Logistic Regression

1 Logistic Regression

1.1 Imports

[30]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

1.2 Dados

Um experimento foi realizado em 5.000 participantes para estudar os efeitos da idade e da saúde física na perda auditiva, especificamente a capacidade de ouvir tons agudos. Esses dados mostram o resultado do estudo em que os participantes foram avaliados e pontuados quanto à capacidade física e, em seguida, tiveram que fazer um teste de áudio (passou/não passou) que avaliou sua capacidade de ouvir altas frequências. A idade do usuário também foi anotada. É possível construir um modelo que preveja a probabilidade de alguém ouvir o som de alta frequência com base apenas em suas características (idade e pontuação física)?

Características

age - Idade do participante em anos physical_score - Pontuação obtida durante o exame físico Rótulo/Alvo

test_result - 0 se não for aprovado, 1 se o teste for aprovado

1.3 Data

An experiment was conducted on 5000 participants to study the effects of age and physical health on hearing loss, specifically the ability to hear high pitched tones. This data displays the result of the study in which participants were evaluated and scored for physical ability and then had to take an audio test (pass/no pass) which evaluated their ability to hear high frequencies. The age of the user was also noted. Is it possible to build a model that would predict someone's liklihood to hear the high frequency sound based solely on their features (age and physical score)?

- Features
  - age - Age of participant in years
  - physical_score - Score achieved during physical exam
- Label/Target

– test_result - 0 if no pass, 1 if test passed

[31]:
```
df = pd.read_csv('../DATA/hearing_test.csv')
```

[32]:
```
df.head()
```

[32]:

	age	physical_score	test_result
0	33.0	40.7	1
1	50.0	37.2	1
2	52.0	24.7	0
3	56.0	31.0	0
4	35.0	42.9	1

1.3.1 Exploratory Data Analysis and Visualization

[33]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             5000 non-null   float64
 1   physical_score  5000 non-null   float64
 2   test_result     5000 non-null   int64
dtypes: float64(2), int64(1)
memory usage: 117.3 KB
```

[34]:
```
df.describe()
```

[34]:

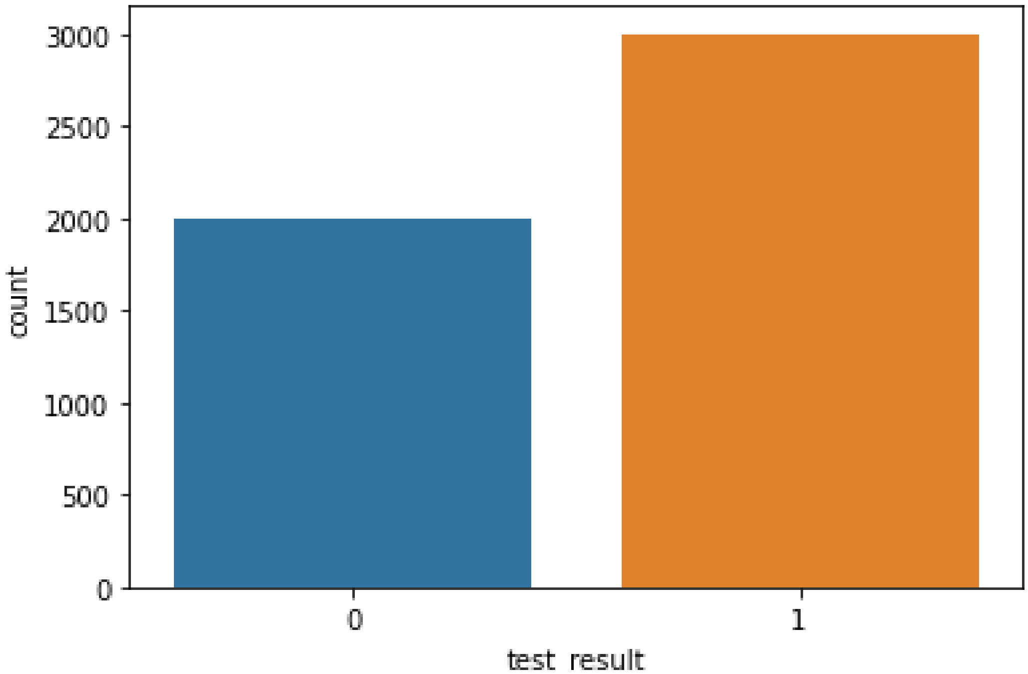
	age	physical_score	test_result
count	5000.000000	5000.000000	5000.000000
mean	51.609000	32.760260	0.600000
std	11.287001	8.169802	0.489947
min	18.000000	-0.000000	0.000000
25%	43.000000	26.700000	0.000000
50%	51.000000	35.300000	1.000000
75%	60.000000	38.900000	1.000000
max	90.000000	50.000000	1.000000

[35]:
```
df['test_result'].value_counts()
```

[35]:
```
1    3000
0    2000
Name: test_result, dtype: int64
```

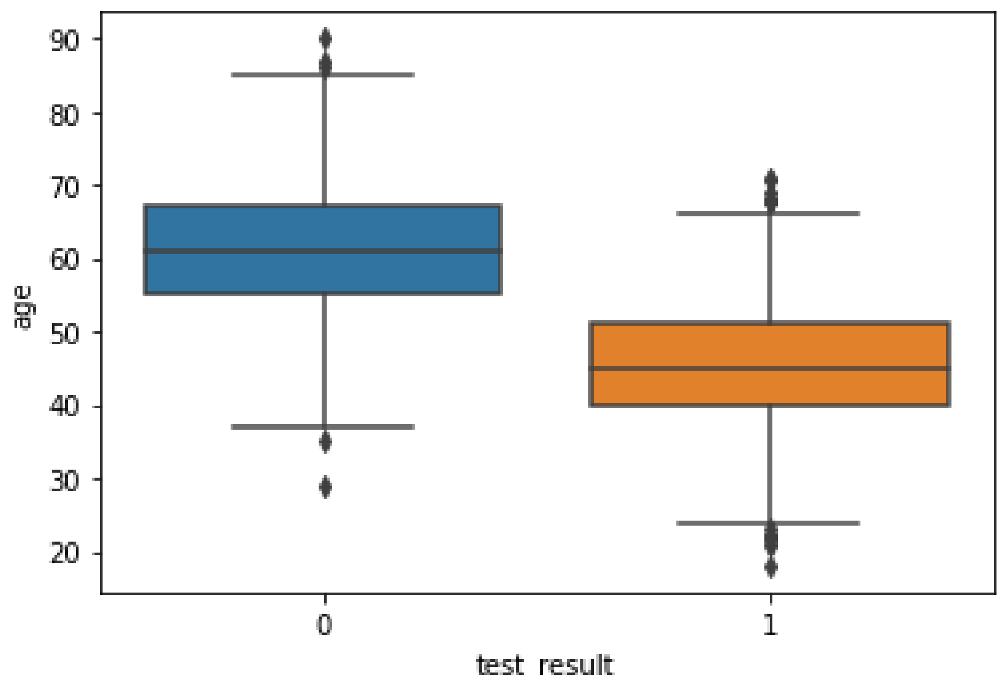
[36]:
```
sns.countplot(data=df,x='test_result')
```

[36]: <AxesSubplot:xlabel='test_result', ylabel='count'>



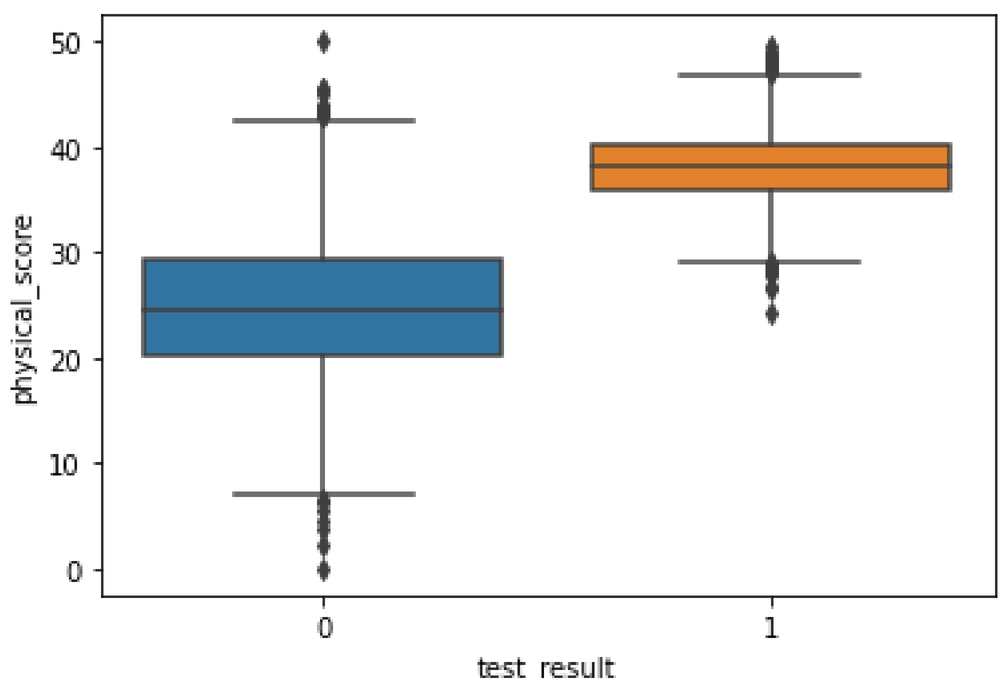
[37]:
```
sns.boxplot(x='test_result',y='age',data=df)
```

[37]: <AxesSubplot:xlabel='test_result', ylabel='age'>



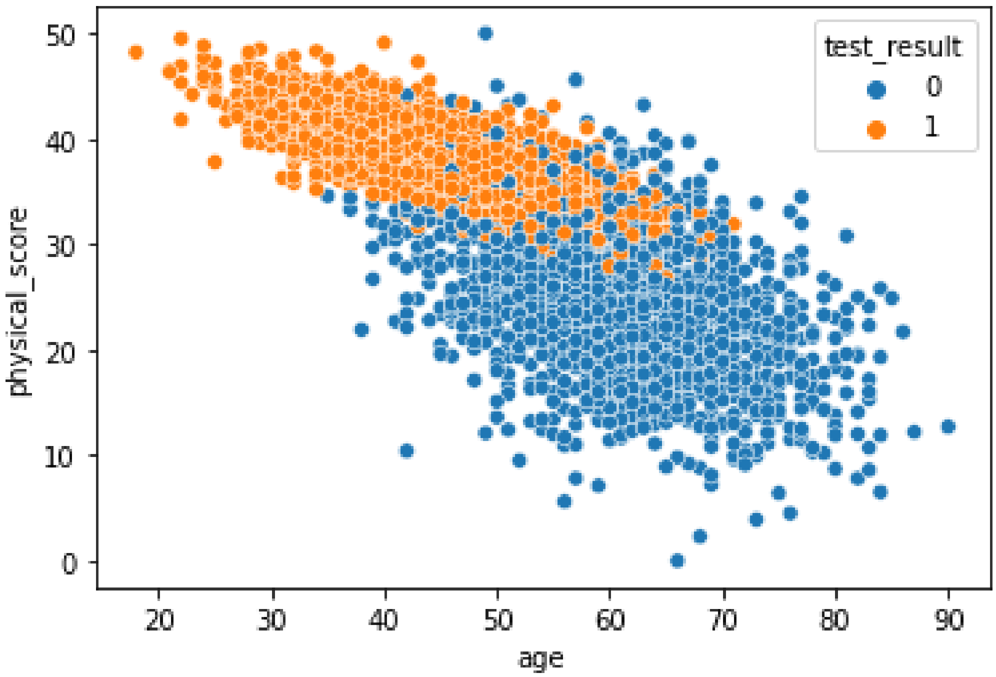

```
[38]: sns.boxplot(x='test_result',y='physical_score',data=df)
```

[38]: <AxesSubplot:xlabel='test_result', ylabel='physical_score'>



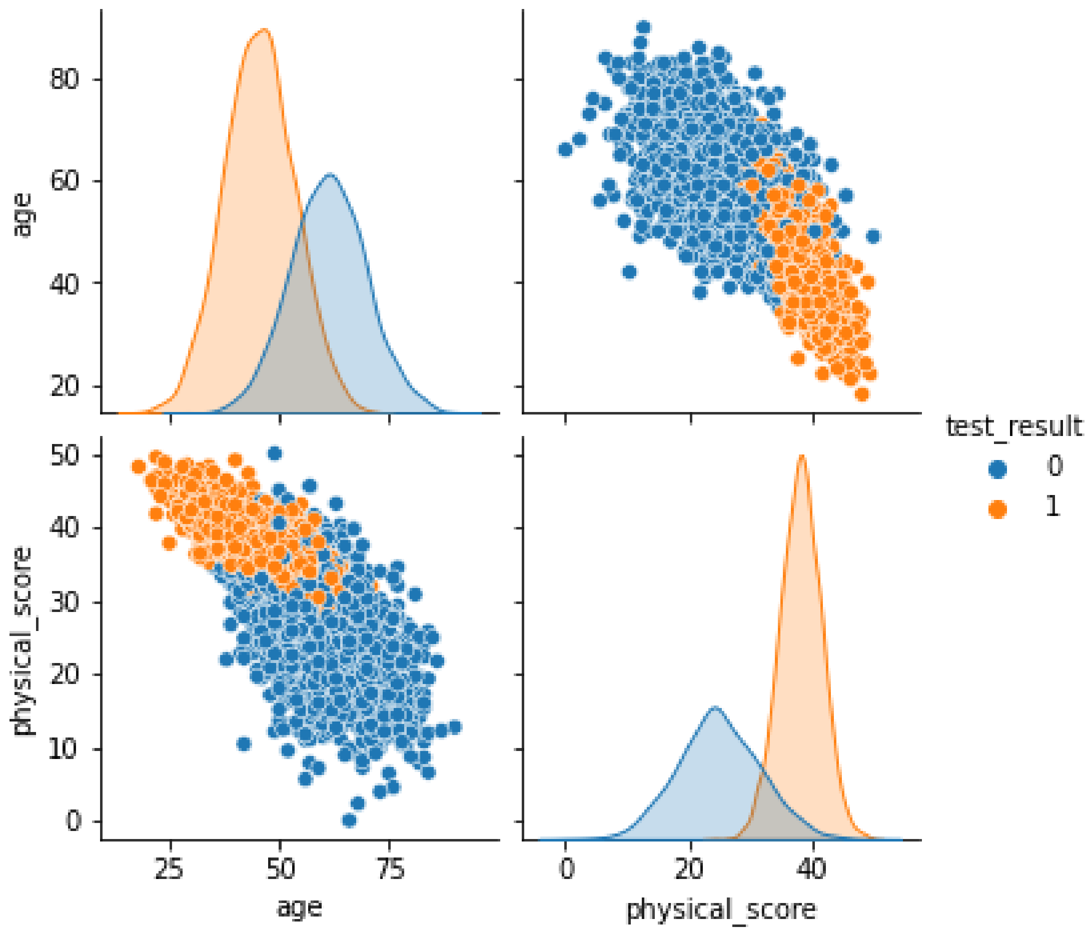
[39]:
```
sns.scatterplot(x='age',y='physical_score',data=df,hue='test_result')
```

[39]: <AxesSubplot:xlabel='age', ylabel='physical_score'>



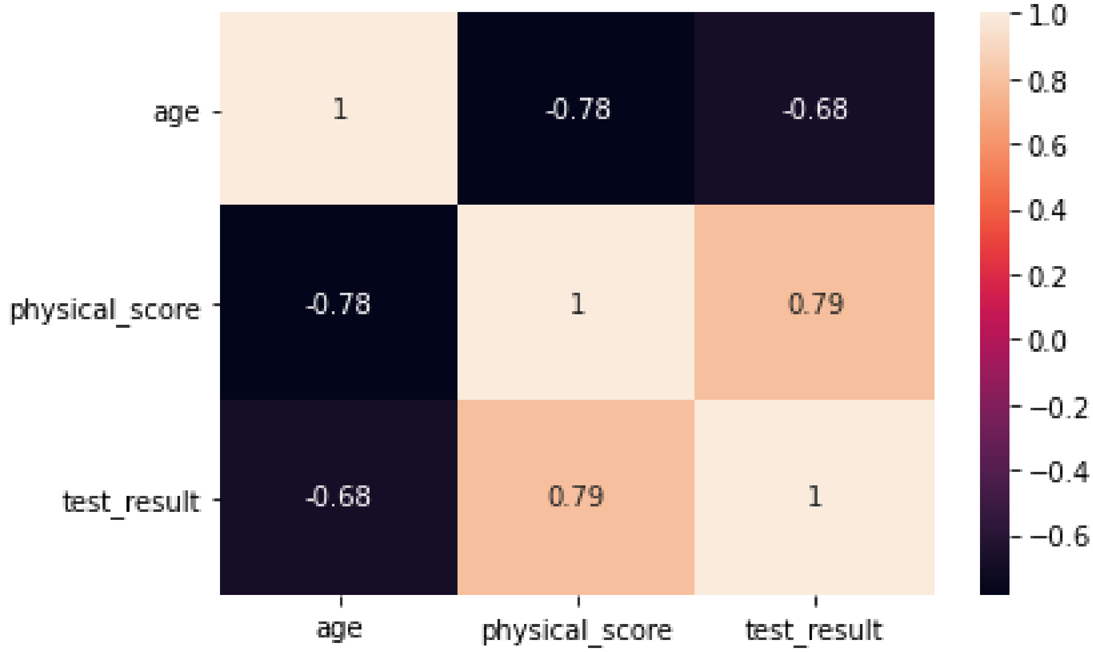
[40]:
```
sns.pairplot(df,hue='test_result')
```

[40]: <seaborn.axisgrid.PairGrid at 0x19ceae2fd08>



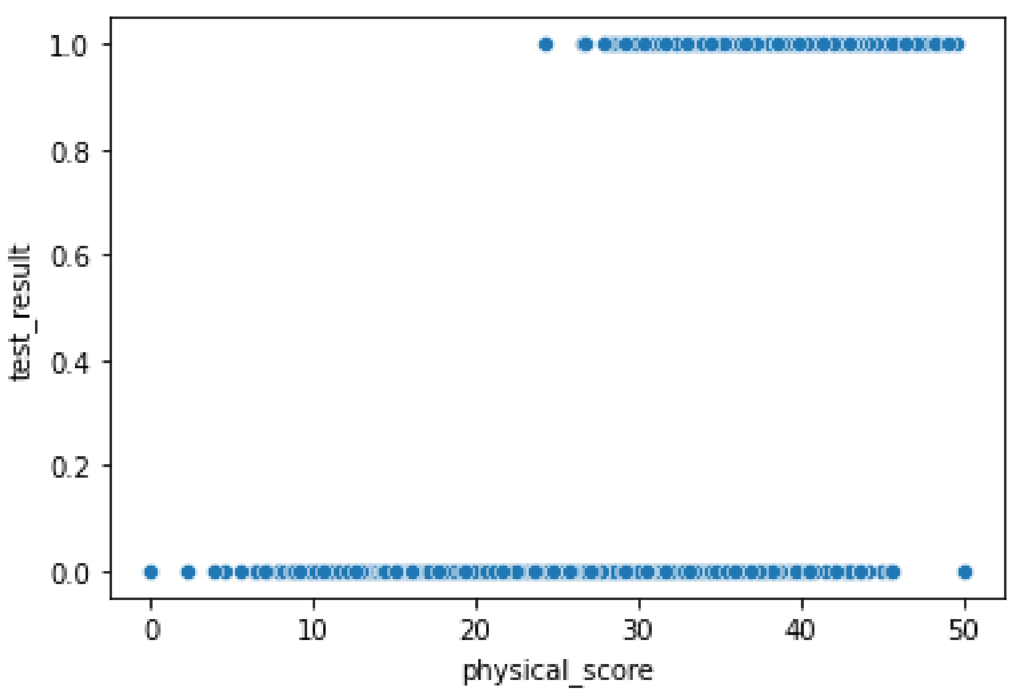
[41]:
```
sns.heatmap(df.corr(),annot=True)
```

[41]: <AxesSubplot:>



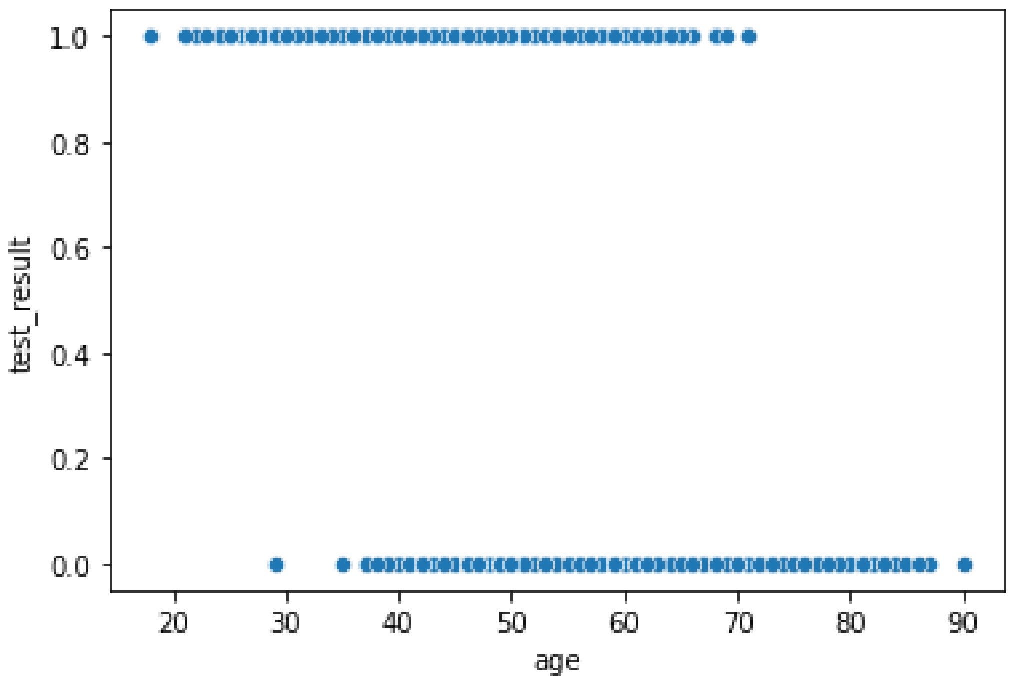
[42]:
```
sns.scatterplot(x='physical_score',y='test_result',data=df)
```

[42]: <AxesSubplot:xlabel='physical_score', ylabel='test_result'>



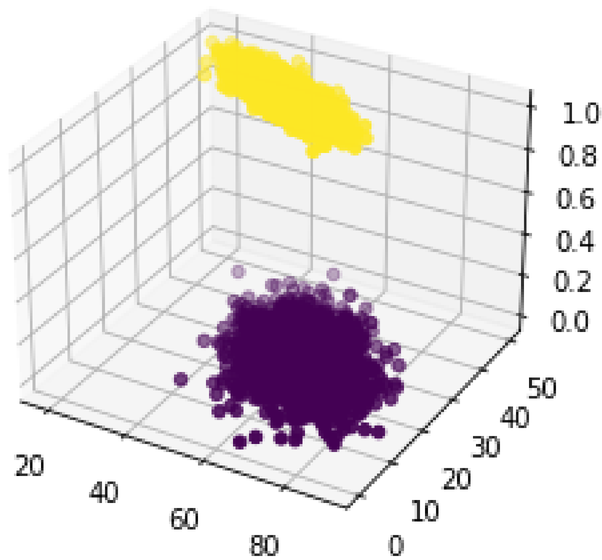
[43]:
```
sns.scatterplot(x='age',y='test_result',data=df)
```

[43]: <AxesSubplot:xlabel='age', ylabel='test_result'>



[44]:
```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['age'],df['physical_score'],df['test_result'],c=df['test_result'])
```

[44]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x19ceaf878c8>



1.3.2 Train | Test Split and Scaling

[45]:
```
X = df.drop('test_result',axis=1)
y = df['test_result']
```

[46]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

[47]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,␣
↪random_state=101)
```

[48]:
```
scaler = StandardScaler()
```

[49]:
```
scaled_X_train = scaler.fit_transform(X_train)
scaled_X_test = scaler.transform(X_test)
```

1.4 Logistic Regression Model

[50]:
```
from sklearn.linear_model import LogisticRegression
```

[51]:
```
# help(LogisticRegression)
```

[52]:
```
# help(LogisticRegressionCV)
```

[53]:
```
log_model = LogisticRegression()
```

[54]:
```
log_model.fit(scaled_X_train,y_train)
```

[54]:
```
LogisticRegression()
```

1.4.1 Coefficient Interpretation

Things to remember:

- These coeffecients relate to the *odds* and can not be directly interpreted as in linear regression.
- We trained on a *scaled* version of the data
- It is much easier to understand and interpret the relationship between the coefficients than it is to interpret the coefficients relationship with the probability of the target/label class.

1.4.2 The odds ratio

For a continuous independent variable the odds ratio can be defined as:

1.4.3 Interpretação do Coeficiente

Coisas para lembrar:

- Esses coeficientes referem-se às *probabilidades* e não podem ser interpretados diretamente como na regressão linear.
- Treinamos em uma versão *em escala* dos dados
- É muito mais fácil entender e interpretar a relação entre os coeficientes do que interpretar a relação dos coeficientes com a probabilidade da classe alvo/rótulo.

1.4.4 A razão de chances

Para uma variável independente contínua, a razão de chances pode ser definida como:

This exponential relationship provides an interpretation for

$$\beta_1$$

The odds multiply by

$$e^\beta_1$$

for every 1-unit increase in x.

[55]:
```
log_model.coef_
```

[55]:
```
array([[-0.94953524,  3.45991194]])
```

Isso significa:

- Podemos esperar que as chances de passar no teste diminuam (o coeficiente original era negativo) por unidade de aumento da idade.
- Podemos esperar que as chances de passar no teste aumentem (o coeficiente original era positivo) por unidade de aumento da pontuação física.
- Com base nas proporções entre si, o indicador physical_score é um preditor mais forte do que a idade.

This means: * We can expect the **odds** of passing the test to **decrease** (the original coeff was negative) per unit increase of the age. * We can expect the **odds** of passing the test to **increase** (the original coeff was positive) per unit increase of the physical score. * Based on the ratios with each other, the physical_score indicator is a stronger predictor than age.

1.4.5 Model Performance on Classification Tasks

[56]:
```
from sklearn.metrics import␣
↪accuracy_score,confusion_matrix,classification_report,plot_confusion_matrix
```

[57]:
```
y_pred = log_model.predict(scaled_X_test)
```

[58]:
```
accuracy_score(y_test,y_pred)
```

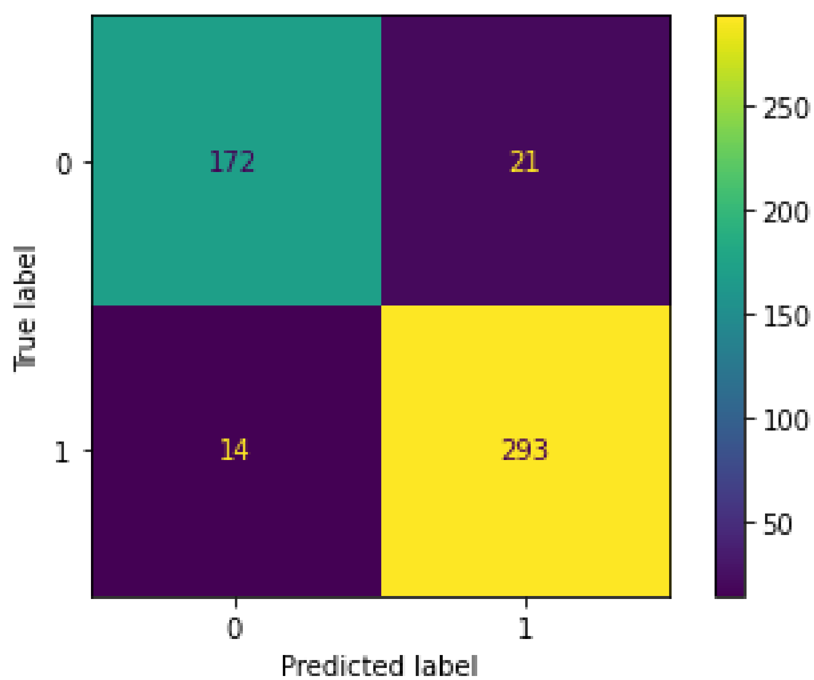
[58]:
```
0.93
```

[59]:
```
confusion_matrix(y_test,y_pred)
```

[59]:
```
array([[172,  21],
       [ 14, 293]], dtype=int64)
```

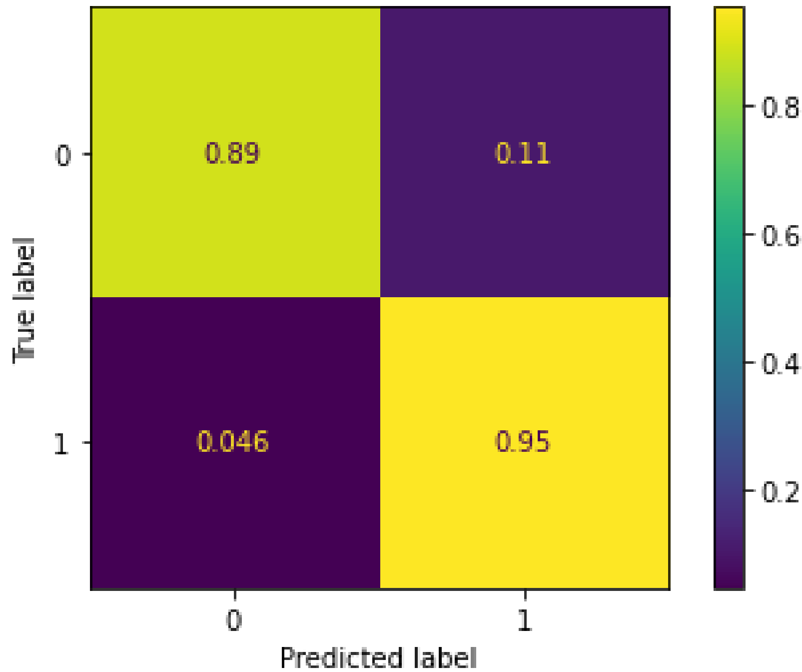
[60]:
```
plot_confusion_matrix(log_model,scaled_X_test,y_test)
```

[60]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x19ceb65e588>
```



[61]:
```python
# Scaled so highest value=1
plot_confusion_matrix(log_model,scaled_X_test,y_test,normalize='true')
```

[61]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x19ceb691b88>



[62]:
```python
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.89      0.91       193
           1       0.93      0.95      0.94       307

    accuracy                           0.93       500
   macro avg       0.93      0.92      0.93       500
weighted avg       0.93      0.93      0.93       500
```

[63]:
```python
X_train.iloc[0]
```

[63]:
```
age               32.0
physical_score    43.0
Name: 141, dtype: float64
```

[64]:
```python
y_train.iloc[0]
```

[64]: 1

2 0% de probabilidade de classe 0

3 100% de probabilidade de 1 classe

[65]:
```
# 0% probability of 0 class
# 100% probability of 1 class
log_model.predict_proba(X_train.iloc[0].values.reshape(1, -1))
```

[65]: array([[0., 1.]])

[66]:
```
log_model.predict(X_train.iloc[0].values.reshape(1, -1))
```

[66]: array([1], dtype=int64)

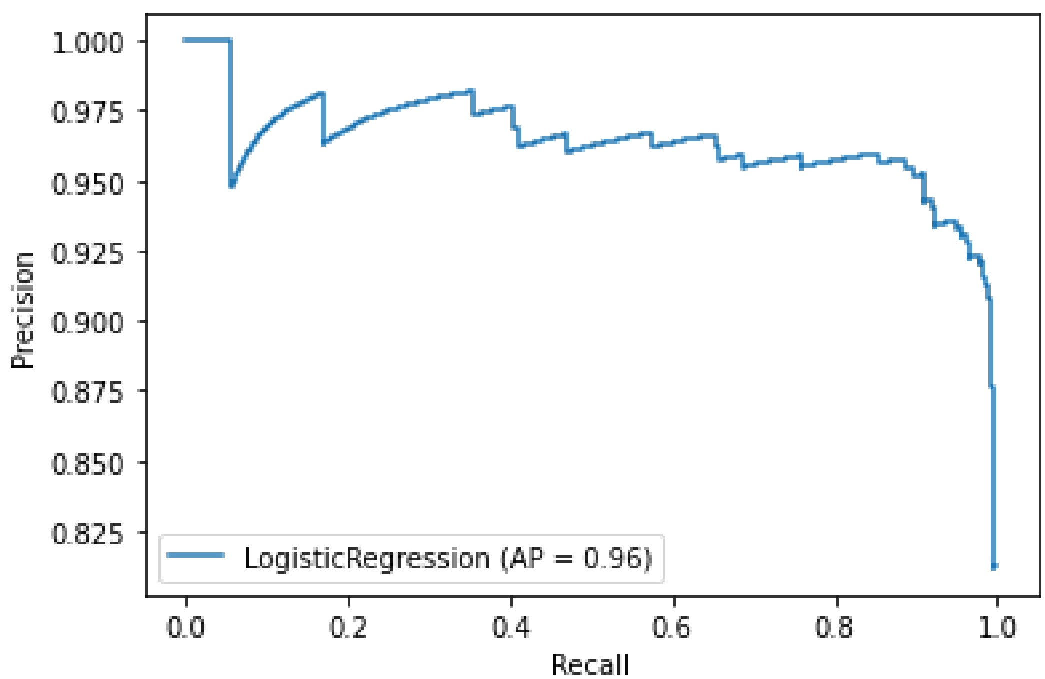
3.1 Evaluating Curves and AUC

Make sure to watch the video on this!

[67]:
```
from sklearn.metrics import␣
↪precision_recall_curve,plot_precision_recall_curve,plot_roc_curve
```

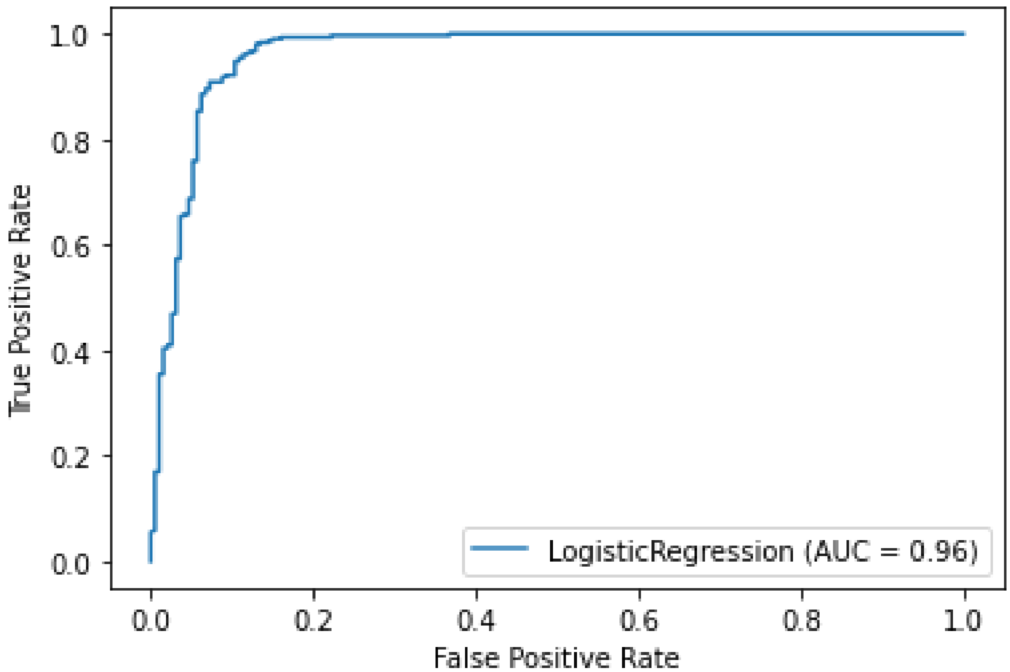
[70]:
```
plot_precision_recall_curve(log_model,scaled_X_test,y_test)
```

[70]: <sklearn.metrics._plot.precision_recall_curve.PrecisionRecallDisplay at 0x19cec76dac8>



[71]:
```
plot_roc_curve(log_model,scaled_X_test,y_test)
```

[71]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x19ceb5c4288>



3.2 ———