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

fonte =  "https://github.com/alura-cursos/imersao-dados-2-2020/blob/master/MICRO

dados = pd.read_csv(fonte)
```

```
provas = ["NU_NOTA_CN", "NU_NOTA_CH", "NU_NOTA_MT", "NU_NOTA_LC", "NU_NOTA_REDAC
dados["NU_NOTA_TOTAL"] = dados[provas].sum(axis = 1)
provas = ["NU_NOTA_CN", "NU_NOTA_CH", "NU_NOTA_MT", "NU_NOTA_LC", "NU_NOTA_REDAC
dados_sem_notas_zero = dados.query("NU_NOTA_TOTAL != 0")
dados_sem_notas_NaN = dados_sem_notas_zero[provas].dropna()
```

```
provas_entrada = ["NU_NOTA_CH", "NU_NOTA_LC","NU_NOTA_CN" ,"NU_NOTA_REDACAO"]
provas_saida = "NU_NOTA_MT"
notas_entrada = dados_sem_notas_NaN[provas_entrada]
notas_saida = dados_sem_notas_NaN[provas_saida]
```

```
x = notas_entrada
y = notas_saida
```

```
from sklearn.model_selection import train_test_split
SEED = 4321
x_treino, x_teste, y_treino, y_teste = train_test_split(x,y, test_size = 0.25, r
```

```
x_teste.shape
```

```
(23135, 4)
```

```
from sklearn.svm import LinearSVR

modelo = LinearSVR(random_state = SEED)
modelo.fit(x_treino, y_treino)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:947: ConvergenceWa
  "the number of iterations.", ConvergenceWarning)
LinearSVR(C=1.0, dual=True, epsilon=0.0, fit_intercept=True,
          intercept_scaling=1.0, loss='epsilon_insensitive', max_iter=1000,
          random_state=4321, tol=0.0001, verbose=0)
```

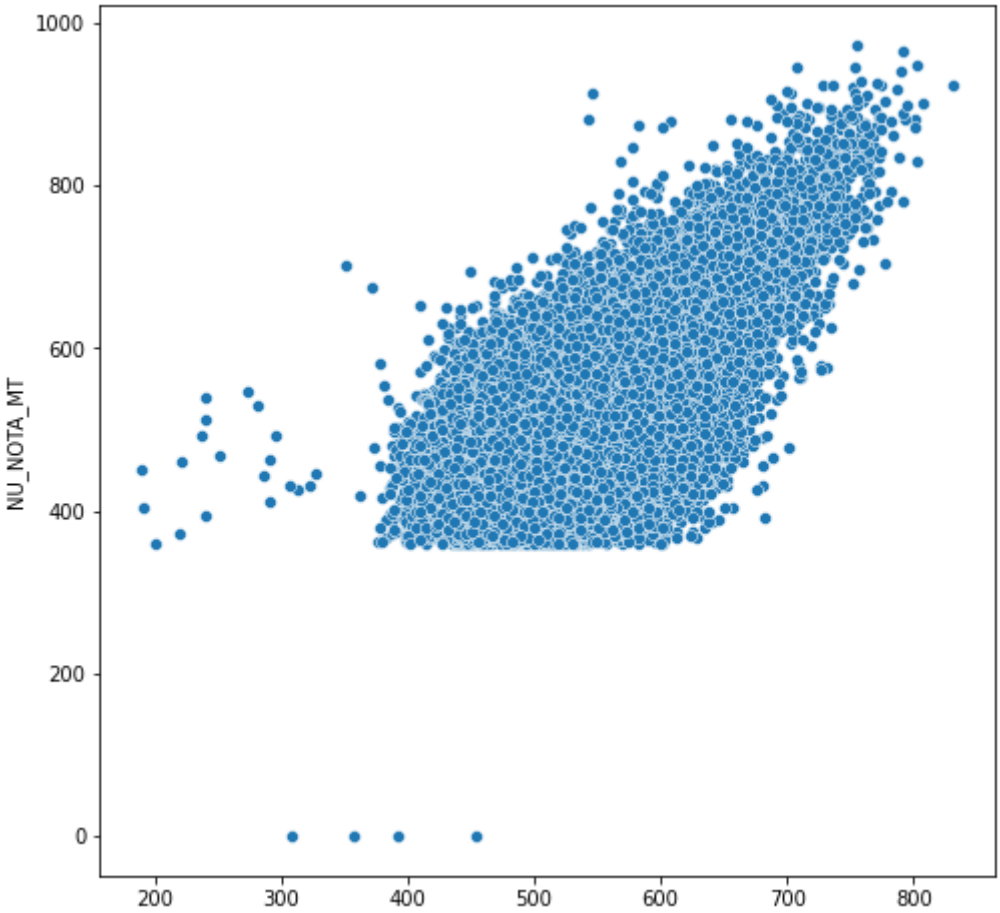
```
predicoes_MT = modelo.predict(x_teste)
```

```
y_teste[:5]
```

```
114991    459.7
104685    617.2
```

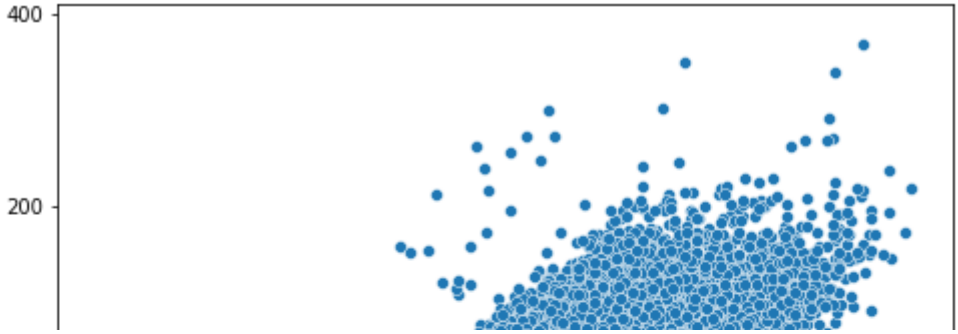
```
plt.figure(figsize=(8,8))
sns.scatterplot(x=predicoes_MT, y = y_teste)


```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f607bc41e10>
```



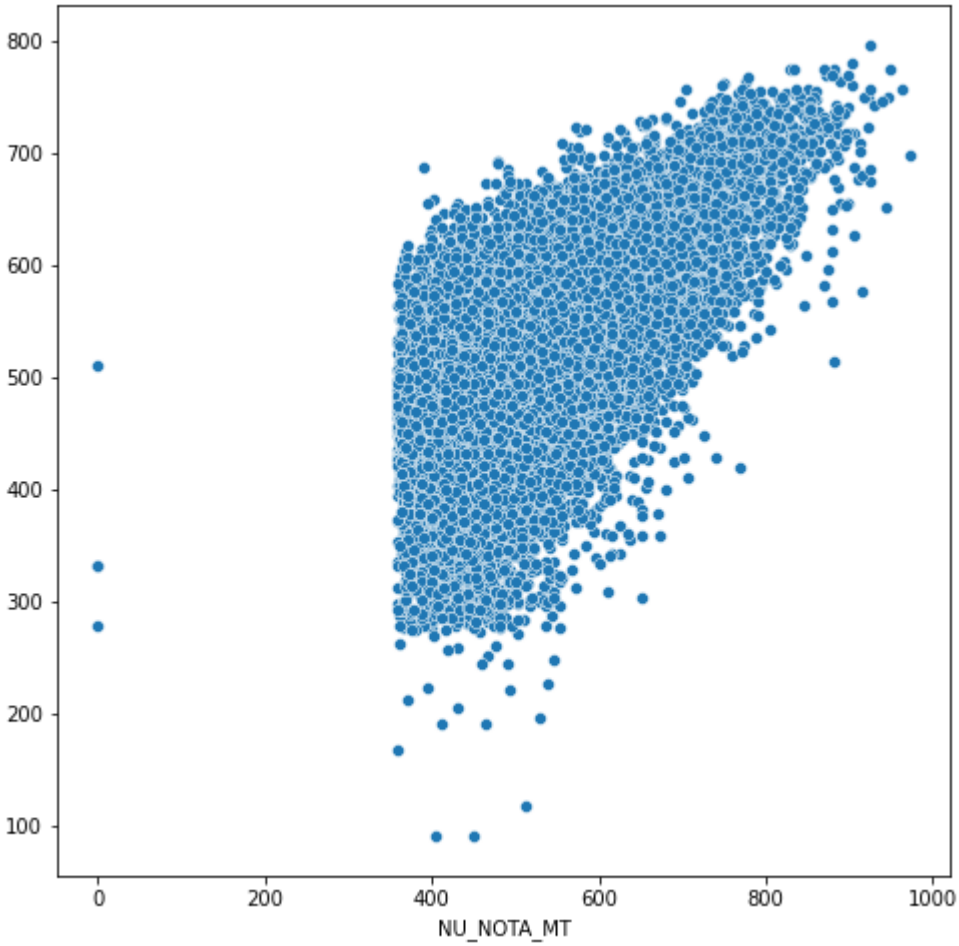

```
plt.figure(figsize=(8,8))
sns.scatterplot(x=y_teste, y = y_teste-predicoes_MT)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f60793a1cf8>
```




```
1 plt.figure(figsize=(8,8))
2 sns.scatterplot(x=y_teste, y = x_teste.mean(axis=1))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f60776d5ba8>
```




```
1 resultados = pd.DataFrame()
2 resultados["Real"] = y_teste
3 resultados["Previsao"]= predicoes_MT
4 resultados["diferenca"] = resultados["Real"] - resultados["Previsao"]
5 resultados["quadrado_diferenca"] = (resultados["Real"] - resultados["Previsao"])
6
```

```
1 resultados
```

	Real	Previsao	diferenca	quadrado_diferenca
114991	459.7	564.486105	-104.786105	10980.127878
104685	617.2	628.305027	-11.105027	123.321625
91028	520.0	509.980986	10.019014	100.380650
115802	703.8	593.681144	110.118856	12126.162420
93303	627.1	579.784588	47.315412	2238.748252
...	...	...	...	...
81393	688.6	626.291851	62.308149	3882.305380
40159	479.1	652.313922	-173.213922	30003.062777
64083	501.4	607.787172	-106.387172	11318.230438
84661	772.5	734.227663	38.272337	1464.771772

```
1 resultados["quadrado_diferenca"].mean()**(1/2)
```

```
81.26755383435341
```

```
1 from sklearn.dummy import DummyRegressor
2
3 modelo_dummy = DummyRegressor()
4 modelo_dummy.fit(x_treino, y_treino)
5 dummy_predicoes = modelo_dummy.predict(x_teste)
6
7 from sklearn.metrics import mean_squared_error
8
9 mean_squared_error(y_teste, dummy_predicoes) #12063.645588509502
```

```
12063.645588509502
```

```
1 mean_squared_error(y_teste, predicoes_MT)#6604.41530621957
```

```
6604.41530621957
```

Desafio1 = procurar outro modelo de ML para treinar e comparar com os modelos criados em aula

```
1 #desafio 1 com decision tree
2 from sklearn import tree
3 modelo_tree = tree.DecisionTreeRegressor()
4 modelo_tree.fit(x_treino, y_treino)
5 tree_predicoes = modelo_tree.predict(x_teste)
6 mean_squared_error(y_teste, tree_predicoes) #11365.011590231252
7
```

```
11382.14146401556
```

```
resultadosdt = pd.DataFrame()
resultadosdt["Real"] = y_teste
resultadosdt["Previsao"]= tree_predicoes
resultadosdt["diferenca"] = resultadosdt["Real"] - resultadosdt["Previsao"]
resultadosdt["quadrado_diferenca"] = (resultadosdt["Real"] - resultadosdt["Previ
resultadosdt
```

	Real	Previsao	diferenca	quadrado_diferenca
114991	459.7	671.0	-211.3	44647.69
104685	617.2	664.6	-47.4	2246.76
91028	520.0	480.8	39.2	1536.64
115802	703.8	553.2	150.6	22680.36
93303	627.1	509.5	117.6	13829.76
...	...	...	...	...
81393	688.6	529.8	158.8	25217.44
40159	479.1	633.4	-154.3	23808.49
64083	501.4	660.0	-158.6	25153.96
84661	772.5	816.3	-43.8	1918.44
79106	369.0	558.7	-189.7	35986.09

23135 rows × 4 columns

```
#desafio 1 com near neighbors
from sklearn.neighbors import KNeighborsRegressor
modelo_neigh = KNeighborsRegressor()
modelo_neigh.fit(x_treino, y_treino)
neigh_predicoes = modelo_neigh.predict(x_teste)
mean_squared_error(y_teste, neigh_predicoes) #6432.6058905554355
```

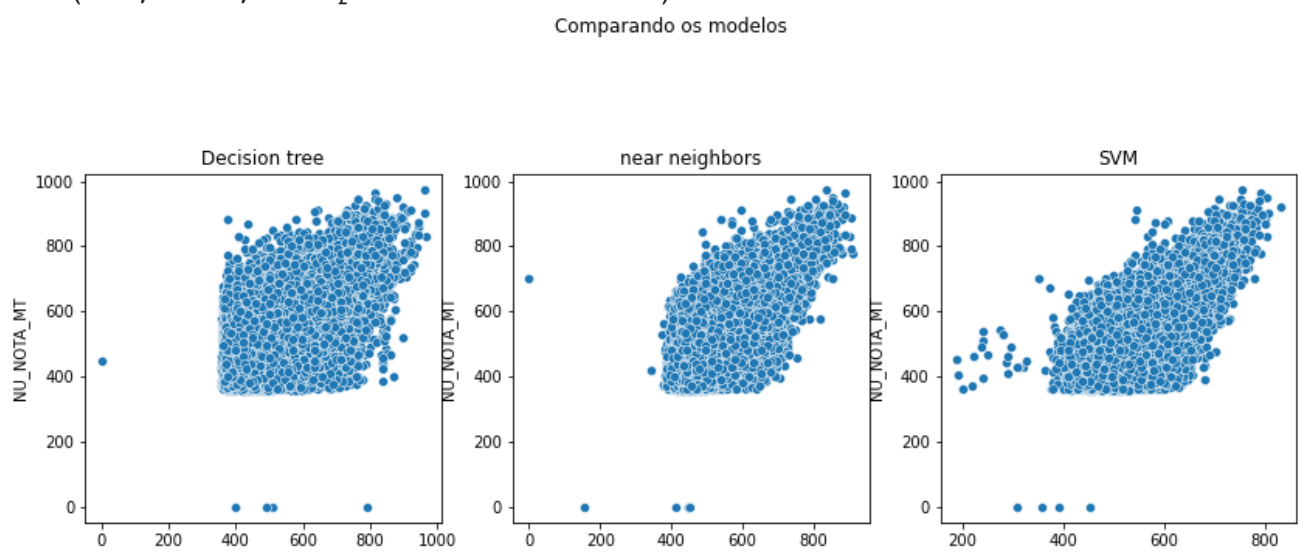
```
6432.6058905554355
```

```
resultadosnn = pd.DataFrame()
resultadosnn["Real"] = y_teste
resultadosnn["Previsao"]= neigh_predicoes
resultadosnn["diferenca"] = resultadosdt["Real"] - resultadosdt["Previsao"]
resultadosnn["quadrado_diferenca"] = (resultadosdt["Real"] - resultadosdt["Previ
resultadosnn
```

	Real	Previsao	diferenca	quadrado_diferenca
114991	459.7	465.70	-211.3	44647.69
104685	617.2	621.74	-47.4	2246.76
91028	520.0	495.72	39.2	1536.64
115802	703.8	506.04	150.6	22680.36
93303	627.1	521.90	117.6	13829.76
...	...	...	...	...
81393	688.6	554.86	158.8	25217.44

```
plt.figure(figsize=(14,14))
plt.subplot(3,3,1)
#arvore de decisoes
sns.scatterplot(x=tree_predicoes, y = y_teste)
plt.title("Decision tree")
#SVM
plt.subplot(3,3,3)
sns.scatterplot(x= predicoes_MT, y = y_teste)
plt.title("SVM")
#near neighbors
plt.subplot(3,3,2)
sns.scatterplot(x= neigh_predicoes, y = y_teste)
plt.title("near neighbors")

plt.suptitle("Comparando os modelos")
```

```
Text(0.5, 0.98, 'Comparando os modelos')
```



Desafio2: Ler a documentacao do Dummy e alterar o metodo de regressao

```
#desafio 2, dummy strategy = mean
```

```
modelo_dummy_mean = DummyRegressor(strategy="mean")
modelo_dummy_mean.fit(x_treino, y_treino)
dummy_predicoes_mean = modelo_dummy_mean.predict(x_teste)
mean_squared_error(y_teste, dummy_predicoes_mean) #12063.645588509502
```

```
12063.645588509502
```

```
#desafio 2, dummy strategy =median
modelo_dummy_median = DummyRegressor(strategy="median")
modelo_dummy_median.fit(x_treino, y_treino)
dummy_predicoes_median = modelo_dummy_median.predict(x_teste)
mean_squared_error(y_teste, dummy_predicoes_median) #12599.969971795981
```

```
12599.969971795981
```

```
#desafio 2, dummy strategy =quantile - 1(maximo)
modelo_dummy_quantile = DummyRegressor(strategy="quantile", quantile = 1.0)
modelo_dummy_quantile.fit(x_treino, y_treino)
dummy_predicoes_quantile = modelo_dummy_quantile.predict(x_teste)
mean_squared_error(y_teste, dummy_predicoes_quantile ) #223890.15624378645
```

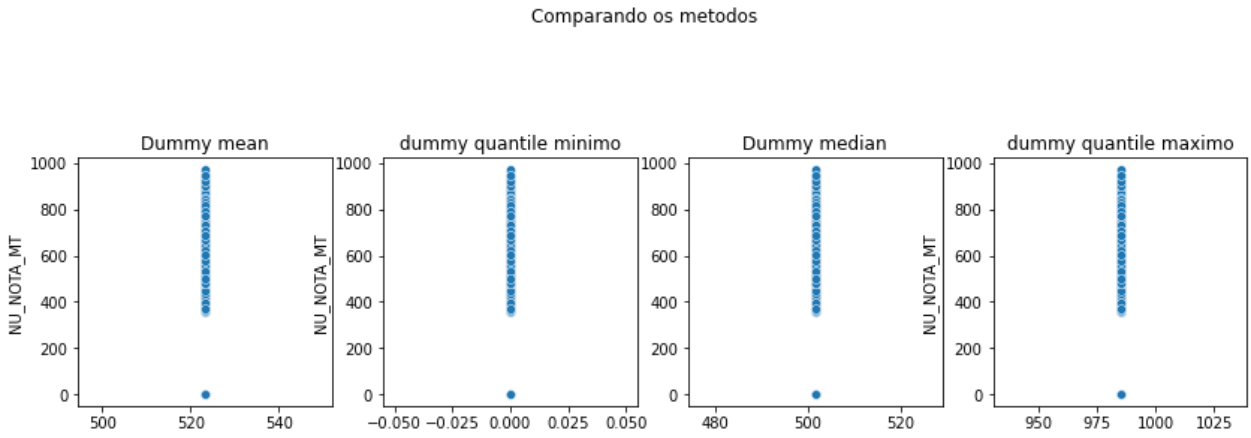
```
223890.15624378645
```

```
#desafio 2, dummy strategy =quantile - 0(minimo)
modelo_dummy_quantile_m = DummyRegressor(strategy="quantile", quantile = 0.0)
modelo_dummy_quantile_m.fit(x_treino, y_treino)
dummy_predicoes_quantile_m = modelo_dummy_quantile_m.predict(x_teste)
mean_squared_error(y_teste, dummy_predicoes_quantile_m ) #287425.98995893664
```

```
287425.98995893664
```

```
plt.figure(figsize=(14,14))
plt.subplot(4,4,1)
#mean
sns.scatterplot(x=dummy_predicoes_mean, y = y_teste)
plt.title("Dummy mean")
#median
plt.subplot(4,4,3)
sns.scatterplot(x= dummy_predicoes_median, y = y_teste)
plt.title("Dummy median")
#minimo
plt.subplot(4,4,2)
sns.scatterplot(x= dummy_predicoes_quantile_m, y = y_teste)
plt.title("dummy quantile minimo")

#maximo
plt.subplot(4,4,4)
sns.scatterplot(x= dummy_predicoes_quantile, y = y_teste)
plt.title("dummy quantile maximo")


plt.suptitle("Comparando os metodos")
```

```
Text(0.5, 0.98, 'Comparando os metodos')
```



Desafio3: Busacar outra métrcia(mean squared root) para avaliar modelos de regressão

```
import sklearn.metrics  as skm
#Metricas para o SVM
m1 = skm.mean_absolute_error(y_teste, predicoes_MT )
m2 = mean_squared_error(y_teste, predicoes_MT)
m3 = skm.max_error(y_teste, predicoes_MT)
m4 = skm.median_absolute_error(y_teste, predicoes_MT )

#metricas Near neighbors
mn1 = skm.mean_absolute_error(y_teste, neigh_predicoes )
mn2 = mean_squared_error(y_teste, neigh_predicoes )
mn3 = skm.max_error(y_teste, neigh_predicoes )
mn4 = skm.median_absolute_error(y_teste, neigh_predicoes )

#metricas Decision tree
dt1 = skm.mean_absolute_error(y_teste, tree_predicoes )
dt2 = mean_squared_error(y_teste, tree_predicoes )
dt3 = skm.max_error(y_teste, tree_predicoes )
dt4 = skm.median_absolute_error(y_teste, tree_predicoes )

#metricas dummy
du1 = skm.mean_absolute_error(y_teste, dummy_predicoes )
du2 = mean_squared_error(y_teste,  dummy_predicoes )
du3 = skm.max_error(y_teste,  dummy_predicoes )
du4 = skm.median_absolute_error(y_teste,  dummy_predicoes )

metri = {"modelos":["SVM", "Near Neighbors", "Decision Tree", "Dummy"],"mean_absc
modelos_regressao = pd.DataFrame(data = metri)
modelos_regressao

```

	modelos	mean_absolute_error	mean_squared_error	max_error	median_absolut
0	SVM	66.045554	6604.415306	453.506823	5
1	Near Neighbors	63.782104	6432.605891	701.600000	5
2	Decision Tree	83.781634	11382.141464	792.500000	6