Bibliotecas

```
import pandas as pd

import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots
import seaborn as sns

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVR
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.dummy import DummyRegressor
```

Dados

```
source = 'https://raw.githubusercontent.com/alura-cursos/imersao-dados-2-2020/ma

data = pd.read_csv(source)

data.head()
```

	NU_INSCRICAO	NU_ANO	CO_MUNICIPIO_RESIDENCIA	NO_MUNICIPIO_RESIDENCIA	CO_
0	190001004661	2019	1506138	Redenção	
1	190001004674	2019	1504208	Marabá	
2	190001004722	2019	1501402	Belém	
3	190001004735	2019	1507300	São Félix do Xingu	
4	190001004776	2019	1500800	Ananindeua	

5 rows × 136 columns

Dia 1 - Desafios

Desafio 1 - Proporção de inscritos por idade

```
# Desafio 1
```

```
2
3 porcentagem_idade = data["NU_IDADE"].value_counts().sort_index() / len(data["NU_
4
5 porcentagem_idade = pd.DataFrame(porcentagem_idade).reset_index(inplace=False)
6
7 porcentagem_idade.columns = ['Idade', 'Porcentagem (%)']
8
9 porcentagem_idade.head()
```

	Idade	Porcentagem (%)
0	13	0.003140
1	14	0.110692
2	15	1.577171
3	16	6.146962
4	17	16.687078

Desafio 2 - Descobrir de quais estados são os inscritos com 13 anos

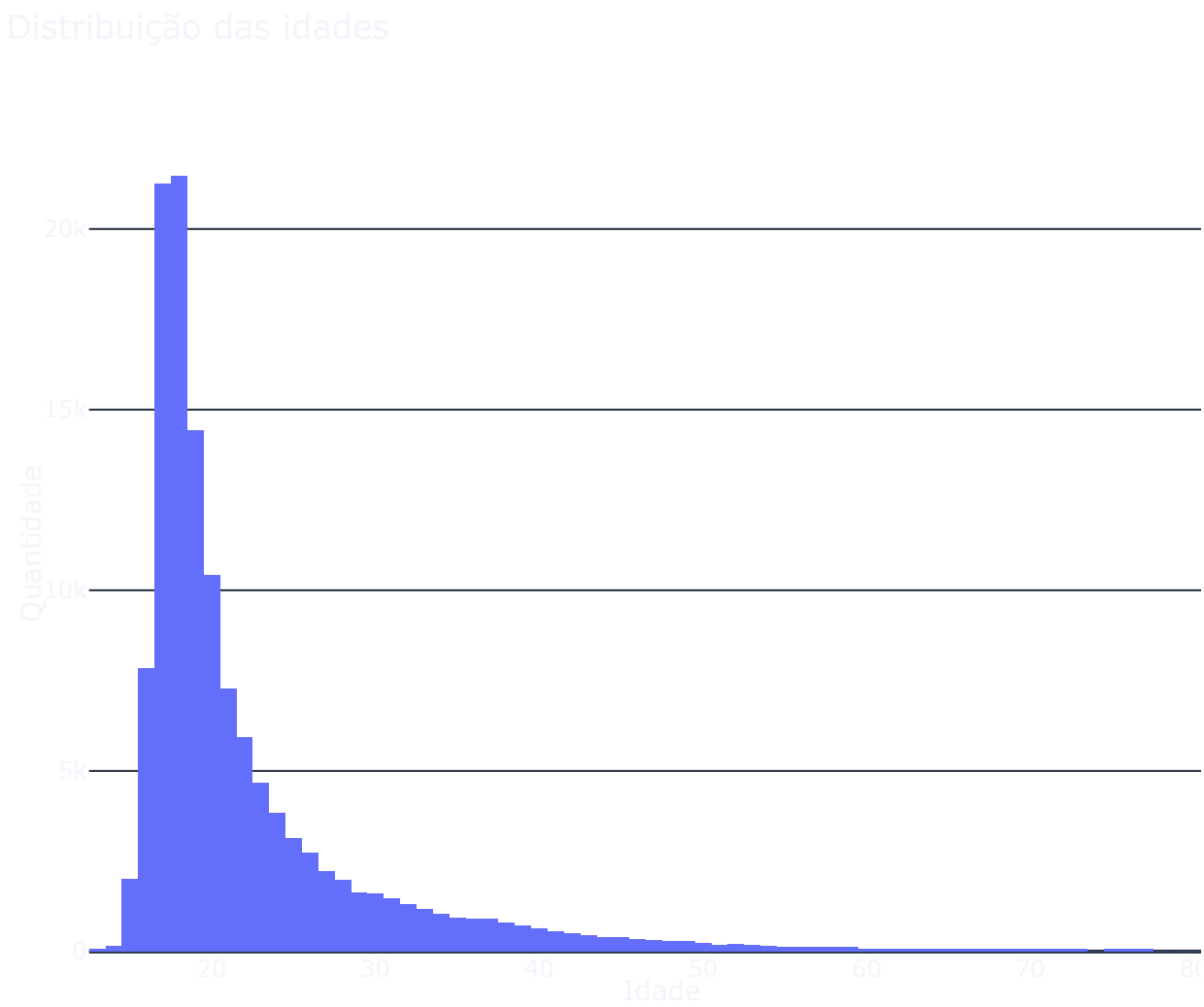
```
1 # Desafio 2
2
3 estados =  data[data["NU_IDADE"] == 13]['SG_UF_RESIDENCIA'].value_counts().index
4
5 print('Estados:', estados)
```

```
Estados: ['AP', 'MT', 'BA', 'SP']
```

Desafio 3 - Distribuição das idades

```
1 # Desafio 3
2
3 fig_idades = go.Figure()
4
5 fig_idades.add_trace(go.Histogram(x = data['NU_IDADE']))
6
7 fig_idades.update_layout(
8     title_text = 'Distribuição das idades',
9     xaxis_title_text = 'Idade',
10    yaxis_title_text = 'Quantidade',
11    template = 'plotly_dark',
12    bargap = .05,
13    width = 750,
14    height = 600
15 )
16
```

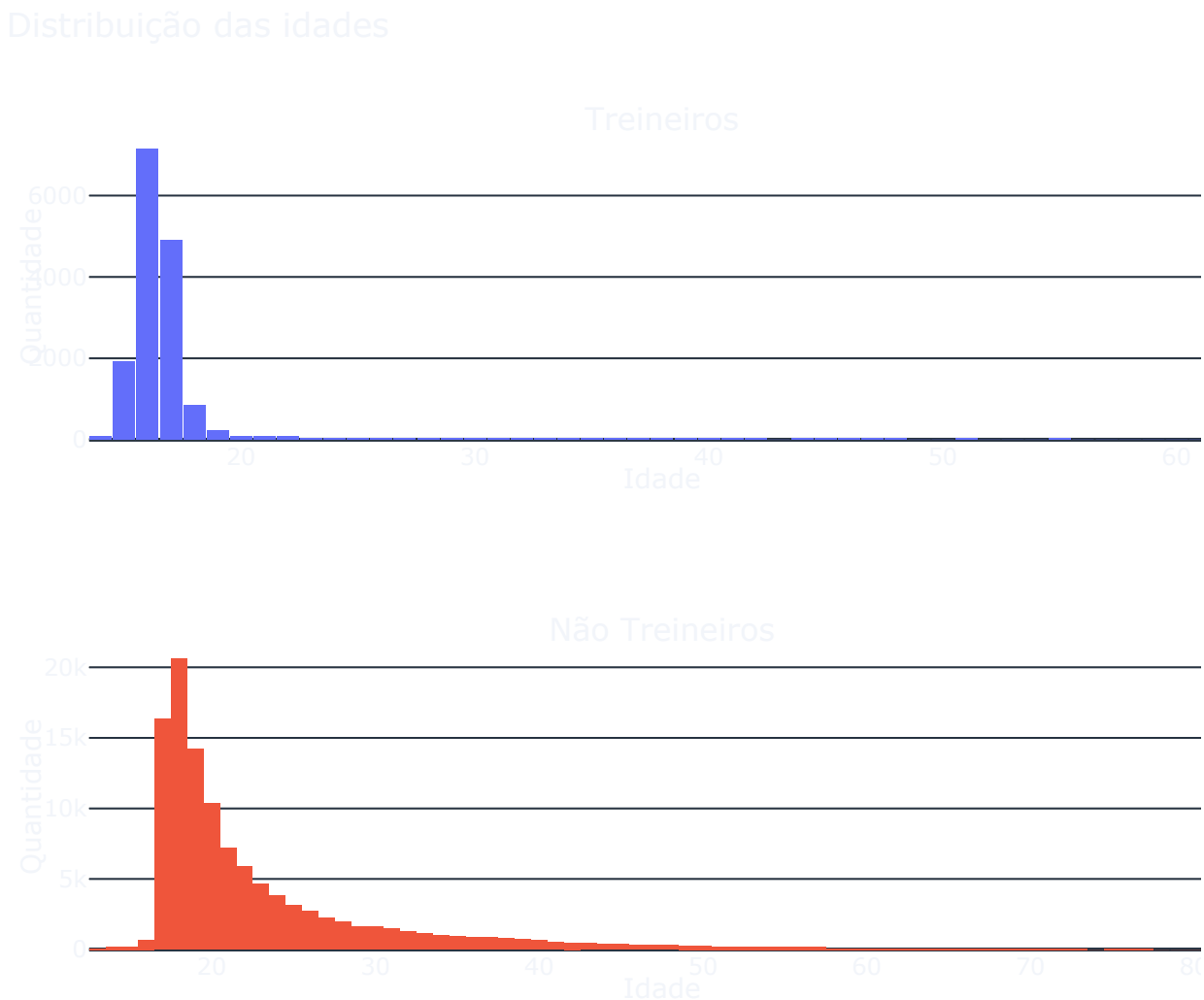
```
fig_idades.show()
```



Desafio 4 - Distribuição das idades dos treineiros e não treineiros

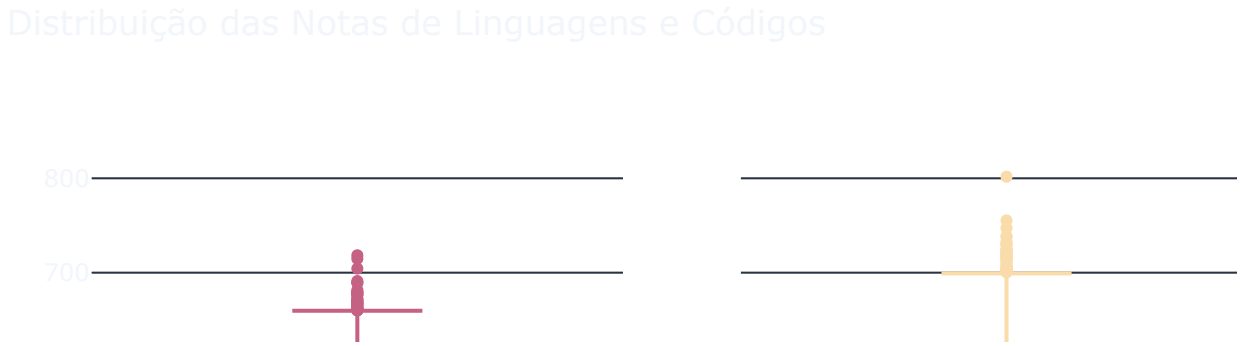
```
# Desafio 4

treineiros = data[data['IN_TREINEIRO'] == 1]
nao_treineiros = data[data['IN_TREINEIRO'] == 0]

fig_idades = make_subplots(rows = 2,
                           cols = 1,
                           subplot_titles = ("Treineiros",
                                             "Não Treineiros"))

fig_idades.add_trace(go.Histogram(x = treineiros['NU_IDADE'],
                                  name = 'Treineiros',
                                  showlegend = False),
                     row = 1, col = 1)

fig_idades.add_trace(go.Histogram(x = nao_treineiros['NU_IDADE'],
```

```
                                    name = 'Não Treineiros',
                                    showlegend = False),
                        row = 2, col = 1)

fig_idades.update_xaxes(title_text='Idade')

fig_idades.update_yaxes(title_text='Quantidade')

fig_idades.update_layout(
    title_text = 'Distribuição das idades',
    template = 'plotly_dark',
    bargap = .05,
    width = 750,
    height = 600
)

fig_idades.show()
```



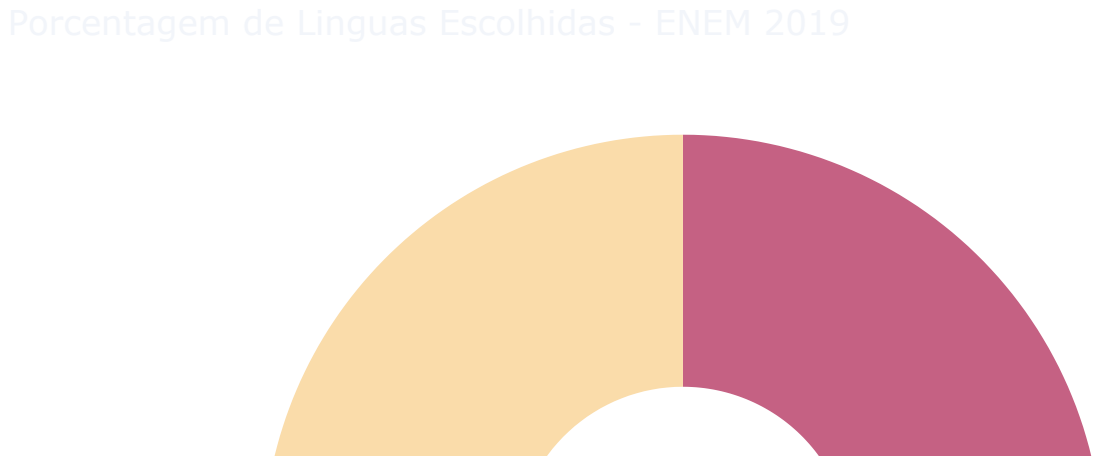
Desafio 5 - Distrubuição das notas de linguagens e códigos nas diferentes línguas (inglês e espanhol).

```
# Desafio 05

prova_espanhol = data[data['TP_LINGUA'] == 1]
prova_ingles = data[data['TP_LINGUA'] == 0]

fig_nota_lc = make_subplots(rows = 1,
                            cols = 2,
                            shared_yaxes = True)

fig_nota_lc.add_trace(go.Box(y = prova_espanhol['NU_NOTA_LC'],
                             name = 'Espanhol',
                             marker_color = '#c56183',
                             showlegend = False),
                      row = 1, col = 1)

fig_nota_lc.add_trace(go.Box(y = prova_ingles['NU_NOTA_LC'],
                             name = 'Ingles',
                             marker_color = '#fadcaa',
                             showlegend = False),
                      row = 1, col = 2)

fig_nota_lc.update_layout(
    title_text = 'Distribuição das Notas de Linguagens e Códigos',
    yaxis_title_text = 'Nota de Linguagens e Códigos',
    template = 'plotly_dark',
    bargap = .05,
    width = 750,
    height = 600
)

fig_nota_lc.show()
```



Desafio 6 - Gráfico Extra

```
colors= ['#c56183', '#fadcaa']

labels = ['Espanhol', 'Inglês']

values = [len(prova_espanhol),
          len(prova_ingles)]

fig_linguas = go.Figure()

fig_linguas.add_trace(go.Pie(labels=labels, values=values,
                             hole=.4, marker_colors=colors))

fig_linguas.update_layout(
    title_text='Porcentagem de Linguas Escolhidas - ENEM 2019',
    template = 'plotly_dark',
    width = 750,
    height = 600
)

fig_linguas.show()
```



Dia 2 - Desafios

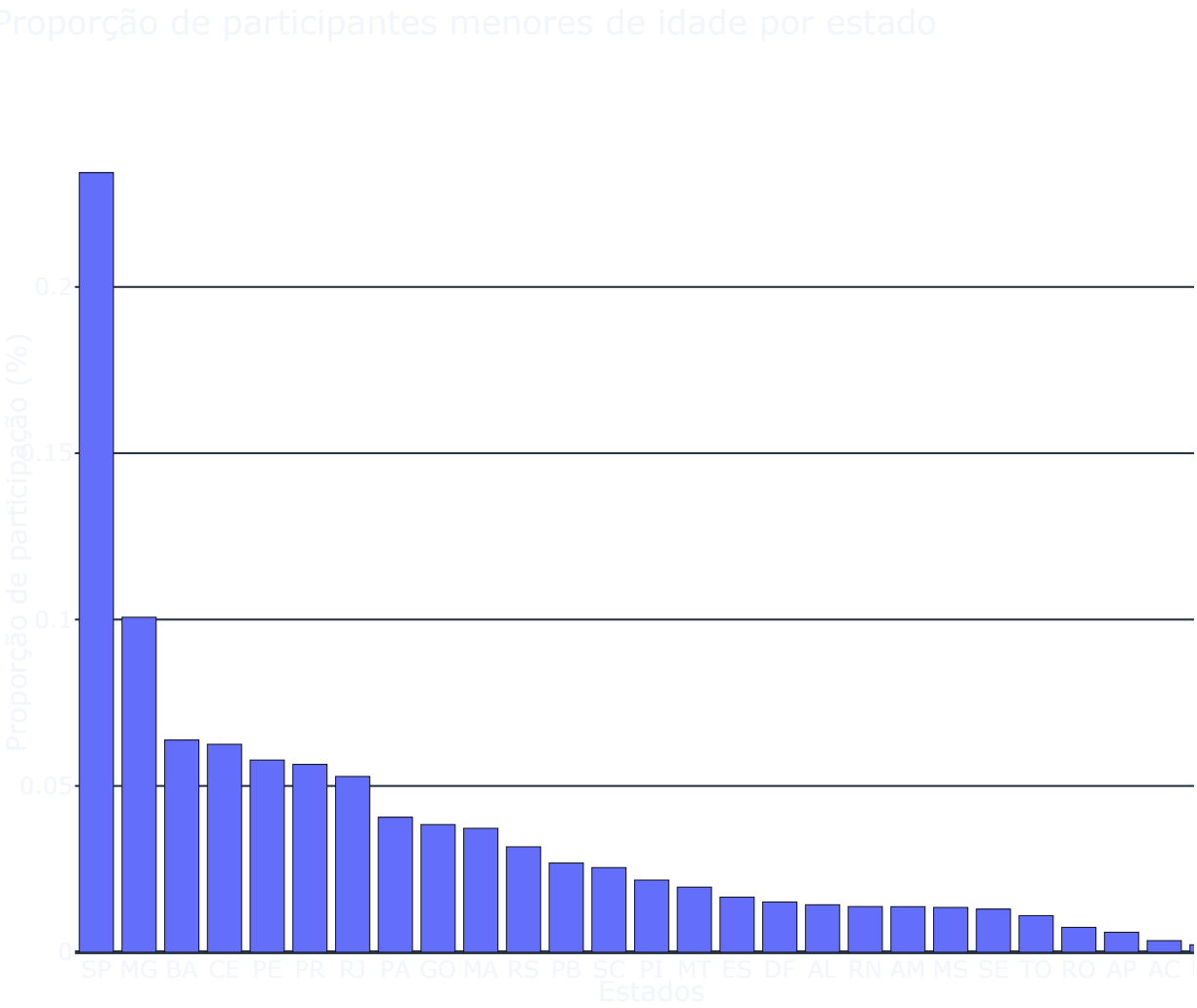
Double-click (or enter) to edit

```
provas = ['NU_NOTA_CN', 'NU_NOTA_CH',
          'NU_NOTA_MT', 'NU_NOTA_LC',
          'NU_NOTA_REDACAO']

presenca_provas = ['TP_PRESENCA_CN', 'TP_PRESENCA_CH',
                   'TP_PRESENCA_MT', 'TP_PRESENCA_LC']


data['NU_NOTA_TOTAL'] = data[provas].sum(axis=1)

data_sem_nota_zero = data[data['NU_NOTA_TOTAL'] != 0]
```

Desafio do Gui Bonzinho - Proporção de participantes menores de idade por estado

```
menores_de_idade = data[data['NU_IDADE'] <= 17]

proporcao_estado = menores_de_idade['SG_UF_RESIDENCIA'].value_counts(normalize =

fig_menores_de_idade = go.Figure()

fig_menores_de_idade.add_trace(go.Bar(x = proporcao_estado.index,
                                      y = proporcao_estado.values))

fig_menores_de_idade.update_layout(
    title_text = 'Proporção de participantes menores de idade por estado',
    xaxis_title_text = 'Estados',
    yaxis_title_text = 'Proporção de participação (%)',
    template = 'plotly_dark',
    width = 750,
```

```
    height = 600
)

fig_menores_de_idade.show()
```

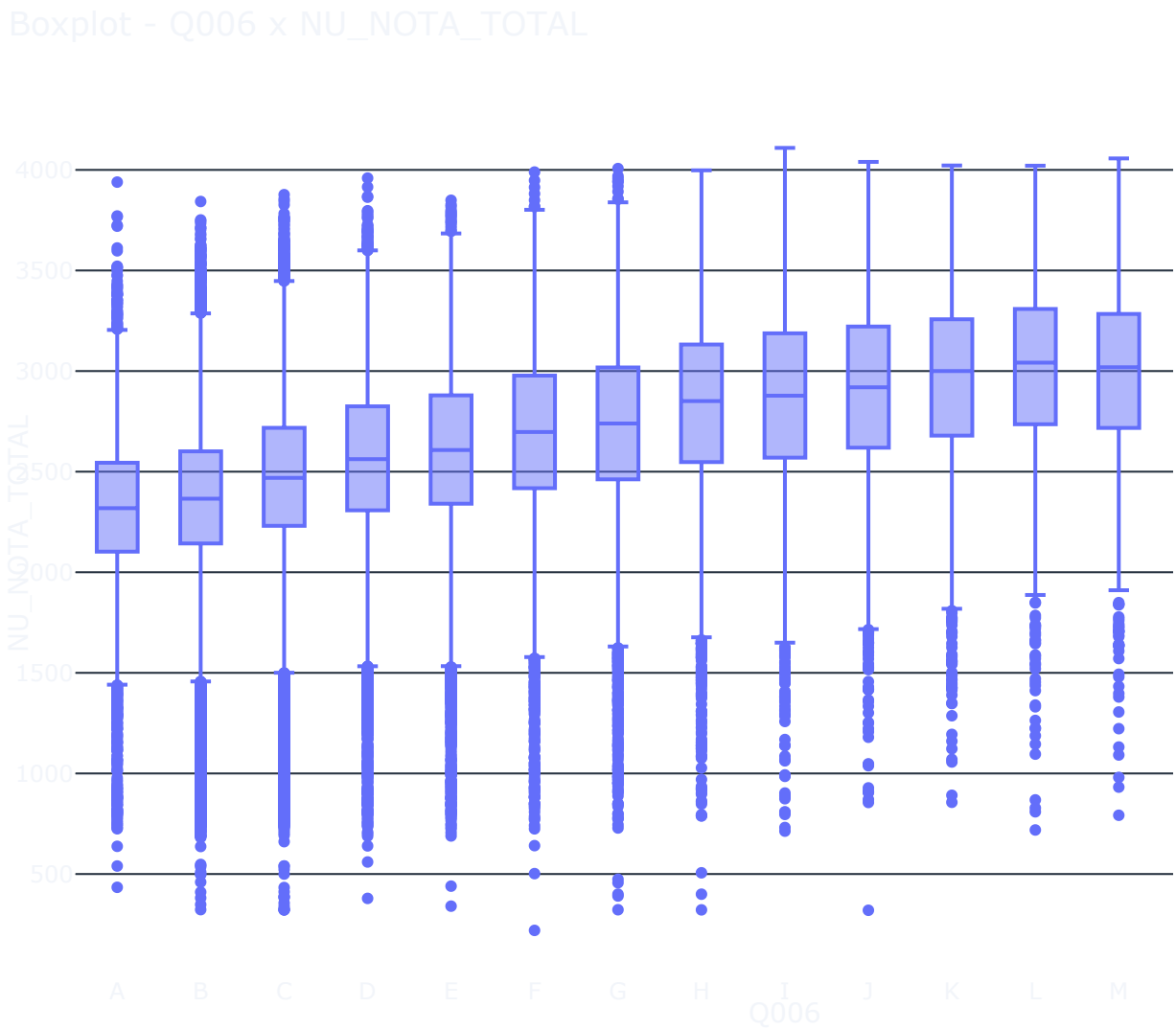


Desafio 3 - Criar uma função para gerar boxplots

```
def gerar_boxplot(dados, x, y, ordenar = False):

  '''
  Retorna um BoxPlot

  Parâmetros:
  dados : DataFrame
    Dados a serem utilizados
  x : String
    Nome da coluna que contem os valores desejados para o eixo X
  y : String
    Nome da coluna que contem os valores desejados para o eixo Y
  ordenar : Booleano, valor padrão False
```

```
    Define se os valores exibidos no eixo X serão ordenados ou não
'''

if ordenar:
  dados_ordenados = dados.sort_values(x, ascending = True)
  boxplot = px.box(data_frame=dados_ordenados, x = x, y = y)
else:
  boxplot = px.box(data_frame=dados, x = x, y = y)

boxplot.update_layout(
  title_text = ('Boxplot - %s x %s' %(x, y)),
  xaxis_title_text = ('%s' %x),
  yaxis_title_text = ('%s' %y),
  template = 'plotly_dark',
  width = 900,
  height = 600
)

boxplot.show()
```

```
gerar_boxplot(data_sem_nota_zero, 'Q006', 'NU_NOTA_TOTAL', ordenar=True)
```



Desafio 4 - Verficiar se quem tirou a nota total igual a zero foi eleminado ou não estava presente

```
1 situacao_participantes = data[data['NU_NOTA_TOTAL'] == 0][presenca_provas]
2
3 situacao_participantes.value_counts()
```

```
TP_PRESENCA_CN  TP_PRESENCA_CH  TP_PRESENCA_MT  TP_PRESENCA_LC
0               0               0               0                28998
                2               0               2                  120
                1               0               1                   39
1               0               1               0                    3
                1               1               1                    2
dtype: int64
```

```
1 round(situacao_participantes.value_counts(normalize=True) * 100 , 2)
```

```
TP_PRESENCA_CN  TP_PRESENCA_CH  TP_PRESENCA_MT  TP_PRESENCA_LC
0               0               0               0                99.44
                2               0               2                 0.41
                1               0               1                 0.13
1               0               1               0                 0.01
                1               1               1                 0.01
dtype: float64
```

Resultado Desafio 4

Total de participantes com nota total igual a zero na amostra: 29.162

Situações possíveis para nota total igual a zero na amostra:

- Participante faltou em todas as provas realizadas
  - Quantidade: 28.998
  - Porcentagem correspondente: 99.44%
- Participante faltou em algumas provas e foi eliminado em outras
  - Quantidade: 120
  - Porcentagem correspondente: 0.41%
- Participante faltou em alguma das provas e fez outras
  - Quantidade: 42
  - Porcentagem correspondente: 0.14%
- Participante foi em todas as provas objetivas e mesmo assim zerou.
  - Quantidade: 2
  - Porcentagem correspondente: 0.01%

Desafio 5 - Quem foi eliminado tira zerou ou NaN

```
1 data[data['TP_PRESENCA_LC'] == 2][provas]
```

	NU_NOTA_CN	NU_NOTA_CH	NU_NOTA_MT	NU_NOTA_LC	NU_NOTA_REDACAO
77	NaN	NaN	NaN	NaN	NaN
396	NaN	NaN	NaN	NaN	NaN
446	NaN	NaN	NaN	NaN	NaN
609	NaN	NaN	NaN	NaN	NaN
728	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...
124695	NaN	NaN	NaN	NaN	NaN
125216	NaN	NaN	NaN	NaN	NaN
125784	NaN	NaN	NaN	NaN	NaN
126769	NaN	NaN	NaN	NaN	NaN
127122	NaN	NaN	NaN	NaN	NaN

126 rows × 5 columns

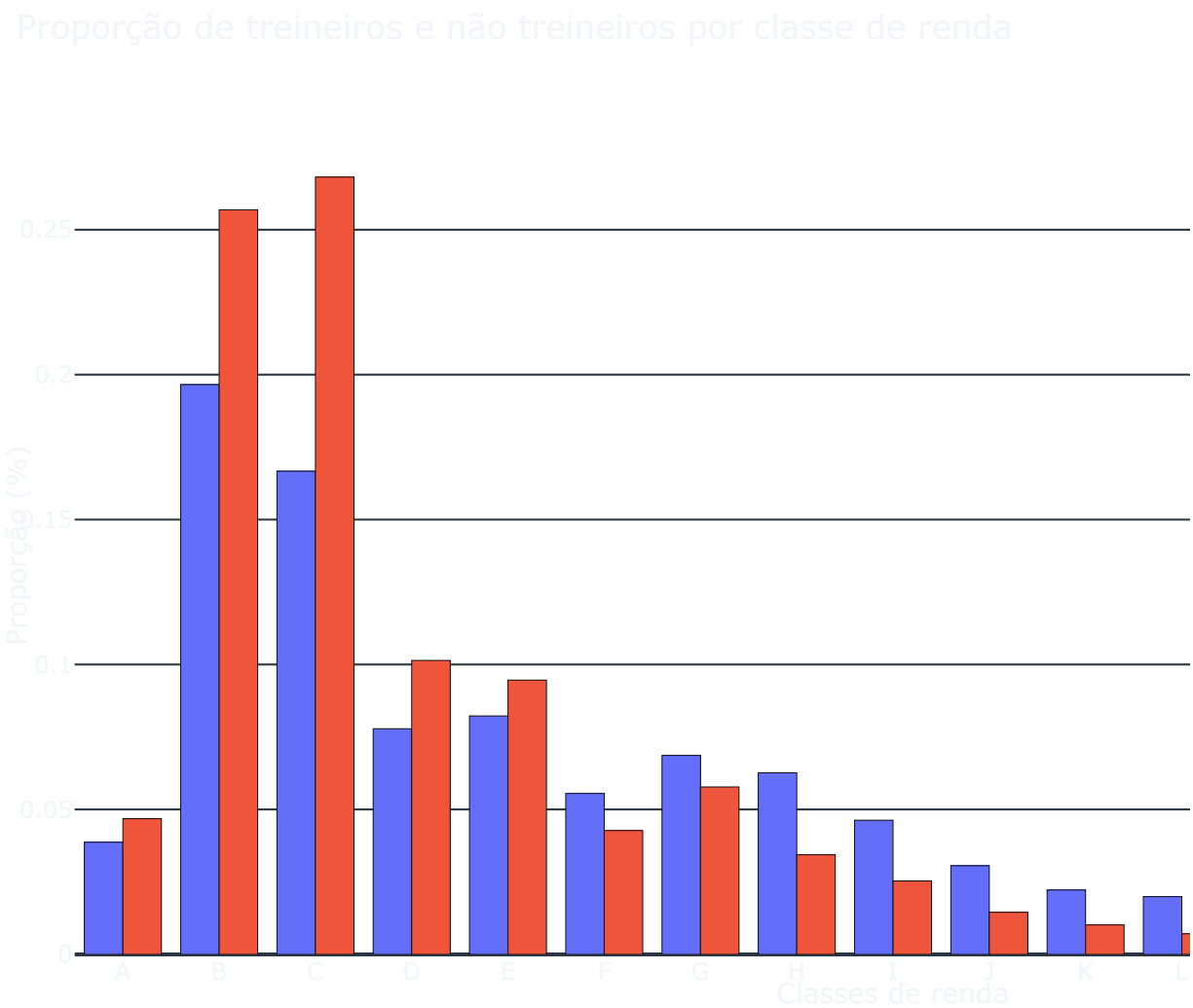
Resultado Desafio 5

Aos participantes eliminados não há registro de notas (NaN)

Desafio 6 - Verificar a proporção dos participantes de rendas mais altas e mais baixas como treineiro e não treineiro.

```
 1 treineiros = data[data['IN_TREINEIRO'] == 1]
 2 nao_treineiros = data[data['IN_TREINEIRO'] == 0]
 3
 4 proporcao__treineiros_estado = treineiros['Q006'].value_counts(normalize = True)
 5 proporcao__nao_treineiros_estado = nao_treineiros['Q006'].value_counts(normalize
 6
 7 fig_treineiros_renda = go.Figure()
 8
 9 fig_treineiros_renda.add_trace(go.Bar(x = proporcao__treineiros_estado.index,
10                                       y = proporcao__treineiros_estado.values,
11                                       name = 'Treineiros'))
12
13 fig_treineiros_renda.add_trace(go.Bar(x = proporcao__nao_treineiros_estado.index
14                                       y = proporcao__nao_treineiros_estado.valu
15                                       name = 'Não Treineiros'))
```

```
15                                                    name = 'Não treineiros' ))
16
17 fig_treineiros_renda.update_layout(
18     title_text = 'Proporção de treineiros e não treineiros por classe de renda',
19     xaxis_title_text = 'Classes de renda',
20     yaxis_title_text = 'Proporção (%)',
21     template = 'plotly_dark',
22     width = 1000,
23     height = 600
24 )
25
26 fig_treineiros_renda.show()
```



Desafio 7 - Boxplot olhando para a Q0025 (Acesso a internet)

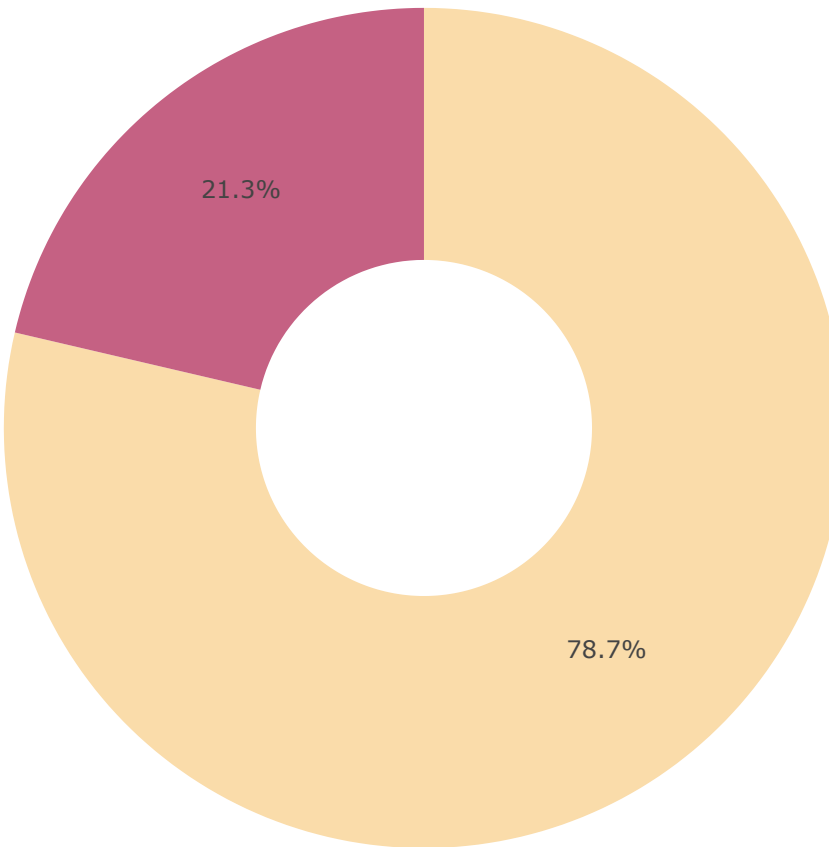
Situações para Q025:

- Não possui acesso a internet (A)
- Possui acesso a internet (B)

```
1 sem_acesso = dados_sem_nota_zero[dados_sem_nota_zero['Q025'] == 'A']
```

```
sem_acesso = data_sem_nota_zero[data_sem_nota_zero['Q025'] == 'A']
acesso = data_sem_nota_zero[data_sem_nota_zero['Q025'] == 'B']

fig_acesso_internet = go.Figure()


fig_acesso_internet.add_trace(go.Box(y = acesso['NU_NOTA_TOTAL'],
                                     name = 'Acesso à Internet',
                                     marker_color = '#fadcaa'))

fig_acesso_internet.add_trace(go.Box(y = sem_acesso['NU_NOTA_TOTAL'],
                                     name = 'Sem Acesso à Internet',
                                     marker_color = '#c56183'))

fig_acesso_internet.update_layout(
    title_text = 'Boxplot - Acesso a Internet x Nota Total',
    xaxis_title_text = 'Q025',
    yaxis_title_text = 'Nota Total',
    template = 'plotly_dark',
    width = 900,
    height = 600
)

```

```
colors= ['#fadcaa', '#c56183']

labels = ['Acesso à Internet', 'Sem Acesso à Internet']

values = [len(acesso),
          len(sem_acesso)]

fig_linguas = go.Figure()

fig_linguas.add_trace(go.Pie(labels=labels, values=values,
                             hole=.4, marker_colors=colors))

fig_linguas.update_layout(
    title_text='Porcentagem de participantes - ENEM 2019',
    template = 'plotly_dark',
    width = 900,
    height = 600
)

fig_linguas.show()
```



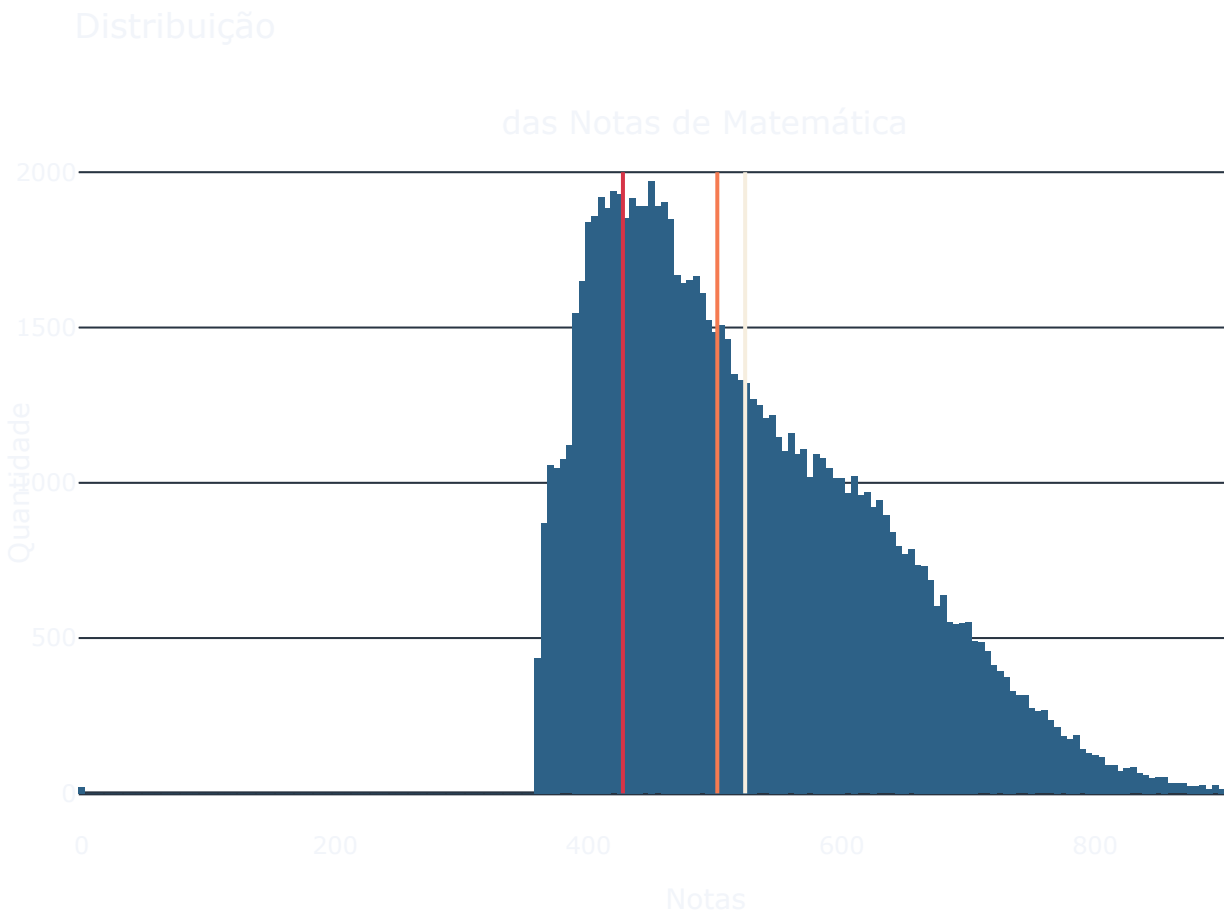
Dia 3 - Desafios

Desafio 1 - Exibir o gráfico de distribuição das notas LC e MT juntamente as suas médias, medianas e moda.

```
fig = make_subplots(rows = 1, cols = 2,
                    subplot_titles = ('das Notas de Matemática',
                                      'das Notas de Linguagens e Códigos'))

# Nota matemática

fig.add_trace(go.Histogram(x = data_sem_nota_zero['NU_NOTA_MT'], name = 'Notas M
                           marker_color = '#2d6187'), row = 1, col = 1)


fig.add_trace(go.Scatter(x=[data_sem_nota_zero['NU_NOTA_MT'].median(), data_sem_
                            y=[0,2000],
                            mode="lines",
                            legendgroup="Mediana",
                            showlegend=True,
                            marker=dict(size=12,
                                        line=dict(width=0.8),
                                        color="#f57b51"
                                        ),
                            name="Mediana"
                            ), row = 1, col = 1,)

fig.add_trace(go.Scatter(x=[data_sem_nota_zero['NU_NOTA_MT'].mean(), data_sem_no
                            y=[0,2000],
                            mode="lines",
                            legendgroup="Média",
                            showlegend=True,
                            marker=dict(size=12,
                                        line=dict(width=0.8),
                                        color="#f6eedf"
                                        ),
                            name="Média"
                            ), row = 1, col = 1,)

fig.add_trace(go.Scatter(x=[data_sem_nota_zero['NU_NOTA_MT'].mode()[0], data_sem
                            y=[0,2000],
                            mode="lines",
                            legendgroup="Moda",
                            showlegend=True,
                            marker=dict(size=12,
                                        line=dict(width=0.8),
                                        color="#d63447"
                                        ),
                            name="Moda"
```

```
                            ), row = 1, col = 1,)

# Nota Lc

fig.add_trace(go.Histogram(x = data_sem_nota_zero['NU_NOTA_LC'], name = 'Notas L
                           marker_color = '#2d6187'), row = 1, col = 2)


fig.add_trace(go.Scatter(x=[data_sem_nota_zero['NU_NOTA_LC'].median(), data_sem_
                            y=[0,1400],
                            mode="lines",
                            legendgroup="Mediana",
                            showlegend=False,
                            marker=dict(size=12,
                                        line=dict(width=0.8),
                                        color="#f57b51"
                                        ),
                            name="Mediana"
                            ), row = 1, col = 2,)

fig.add_trace(go.Scatter(x=[data_sem_nota_zero['NU_NOTA_LC'].mean(), data_sem_no
                            y=[0,1400],
                            mode="lines",
                            legendgroup="Média",
                            showlegend=False,
                            marker=dict(size=12,
                                        line=dict(width=0.8),
                                        color="#f6eedf",
                                        ),
                            name="Média"
                            ), row = 1, col = 2,)

fig.add_trace(go.Scatter(x=[data_sem_nota_zero['NU_NOTA_LC'].mode()[0], data_sem
                            y=[0,1400],
                            mode="lines",
                            legendgroup="Moda",
                            showlegend=False,
                            marker=dict(size=12,
                                        line=dict(width=0.8),
                                        color="#d63447"
                                        ),
                            name="Moda"
                            ), row = 1, col = 2)

fig.update_xaxes(title_text='Notas')

fig.update_yaxes(title_text='Quantidade')

fig.update_layout(
    title_text = 'Distribuição',
    template = 'plotly_dark',
    bargap = .05,
)

fig.show()
```

100



Resultado Desafio 1

- As notas de Matemática apresentam uma distribuição assimétrica positiva
- As notas de Linguagens e Códigos apresentam uma distribuição assimétrica negativa

Desafio 2 e 3 - Melhorar a visualização da matriz de correlação, realizar uma análise mais detalhada sobre as correlações e comparar elas entre o Brasil e seu estado.

```
correlaca_prova_brasil = data_sem_nota_zero[provas].corr()

correlaca_prova_sp = data_sem_nota_zero[data_sem_nota_zero['SG_UF_RESIDENCIA'] =
```

```
fig_corr = make_subplots(rows = 1, cols = 2,
                         subplot_titles = ('Brasil', 'São Paulo'),
                         shared_yaxes = True)

```

```
fig_corr.add_trace(go.Heatmap(
                   z=correlaca_prova_brasil,
                   x=['Ciências Naturais', 'Ciências Humanas', 'Matemática', 'Li
                   y=['Ciências Naturais', 'Ciências Humanas', 'Matemática', 'Li
                   colorscale = 'Peach',
                   showscale=False,
                   xgap=1, ygap=1,
                   colorbar_thickness=20,
                   colorbar_ticklen=3,
                   hoverongaps = False),row = 1, col = 1)


fig_corr.add_trace(go.Heatmap(
                   z=correlaca_prova_sp,
                   x=['Ciências Naturais', 'Ciências Humanas', 'Matemática', 'Li
                   y=['Ciências Naturais', 'Ciências Humanas', 'Matemática', 'Li
                   colorscale = 'Peach',
                   xgap = 1, ygap = 1,
                   colorbar_thickness=20,
                   colorbar_ticklen=3,
                   hoverongaps = False),row = 1, col = 2)

fig_corr.update_layout(
    title_text = 'Correlação - Notas das Provas',
    template = 'plotly_dark',
)

fig_corr.show()
```

Resultado Desafio 2 e 3

Como podemos observar a correlação entre as notas das diversas provas no estado de São Paulo se comporta de forma bem semelhante ao restante do Brasil.

Desafio 4 - Análisar a correlação entre matemática e linguagens.

A existência de uma correlção de 58% entre as notas de matemática e linguagens pode estar associada a casualidade, contudo podemos considerar que esta correlação se faz presente pois os alunos que estudaram para uma das provas tendem a estudar para todas as outras, ou seja, possivelmente um participante pode apresentar um desemepenho semelhante nas diferentes provas

Dia 4 - Desafios

```
1 provas_entrada = ['NU_NOTA_CH', 'NU_NOTA_LC', 'NU_NOTA_CN', 'NU_NOTA_REDACAO']
2 provas_saida = 'NU_NOTA_MT'
3
4 data_sem_nota_zero = data_sem_nota_zero[provas].dropna()
```

```
1 X  = data_sem_nota_zero[provas_entrada]
2
3 y =  data_sem_nota_zero[provas_saida]
4
5 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Criando modelos

Modelo - Linear SVR

Baseia-se na elaboração de um equação linear utilizando variáveis independente para predizer uma variável dependente.

- Equação linear: $y = a_1 x_1 + b$
- No nosso caso:

$$NotaMT = Coef_{NotaCH} * NotaCH + Coef_{NotaCN} * NotaCN + Coef_{NotaLC} * Nota$$

```
1 linear_svr = LinearSVR().fit(X_train, y_train)
2
```

```
y_pred_linear_svr = linear_svr.predict(X_test)

mae_linear_svr = mean_absolute_error(y_test, y_pred_linear_svr)

mse_linear_svr = mean_squared_error(y_test, y_pred_linear_svr)

rsq_linear_svr = r2_score(y_test, y_pred_linear_svr)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:947: ConvergenceWa

Liblinear failed to converge, increase the number of iterations.
```

Modelo - Polynomial Regression

É um extensão ao modelo de regressão linear, pois é capaz de agregar preditores adicionais ao modelo.

Este processo funciona elevando os preditores originais a uma nova potência, possibilitando que o modelo se ajuste a dados não lineares.

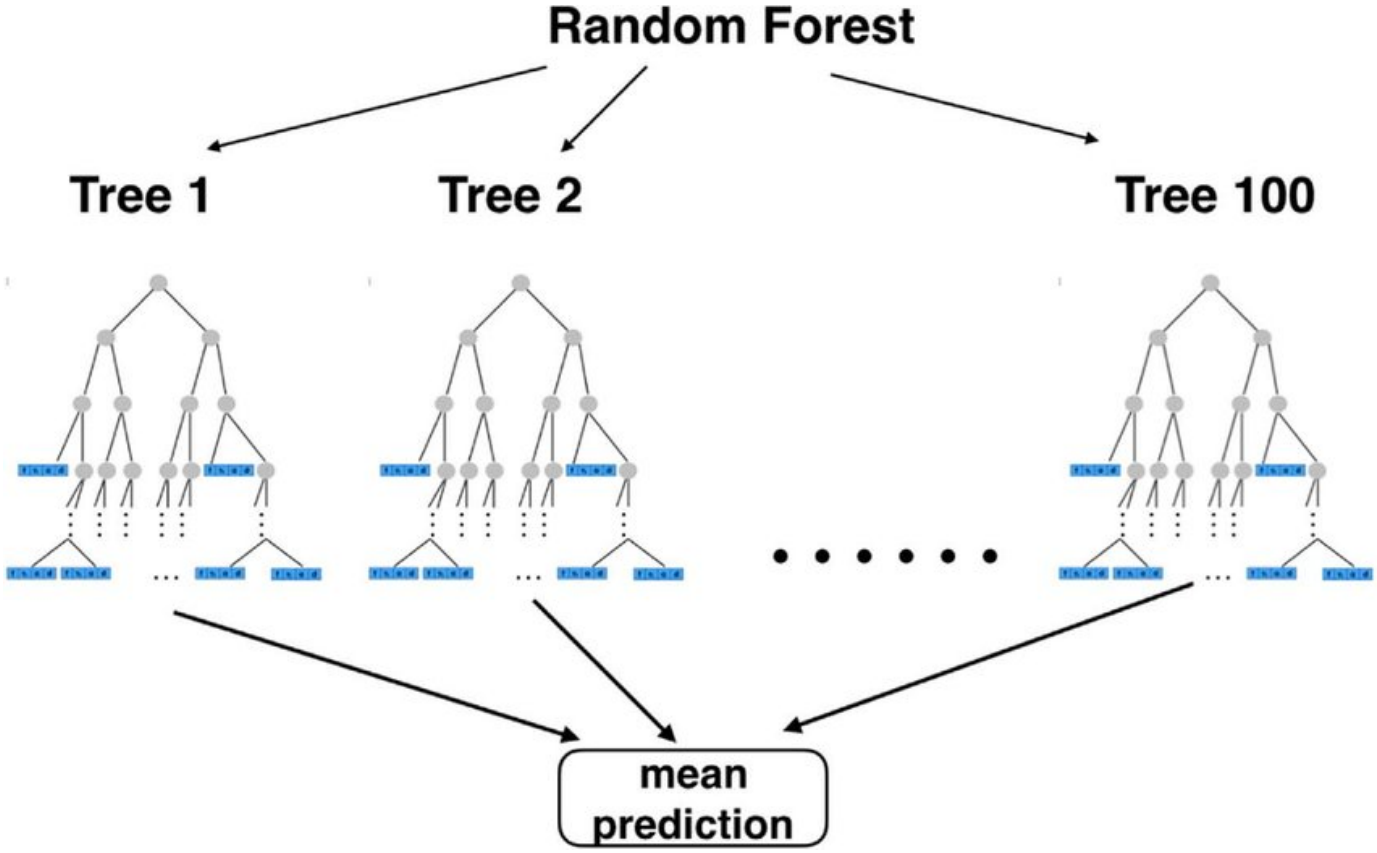
Exemplo:

- Equção linear original: $y = a_1 x_1 + b$
- Equação polinomial criada: $y = a_1 x_1 + a_2 x^2{}_1 + b$
- Caso utilizemos uma potência quadrada, nosso caso ficará assim:

$$NotaMT = Coef_{NotaCH} * NotaCH + Coef_{2NotaCH} * NotaCH^2 + Coef_{NotaCN} * Not$$
$$+ Coef_{NotaLC} * NotaLC + Coef_{2NotaLC} * NotaLC^2 + Coef_{NotaRed} * NotaRED + C$$

```
polynomial_features = PolynomialFeatures(degree = 2)

polynomial_features.fit(X_train)

X_train_poly = polynomial_features.transform(X_train)
X_test_poly = polynomial_features.transform(X_test)

X_poly = polynomial_features.fit_transform(X)

polynomial_regression = LinearRegression().fit(X_train_poly, y_train)

y_pred_poly = polynomial_regression.predict(X_test_poly)

mae_poly = mean_absolute_error(y_test, y_pred_poly)

mse_poly = mean_squared_error(y_test, y_pred_poly)

rsq_poly =  r2_score(y_test, y_pred_poly)
```

Modelo - Random Forest Regressor




```
rfr = RandomForestRegressor().fit(X_train, y_train)

y_pred_rfr = rfr.predict(X_test)

mae_rfr = mean_absolute_error(y_test, y_pred_rfr)

mse_rfr = mean_squared_error(y_test, y_pred_rfr)

rsq_rfr =  r2_score(y_test, y_pred_rfr)
```

Modelo - Dummy Regressor Median

Realiza predições baseando-se em métricas simples. Neste caso utilizarei a mediana.

```
dummy_regressor = DummyRegressor(strategy='median')

dummy_regressor.fit(X_train, y_train)

y_pred_dummy = dummy_regressor.predict(X_test)

mae_dummy_reg = mean_absolute_error(y_test, y_pred_dummy)

mse_dummy_reg = mean_squared_error(y_test, y_pred_dummy)

rsq_dummy_reg =  r2_score(y_test, y_pred_dummy)
```

Comparando modelos

Métricas utilizadas:

- MAE: É a média da diferença absoluta entre os valores reais e os valores preditos.
- MSE: É a média da diferença quadrada entre os valores reais e os valores preditos.
  - Valor sempre positivo e quanto menor melhor
- R²: Quantifica a diferença entre a linha elaborada pelo equação do modelo de regressão e a média.
  - Valor que varia de 0 a 1, sendo 0 o indicador que o modelo proposto não melhora a predição e 1 o indicador de uma predição perfeita

Através de um DataFrame

```
modelos = [('Linear SVR', mae_linear_svr, mse_linear_svr,rsq_rfr),
           ('Polynomial Regression', mae_poly, mse_poly, rsq_poly),
           ('Random Forest Regressor', mae_rfr, mse_rfr, rsq_rfr),
           ('Dummy Regressor', mae_dummy_reg, mse_dummy_reg, rsq_dummy_reg)]

df_comparando_modelos = pd.DataFrame(modelos, columns = ['Modelo', 'MAE', 'MSE',

df_comparando_modelos.head()
```

	Modelo	MAE	MSE	R²
0	Linear SVR	74.069213	8754.081890	0.515917
1	Polynomial Regression	58.782090	5397.478485	0.542046
2	Random Forest Regressor	60.318493	5705.436794	0.515917
3	Dummy Regressor	88.105046	12274.403152	-0.041434

Gráficamente

```
fig_prediction = make_subplots(rows = 2, cols = 2,

                               subplot_titles = (("Linear SVR (R² = %.4f)" %rsq_
                                                 ("Polynomial (R² = %.4f)" %rsq_
                                                 ("RFR (R² = %.4f)" %rsq_rfr),
                                                 ("Dummy Median (R² = %.4f)" %rs

# Linear Regression

fig_prediction.add_trace(go.Scatter(x = y_test,
                                    y = y_test,
                                    marker_color = '#3da4ff',
                                    name = 'Predição Perfeita'),
                         row = 1, col = 1)
```

```
15
16 fig_prediction.add_trace(go.Scatter(x = y_test,
17                                       y = y_pred_linear_svr,
18                                       marker_color = '#ff3e3b',
19                                       mode = 'markers',
20                                       name = 'Predições Realizadas'),
21                            row = 1, col =1)
22
23 # Polynomial Regression
24
25 fig_prediction.add_trace(go.Scatter(x = y_test,
26                                       y = y_test,
27                                       marker_color = '#3da4ff',
28                                       name = 'Predição Perfeita',
29                                       showlegend = False),
30                            row = 1, col = 2)
31
32
33 fig_prediction.add_trace(go.Scatter(x = y_test,
34                                       y = y_pred_poly,
35                                       marker_color = '#ff3e3b',
36                                       mode = 'markers',
37                                       name = 'Predição Realizadas',
38                                       showlegend = False),
39                            row = 1, col = 2)
40
41 # RFR
42
43 fig_prediction.add_trace(go.Scatter(x = y_test,
44                                       y = y_test,
45                                       marker_color = '#3da4ff',
46                                       name = 'Predição Perfeita',
47                                       showlegend = False),
48                            row = 2, col = 1)
49
50
51 fig_prediction.add_trace(go.Scatter(x = y_test,
52                                       y = y_pred_rfr,
53                                       marker_color = '#ff3e3b',
54                                       mode = 'markers',
55                                       name = 'Predições Realizadas',
56                                       showlegend = False),
57                            row = 2, col = 1)
58
59 # Dummy Regressor
60
61 fig_prediction.add_trace(go.Scatter(x = y_test,
62                                       y = y_test,
63                                       marker_color = '#3da4ff',
64                                       name = 'Predição Perfeita',
65                                       showlegend = False),
66                            row = 2, col = 2)
67
68
69 fig_prediction.add_trace(go.Scatter(x = y_test,
```

```
                                    y = y_pred_dummy,
                                    marker_color = '#ff3e3b',
                                    mode = 'markers',
                                    name = 'Predições Realizadas',
                                    showlegend = False),
                            row = 2, col = 2)

fig_prediction.update_xaxes(title_text='Nota Real')

fig_prediction.update_yaxes(title_text='Nota Prevista')

fig_prediction.update_layout(

    template = 'plotly_dark',
       width = 1150,
    height = 600
)


fig_prediction.show()
```