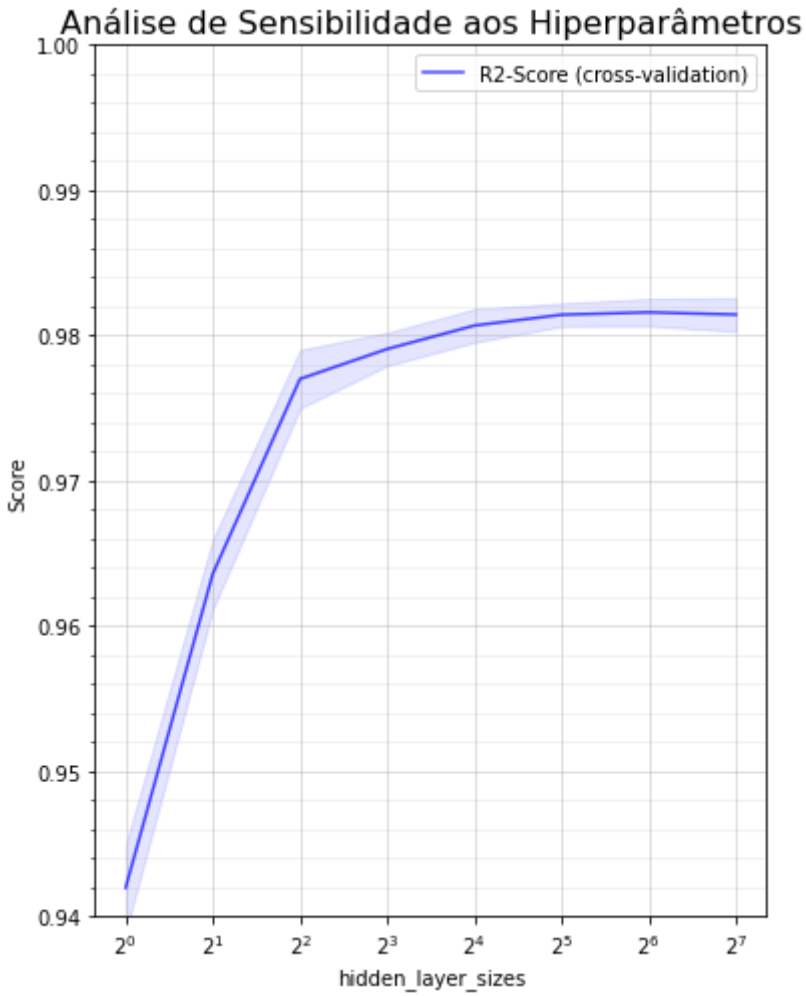
In [1]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.simplefilter(action='ignore')#, category=FutureWarning)
```

In [2]:
```
pd.set_option('display.max_columns', None)
pd.set_option("max_rows", None)
#pd.reset_option("max_columns")
#pd.reset_option("max_rows")
```

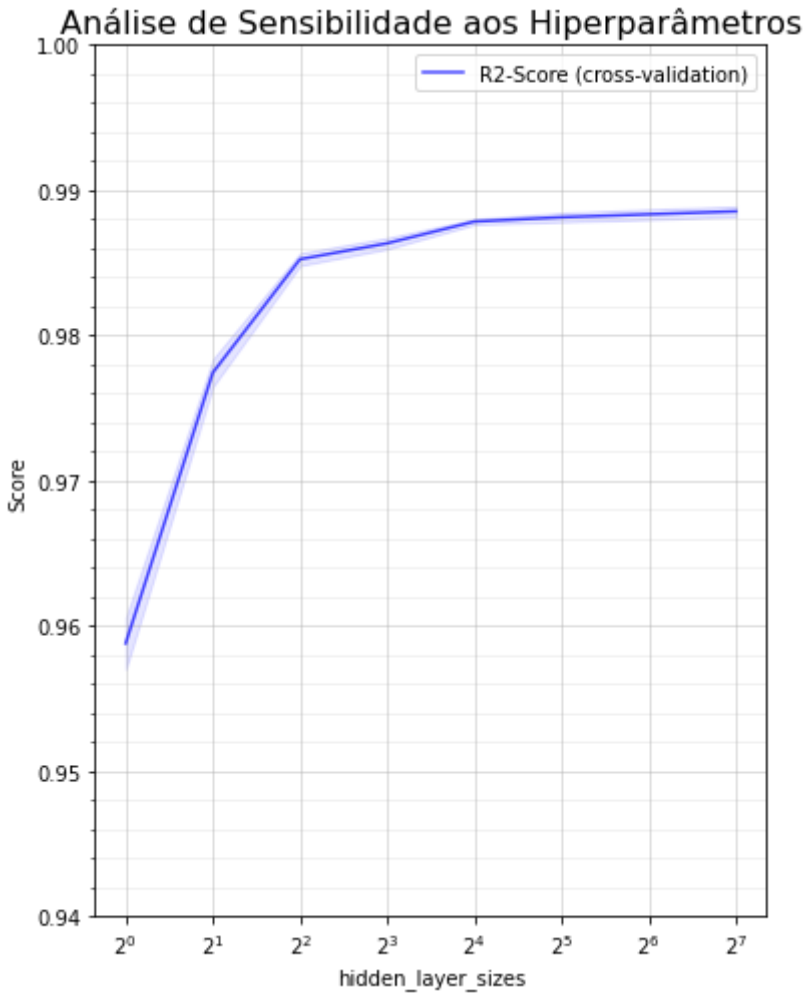
In [3]:
```
def plotar (plot_x, plot_y, plot_y_err, parameter, log=False, base=10, yinf=0.9, aut

    plt.figure(figsize=(6, 8))
    plt.title("Análise de Sensibilidade aos Hiperparâmetros", fontsize=16)

    plt.xlabel(parameter)
    plt.ylabel("Score")

    ax = plt.gca()
    #ax.set_xlim(0, 402)
    ax.set_ylim(yinf, 1)

    if autoxtick==False:
        x_ticks = plot_x
        x_ticks_minor = plot_x

        ax.set_xticks(x_ticks)
        ax.set_xticks(x_ticks_minor, minor=True)

    y_ticks = np.arange(yinf, 1.001, 0.01)
    y_ticks_minor = np.arange(yinf, 1.00, 0.002)
    ax.set_yticks(y_ticks)
    ax.set_yticks(y_ticks_minor, minor=True)
    ax.grid(which='both')
    ax.grid(which='minor', alpha=0.2)
    ax.grid(which='major', alpha=0.5)

    if(log):
        ax.set_xscale('log', basex=base)


    #X_axis = np.array(plot_x.data, dtype=object)
    X_axis = np.array(plot_x.data , dtype=float)
    sample_score_mean = np.array(plot_y.data, dtype=float)
    sample_score_std = np.array(plot_y_err.data , dtype=float)


    ax.fill_between(X_axis, sample_score_mean - sample_score_std,
                            sample_score_mean + sample_score_std, color='b', alpha=0

    ax.plot(X_axis, sample_score_mean, color='b',
                    alpha=0.7,
                    label="R2-Score (cross-validation)")


    plt.legend(loc="best")
    plt.grid(True)
    plt.show()
```

In [4]:
```
def plot_category (plot_x, plot_y, plot_y_err, parameter, yinf=0.9):
    plt.figure(figsize=(6, 8))
    plt.title("Análise de Sensibilidade aos Hiperparâmetros", fontsize=16)

    plt.xlabel(parameter)
    plt.ylabel("Score")

    ax = plt.gca()
    #ax.set_xlim(0, 402)
    ax.set_ylim(yinf, 1)

    #if autoxtick==False:
    #    x_ticks = plot_x
    #    x_ticks_minor = plot_x

    #    ax.set_xticks(x_ticks)
    #    ax.set_xticks(x_ticks_minor, minor=True)

    y_ticks = np.arange(yinf, 1.001, 0.01)
    y_ticks_minor = np.arange(yinf, 1.00, 0.002)
    ax.set_yticks(y_ticks)
    ax.set_yticks(y_ticks_minor, minor=True)
    ax.grid(which='both')
    ax.grid(which='minor', alpha=0.2)
    ax.grid(which='major', alpha=0.5)

    #if(log):
    #    ax.set_xscale('log', basex=base)


    #X_axis = np.array(plot_x.data, dtype=object)

    X_axis = list(plot_x)
    sample_score_mean = np.array(plot_y.data, dtype=float)
    sample_score_std = np.array(plot_y_err.data , dtype=float)

    x_pos = np.arange(len(X_axis))
    y_err = sample_score_std
    #y_err = [0.001, 0.002, 0.003]
    ax.bar(x_pos, sample_score_mean, yerr=y_err, align='center', alpha=0.7, ecolor='
    #ax.bar(x_pos, sample_score_mean, y_err, align='center', alpha=0.5, ecolor='blac
    ax.set_xticks(x_pos)
    ax.set_xticklabels(X_axis)

    plt.legend(loc="best")
    plt.grid(True)
    plt.show()
```

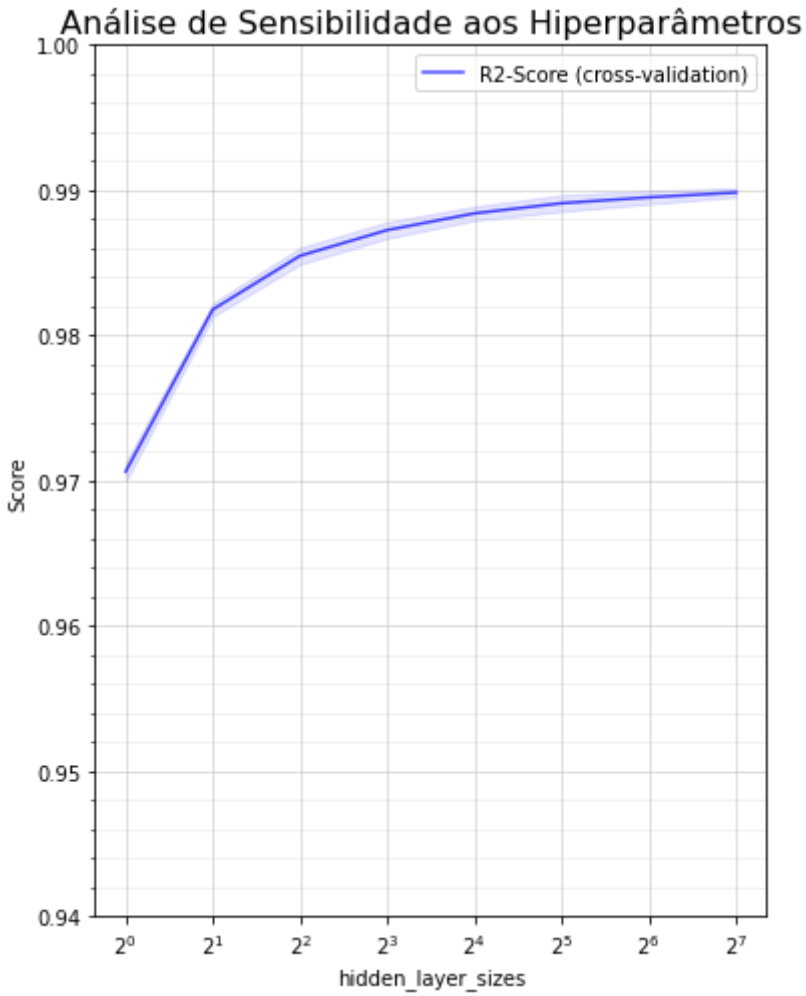
In [5]:
```
#results_linear
results_lin = pd.read_pickle("results_8_linear")
parameter = 'hidden_layer_sizes'
#plot_x = results_lin['param_hidden_layer_sizes']
plot_x = pd.Series([1, 2, 4, 8, 16, 32, 64, 128])
plot_y = results_lin['mean_test_score']
plot_y_err = results_lin['std_test_score']

plot_x.describe()

plotar(plot_x, plot_y, plot_y_err, parameter, log=True, base=2, yinf=0.94)
```



In [6]:
```
#results_sqrt
results_sqrt = pd.read_pickle("results_8_sqrt")
parameter = 'hidden_layer_sizes'
#plot_x = results_sqrt['param_hidden_layer_sizes']
plot_x = pd.Series([1, 2, 4, 8, 16, 32, 64, 128])
plot_y = results_sqrt['mean_test_score']
plot_y_err = results_sqrt['std_test_score']

plot_x.describe()

plotar(plot_x, plot_y, plot_y_err, parameter, log=True, base=2, yinf=0.94)
```



In [7]:
```
#results_log
results_log = pd.read_pickle("results_8_log")
parameter = 'hidden_layer_sizes'
plot_x = pd.Series([1, 2, 4, 8, 16, 32, 64, 128])
plot_y = results_log['mean_test_score']
plot_y_err = results_log['std_test_score']
results_log
plotar(plot_x, plot_y, plot_y_err, parameter, log=True, base=2, yinf=0.94)
```



In [8]:
```
results = pd.read_pickle("results_72")

solver = results[results['param_solver'].notnull()]
learning_rate_init = results[results['param_learning_rate_init'].notnull()]
activation = results[results['param_activation'].notnull()]
early_stopping = results[results['param_validation_fraction'].notnull()]
alpha = results[results['param_alpha'].notnull()]
tol = results[27:35]
layer1 = results[35:45]
layer2 = results[45:55]
layer3 = results[55:63]
layer4 = results[63:]

learning_rate_init = learning_rate_init.sort_values('param_learning_rate_init').rese
early_stopping = early_stopping.sort_values('param_validation_fraction').reset_index
alpha = alpha.sort_values('param_alpha').reset_index()
tol = tol.sort_values('param_tol').reset_index()

results
```

Out[8]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_hidden_layer_sizes	param_m
0	4.624305	0.543939	0.006001	4.238221e-07	(32, 32)	
1	3.441179	0.330590	0.006802	3.997566e-04	(32, 32)	
2	21.229384	10.845027	0.027405	2.498817e-03	NaN	

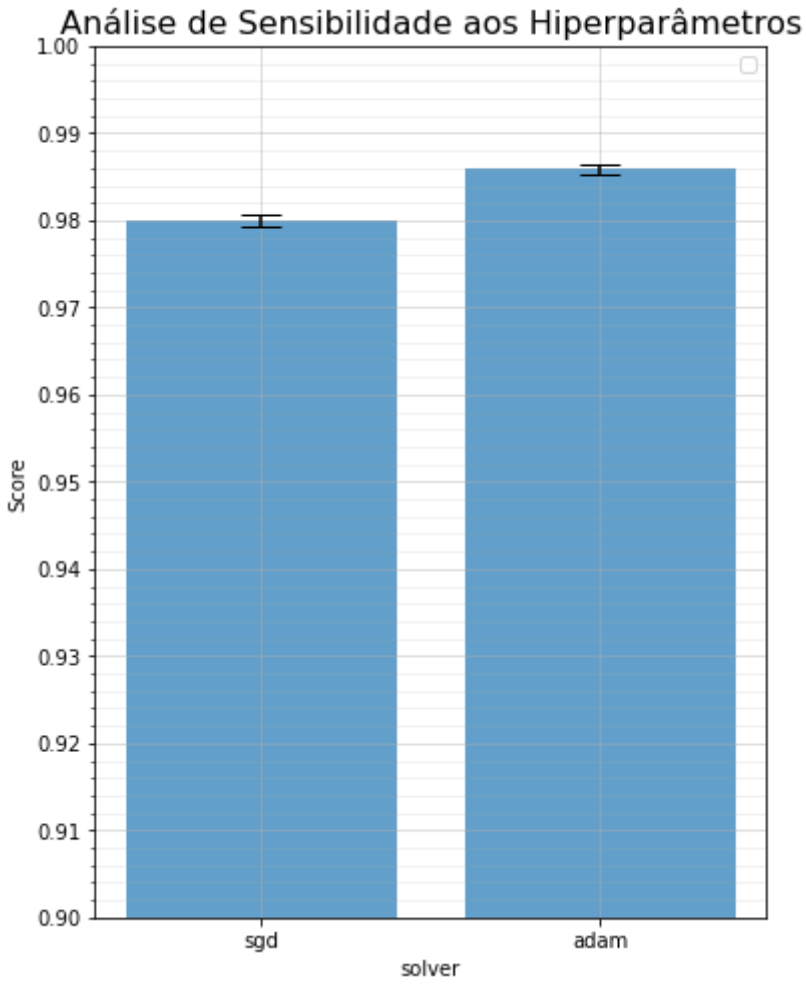
	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_hidden_layer_sizes	param_m
3	28.756789	8.842553	0.021204	6.008393e-04	NaN	
4	79.947864	24.693997	0.022704	2.193557e-03	NaN	
5	135.376774	13.820351	0.021404	1.562569e-03	NaN	
6	388.281841	26.878442	0.020004	1.483888e-03	NaN	
7	72.058172	15.033945	0.021804	3.999238e-04	NaN	
8	64.082514	11.774816	0.023305	6.408359e-04	NaN	
9	73.512640	24.181902	0.018004	2.098421e-03	NaN	
10	22.844269	7.991456	0.020805	1.939485e-03	NaN	
11	26.413152	6.095513	0.020705	1.552990e-03	NaN	
12	36.306733	7.777258	0.021605	1.685770e-03	NaN	
13	33.227144	9.090286	0.021405	1.428603e-03	NaN	
14	31.843356	7.849223	0.021605	1.356653e-03	NaN	
15	28.891190	11.254727	0.022105	1.868690e-03	NaN	
16	24.313395	5.561011	0.021505	1.858509e-03	NaN	
17	16.138428	6.639658	0.020005	1.549697e-03	NaN	
18	98.898021	30.386532	0.023505	1.360233e-03	NaN	
19	85.579914	19.539381	0.021705	7.812494e-04	NaN	
20	53.734380	8.077629	0.019604	6.636077e-04	NaN	
21	50.516964	15.811061	0.018104	1.135515e-03	NaN	
22	31.828160	9.439938	0.018204	1.076650e-03	NaN	
23	96.759644	27.729123	0.023205	6.003465e-04	NaN	
24	113.141715	43.556033	0.023405	8.006819e-04	NaN	
25	15.487906	2.058597	0.018004	1.183413e-03	NaN	
26	12.020520	1.872456	0.018004	1.095866e-03	NaN	

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_hidden_layer_sizes	param_m
27	14.933581	2.393992	0.020003	7.746954e-04	NaN	
28	5.318003	0.192154	0.019504	1.688051e-03	NaN	
29	5.033169	0.112931	0.019404	1.428222e-03	NaN	
30	4.679101	0.017231	0.018805	7.489213e-04	NaN	
31	52.616517	6.795672	0.021705	1.552575e-03	NaN	
32	66.692769	15.732382	0.021905	1.044560e-03	NaN	
33	67.728537	7.412874	0.021504	8.065227e-04	NaN	
34	64.552614	18.321635	0.022305	1.676905e-03	NaN	
35	3.242290	0.859432	0.001200	4.003050e-04	1	
36	5.991635	2.245977	0.001300	4.583806e-04	2	
37	9.830764	1.891321	0.001700	4.585626e-04	4	
38	12.373168	2.757667	0.002300	4.583909e-04	8	
39	16.138333	5.083965	0.002901	3.002726e-04	16	
40	16.998027	2.923482	0.004201	3.999180e-04	32	
41	28.811189	7.070130	0.007002	7.748483e-04	64	
42	38.406209	8.314615	0.012003	6.326361e-04	128	
43	72.958631	21.809590	0.022211	2.116427e-03	256	
44	133.004141	27.381712	0.043809	3.060111e-03	512	
45	6.089676	1.902512	0.001499	5.010874e-04	(1, 1)	
46	11.649100	3.970217	0.001799	3.997526e-04	(2, 2)	
47	16.804680	4.996749	0.001999	2.391803e-06	(4, 4)	
48	20.339432	7.572448	0.002999	4.453238e-04	(8, 8)	
49	22.500391	6.801515	0.004599	4.891330e-04	(16, 16)	
50	23.874117	5.155415	0.007902	5.358815e-04	(32, 32)	
51	41.830853	12.964848	0.013208	3.108316e-03	(64, 64)	
52	84.859930	17.726487	0.028906	1.043691e-03	(128, 128)	

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_hidden_layer_sizes	param_m
53	194.420734	65.654894	0.062713	3.495451e-03	(256, 256)	
54	972.782559	358.883258	0.139032	1.308754e-02	(512, 512)	
55	7.051556	1.604595	0.001900	5.396175e-04	(1, 1, 1)	
56	12.704450	4.920776	0.002200	6.002827e-04	(2, 2, 2)	
57	21.151092	13.987520	0.002900	5.388462e-04	(4, 4, 4)	
58	22.036856	3.469797	0.003901	1.374735e-03	(8, 8, 8)	
59	29.835446	6.949364	0.007201	4.005444e-04	(16, 16, 16)	
60	32.755851	12.132865	0.011101	8.310556e-04	(32, 32, 32)	
61	67.973725	22.708535	0.022305	1.005164e-03	(64, 64, 64)	
62	132.973205	35.568734	0.046810	3.919891e-03	(128, 128, 128)	
63	7.298700	1.571234	0.001500	6.709319e-04	(1, 1, 1, 1)	
64	12.363802	4.024096	0.002000	3.380167e-07	(2, 2, 2, 2)	
65	21.409846	3.915015	0.002601	4.891665e-04	(4, 4, 4, 4)	
66	26.061454	7.875482	0.004699	4.566364e-04	(8, 8, 8, 8)	
67	31.576338	6.413834	0.007105	1.432375e-03	(16, 16, 16, 16)	
68	40.538161	13.127391	0.013109	1.792779e-03	(32, 32, 32, 32)	
69	95.836064	21.947337	0.031307	2.759279e-03	(64, 64, 64, 64)	
70	179.305355	37.680813	0.060313	4.776838e-03	(128, 128, 128, 128)	

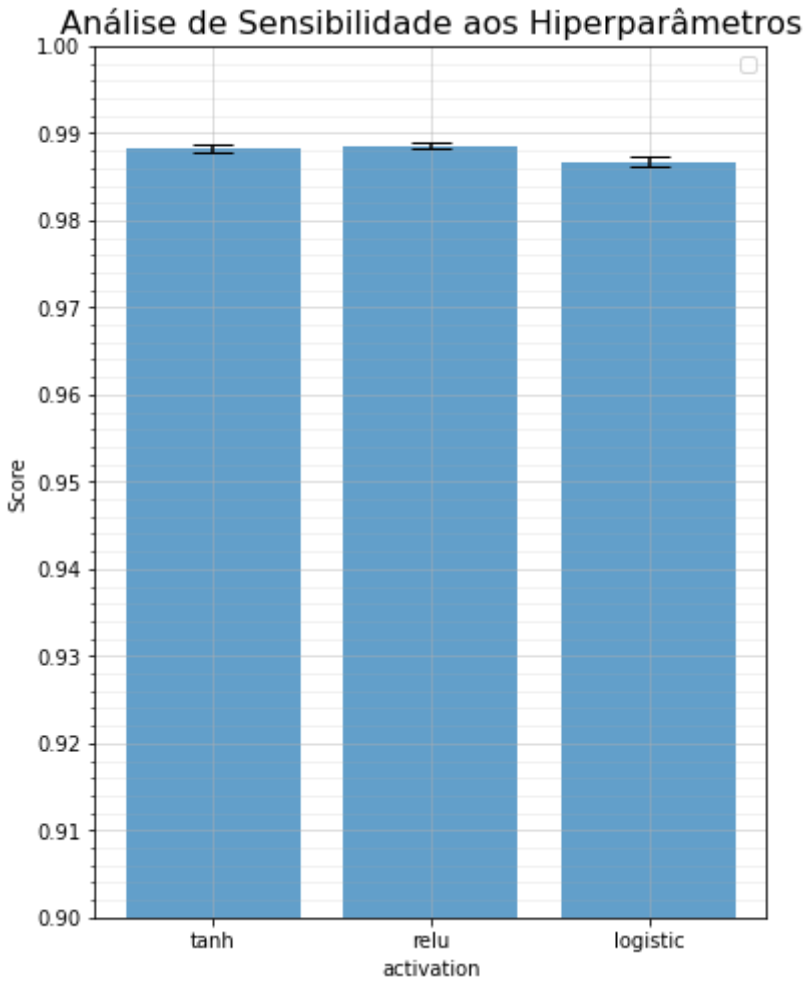
In [9]:
```
#solver
solver

parameter = 'solver'
plot_x = list(solver['param_solver'])
plot_y = solver['mean_test_score']
plot_y_err = solver['std_test_score']

plot_category(plot_x, plot_y, plot_y_err, parameter, yinf=0.90)
```

```
No handles with labels found to put in legend.
```

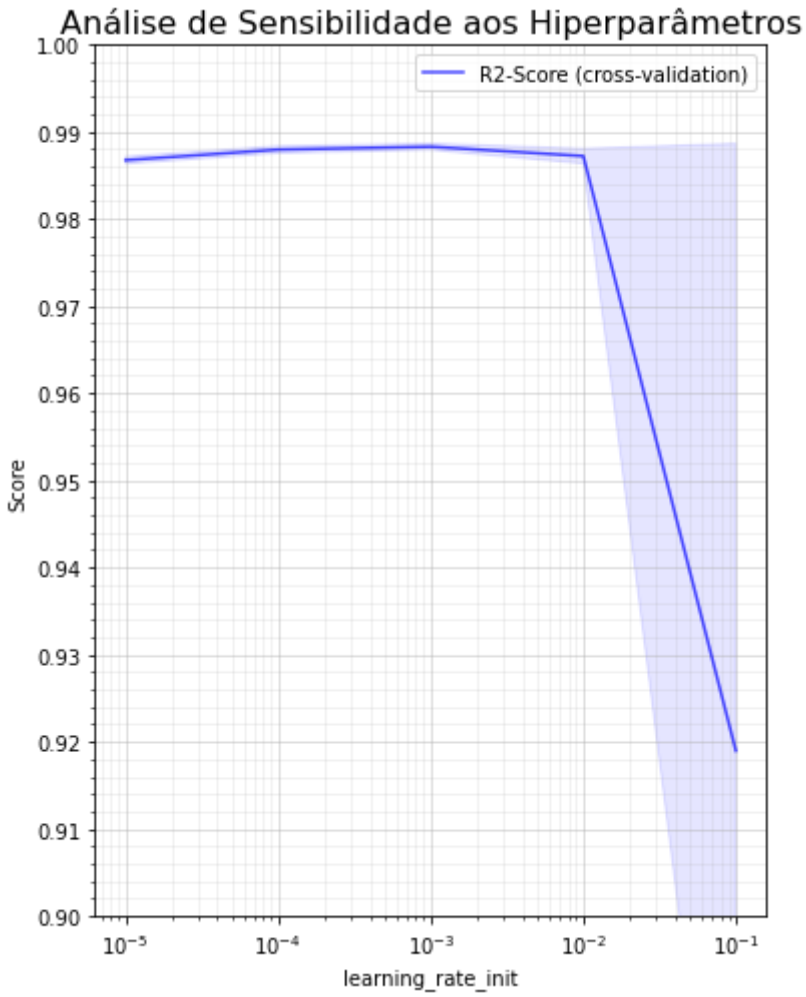


In [10]:
```
#activation
activation

parameter = 'activation'
plot_x = list(activation['param_activation'])
plot_y = activation['mean_test_score']
plot_y_err = activation['std_test_score']

plot_category(plot_x, plot_y, plot_y_err, parameter, yinf=0.90)
```

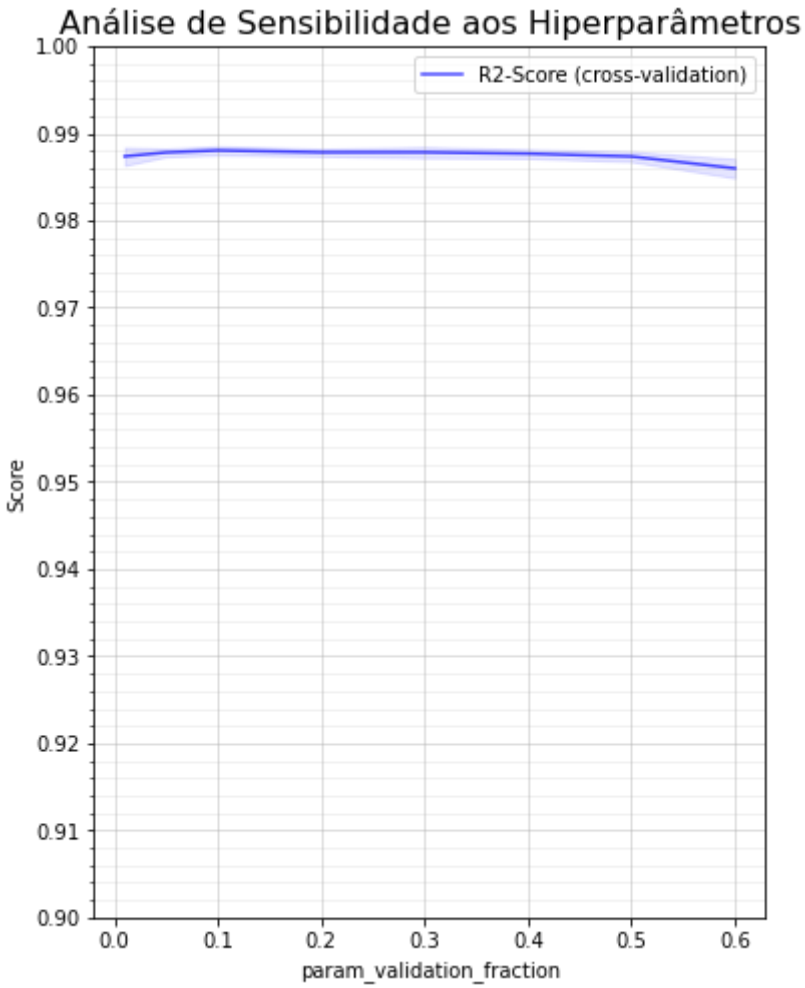
```
No handles with labels found to put in legend.
```



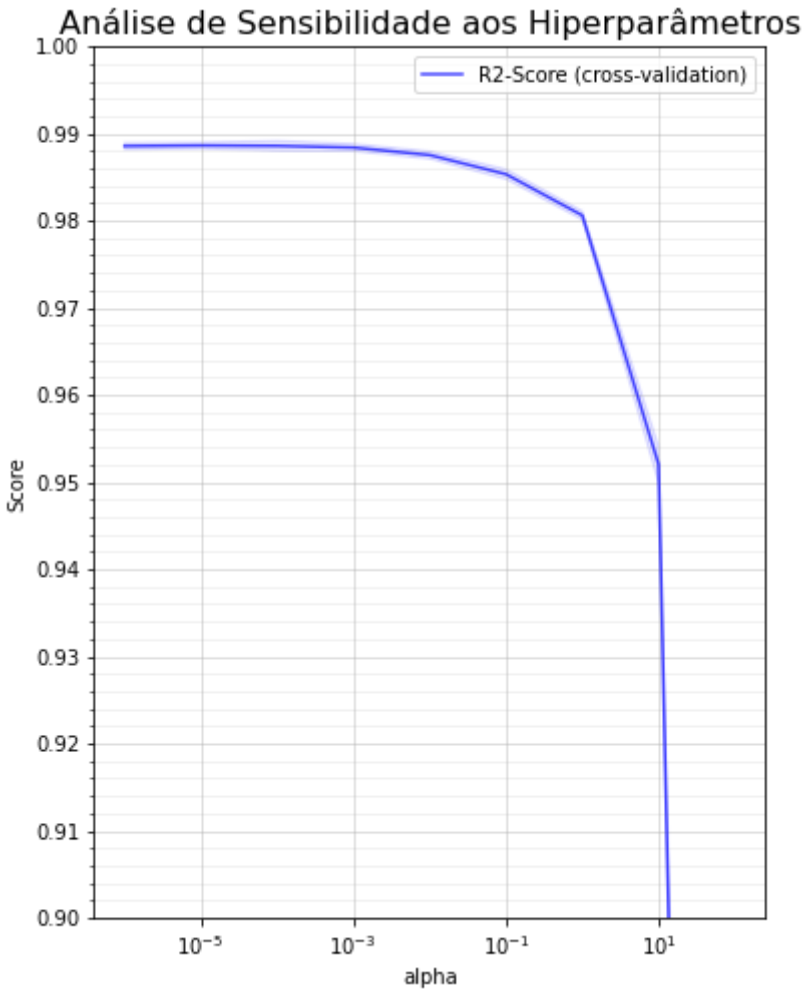
In [11]:
```
#learning_rate_init
learning_rate_init


parameter = 'learning_rate_init'
plot_x = learning_rate_init['param_learning_rate_init']
plot_y = learning_rate_init['mean_test_score']
plot_y_err = learning_rate_init['std_test_score']

plotar(plot_x, plot_y, plot_y_err, parameter, log=True, base=10, yinf=0.9)

#learning_rate_init
```

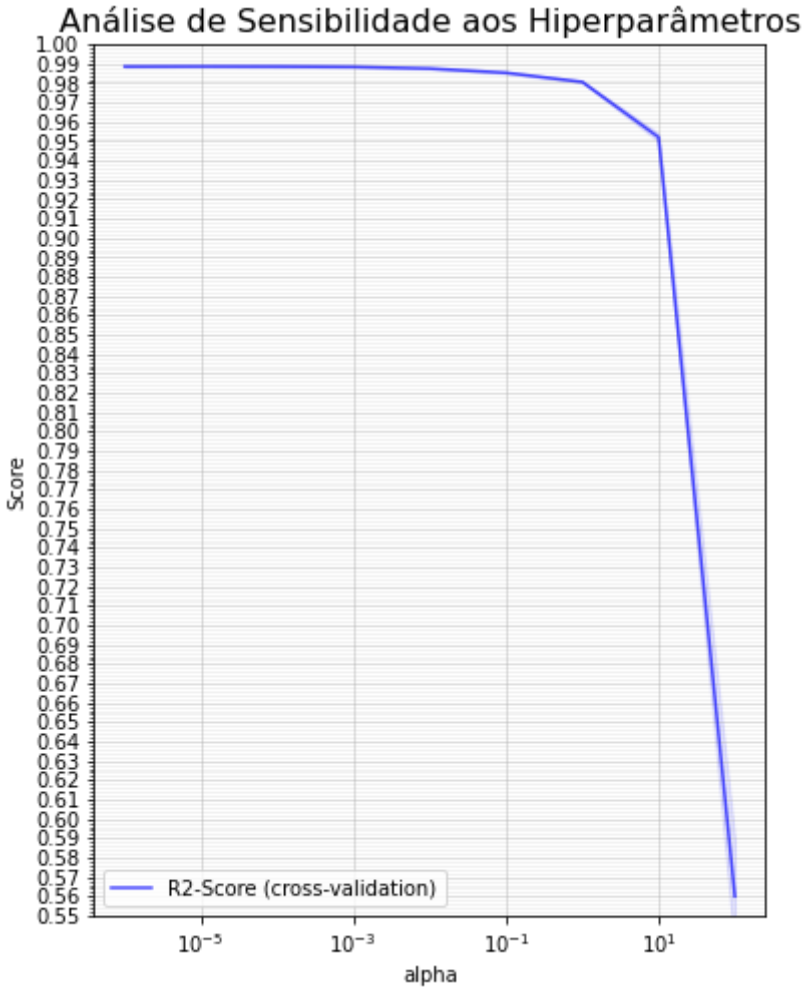


In [12]:
```
#early_stopping
early_stopping

parameter = 'param_validation_fraction'
plot_x = early_stopping['param_validation_fraction']
plot_x = pd.Series(plot_x)
plot_y = early_stopping['mean_test_score']
plot_y_err = early_stopping['std_test_score']

plot_x

plotar(plot_x, plot_y, plot_y_err, parameter, log=False, autoxtick=True)

#early_stopping
```



In [13]:
```
#alpha
alpha

parameter = 'alpha'
plot_x = alpha['param_alpha']
plot_x = pd.Series([1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100])
plot_y = alpha['mean_test_score']
plot_y_err = alpha['std_test_score']

plotar(plot_x, plot_y, plot_y_err, parameter, log=True, base=10, yinf=0.9)
plotar(plot_x, plot_y, plot_y_err, parameter, log=True, base=10, yinf=0.55)

#alpha
plot_x
```

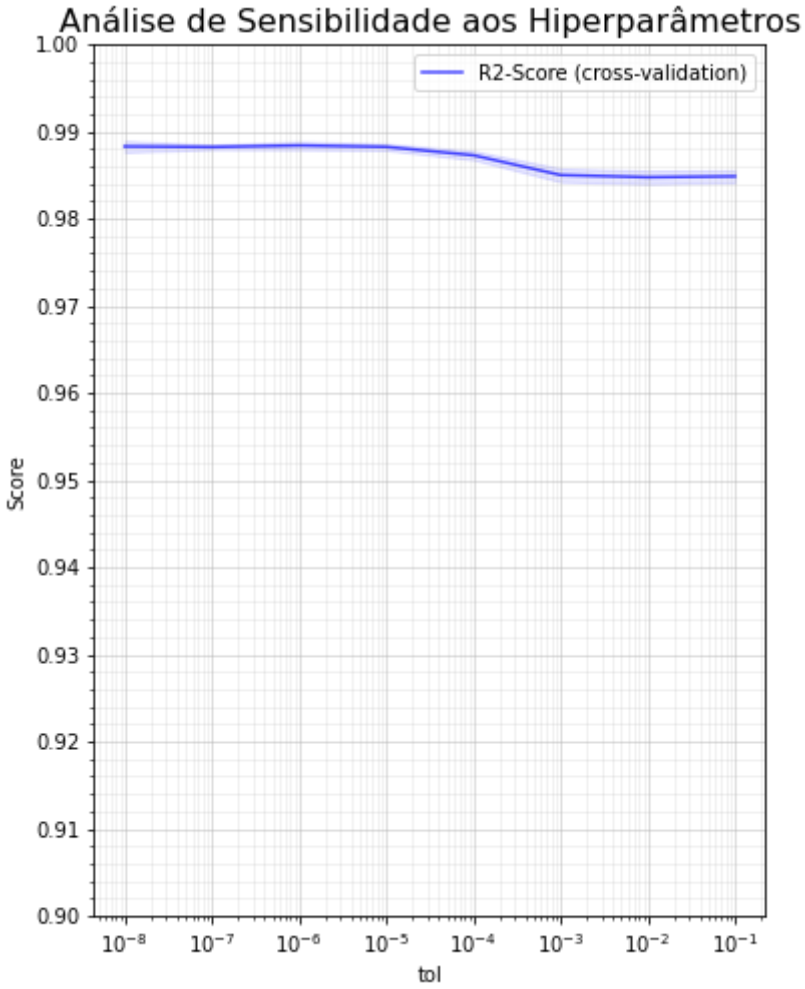




Out[13]: 0 0.000001

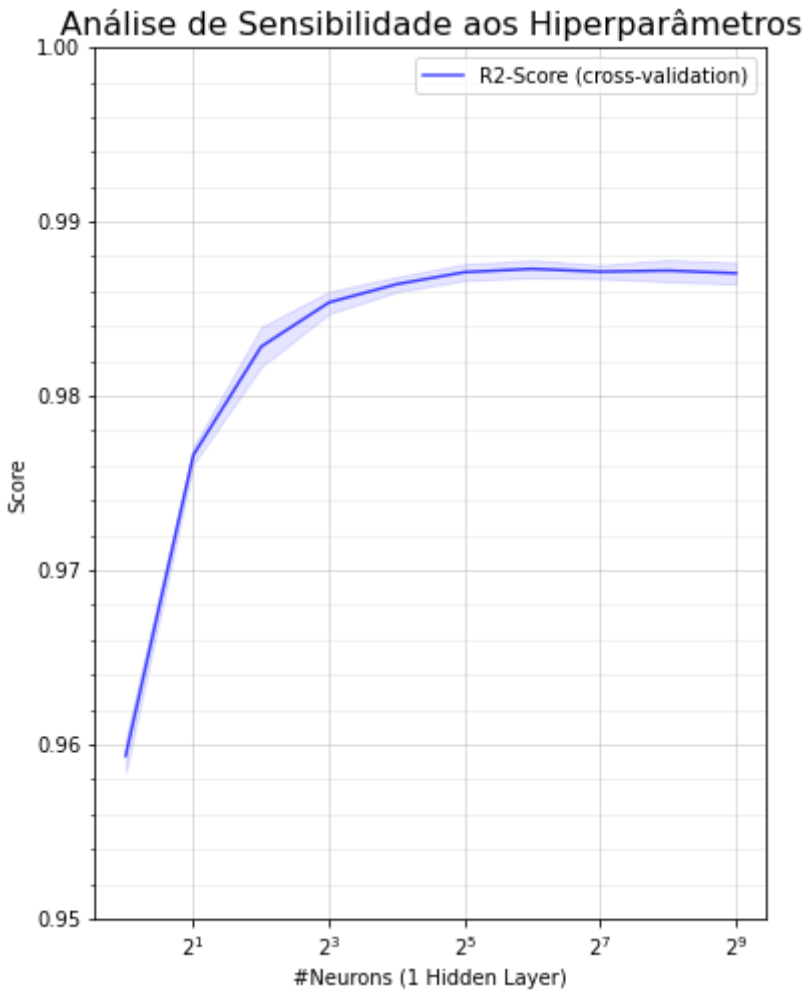
```
1      0.000010
2      0.000100
3      0.001000
4      0.010000
5      0.100000
6      1.000000
7     10.000000
8    100.000000
dtype: float64
```

In [14]:
```
#tol
tol

parameter = 'tol'
plot_x = tol['param_tol']
plot_y = tol['mean_test_score']
plot_y_err = tol['std_test_score']

plotar(plot_x, plot_y, plot_y_err, parameter, log=True, base=10)

#tol
```

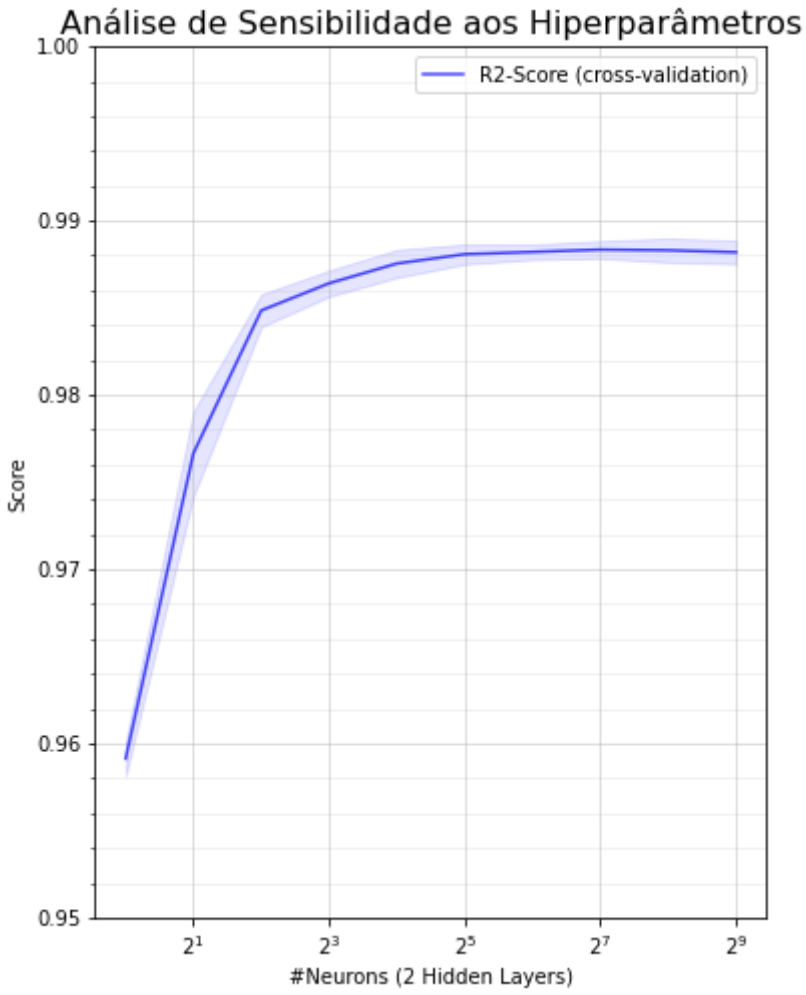


In [15]:
```
# 1 Hidden Layer
layer1

parameter = '#Neurons (1 Hidden Layer)'

plot_x = pd.Series([1, 2, 4, 8, 16, 32, 64, 128, 256, 512])
plot_y = layer1['mean_test_score']
plot_y_err = layer1['std_test_score']

#plot_x.describe()
```

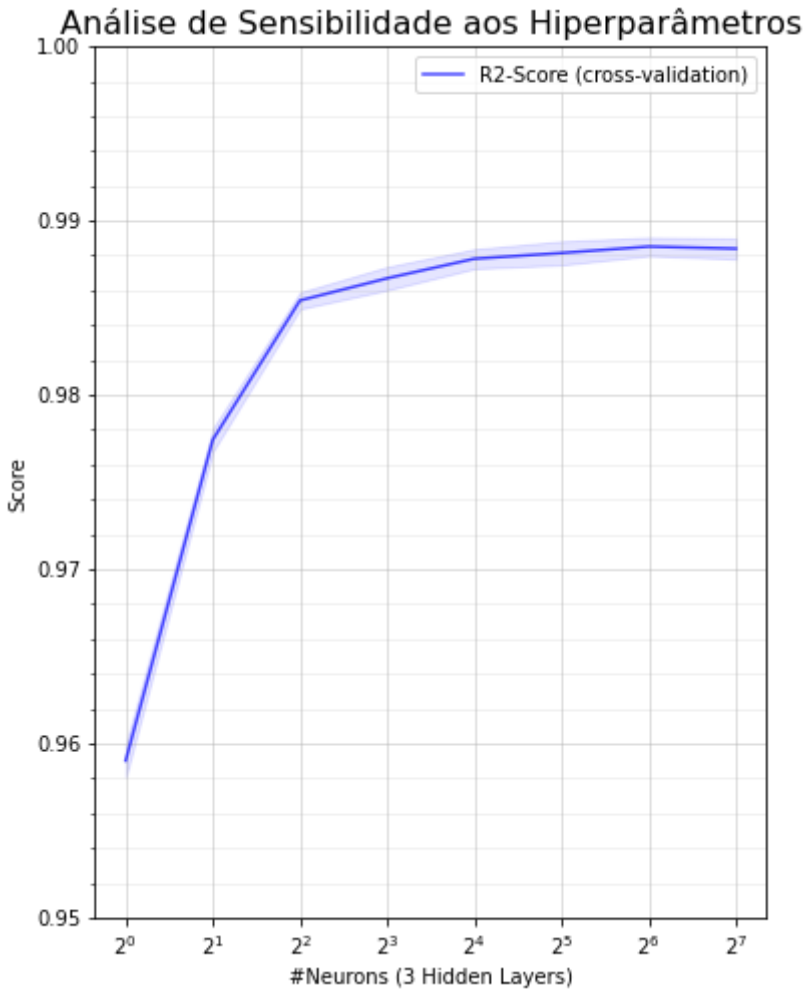
```
plotar(plot_x, plot_y, plot_y_err, parameter, log=True, base=2, yinf=0.95)
```



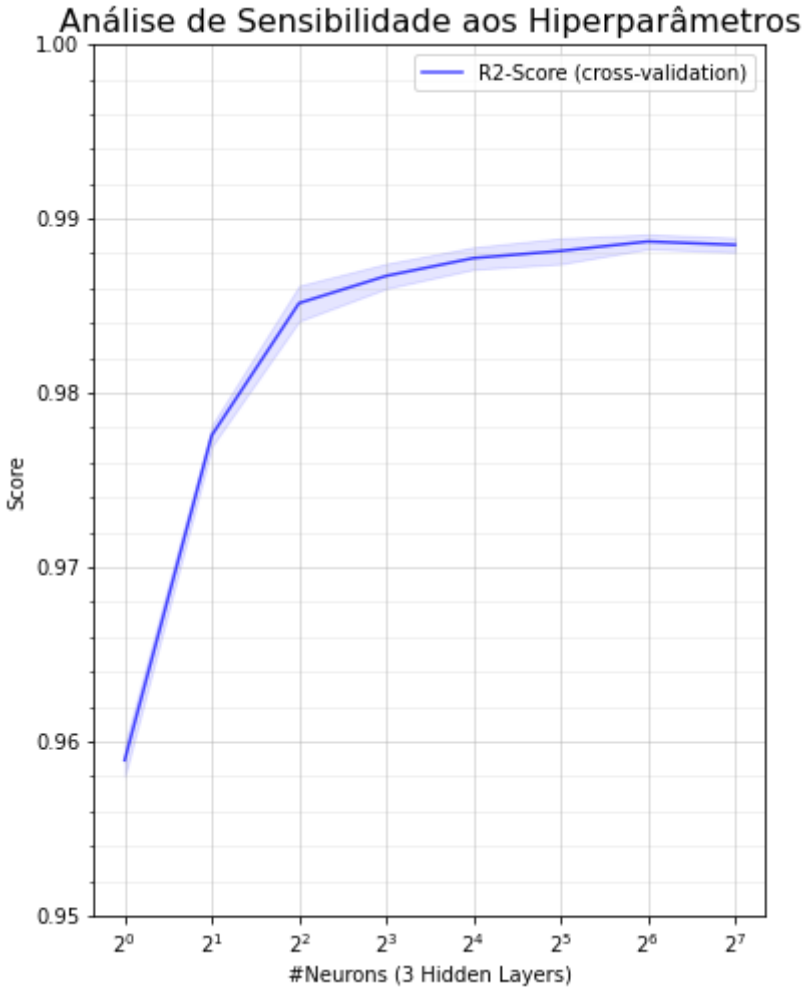
In [16]:
```
# 2 Hidden Layer
layer2

parameter = '#Neurons (2 Hidden Layers)'

plot_x = pd.Series([1, 2, 4, 8, 16, 32, 64, 128, 256, 512])
plot_y = layer2['mean_test_score']
plot_y_err = layer2['std_test_score']

#plot_x.describe()

plotar(plot_x, plot_y, plot_y_err, parameter, log=True, base=2, yinf=0.95)
```



In [17]:
```
# 3 Hidden Layer
layer3

parameter = '#Neurons (3 Hidden Layers)'

plot_x = pd.Series([1, 2, 4, 8, 16, 32, 64, 128])
plot_y = layer3['mean_test_score']
plot_y_err = layer3['std_test_score']

#plot_x.describe()

plotar(plot_x, plot_y, plot_y_err, parameter, log=True, base=2, yinf=0.95)
```



In [18]:
```
# 4 Hidden Layer
layer4

parameter = '#Neurons (3 Hidden Layers)'

plot_x = pd.Series([1, 2, 4, 8, 16, 32, 64, 128])
plot_y = layer4['mean_test_score']
plot_y_err = layer4['std_test_score']

#plot_x.describe()

plotar(plot_x, plot_y, plot_y_err, parameter, log=True, base=2, yinf=0.95)
```


Análise de Sensibilidade aos Hiperparâmetros
R2-Score (cross-validation)
Score
1.00
0.99
0.98
0.97
0.96
0.95
2^0
2^1
2^2
2^3
2^4
2^5
2^6
2^7
#Neurons (3 Hidden Layers)