0 1 1 1 1 0 0 0 1 0 1 0 0 1 1
 0 0 1 0 0 0 1 1 0 1 0 0 1 1 0 1 0 1 0 1 1 1 1 0 1 1 0 0 0 0 1 0 1 1 1 0 0
 0 1 1 1 0 1 1 1 0 1 0 1 1 0 1 0 1 0 1 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 1 1 1
 0 1 1 0 1 1 0 0 0 1 1 1 1 0 1 1 0 1 0 1 1 1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1
 0 1 1 0 0 0 1 0 0 1 0 0 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 1 0 1 1 0 1 0 1 1 1
 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 0
 1 0 1 1 1 1 0 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 1 0 1 1 1 0 1 1 1 0 0 0
 1 0 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 1 1
 1 0 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 0 0 1 1 0
 1 0 0 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 1 1 1 1 1 0 0 1 0 1 1
 1 1 1 0 1 0 1 0 0 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 0 1
 1 0 0 1 0 1 1 0 1 0 0 1 1 1 1 1 0 0 0 1 0 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 1
 0 1 0 1 0 1 1 1 0 0 1 1 0 1 0 1 1 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 1 0 1 1 0
 0 1 1 1 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 0 0 1 0 1 1 1 1 1 1
 0 1 0 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1
 1 0 0 1 1 0 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 0 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1
 0 0 1 1 0 1 0 1 1 1 1 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 0
 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 1 0 1 1 0 1 0 1 0 0 1 0 1 1 1 1 0 1 0 1 1 1
 1 1 1 0 0 1 0 0 0 0 1 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0
 1 1 1 1 0 1 0 0 1 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1
 0 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 1 0 1 1 1 0 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0
 1 0 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0
 0 0 1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 1 0 1 1 1 1 0 1 1 0 1 0 1 1 1 1
 1 1 1 1 1 0 1 0 0 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1
 1 1 0 0 1 1 1 0 0 1 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 0 1 0 0 0 1 1 0 1 1 0
 1]
[[0.02436668 0.97563332]
 [0.02672574 0.97327426]
 [0.9893468  0.0106532 ]
 ...
 [0.02396838 0.97603162]
 [0.94905149 0.05094851]
 [0.22356825 0.77643175]]
0.917
```




```
# confusion matrix including  normalization
confusion_matrix_2 = confusion_matrix(y_test, prediction, normalize="all")

# plot
ConfusionMatrix_Display_2 = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix_2)
ConfusionMatrix_Display_2.plot()

plt.show()
```



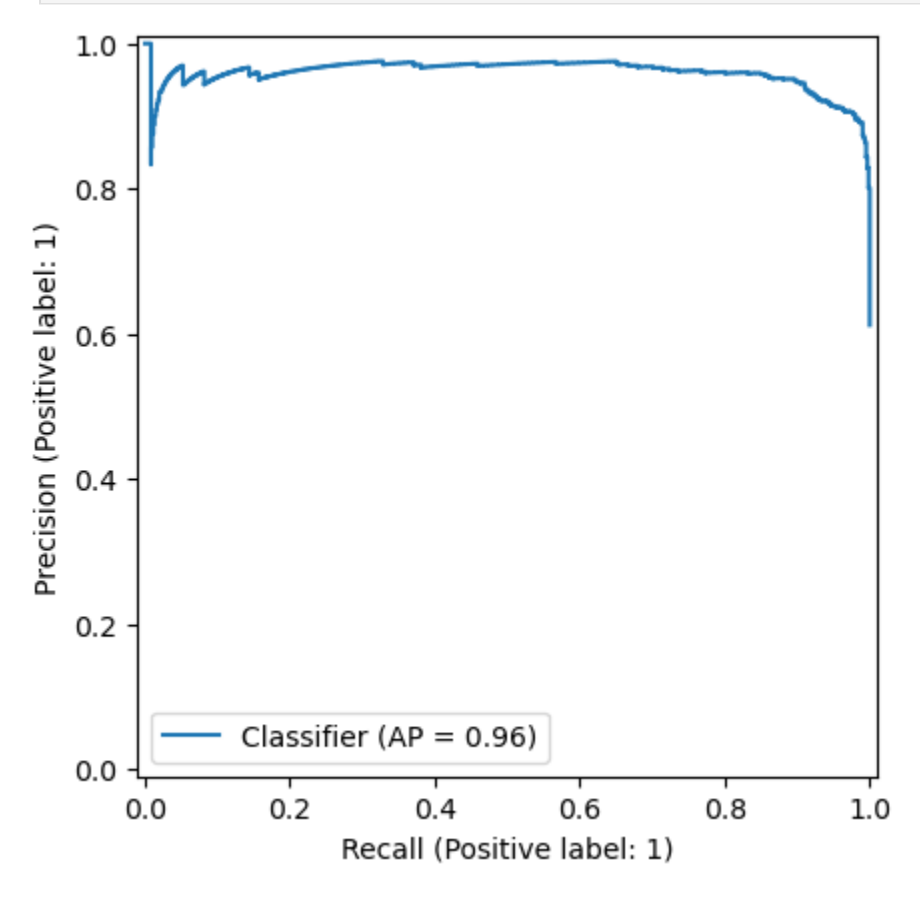

```
from sklearn.metrics import classification_report

classification_report(y_test, prediction)
print(classification_report(y_test, prediction))
```

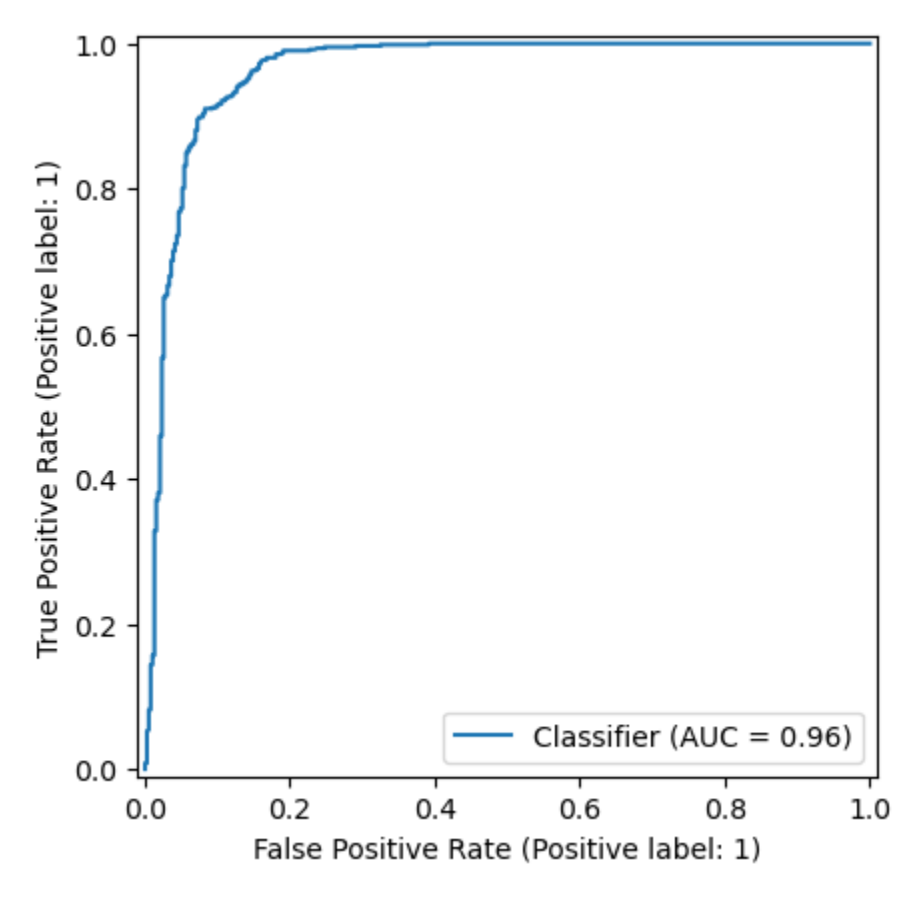
	precision	recall	f1-score	support
0	0.92	0.86	0.89	388
1	0.91	0.96	0.93	612
accuracy			0.92	1000
macro avg	0.92	0.91	0.91	1000
weighted avg	0.92	0.92	0.92	1000

```
from sklearn.metrics import precision_score, recall_score


print(precision_score(y_test, prediction))

print(recall_score(y_test, prediction))
```

```
0.9126365054602185
0.9558823529411765
```

```
# Load Precision Recall
from sklearn.metrics import PrecisionRecallDisplay

y_predict_proba = logis_model.predict_proba(X_test_scal)[:, 1]

PrecisionRecallDisplay.from_predictions(y_test, y_predict_proba)

plt.show()

# Load Roc Curve
from sklearn.metrics import RocCurveDisplay
RocCurveDisplay.from_predictions(y_test, y_predict_proba)
plt.show()
```






```
# Find first predicted probability
print(logis_model.predict_proba(X_test_scal))


# Look at second predicted probability
print(logis_model.predict_proba(X_test_scal)[1])
```

```
[[0.02436668 0.97563332]
 [0.02672574 0.97327426]
 [0.9893468  0.0106532 ]
 ...
 [0.02396838 0.97603162]
 [0.94905149 0.05094851]
 [0.22356825 0.77643175]]
[0.02672574 0.97327426]
```

In [ ]: