

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Wagner Alberto Soares Junior

Uma análise do impacto da educação básica na porta de entrada do ensino superior

Belo Horizonte
2022

Wagner Alberto Soares Junior

Uma análise do impacto da educação básica na porta de entrada do ensino superior

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Belo Horizonte

2022

SUMÁRIO

1. Introdução	4
1.1. Contextualização.....	4
1.2. O problema proposto	4
1.3. Objetivos	5
3. Processamento/Tratamento de Dados	10
4. Análise e Exploração dos Dados	19
5. Criação de Modelos de Machine Learning.....	22
6. Interpretação dos Resultados.....	29
7. Apresentação dos Resultados.....	10
8. Links	32
REFERÊNCIAS	33
APÊNDICE	34

1. Introdução

1.1. Contextualização

A principal porta de entrada das universidades públicas e particulares é o ENEM (Exame Nacional do Ensino Médio). Entender se ele dá a mesma oportunidade para todos os Brasileiros, é importante. É claro que os últimos anos que antecedem o exame são os mais importantes, mas será que somente um ensino de qualidade nesses últimos anos, é capaz de alterar o desempenho no exame? Ou o ensino básico, dado nos anos iniciais de estudo do aluno é tão importante quanto? Este trabalho tenta questionar o impacto que o ensino básico possui dentro do desempenho do ENEM.

1.2. O problema proposto

Proponho descobrir o impacto que a qualidade do ensino da educação básica influencia nos exames de ingresso da faculdade. Com esse impacto comprovado, é fácil justificar para os governantes a importância de investir mais nessa fase da educação.

As bases que serão analisadas nesse trabalho, serão duas: A base de micro dados do censo escola de educação básica de 2021, que estão no seguinte endereço <https://www.gov.br/inep/pt-br/aceso-a-informacao/dados-abertos/microdados/censo-escolar>. E a base de dados do ENEM de 2021, que é provida no seguinte link <https://www.gov.br/inep/pt-br/aceso-a-informacao/dados-abertos/microdados/enem> .

Os anos de ambas as bases serão de 2021, é claro que não serão os mesmos alunos fazendo ambas as provas, mas o que está procurando ser medido, é o comportamento médio dos estudantes daquele município. Com a tendência do comportamento médio dos estudantes, é fácil tentar entender, se a educação básica tem impacto no ENEM.

Os dados tanto do ENEM quanto do censo de educação básica, são do ministério da educação e foram disponibilizados graças a lei de acesso de informação. Os dados de educação básica possuem informação a nível de colégios, eles registram informações por exemplo : tipo de financiamento do colégio, quantidade de professores, quantidade de salas de aula e etc.

Os dados do ENEM possuem informação a nível de aluno. Os dados contêm informação censurada de identificação, porém possuem dados como faixa etária, estado civil, sexo e informações sobre sua faixa de renda.

As duas bases foram extraídas do ano de 2021. Por isso, os alunos analisados não são os mesmos, porém não estamos olhando para um aluno individualmente e sim a tendência ao longo do tempo. Uma sugestão que fica para o Ministério da educação é a seguinte, poderia ser possível criar um identificador único para cada aluno, que acompanhasse ele em todas as bases desidentificadas. Com isso poderíamos cruzar exatamente os dados de desempenho daquele aluno ao longo do tempo.

1.3. Objetivos

O principal objetivo nesse trabalho é tentar entender como a educação básica têm a capacidade de influenciar o desempenho do aluno ao longo do tempo. É claro que a educação básica não é o único ponto que influencia o desempenho de um aluno para entrar em uma faculdade. Mas tentamos descobrir se há alguma tendência de influência nesse desempenho.

No trabalho por decisão ética, retiramos ao máximo variáveis que identifiquem pessoas por gênero, raça ou cor. Focamos totalmente a análise em variáveis que podem ser influenciadas por políticas públicas.

2. Coleta de Dados

Os dados vieram do Governo, do INEP. O Instituto Nacional de Estudos e Pesquisa Anísio Teixeira. Os dados são disponibilizados graças a Lei de acesso à informação. Ambos os datasets, foram disponibilizados no site de dados abertos. O site de dados aberto pode ser acessado com o seguinte link <https://www.gov.br/inep/pt-br/acesso-a-informacao/dados-abertos/microdados> . Os dados foram acessados no dia 20/11/2022 no período noturno. Vou disponibilizar na tabela abaixo os dados que foram usados na versão final do modelo de cada conjunto de dados.

O primeiro conjunto de dados são os dados do ensino básico, vou descrever aqui o subconjunto de colunas que eu selecionei para a construção do modelo. O dicionário de dados completo está anexado no final desse trabalho.

Dados da educação básica

Nome da coluna/campo	Descrição	Tipo
CO_MUNICIPIO	Código do Município	Numérico
TP_DEPENDENCIA	Dependência Administrativa	Numérico
TP_LOCALIZACAO	Localização, se urbana ou rural	Numérico
IN_AGUA_POTAVEL	Fornece água potável para o consumo humano	Numérico
IN_ENERGIA_INEXISTENTE	Abastecimento de energia elétrica - Não há energia elétrica	Numérico
IN_ESGOTO_REDE_PUBLICA	Esgoto sanitário - Rede pública	Numérico

IN_BIBLIOTECA_SALA_LEITURA	Dependências físicas existentes e utilizadas na escola - Biblioteca e/ou Sala de leitura	Numérico
IN_QUADRA_ESPORTES	Dependências físicas existentes e utilizadas na escola - Quadra de esportes coberta ou descoberta	Numérico
IN_SALA_ESTUDIO_DANCA	Dependências físicas existentes e utilizadas na escola - Sala/estúdio de dança	Numérico
QT_SALAS_UTILIZADAS	Número de salas de aula utilizadas na escola (dentro e fora do prédio)	Numérico
QT_DESKTOP_ALUNO	Quantidade de computadores em uso pelos alunos - Computador de mesa (desktop)	Numérico
QT_COMP_PORTATIL_ALUNO	Quantidade de computadores em uso pelos alunos - Computador portátil	Numérico
IN_INTERNET	Acesso à Internet	Numérico
TP_ATIVIDADE_COMPLEMENTAR	Atividade Complementar	Numérico

Esse subconjunto de dados, foi selecionado a partir de análise exploratória. Após uma rodada inicial de cálculos de correlação, essas variáveis foram as que sobressaíram, tanto do ponto de vista de importância, através de serem acessíveis em serem alteradas a partir de políticas públicas, mas também porque mostram características físicas do colégio que podem ser mensuradas.

Outra base de dados que foi utilizada nesse relatório foi a do ENEM (Exame nacional do ensino médio). O ENEM é uma prova de nível médio, que é aplicada em todo o Brasil, ao mesmo tempo. Essa prova tem o objetivo de ser a porta de entrada unificada das Universidades públicas para todos os estudantes do ensino médio. O ENEM é aplicado no mesmo dia e no mesmo horário para todos os estudantes em todo o país. É uma operação gigantesca a nível nacional. Neste trabalho o principal objetivo é justamente entender o desempenho dos estudantes no ENEM de forma agrupada por município, com base no ensino básico que eles tiveram. Para isso ser possível, foi adotada a seguinte decisão, não seria utilizado nenhum dado do ENEM, exceto o código do município e a nota média das provas, os tratamentos necessários para isso ocorrer serão apresentados no próximo capítulo.

Dados do Enem 2021

Nome da coluna/campo	Descrição	Tipo
CO_MUNICIPIO_ESC	Código da escola do Município	Numérico
TP_PRESENCA_CN	Presença na prova objetiva de Ciências da Natureza	Numérico
TP_PRESENCA_CH	Presença na prova objetiva de Ciências Humanas	Numérico
TP_PRESENCA_LC	Presença na prova objetiva de Linguagens e Códigos	Numérico
TP_PRESENCA_MT	Presença na prova objetiva de Matemática	Numérico

NU_NOTA_CN	Nota da prova de Ciências da Natureza	Numérico
NU_NOTA_CH	Nota da prova de Ciências Humanas	Numérico
NU_NOTA_LC	Nota da prova de Linguagens e Códigos	Numérico
NU_NOTA_MT	Nota da prova de Matemática	Numérico

3. Processamento/Tratamento de Dados

O primeiro passo foi entender os datasets, ambos os dicionários de dados passaram por uma leitura. Nesse processo da leitura e entendimento das variáveis de interesse, algumas foram analisadas através de histogramas e construção de gráficos de distribuição. Além de entender a quantidade de registros nulos. Nessa etapa foram tomadas as seguintes decisões, tentar ao máximo usar colunas que possuam uma grande quantidade de registros presentes, para evitar ao máximo subjetividade ao preencher os dados faltantes. Outro ponto principal também foi evitar características das pessoas e focar nos dados que qualificavam os colégios de educação básica.

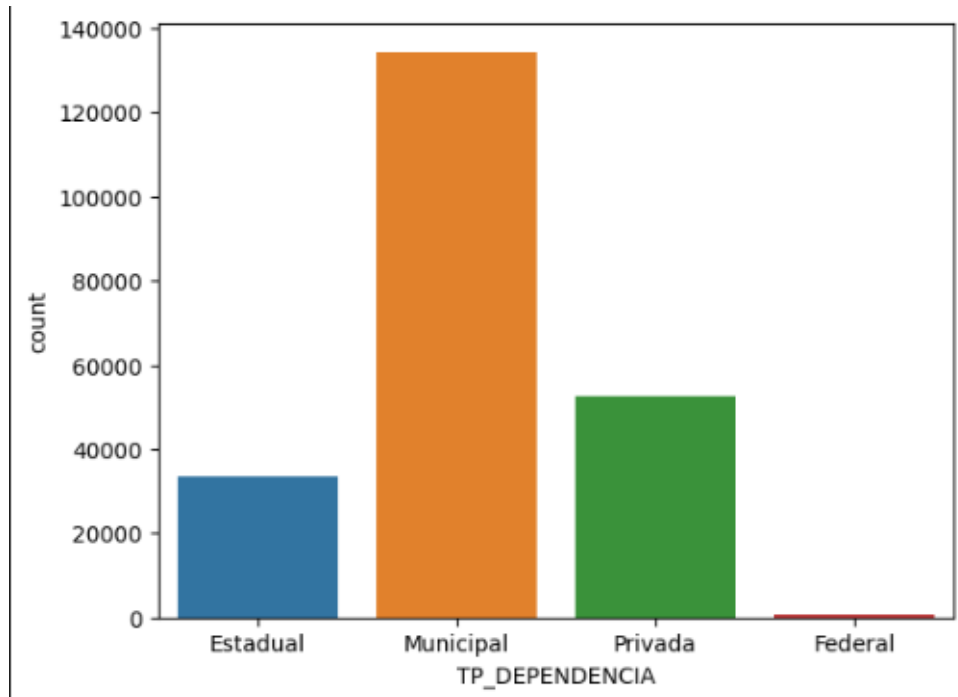
Após essa primeira etapa de análise, um pouco subjetiva, variáveis selecionadas passaram pelo processo de teste de correlação e distribuição, foram feitos alguns testes com variáveis que não estão presentes aqui, porém vou focar na decisão das variáveis que foram utilizadas.

Um dado importante, é que o dataset completo de educação básica possui 221140 linhas.

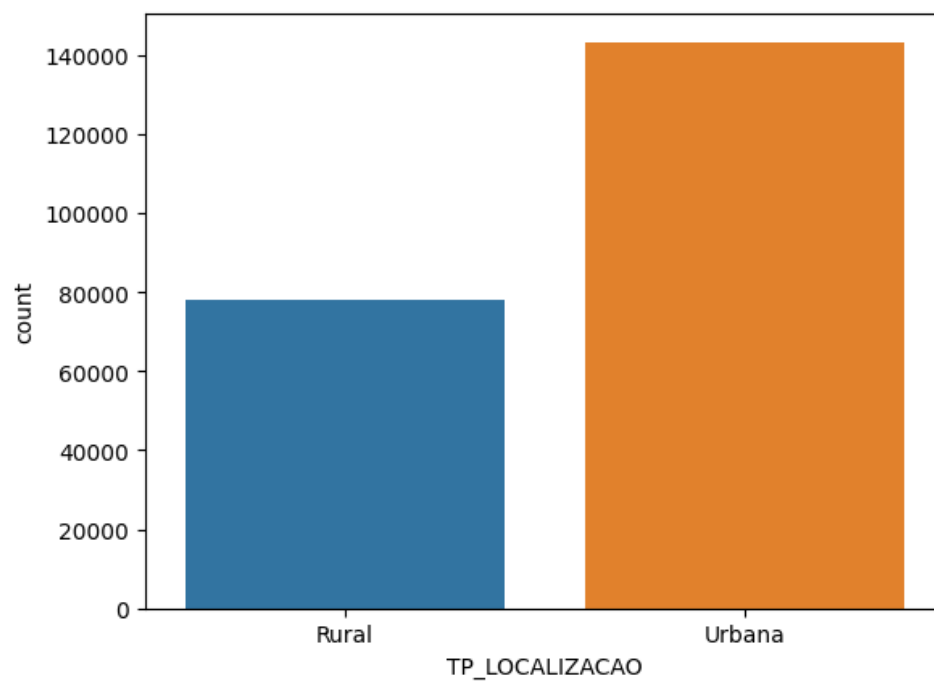
No dataset de educação básica, as seguintes variáveis se sobressaíram : CO_MUNICIPIO, TP_DEPENDENCIA, TP_LOCALIZACAO, IN_AGUA_POTAVEL, IN_ENERGIA_INEXISTENTE, IN_ESGOTO_REDE_PUBLICA, IN_BIBLIOTECA_SALA_LEITURA, IN_QUADRA_ESPORTES, IN_SALA_ESTUDIO_DANCA, QT_SALAS_UTILIZADAS, QT_DESKTOP_ALUNO, QT_COMP_PORTATIL_ALUNO, IN_BANDA_LARGA. Abordarei agora as informações de cada uma delas.

CO_MUNICIPIO : Código do Município, é um identificador único. Cada município Brasileiro possui um código de 7 dígitos. Ele é necessário para ser feito o enriquecimento de uma base com a outra. Possui 5570 códigos de municípios únicos a base.

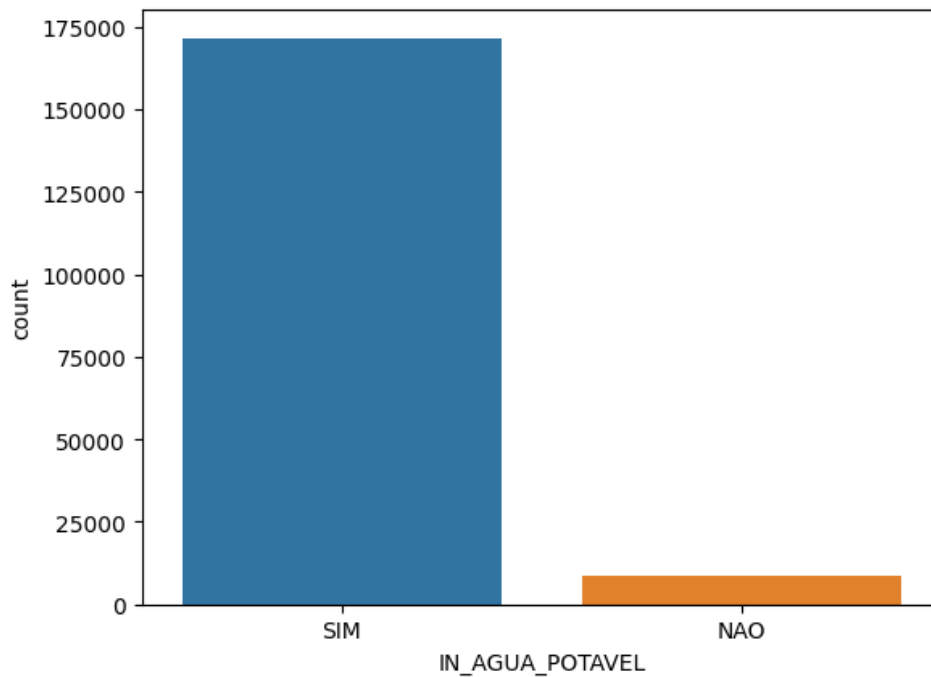
TP_DEPENDENCIA : Dependência administrativa, indica de qual é o tipo de administração do colégio. O gráfico abaixo mostra os tipos possíveis, porém as categorias Estadual / Municipal / Federal, foram categorizados como público e a categoria privada, alterou-se para privado.



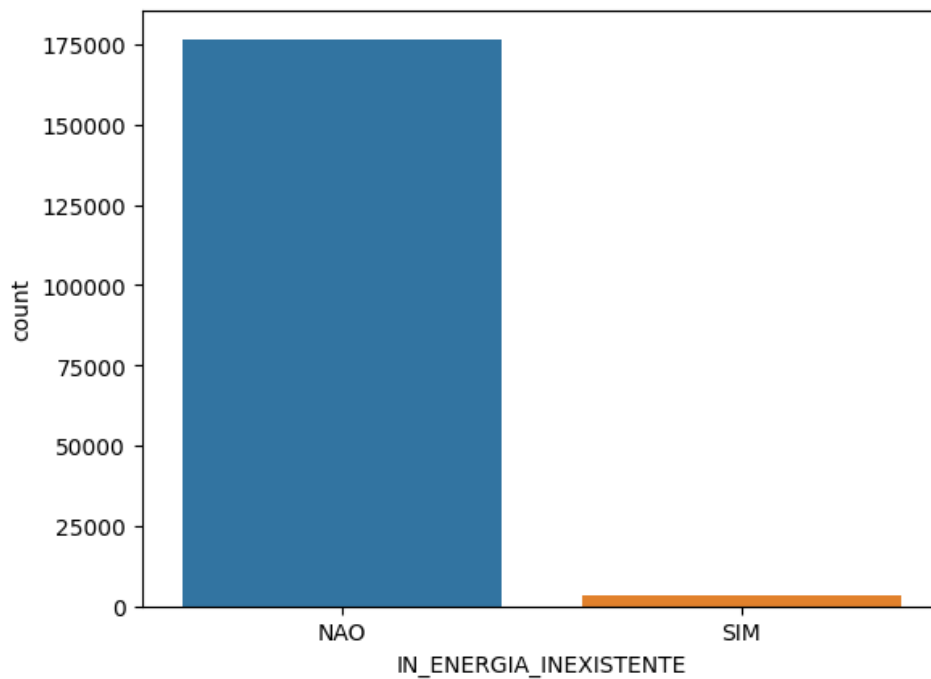
TP_LOCALIZACAO: Localização da escola, esse dado indica se o colégio está na zona rural ou urbana. Possui duas categorias, Rural ou Urbana.



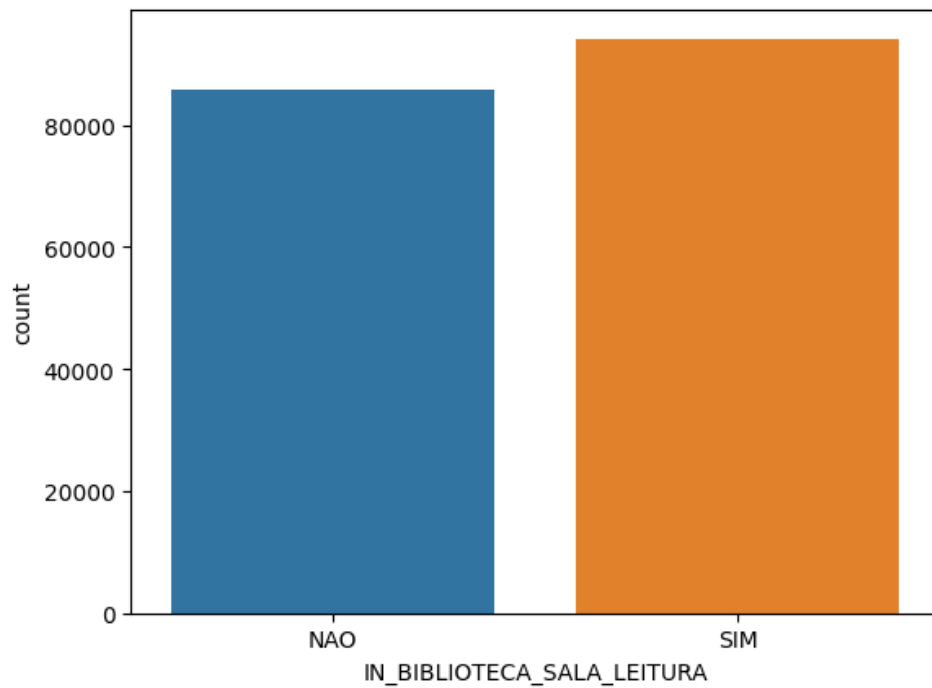
IN_AGUA_POTAVEL : Fornece água potável para o consumo humano, esse dado mostra a condição básica de fornecer água potável para os alunos.



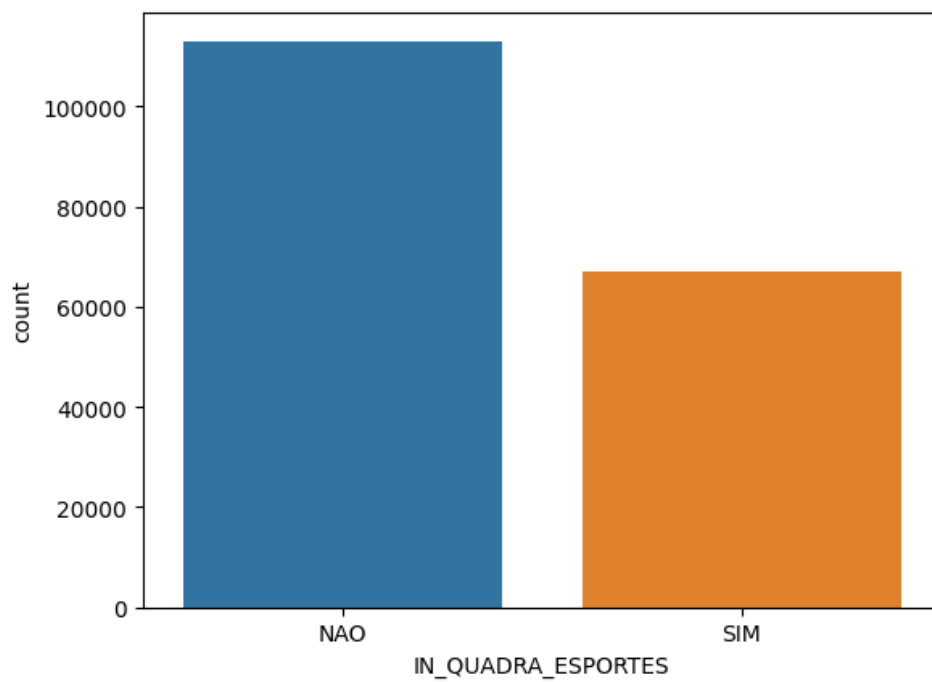
IN_ENERGIA_INEXISTENTE: Abastecimento de energia elétrica - Não há energia elétrica. Indica os lugares que não possuem energia elétrica para dar aulas.



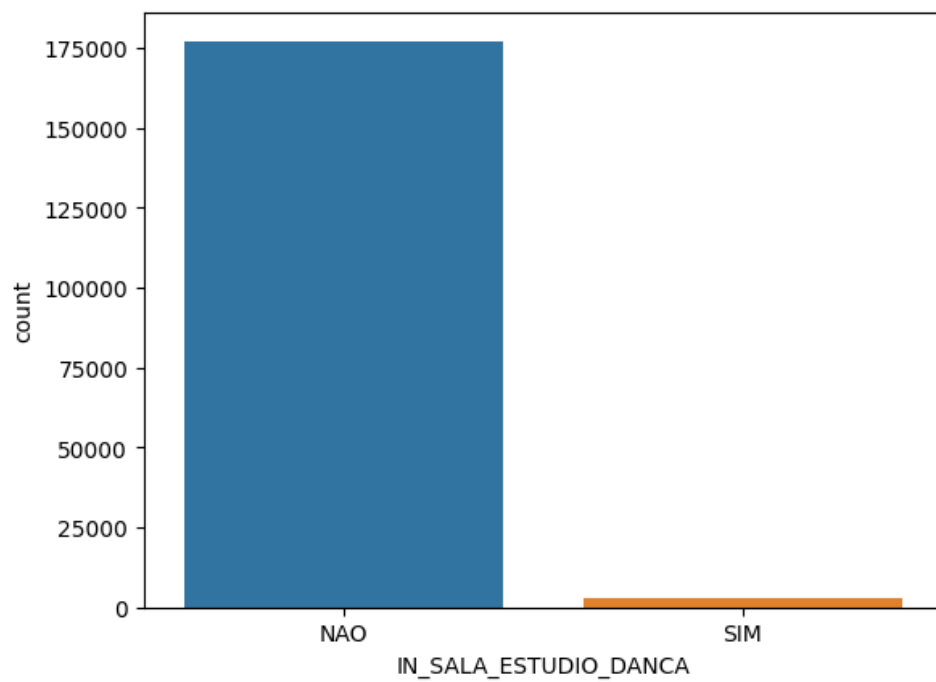
IN_BIBLIOTECA_SALA_LEITURA: Dependências físicas existentes e utilizadas na escola - Biblioteca e/ou Sala de leitura.



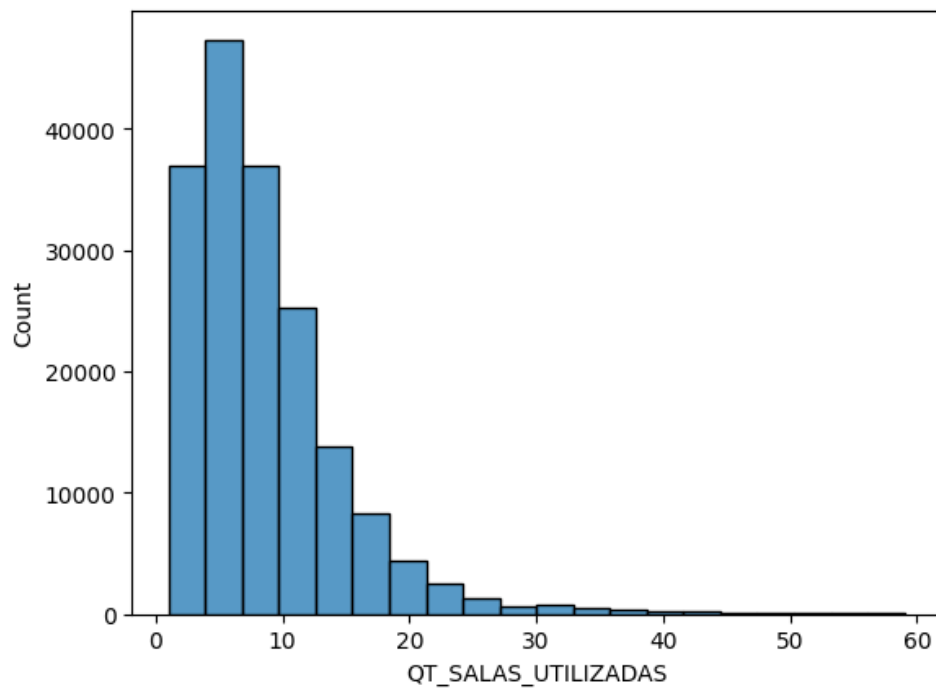
IN_QUADRA_ESPORTES : Dependências físicas existentes e utilizadas na escola - Quadra de esportes coberta ou descoberta.



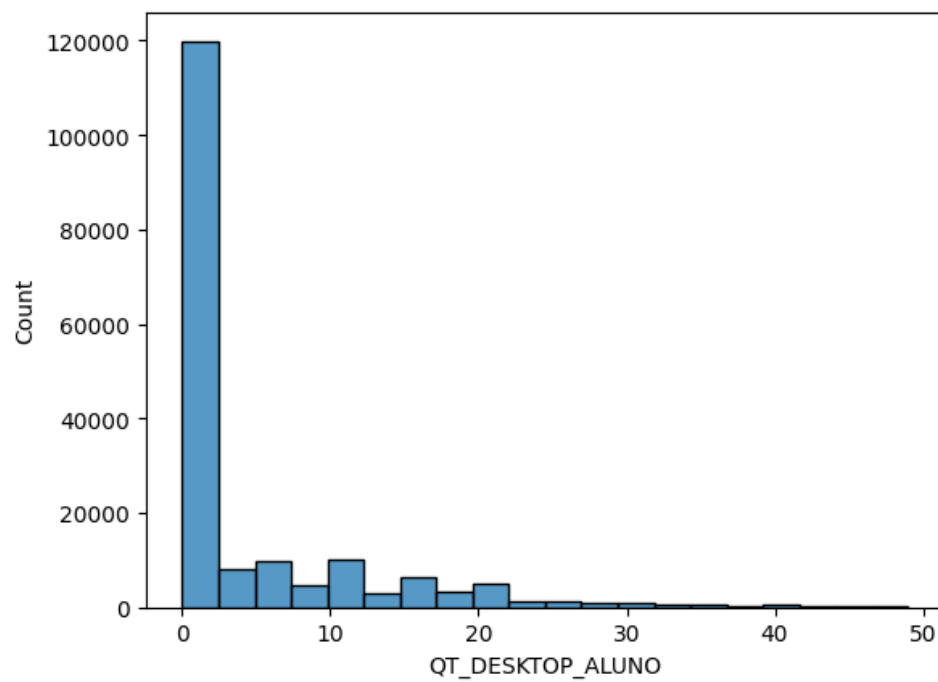
IN_SALA_ESTUDIO_DANCA : Dependências físicas existentes e utilizadas na escola - Sala/estúdio de dança



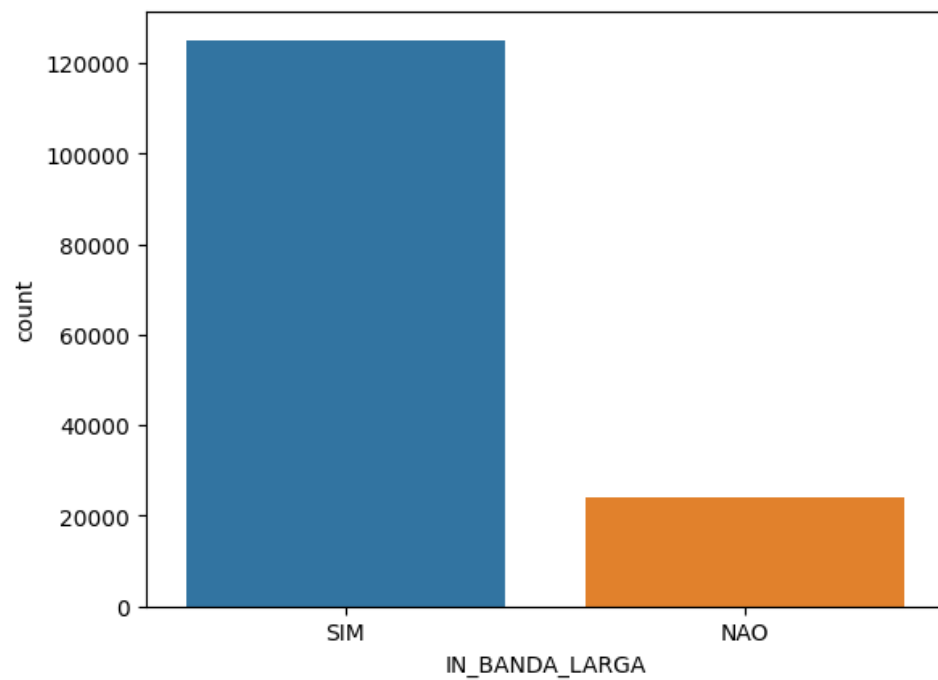
QT_SALAS_UTILIZADAS: Número de salas de aula existentes na escola



QT_DESKTOP_ALUNO : Quantidade de computadores em uso pelos alunos - Computador de mesa (desktop).



IN_BANDA_LARGA : Internet Banda Larga.



Como foi interesse de analisar os dados a nível de município, todos os dados foram agrupados para o nível de município. Para fazer isso, foram criadas duas funções que estão no arquivo utils.py. A primeira calcula a porcentagem de uma certa categoria para todos os municípios. É a função anexada abaixo.

```
def calculate_percentage_of_group(
    df, id_column, indicator_column, group_interest, new_column_name
):
    df = df[[id_column, indicator_column]]
    cols_to_group = [id_column, indicator_column]
    df_grouped = df.groupby(cols_to_group, as_index=False).size()
    df_grouped[new_column_name] = (
        100
        * df_grouped["size"]
        / df_grouped.groupby([id_column])["size"].transform("sum")
    )
    df_grouped = df_grouped[df_grouped[indicator_column] == group_interest]
    return df_grouped
```

A segunda função cria quantitativos a nível de município, no caso o somatório.

```
def calculate_sum_of_group(df, id_column, indicator_column, new_column_name):
    df = df[[id_column, indicator_column]]
    df = df.groupby(id_column, as_index=False).agg(np.sum)
    df = df.rename(columns={indicator_column: new_column_name})
    return df
```

Os filtros e tratamentos que foram utilizados nas variáveis escolhidas foram os seguintes :

CO_MUNICIPIO: Nenhum tratamento aplicado, foi a variável utilizada para fazer JOIN com a outra base e agrupamento.

TP_DEPENDENCIA : Tipo dependência foi mapeado seguindo a seguinte tabela:

Valor inicial	Valor convertido
1	publico
2	publico
3	Publico
4	Privado

TP_LOCALIZACAO: Tipo de localização, foi mapeado seguindo a seguinte tabela:

Valor inicial	Valor convertido
1	urbana
2	rural

TP_ATIVIDADE_COMPLEMENTAR : Tipo de atividade complementar foi mapeado seguindo a seguinte tabela:

Valor inicial	Valor convertido
0	não
1	sim
2	sim

PS: os valores nulos da coluna TP_ATIVIDADE_COMPLEMENTAR, foram completados com o valor não.

As variáveis a seguir, não foram mapeadas ou completadas anteriormente ao cálculo, elas foram completadas após o cálculo de porcentagem de cada uma para cada município.

IN_AGUA_POTAVEL, IN_ENERGIA_INEXISTENTE, IN_ESGOTO_REDE_PUBLICA, IN_BIBLIOTECA_SALA_LEITURA, IN_QUADRA_ESPORTES, IN_SALA_ESTUDIO_DANCA, QT_SALAS_UTILIZADAS, QT_DESKTOP_ALUNO, QT_COMP_PORTATIL_ALUNO, IN_BANDA_LARGA.

Filtramos as linhas que não seguiam as seguintes faixas de dados:

- Tiramos escolas que possuíam mais de 60 salas utilizadas, pois eram outliers claros.
- Retiramos escolas que possuíam mais de 50 desktops por alunos
- Retiramos escolas que possuem mais do que 50 computadores portáteis por aluno.

Após calcular o percentual de cada município e criar novas variáveis que levam os nomes das variáveis aqui declaradas anteriormente, porém com ou PERC_ ou SUM_ no prefixo do nome. Essas novas variáveis foram preenchidas da seguinte maneira.

Variável	Valor nulo substituído por
PERC_TP_DEPENDENCIA	0
PERC_TP_LOCALIZACAO	0
PERC_IN_AGUA_POTAVEL	100
PERC_IN_ENERGIA_INEXISTENTE	0
PERC_IN_ESGOTO_REDE_PUBLICA	média
PERC_IN_BIBLIOTECA_SALA_LEITURA	0

PERC_IN_QUADRA_ESPORTES	média
PERC_IN_SALA_ESTUDIO_DANCA	0
PERC_IN_BANDA_LARGA	média
PERC_TP_ATIVIDADE_COMPLEMENTAR	0
SUM_QT_SALAS_UTILIZADAS	média
SUM_QT_DESKTOP_ALUNO	média
SUM_QT_PORTATIL_ALUNO	média

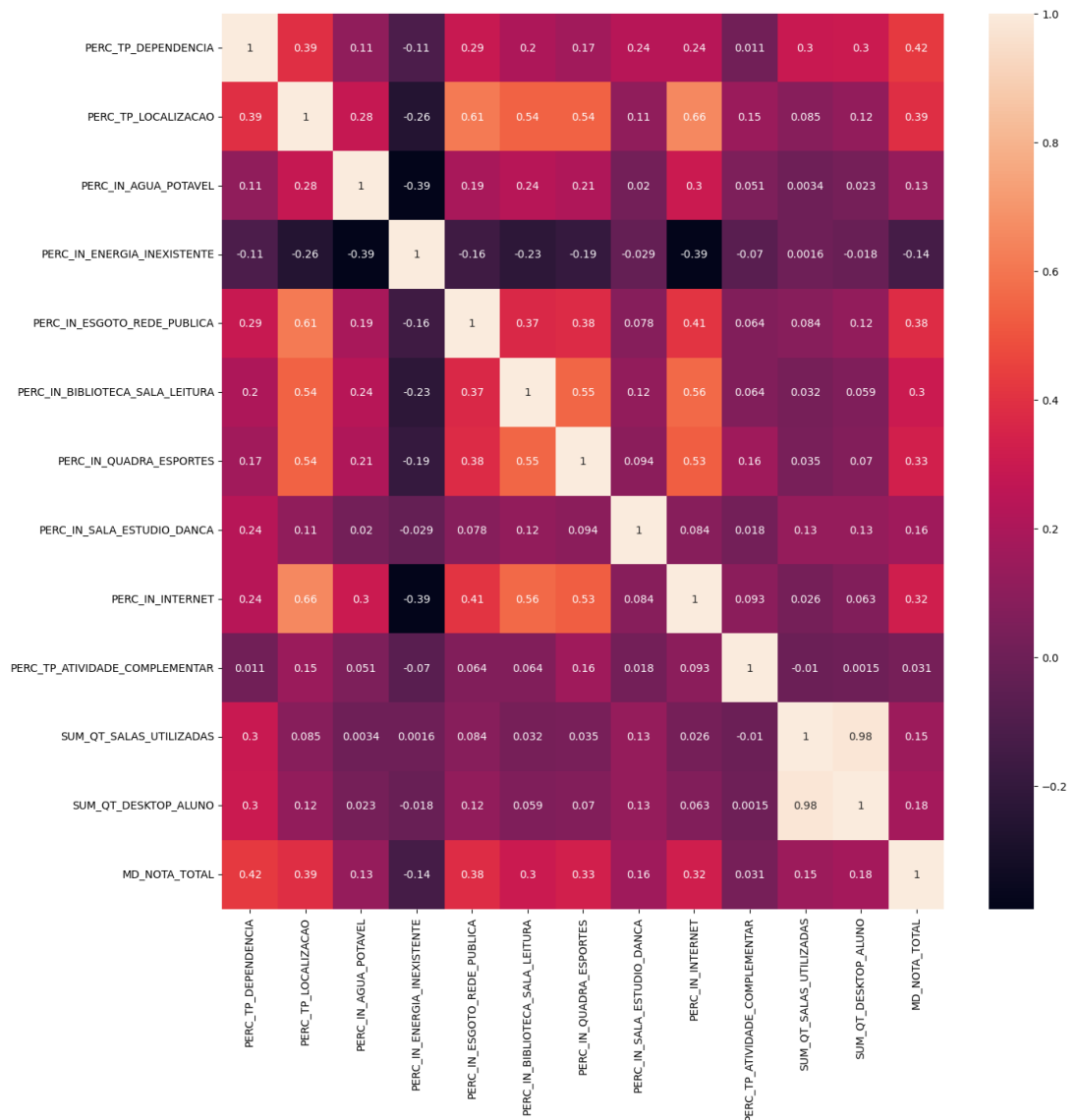
4. Análise e Exploração dos Dados

A análise de dados começou primeiro entendendo as bases em si, então foram construídos gráficos de distribuição. Inicialmente das variáveis sem agregação, e posteriormente da base agregada. Pois como os dados foram construídos para serem apresentados a nível de município, o foco foi nos indicadores a nível de município.

Foram analisados tanto a distribuição, quanto a correlação dos indicadores com a variável final, esse processo é importante para poder validar as variáveis que serão levadas para o modelo.

Todo o conjunto de variáveis que será apresentado nesse capítulo, serão as já escolhidas, a base de educação básica é muito extensa e apresenta mais de 100 variáveis. Foi escolhido um subconjunto das variáveis que apresentaram a maior correlação possível.

O gráfico abaixo mostra a correlação das variáveis. É possível ver em maior qualidade, no notebook chamado `modelo.ipynb`



Com base na correlação foram selecionadas as variáveis de interesse para a construção do modelo. Após a limpeza e do join entre os dois datasets, que podem ser observados nos scripts. Na seguinte sequência.

Ae_01.ipynb

Ae_enem.ipynb

Join_datasets.ipynb

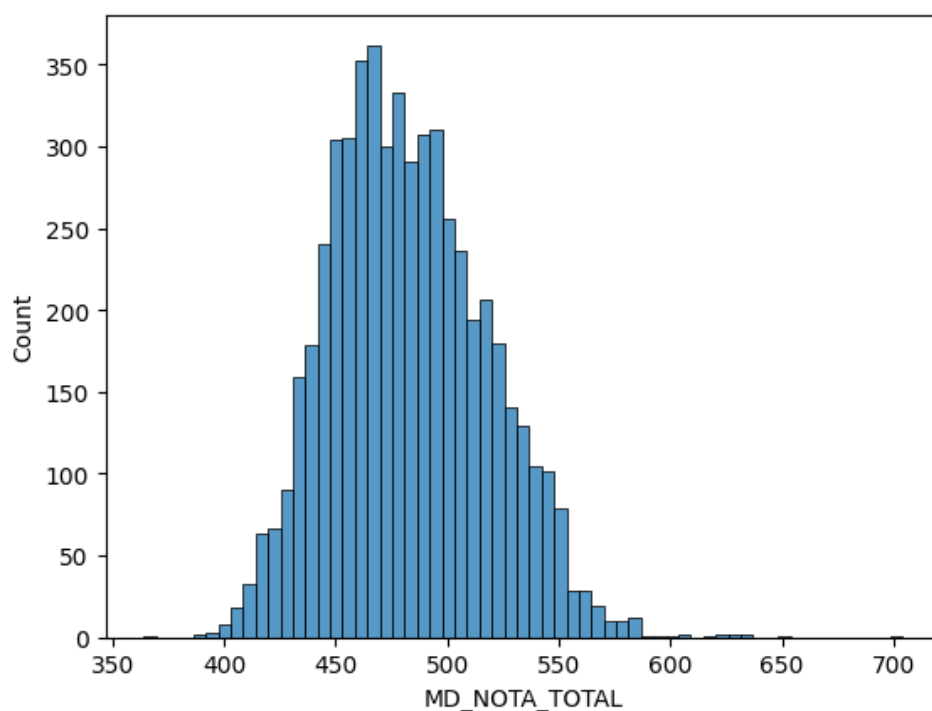
Modelo.ipynb

5. Criação de Modelos de Machine Learning

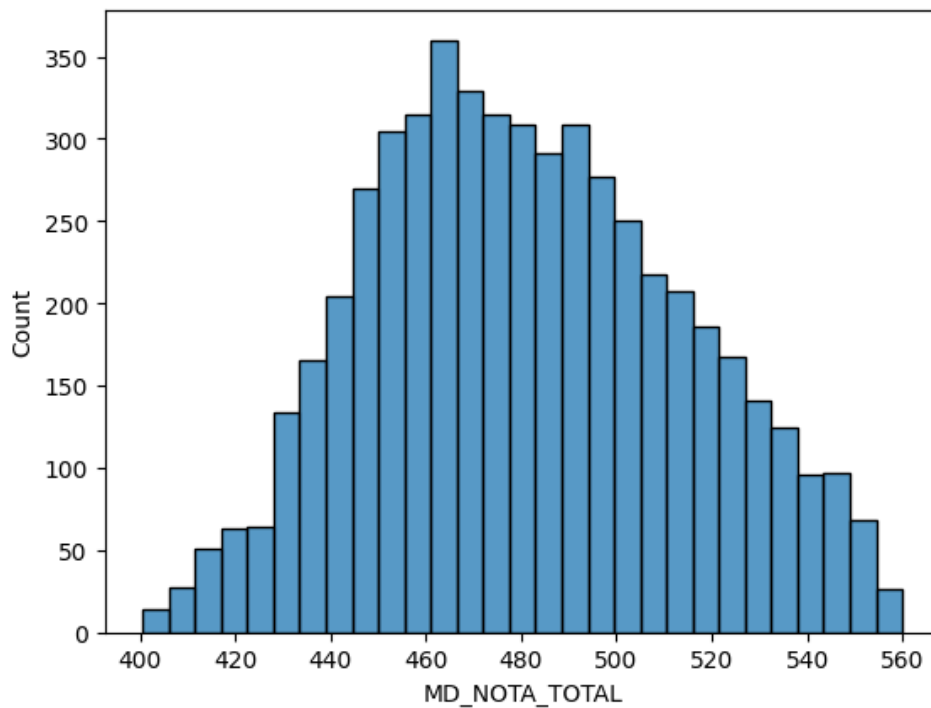
Como o objetivo é prever a nota do ENEM a partir das variáveis do município, é necessário o uso de um Regressor. O regressor consegue prever um valor através das variáveis fornecidas após ser treinado.

A variável `train_data`, será meu conjunto completo de dados, que será utilizado para teste e treino. Durante o treinamento dos modelos. Após todos os treinamentos, irei validar minha metodologia com o uso do `valid_data`.

Também foram retirados os valores de outlier, que eram notas maiores que 560 e menores que 400. Podemos ver o gráfico a seguir, antes da retirada dos outliers da variável `target`.



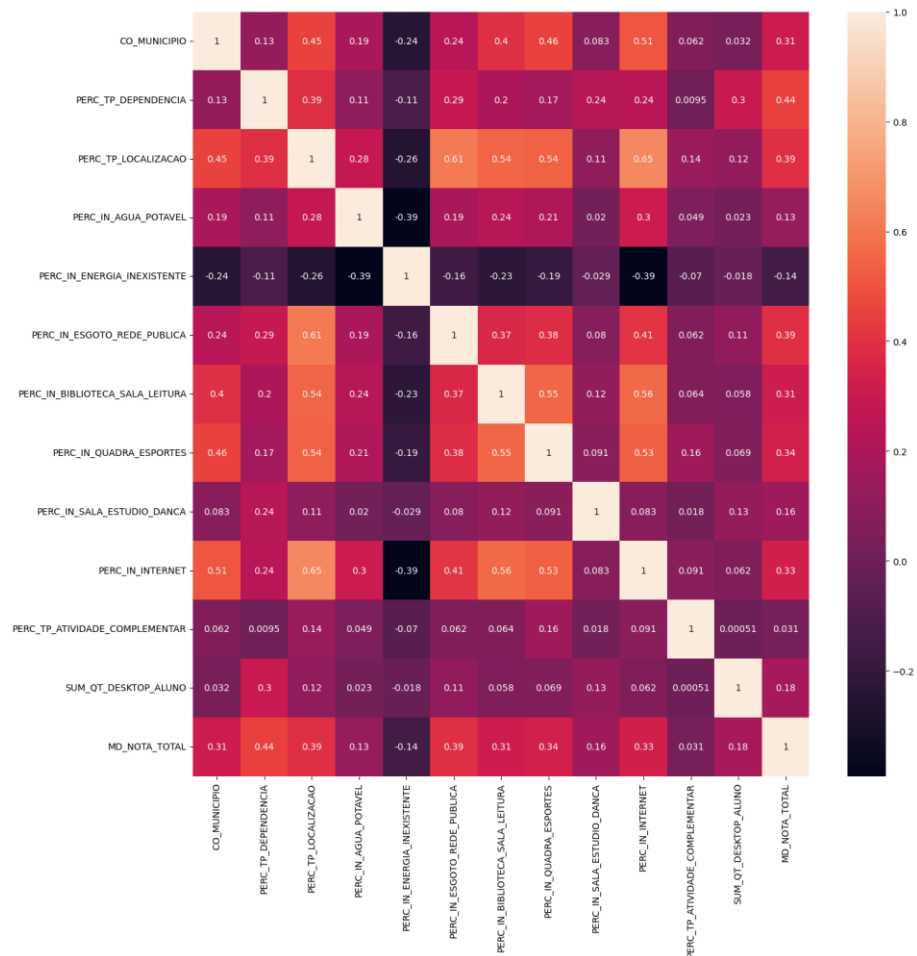
Após retirarmos os outliers, podemos ver que a distribuição fica mais próxima da forma normal, o que vai ajudar a prever com mais exatidão os dados. E ainda continuamos com uma grande quantidade de códigos, temos 5382 linhas ainda.



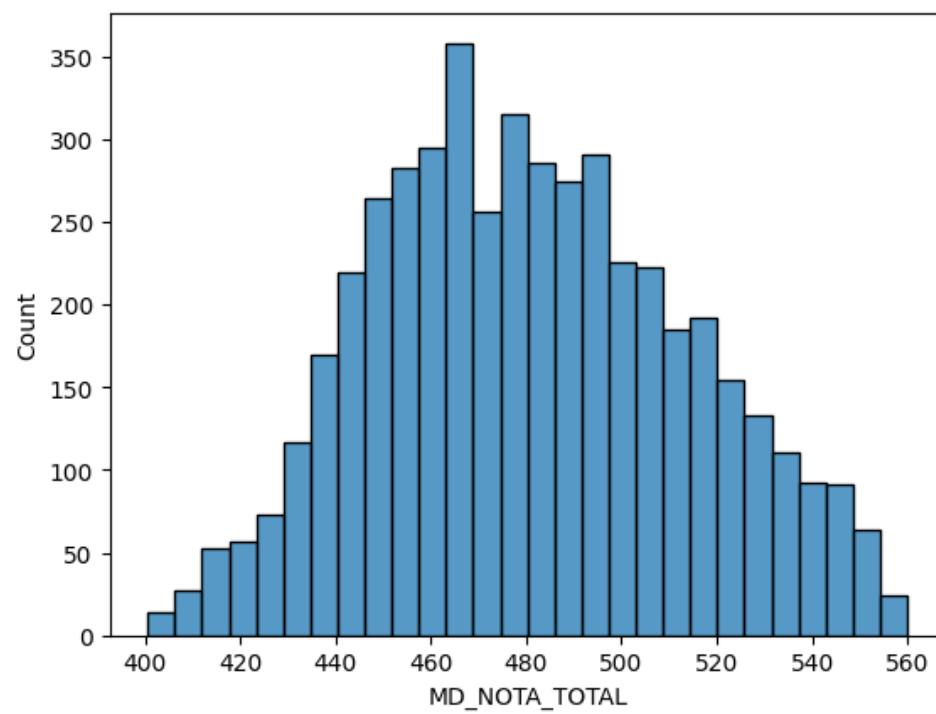
Existem diversos modelos de repressores, diversas metodologias. Porém, a estratégia inicial para validação de todos é a mesma. Vou dividir meu dataset em 3 partes. 70% dos dados serão utilizados para treino, 20% para teste e 10% para validação final. Esses 10% da validação final, são dados que o modelo nunca viu, nem no momento do cross validation, para garantir que meu modelo não teve nenhum overfitting. Os dados foram divididos utilizando o seguinte trecho abaixo de código.

```
# pegando 90% dos dados para treinar e 10% para validar no final.  
train_data = full_data.sample(frac=0.9, random_state=42)  
valid_data = full_data.loc[~full_data.index.isin(train_data.index)]  
|  
✓ 0.1s
```

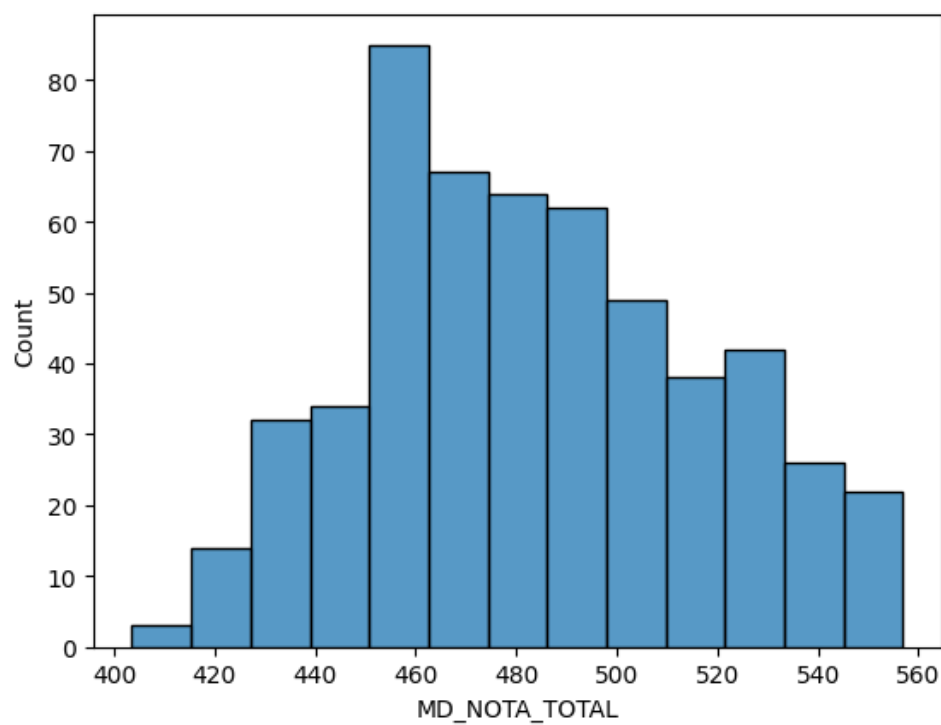
Também vão ser retiradas duas colunas do modelo, as colunas SUM_QT_DESKTOP_ALUNO e SUM_QT_COMP_PORTATIL_ALUNO, pois ambas têm uma correlação alta de mais de 90% com a variável SUM_QT_SALAS_UTILIZADAS. Como pode ser visto na imagem a seguir.



Uma coisa que é importante mostrar para garantir a confiabilidade do trabalho, é garantir que ambas as amostras, sigam uma distribuição próxima de normal, o que podemos ver com esses gráficos a seguir. O gráfico a seguir mostra a distribuição da variável `train_data`, ou seja, o dataset que será utilizado para treinar o modelo.



Já a próxima imagem, mostra o dataset que será utilizado para validar o modelo, um dataset que o modelo não conheceu em nenhum momento, são 10% de dados totalmente separados.



Podemos ver também pelos prints, que não foram perdidos dados, durante a divisão dos datasets.

```
print(f"len full data {len(full_data)}")
print(f"len train data {len(train_data)} ")
print(f"len train data {len(valid_data)} ")
✓ 0.2s
```

```
len full data 5382
len train data 4844
len train data 538
```

Para validar os modelos, iremos utilizar os métodos de treino aliados ao uso de validação cruzada, isso vai garantir que os modelos, não estejam simplesmente memorizando o dado, além de nos ajudar a buscar o melhor potencial de cada modelo.

```
• kf = KFold(n_splits= 5, shuffle=True, random_state=42, )
  cnt = 1
  for train_index, test_index in kf.split(x,y):
    print(f'Fold:{cnt}, Train set: {len(train_index)}, Test set:{len(test_index)}')
    cnt += 1
✓ 0.2s
```

```
Fold:1, Train set: 3943, Test set:986
Fold:2, Train set: 3943, Test set:986
Fold:3, Train set: 3943, Test set:986
Fold:4, Train set: 3943, Test set:986
Fold:5, Train set: 3944, Test set:985
```

O primeiro modelo testado foi a regressão linear.

```
score = cross_val_score(linear_model.LinearRegression(), x, y, cv= kf, scoring="neg_mean_squared_error")
print(f'Scores for each fold: {score}')
rmse(score.mean())
```

0.8s

Na média ela teve RMSE de 27,43.

O segundo modelo a ser testado foi o random Forest, que performou um pouco melhor com atingindo um RMSE de 26,68.

```
score = cross_val_score(ensemble.RandomForestRegressor(random_state= 42), x, y, cv= kf, scoring="neg_mean_squared_error", )
print(f'Scores for each fold are: {score}')
rmse(score.mean())
```

✓ 5.1s Python

Scores for each fold are: [-693.02993937 -737.54645682 -732.89359804 -717.46626784 -678.82046678]

rmse= 26.68

Apesar dos dois modelos estarem próximos, também testei o modelo do xgboost. No teste inicial, ele foi o que se mostrou com o melhor desempenho. E então resolvi gastar mais esforço otimizando o modelo. Para otimizar o modelo do xgboost, eu defino uma classe de valores que o modelo pode assumir, e ele vai testar aleatoriamente, toda essa combinação parâmetros para o modelo.

```
params = { 'max_depth': [3, 5, 6, 10, 15, 20],
           'learning_rate': [0.01, 0.1, 0.2, 0.3],
           'subsample': np.arange(0.5, 1.0, 0.1),
           'colsample_bytree': np.arange(0.4, 1.0, 0.1),
           'colsample_bylevel': np.arange(0.4, 1.0, 0.1),
           'n_estimators': [100, 500, 1000]}

xgbr = xgb.XGBRegressor(seed = 20)

clf = RandomizedSearchCV(estimator=xgbr,
                        param_distributions=params,
                        scoring='neg_mean_squared_error',
                        n_iter=25,
                        verbose=1)

clf.fit(x,y)
```

✓ 0.1s

✓ 0.1s

✓ 0.1s

✓ 3m 8.1s

Fitting 5 folds for each of 25 candidates, totalling 125 fits

Após isso, foi definido o melhor modelo, com base nos parâmetros encontrados.

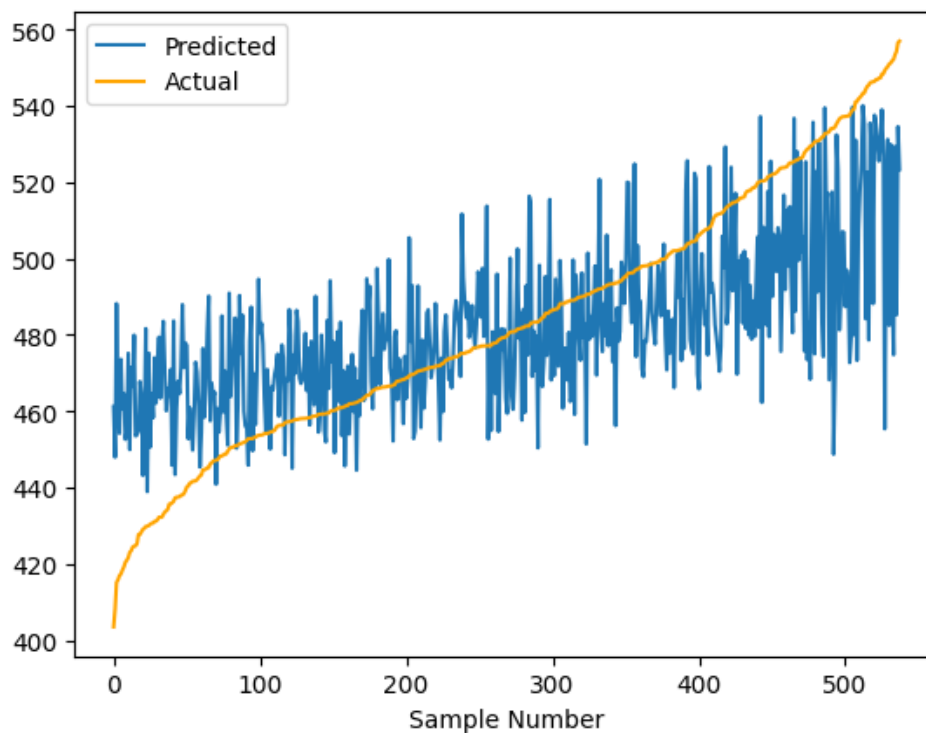
Conseguimos visualizar os parâmetros que foram encontrados.

```
{'subsample': 0.7999999999999999,
 'n_estimators': 1000,
 'max_depth': 3,
 'learning_rate': 0.01,
 'colsample_bytree': 0.4,
 'colsample_bylevel': 0.8999999999999999}
```

E esse modelo, com esses parâmetros, conseguiu ter a média de 25,40 pontos de erro, comparando com o dado totalmente novo. Ou seja, como se fosse um modelo em produção.

```
mse**(1/2)
✓ 0.2s
25.40104385674488
```

Com o gráfico a seguir, conseguimos ver o resultado do modelo, ele consegue prever bem a faixa do meio das pontuações, com um erro de 25 pontos em média. Porém os outliers ele não consegue prever muito bem. Poderia ter retirado os outliers, mas acho importante para mostrar as limitações do modelo.

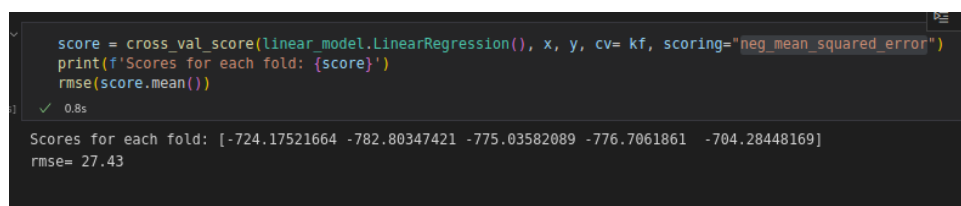


6. Interpretação dos Resultados / Apresentação dos Resultados

Os resultados mostram que é possível avaliar com uma precisão razoável o desempenho do município, baseado na sua educação básica. As variáveis utilizadas para a análise, todas tem a ver com características físicas do colégio. O que é possível de ser alterado com políticas públicas.

Foram três modelos analisados:

1. O modelo de regressão linear, esse teve um erro médio quadrático de 27,43, isso significa que a cada previsão feita, ele erra em média 27,43 pontos. Com base nas variáveis que foram providas.



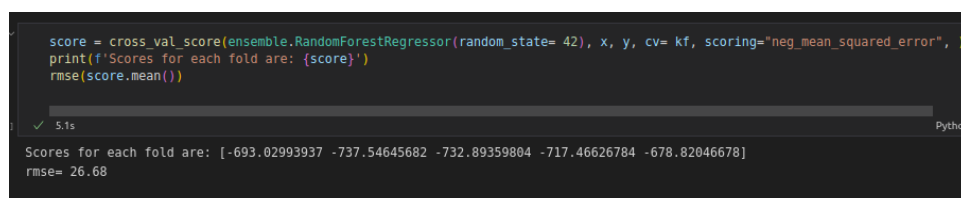
```
score = cross_val_score(linear_model.LinearRegression(), x, y, cv= kf, scoring="neg_mean_squared_error")
print(f'Scores for each fold: {score}')
rmse(score.mean())
```

✓ 0.8s

Scores for each fold: [-724.17521664 -782.80347421 -775.03582089 -776.7061861 -704.28448169]

rmse= 27.43

2. O segundo modelo escolhido, foi o modelo de florestas aleatórias, esse modelo foi um pouco melhor com um erro médio quadrático de 26,68 pontos.



```
score = cross_val_score(ensemble.RandomForestRegressor(random_state= 42), x, y, cv= kf, scoring="neg_mean_squared_error", )
print(f'Scores for each fold are: {score}')
rmse(score.mean())
```

✓ 5.1s

Scores for each fold are: [-693.02993937 -737.54645682 -732.89359804 -717.46626784 -678.82046678]

rmse= 26.68

3. O terceiro modelo escolhido foi o XGBOOST. Ele foi o modelo que sem nenhuma otimização apresentou o melhor resultado, mesmo com poucos pontos, ele havia gerado 26,3 de erro médio quadrático. É uma diferença muito pequena, porém como foi o que de início apresentou o melhor resultado, ele foi o que gastei mais tempo fazendo otimizações. E no final foi possível alcançar um erro de 25,40 pontos em média.

```

best_xgb = clf.best_estimator_
149] ✓ 0.1s

x_valid, y_valid = (valid_data.iloc[:, :-1], valid_data.iloc[:, -1])
150] ✓ 0.1s

y_predicted = best_xgb.predict(x_valid)
154] ✓ 0.9s

mse = mean_squared_error(y_valid, y_predicted)
155] ✓ 0.1s

mse*(1/2)
157] ✓ 0.2s
... 25.40104385674488

>
158] ✓ 0.2s
... {'subsample': 0.7999999999999999,
'n_estimators': 1000,
'max_depth': 3,
'learning_rate': 0.01,
'colsample_bytree': 0.4,
'colsample_bylevel': 0.8999999999999999}

```

Todo o treinamento está presente no arquivo modelo.ipynb.

Os resultados nos trazem alguns pontos importantes. A prova do enem possui 1000 pontos, percebemos que a média dos municípios anda em torno de 500 pontos. Ou seja apenas 50% das provas, algo que ainda não é muito satisfatório. Porém com a aplicação de políticas públicas é possível alterar essa realidade. Já que todas as variáveis que foram utilizadas nesse trabalho têm relação com elementos de infraestrutura, o que é possível de ser alterado com o remanejamento de orçamento.

Já que utilizamos o método do XGBOOST, é possível entender quais variáveis foram mais importantes. O seguinte trecho de código consegue fazer o modelo nos contar a importância de cada Feature.

```

features = [[key, value] for key, value in best_xgb.get_booster().get_score(importance_type="gain").items() ]
features = pd.DataFrame(features, columns=["Variavel", "importancia"])
features = features.sort_values(["importancia"], ascending=False)
features

```

	Variavel	importancia
0	PERC_TP_DEPENDENCIA	22751.445312
4	PERC_IN_ESGOTO_REDE_PUBLICA	20918.789062
10	SUM_QT_SALAS_UTILIZADAS	15493.189453
6	PERC_IN_QUADRA_ESPORTES	10644.326172
1	PERC_TP_LOCALIZACAO	9927.051758
5	PERC_IN_BIBLIOTECA_SALA_LEITURA	8597.077148
7	PERC_IN_SALA_ESTUDIO_DANCA	7839.082031
8	PERC_IN_INTERNET	5264.391113
9	PERC_TP_ATIVIDADE_COMPLEMENTAR	4903.759277
2	PERC_IN_AGUA_POTAVEL	2356.984131
3	PERC_IN_ENERGIA_INEXISTENTE	2182.372070

Observando as Features, em ordem decrescente de importância, observamos que a mais importante é a variável PERC_TP_DEPENDENCIA, essa variável, fala qual é a porcentagem de colégios naquele município que é particular. Podemos extrapolar a explicação e concluir que o ensino público, infelizmente ainda está muito longe do ensino privado, ao menos no nível de educação básica. Fato que não existe no ensino superior, onde é consenso que o ensino superior público é mais estruturado.

A próxima variável na lista de importância é o PERC_IN_ESGOTO_REDE_PUBLICA, variável que conta a realidade de cidades mais pobres, ou que estão em locais mais rurais. O ensino de qualidade, necessita que os alunos tenham acesso ao mínimo como um saneamento básico de qualidade.

As próximas variáveis que devo destacar são o percentual de bibliotecas ou salas de leitura e o percentual de estúdios de dança nos colégios, essas variáveis entram em uma segunda função do colégio, que além do ensino, é importante dar cultura para os alunos. Muitas vezes a cultura dentro do colégio é visto como segunda prioridade, mas podemos ver que ela é uma variável importante, para a média do ensino ser bem qualificada naquele município. E elas tem mais importância do que a presença de internet, cultura e um ambiente agradável são capazes de manter os alunos matriculados por mais tempo e dispostos em ter um ensino de qualidade.

8. Links

Aqui você deve disponibilizar os links para o vídeo com sua apresentação de 5 minutos e para o repositório contendo os dados utilizados no projeto, scripts criados, etc.

Link para o vídeo: https://www.youtube.com/watch?v=OUw_iWEwcmY

Link para o repositório: https://github.com/wagneralbjr/tcc-puc_minas

REFERÊNCIAS

APÊNDICE

Programação/Scripts

São os seguintes scripts:

ae_educacao_basica.ipynb

criacao_modelo.ipynb

datasets.py

join_datasets.ipynb

tratando_ed_basica.ipynb

tratando_enem.ipynb

utils.py

ae_educacao_basica.py

```
import pandas as pd
```

```
from utils import start_notebook
```

```
from datasets import get_ed_basica, treatment_ed_basica, get_unique_municipios
```

```
import seaborn as sns
```

```
start_notebook()
```

```
ed_basica = get_ed_basica()
```

```
ed_basica["CO_MUNICIPIO"].drop_duplicates().count()
```

```
tp_dependencia_to_plot = ed_basica["TP_DEPENDENCIA"].map({
```

```
    1 : "Federal",
```

```
    2 : "Estadual",
```

```
    3 : "Municipal",
```

```
    4 : "Privada"
```

```
})
```

```
tp_dependencia_to_plot = pd.DataFrame(tp_dependencia_to_plot)
```

```
tp_dependencia_to_plot.value_counts()
```

```
sns.countplot(x='TP_DEPENDENCIA', data=tp_dependencia_to_plot, )
```

```
ed_basica[["CO_MUNICIPIO", "TP_DEPENDENCIA"]].info() print(ed_basica[["CO_MUNICIPIO", "TP_LOCALIZACAO"]].info())
```

```

tp_localizacao_to_plot = ed_basica["TP_LOCALIZACAO"].map({
    1 : "Urbana",
    2 : "Rural"
})
tp_localizacao_to_plot = pd.DataFrame(tp_localizacao_to_plot)
sns.countplot(x="TP_LOCALIZACAO", data = tp_localizacao_to_plot) print(ed_ba-
sica[["CO_MUNICIPIO","IN_AGUA_POTAVEL"]].info())
IN_AGUA_POTAVEL_to_plot = ed_basica["IN_AGUA_POTAVEL"].map({
    1 : "SIM",
    0 : "NAO"
})
IN_AGUA_POTAVEL_to_plot = pd.DataFrame(IN_AGUA_POTAVEL_to_plot)
sns.countplot(x="IN_AGUA_POTAVEL", data = IN_AGUA_POTAVEL_to_plot)
print(ed_basica[["CO_MUNICIPIO","IN_ENERGIA_INEXISTENTE"]].info())
IN_ENERGIA_INEXISTENTE_to_plot = ed_basica["IN_ENERGIA_INEXISTENTE"].map({
    1 : "SIM",
    0 : "NAO"
})
IN_ENERGIA_INEXISTENTE_to_plot = pd.DataFrame(IN_ENERGIA_INEXISTENTE_to_plot)
sns.countplot(x="IN_ENERGIA_INEXISTENTE", data = IN_ENERGIA_INEXISTENTE_to_plot)
print(ed_basica[["CO_MUNICIPIO","IN_ESGOTO_REDE_PUBLICA"]].info())
IN_ESGOTO_REDE_PUBLICA_to_plot = ed_basica["IN_ESGOTO_REDE_PUBLICA"].map({
    1 : "SIM",
    0 : "NAO"
})
IN_ESGOTO_REDE_PUBLICA_to_plot = pd.DataFrame(IN_ESGOTO_REDE_PUBLICA_to_plot)
sns.countplot(x="IN_ESGOTO_REDE_PUBLICA", data = IN_ESGOTO_REDE_PUBLICA_to_plot)
print(ed_basica[["CO_MUNICIPIO","IN_BIBLIOTECA_SALA_LEITURA"]].info())
IN_BIBLIOTECA_SALA_LEITURA_to_plot = ed_basica["IN_BIBLIOTECA_SALA_LEITURA"].map({
    1 : "SIM",
    0 : "NAO"
})

```

```
IN_BIBLIOTECA_SALA_LEITURA_to_plot = pd.DataFrame(IN_BIBLIOTECA_SALA_LEI-
TURA_to_plot)
```

```
sns.countplot(x="IN_BIBLIOTECA_SALA_LEITURA", data = IN_BIBLIOTECA_SALA_LEI-
TURA_to_plot)
```

```
print(ed_basica[["CO_MUNICIPIO","IN_QUADRA_ESPORTES"]].info())
```

```
IN_QUADRA_ESPORTES_to_plot = ed_basica["IN_QUADRA_ESPORTES"].map({
    1 : "SIM",
    0 : "NAO"
})
```

```
IN_QUADRA_ESPORTES_to_plot = pd.DataFrame(IN_QUADRA_ESPORTES_to_plot)
```

```
sns.countplot(x="IN_QUADRA_ESPORTES", data = IN_QUADRA_ESPORTES_to_plot)
```

```
print(ed_basica[["CO_MUNICIPIO","IN_SALA_ESTUDIO_DANCA"]].info())
```

```
IN_SALA_ESTUDIO_DANCA_to_plot = ed_basica["IN_SALA_ESTUDIO_DANCA"].map({
    1 : "SIM",
    0 : "NAO"
})
```

```
IN_SALA_ESTUDIO_DANCA_to_plot = pd.DataFrame(IN_SALA_ESTUDIO_DANCA_to_plot)
```

```
sns.countplot(x="IN_SALA_ESTUDIO_DANCA", data = IN_SALA_ESTUDIO_DANCA_to_plot)
```

```
ed_basica.value_counts()
```

```
sns.histplot(x="QT_SALAS_UTILIZADAS",\
    data = ed_basica.query("QT_SALAS_UTILIZADAS < 60"),
    bins=20)
```

```
#QT_DESKTOP_ALUNO
```

```
sns.histplot(x="QT_DESKTOP_ALUNO",\
    data = ed_basica.query("QT_DESKTOP_ALUNO < 50"),
    bins=20)
```

```
#QT_DESKTOP_ALUNO
```

```
sns.histplot(x="QT_COMP_PORTATIL_ALUNO",\
    data = ed_basica.query("QT_COMP_PORTATIL_ALUNO < 50"),
```

```

        bins=10)
#IN_BANDA_LARGA.

print(ed_basica[["CO_MUNICIPIO","IN_BANDA_LARGA"]].info())
IN_BANDA_LARGA_to_plot = ed_basica["IN_BANDA_LARGA"].map({
    1 : "SIM",
    0 : "NAO"
})
IN_BANDA_LARGA_to_plot = pd.DataFrame(IN_BANDA_LARGA_to_plot)

sns.countplot(x="IN_BANDA_LARGA", data = IN_BANDA_LARGA_to_plot)

```

Criando_modelo.ipynb

```

from utils import start_notebook
import pandas as pd
import seaborn as sns
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score
from sklearn import linear_model, tree, ensemble
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from xgboost.sklearn import XGBClassifier,XGBRegressor
from sklearn.metrics import mean_squared_error
import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV
from pprint import pprint

full_data          =          pd.read_parquet("datasets/tratados/base_completa.parquet")
sns.histplot(full_data,                                x="MD_NOTA_TOTAL")

# removendo outliers
full_data = full_data.query(" MD_NOTA_TOTAL > 400 & MD_NOTA_TOTAL < 560")

#plotando o histograma após retirada e outliers

```

```

sns.histplot(full_data, x="MD_NOTA_TOTAL")
full_data.columns
full_data = full_data.drop(["CO_MUNICIPIO"], axis =1)
sns.histplot(full_data, x="MD_NOTA_TOTAL")
fig, ax = plt.subplots(figsize=(15,15)) # Sample figsize in inches
sns.heatmap(full_data.corr(), annot=True, ax=ax)
full_data = full_data.drop(["SUM_QT_DESKTOP_ALUNO", "SUM_QT_COMP_PORaTA-
TIL_ALUNO"], axis = 1)
# pegando 90% dos dados para treinar e 10% para validar no final.
train_data = full_data.sample(frac=0.9, random_state=42)
valid_data = full_data.loc[~full_data.index.isin(train_data.index)]
train_data.columns
sns.histplot(data=train_data, x = "MD_NOTA_TOTAL")
sns.histplot(data=valid_data, x = "MD_NOTA_TOTAL")
print(f"len full data {len(full_data)}")
print(f"len train data {len(train_data)} ")
print(f"len train data {len(valid_data)} ")
y = train_data["MD_NOTA_TOTAL"]
train_data = train_data.drop(["MD_NOTA_TOTAL"], axis = 1)
x = train_data.copy()
kf = KFold(n_splits= 5, shuffle=True, random_state=42, )
cnt = 1
for train_index, test_index in kf.split(x,y):
    print(f'Fold:{cnt}, Train set: {len(train_index)}, Test set:{len(test_index)}')
    cnt += 1
def rmse(score):
    rmse = np.sqrt(-score)
    print(f'rmse= {rmse:.2f}'.format(rmse))
score = cross_val_score(linear_model.LinearRegression(), x, y, cv= kf,
scoring="neg_mean_squared_error")
print(f'Scores for each fold: {score}')

```

```

rmse(score.mean())
score = cross_val_score(ensemble.RandomForestRegressor(random_state= 42), x, y, cv= kf,
scoring="neg_mean_squared_error", )
print(f'Scores for each fold are: {score}')
rmse(score.mean())
params = { 'max_depth': [3, 5, 6, 10, 15, 20],
          'learning_rate': [0.01, 0.1, 0.2, 0.3],
          'subsample': np.arange(0.5, 1.0, 0.1),
          'colsample_bytree': np.arange(0.4, 1.0, 0.1),
          'colsample_bylevel': np.arange(0.4, 1.0, 0.1),
          'n_estimators': [100, 500, 1000]}
xgbr = xgb.XGBRegressor(seed = 20)
clf = RandomizedSearchCV(estimator=xgbr,
                        param_distributions=params,
                        scoring='neg_mean_squared_error',
                        n_iter=100,
                        verbose=1,
                        n_jobs=-1)
clf.fit(x,y)
best_xgb = clf.best_estimator_
x_valid, y_valid = (valid_data.iloc[:, :-1], valid_data.iloc[:, -1])
y_predicted = best_xgb.predict(x_valid)
mse = mean_squared_error(y_valid, y_predicted)
mse**(1/2)
clf.best_params_
target = pd.Series(y_valid).reset_index(drop=True).sort_values()
predictions = pd.Series(y_predicted)
predictions = predictions.iloc[target.index].reset_index(drop=True)
target = target.reset_index(drop=True)
plt.plot(predictions, label='Predicted')
plt.plot(target, label='Actual', color='orange')
plt.legend(['Predicted', 'Actual'])

```

```

plt.xlabel('Sample Number')
plt.show()
features = [[key, value] for key, value in
best_xgb.get_booster().get_score(importance_type="gain").items() ]
features = pd.DataFrame(features, columns=["Variavel", "importancia"])
features = features.sort_values(["importancia"], ascending=False)
features

```

Datasets.py

```
import pandas as pd
```

```
def get_ed_basica() -> "pd.DataFrame":
```

```
    df = pd.read_csv(
```

```
        "datasets/microdados_censo_escolar_2021/2021/dados/microdados_ed_basica_2021.csv",
        encoding="iso-8859-1",
        sep=";",
    )
```

```
    return df
```

```
def treatment_ed_basica(df):
```

```
    interest_columns = [
        "CO_MUNICIPIO",
        "TP_DEPENDENCIA",
        "TP_LOCALIZACAO",
        "IN_AGUA_POTAVEL",
        "IN_ENERGIA_INEXISTENTE",
    ]

```



```

"IN_ESGOTO_REDE_PUBLICA",
"IN_BIBLIOTECA_SALA_LEITURA",
"IN_QUADRA_ESPORTES",
"IN_SALA_ESTUDIO_DANCA",
"QT_SALAS_UTILIZADAS",
"QT_DESKTOP_ALUNO",
"QT_COMP_PORTATIL_ALUNO",
"IN_INTERNET",
"TP_ATIVIDADE_COMPLEMENTAR", # prestar atenção nesse caso, pois é 0,1,2
]

# dependencia administrativa
df = df.loc[:, interest_columns]
df["TP_DEPENDENCIA"] = df["TP_DEPENDENCIA"].map(
    {1: "publico", 2: "publico", 3: "publico", 4: "privado"}
)

# retirando escolas com mais de 60 salas, outliers
df = df[df["QT_SALAS_UTILIZADAS"] <= 60]

# retirando escolas com mais de 50 desktops por aluno
df = df[df["QT_DESKTOP_ALUNO"] <= 50]

# retirando escola com mais de 50 computadores portáteis por aluno
df = df[df["QT_COMP_PORTATIL_ALUNO"] <= 50]

# localizacao
df["TP_LOCALIZACAO"] = df["TP_LOCALIZACAO"].map({1: "urbana", 2: "rural"})

# no caso dessa variável, considere como ter a atividade, já como sim
# 0 - Não oferece
# 1 - Não exclusivamente
# 2 - Exclusivamente

```

```

df["TP_ATIVIDADE_COMPLEMENTAR"] = df["TP_ATIVIDADE_COMPLEMENTAR"].map(
    {0: "nao", 1: "sim", 2: "sim"}
)
df["TP_ATIVIDADE_COMPLEMENTAR"] =
df["TP_ATIVIDADE_COMPLEMENTAR"].fillna("nao")

return df

```

```

def get_unique_municipios(ed_basica):

```

```

    df = ed_basica[["CO_MUNICIPIO"]].drop_duplicates()
    df = df.reset_index(drop=True)
    return df

```

Join_datasets.py

```

from utils import start_notebook
import pandas as pd
ed_basica = pd.read_parquet("datasets/tratados/ed_basica_enriquecida.parquet")
enem = pd.read_parquet("datasets/tratados/enem.parquet")
data = ed_basica.merge(enem, how='inner', on="CO_MUNICIPIO")
data.to_parquet("datasets/tratados/base_completa.parquet")

```

tratando_ed_basica.py

```

import pandas as pd
from datasets import get_ed_basica, treatment_ed_basica, get_unique_municipios
from utils import start_notebook, calculate_percentage_of_group, apply_func_perc,
calculate_sum_of_group, apply_func_sum

import numpy as np
import numbers

```

```

start_notebook()

ed_basica = get_ed_basica()

ed_basica = treatment_ed_basica(ed_basica)

final_df = get_unique_municipios(ed_basica)

final_df = apply_func_perc( final_df, ed_basica, calculate_percentage_of_group,
"CO_MUNICIPIO", "TP_DEPENDENCIA", "privado", "PERC_TP_DEPENDENCIA")

final_df = apply_func_perc( final_df, ed_basica, calculate_percentage_of_group,
"CO_MUNICIPIO", "TP_LOCALIZACAO", "urbana", "PERC_TP_LOCALIZACAO")

final_df = apply_func_perc( final_df, ed_basica, calculate_percentage_of_group,
"CO_MUNICIPIO", "IN_AGUA_POTAVEL", 1, "PERC_IN_AGUA_POTAVEL")

final_df = apply_func_perc( final_df, ed_basica, calculate_percentage_of_group,
"CO_MUNICIPIO", "IN_ENERGIA_INEXISTENTE", 1, "PERC_IN_ENERGIA_INEXISTENTE")

final_df = apply_func_perc( final_df, ed_basica, calculate_percentage_of_group,
"CO_MUNICIPIO", "IN_ESGOTO_REDE_PUBLICA", 1, "PERC_IN_ESGOTO_REDE_PUBLICA")

final_df = apply_func_perc( final_df, ed_basica, calculate_percentage_of_group,
"CO_MUNICIPIO", "IN_BIBLIOTECA_SALA_LEITURA", 1,
"PERC_IN_BIBLIOTECA_SALA_LEITURA")

final_df = apply_func_perc( final_df, ed_basica, calculate_percentage_of_group,
"CO_MUNICIPIO", "IN_QUADRA_ESPORTES", 1, "PERC_IN_QUADRA_ESPORTES")

final_df = apply_func_perc( final_df, ed_basica, calculate_percentage_of_group,
"CO_MUNICIPIO", "IN_SALA_ESTUDIO_DANCA", 1, "PERC_IN_SALA_ESTUDIO_DANCA")

final_df = apply_func_perc( final_df, ed_basica, calculate_percentage_of_group,
"CO_MUNICIPIO", "IN_INTERNET", 1, "PERC_IN_INTERNET")

final_df = apply_func_perc( final_df, ed_basica, calculate_percentage_of_group,
"CO_MUNICIPIO", "TP_ATIVIDADE_COMPLEMENTAR", "sim",
"PERC_TP_ATIVIDADE_COMPLEMENTAR")

final_df = apply_func_sum(final_df, ed_basica, calculate_sum_of_group, "CO_MUNICIPIO",
"QT_SALAS_UTILIZADAS", "SUM_QT_SALAS_UTILIZADAS")

final_df = apply_func_sum(final_df, ed_basica, calculate_sum_of_group, "CO_MUNICIPIO",
"QT_DESKTOP_ALUNO", "SUM_QT_DESKTOP_ALUNO")

final_df = apply_func_sum(final_df, ed_basica, calculate_sum_of_group, "CO_MUNICIPIO",
"QT_COMP_PORTATIL_ALUNO", "SUM_QT_COMP_PORTATIL_ALUNO")

```

```

nan_values = {
    "PERC_TP_DEPENDENCIA" : 0,
    "PERC_TP_LOCALIZACAO" : 0,
    "PERC_IN_AGUA_POTAVEL" : 100,
    "PERC_IN_ENERGIA_INEXISTENTE" : 0,
    "PERC_IN_ESGOTO_REDE_PUBLICA" : "mean",
    "PERC_IN_BIBLIOTECA_SALA_LEITURA" : 0,
    "PERC_IN_QUADRA_ESPORTES" : "mean",
    "PERC_IN_SALA_ESTUDIO_DANCA" : 0,
    "PERC_IN_INTERNET" : "mean",
    "PERC_TP_ATIVIDADE_COMPLEMENTAR" : 0,
    "SUM_QT_SALAS_UTILIZADAS" : "mean",
    "SUM_QT_DESKTOP_ALUNO" : "mean",
    "SUM_QT_COMP_PORTATIL_ALUNO" : "mean",
}

df = final_df
for key, value in nan_values.items():
    if isinstance(value, numbers.Number):
        df[key] = df[key].fillna(value)
    if value == "mean":
        df[key] = df[key].fillna(df[key].mean())
final_df.to_parquet("datasets/tratados/ed_basica_enriquecida.parquet", )

```

Tratando_enem.py

```

import pandas as pd
from utils import start_notebook
import seaborn as sns

start_notebook()

df_enem = pd.read_csv("datasets/microdados_enem_2021/DADOS/MICRODADOS_ENEM_2021.csv",

```

```

sep=";", encoding="iso-8859-1")
df_enem = df_enem.query(
    """
    TP_PRESENCA_CN == 1 & TP_PRESENCA_CH == 1 & TP_PRESENCA_LC == 1 &
    TP_PRESENCA_MT == 1

    """
)
sub = df_enem[["CO_MUNICIPIO_ESC", 'NU_NOTA_CN', 'NU_NOTA_CH', 'NU_NOTA_LC',
'NU_NOTA_MT']]
df_means_municipio = sub.groupby("CO_MUNICIPIO_ESC", as_index=False).agg(
    MD_NU_NOTA_CN = pd.NamedAgg(column= "NU_NOTA_CN", aggfunc="mean"),
    MD_NU_NOTA_CH = pd.NamedAgg(column= "NU_NOTA_CH", aggfunc="mean"),
    MD_NU_NOTA_LC = pd.NamedAgg(column= "NU_NOTA_LC", aggfunc="mean"),
    MD_NU_NOTA_MT = pd.NamedAgg(column= "NU_NOTA_MT", aggfunc="mean"),

)
df_means_municipio.sort_values("MD_NU_NOTA_MT")
df_means_municipio["MD_NOTA_TOTAL"] = (
    df_means_municipio["MD_NU_NOTA_CN"]
    + df_means_municipio["MD_NU_NOTA_CH"]
    + df_means_municipio["MD_NU_NOTA_LC"]
    + df_means_municipio["MD_NU_NOTA_MT"] ) / 4
sns.histplot(df_means_municipio, x = "MD_NOTA_TOTAL")

df_means_municipio = df_means_municipio.drop(["MD_NU_NOTA_CN",
"MD_NU_NOTA_CH", "MD_NU_NOTA_LC", "MD_NU_NOTA_MT"], axis=1)

df_means_municipio["CO_MUNICIPIO_ESC"] =
df_means_municipio["CO_MUNICIPIO_ESC"].apply( lambda x : int(x))

```

```
df_means_municipio = df_means_municipio.rename(columns={"CO_MUNICIPIO_ESC" :
"CO_MUNICIPIO"})
df_means_municipio.to_parquet("datasets/tratados/enem.parquet")
```

utils.py

```
import pandas as pd
import numpy as np
```

```
def start_notebook():
    pd.set_option("display.max_columns", None)
    pd.set_option("display.float_format", lambda x: "%.3f" % x)

    return

def calculate_percentage_of_group(
    df, id_column, indicator_column, group_interest, new_column_name
):
    df = df[[id_column, indicator_column]]
    cols_to_group = [id_column, indicator_column]
    df_grouped = df.groupby(cols_to_group, as_index=False).size()
    df_grouped[new_column_name] = (
        100
        * df_grouped["size"]
        / df_grouped.groupby([id_column])["size"].transform("sum")
    )
    df_grouped = df_grouped[df_grouped[indicator_column] == group_interest]
    return df_grouped

def apply_func_perc(
```

```

final_df,
origin_df,
f_to_apply,
unique_key,
indicator_column,
group_interest,
new_column_name,
):

```

```

"""

```

Parameters:

final_df : the dataframe that joins all the final data,
 origin_df : dataframe with all columns, the origin
 f_to_apply : the function that is applied
 unique_key : the join key
 indicator_column : the column with the indicator
 group_interest : the group indicator that i want to filter
 new_column_name : the new name of the column on the final_df

returns :

the final_df enriched

```

"""

```

```

tmp_df = f_to_apply(
    df=origin_df,
    id_column=unique_key,
    indicator_column=indicator_column,
    group_interest=group_interest,
    new_column_name=new_column_name,
)
final_df = final_df.merge(
    tmp_df[[unique_key, new_column_name]], how="left", on=unique_key
)

```

```
return final_df
```

```
def calculate_sum_of_group(df, id_column, indicator_column, new_column_name):
```

```
    df = df[[id_column, indicator_column]]
```

```
    df = df.groupby(id_column, as_index=False).agg(np.sum)
```

```
    df = df.rename(columns={indicator_column: new_column_name})
```

```
    return df
```

```
def apply_func_sum(
```

```
    final_df, origin_df, f_to_apply, unique_key, indicator_column, new_column_name
```

```
):
```

```
    tmp_df = f_to_apply(origin_df, unique_key, indicator_column, new_column_name)
```

```
    final_df = final_df.merge(tmp_df, how="left", on=unique_key)
```

```
    return final_df
```