Logistic Regression

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

In [1]:
```
# Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('hearing_test.csv')
df.head()
```

Out[2]:

	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

Exploratory Data Analysis and Visualization

Feel free to explore the data further on your own.

In [3]:
```
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
```

In [4]:
```
df.describe()
```

Out[4]:

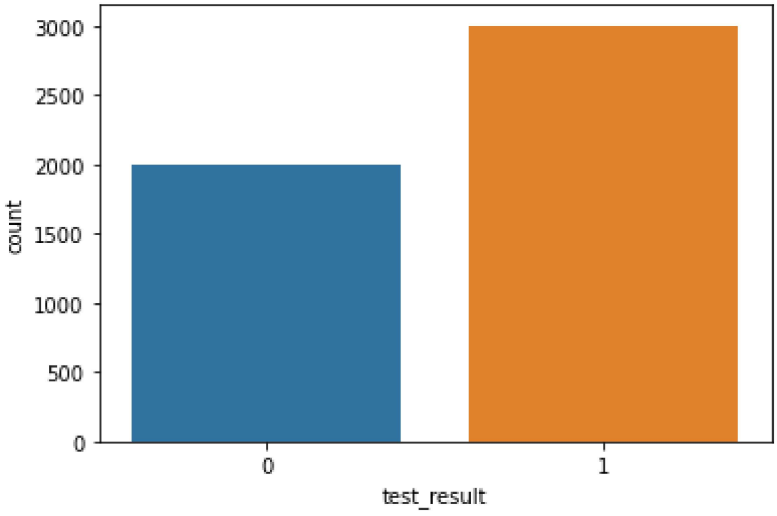
	age	physical_score	test_result
count	5000.000000	5000.000000	5000.000000
mean	51.609000	32.760260	0.600000
std	11.287001	8.169802	0.489947
min	18.000000	-0.000000	0.000000
25%	43.000000	26.700000	0.000000
50%	51.000000	35.300000	1.000000
75%	60.000000	38.900000	1.000000
max	90.000000	50.000000	1.000000

In [5]:
```
df['test_result'].value_counts()
```

Out[5]:
```
1    3000
0    2000
Name: test_result, dtype: int64
```

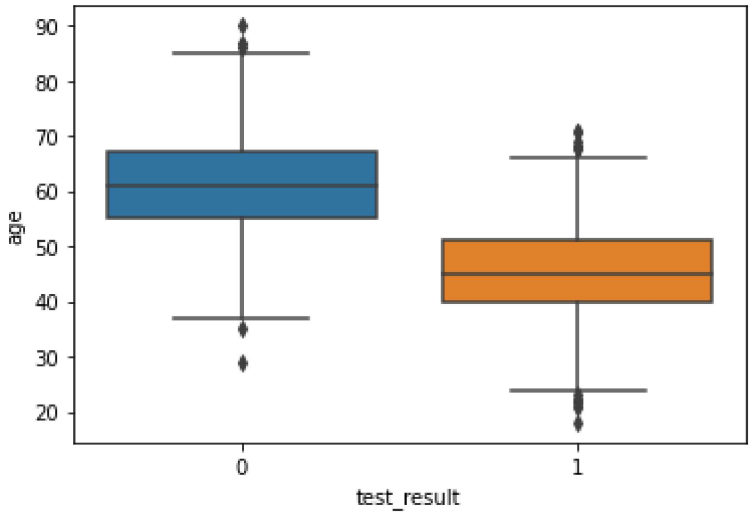
In [6]:
```
sns.countplot(data=df,x='test_result')
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21e18dad1f0>
```



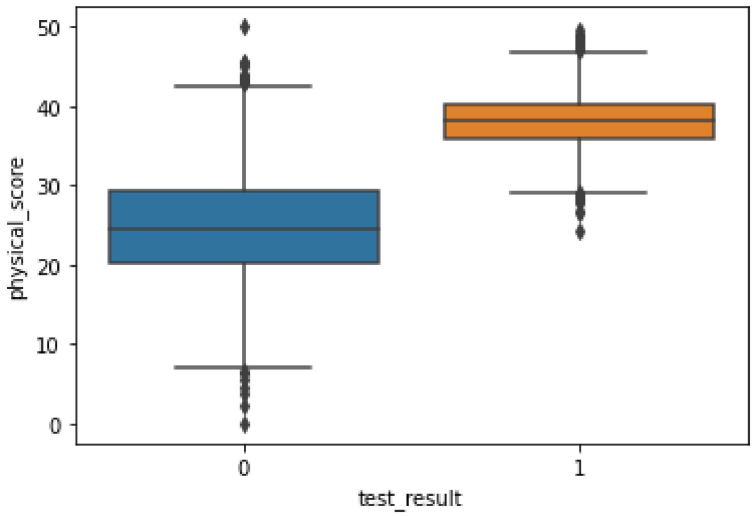
In [7]:
```
sns.boxplot(x='test_result',y='age',data=df)
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x21e194e6520>



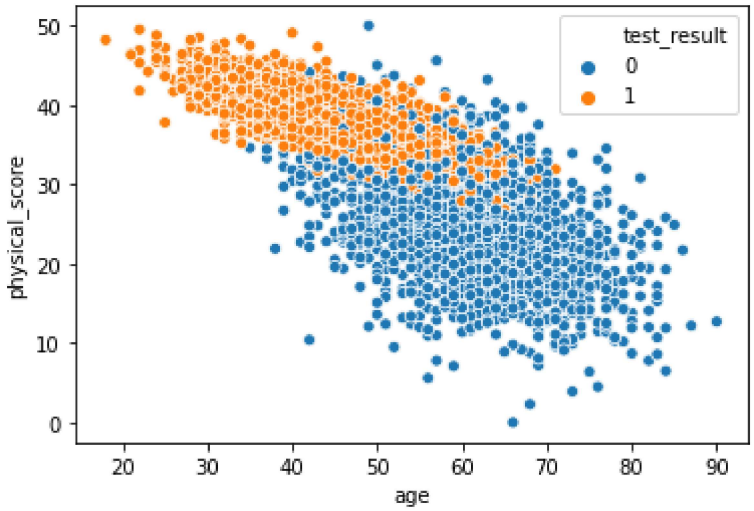
In [8]:
```
sns.boxplot(x='test_result',y='physical_score',data=df)
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x21e1956a820>



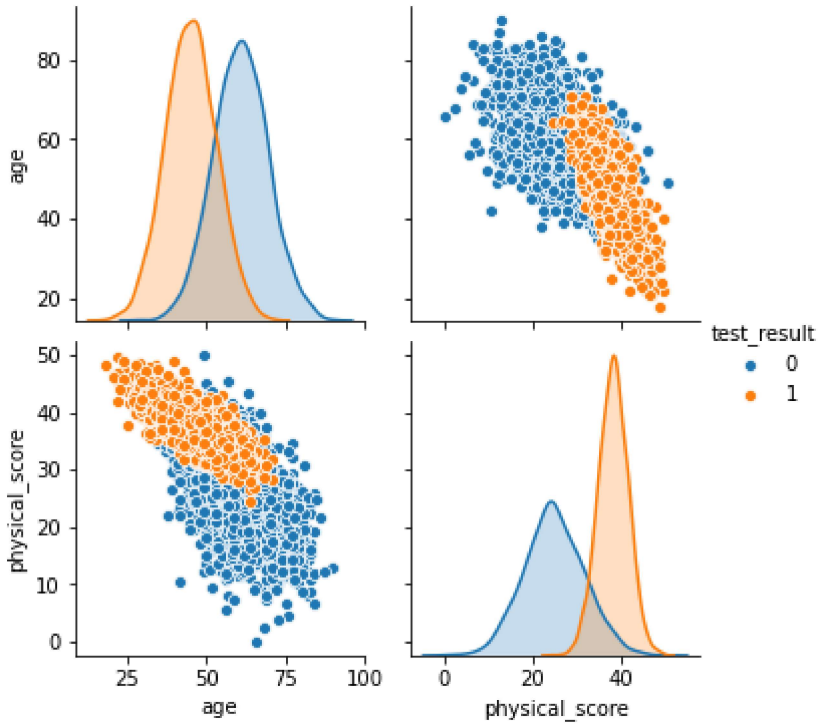
In [9]:
```
sns.scatterplot(x='age',y='physical_score',data=df,hue='test_result')
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x21e195e42e0>



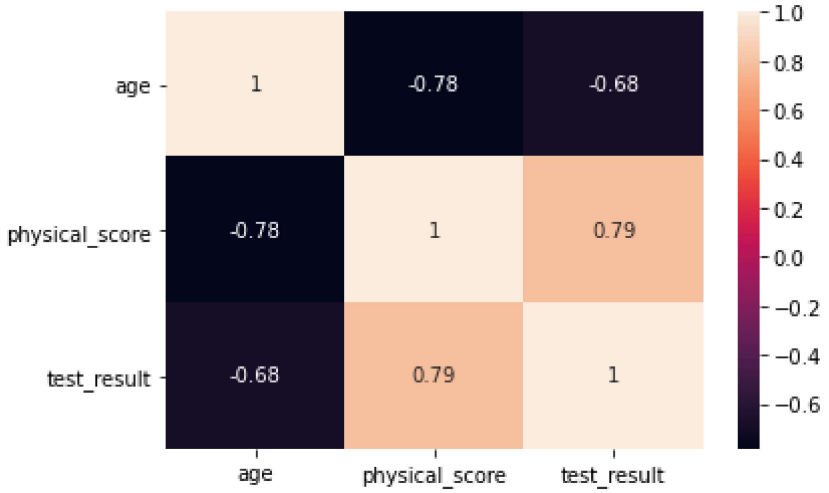
In [10]:
```
sns.pairplot(df,hue='test_result')
```

Out[10]: <seaborn.axisgrid.PairGrid at 0x21e19666400>



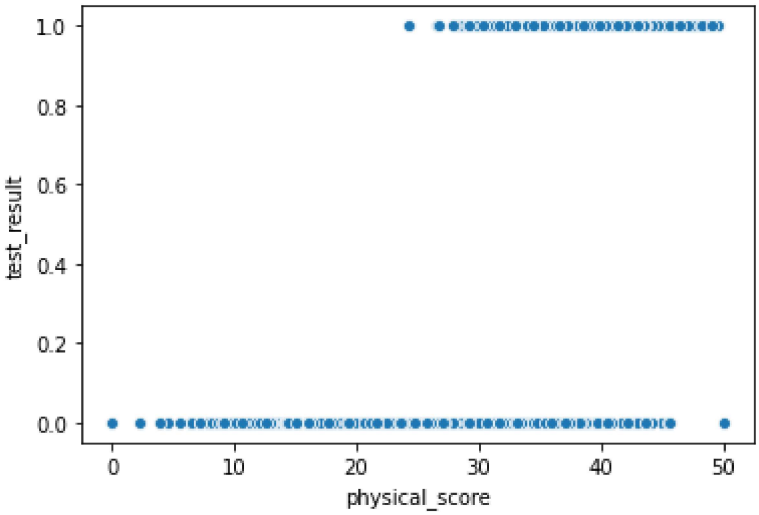
In [11]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x21e1987d2b0>



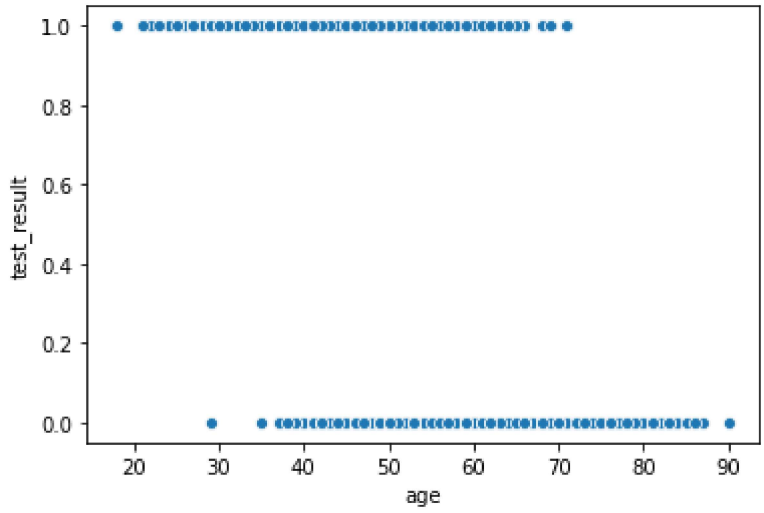
In [12]:
```
sns.scatterplot(x='physical_score',y='test_result',data=df)
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x21e19930d00>



In [13]:
```
sns.scatterplot(x='age',y='test_result',data=df)
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21e19958df0>
```



X & y

In [14]:
```
X = df.drop('test_result',axis=1)
y = df['test_result']
```

Train | Test Split

In [15]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando
m_state=101)
```

Scaling

In [16]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

scaled_X_train = scaler.fit_transform(X_train)
scaled_X_test = scaler.transform(X_test)
```

Logistic Regression Model

In [17]:
```
from sklearn.linear_model import LogisticRegression

log_model = LogisticRegression()

log_model.fit(scaled_X_train,y_train)
```

Out[17]:
```
LogisticRegression()
```

In [18]:
```
log_model.coef_
```

Out[18]:
```
array([[-0.91653034,  3.4506941 ]])
```

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.

Model Performance on Classification Tasks

In [19]:
```
y_pred = log_model.predict(scaled_X_test)
y_pred
```

Out[19]:
```
array([1, 1, 0, ..., 0, 1, 1], dtype=int64)
```

In [20]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
ort,plot_confusion_matrix
```

In [21]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

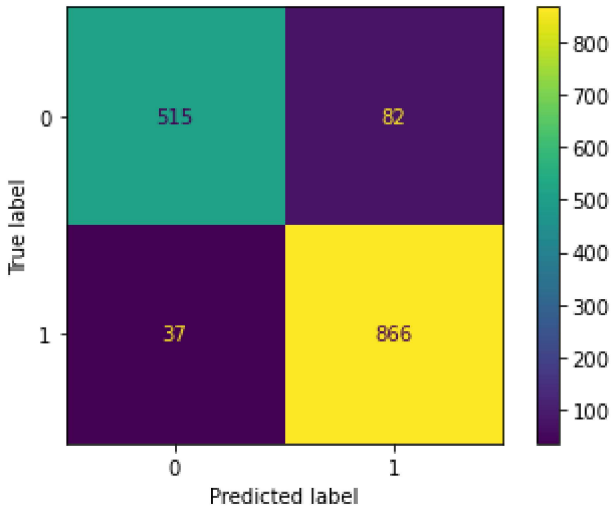
Out[21]:
```
0.9206666666666666
```

In [22]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[22]:
```
array([[515,  82],
       [ 37, 866]], dtype=int64)
```

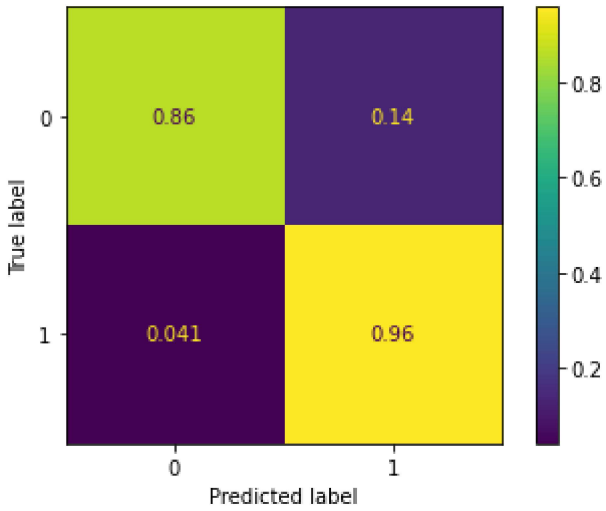
In [23]:
```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(log_model,scaled_X_test,y_test)
```

Out[23]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x21e199004
90>
```



In [24]:
```
# Scaled so highest value=1
plot_confusion_matrix(log_model,scaled_X_test,y_test,normalize='true')
```

Out[24]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x21e1966b5
80>
```



In [25]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.86      0.90       597
           1       0.91      0.96      0.94       903

    accuracy                           0.92      1500
   macro avg       0.92      0.91      0.92      1500
weighted avg       0.92      0.92      0.92      1500
```

AUC

In [26]:
```
from sklearn.metrics import plot_roc_curve
plot_roc_curve(log_model,scaled_X_test,y_test)
```

Out[26]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x21e199c00a0>