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

Victor Beltrão Valente Duarte

Analise do Programa Internacional de Avaliação de Estudantes utilizando modelos de previsão para o desempenho dos alunos em Matemática.

Belo Horizonte
2021

Victor Beltrão Valente Duarte

Analise do Programa Internacional de Avaliação de Estudantes utilizando modelos de previsão para o desempenho dos alunos em Matemática.

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

2021

SUMÁRIO

1. Introdução

O PISA (Programme for International Student Assessment) ou Programa Internacional de Avaliação de Estudantes, representa a mais importante avaliação comparativa de educação no mundo [1]. É uma rede mundial de avaliação de desempenho escolar coordenado pela OCDE (Organização para a Cooperação e Desenvolvimento Econômico -- Organization for Economic Co-operation and Development), com o objetivo de melhorar políticas nacionais de educação. Para tanto, testes são realizados em dezenas de países a cada três anos avaliando, em geral, a capacidade de alunos de cerca de 15 anos em Leitura (no idioma do seu país), Matemática e Ciências.

O PISA 2012 foi a 5º edição realizada, cobrindo 65 países e 510 mil estudantes com idades entre 14,5 e 16 anos. A amostra, embora imperfeita, tentou representar uma população mundial de 28 milhões de estudantes. Seu foco neste ano foi em Matemática. No caso do Brasil, participaram alunos de instituições escolares públicas, particulares, rurais e urbanas.

O PISA se propõe a avaliar estudantes de 15 anos de idade e matriculados a partir do sétimo ano de estudo. Assim sendo, estão perto de concluir sua educação básica e já devem possuir os requisitos educacionais básicos para prosseguir na vida adulta. Particularmente, os conhecimentos em leitura, matemática e ciências [2].

O teste em si, é feito por meio de computadores com a duração de 2 horas e inclui questões discursivas e de múltipla escolha. Estudantes e diretores de escolas também responderam questionários acerca do contexto e história dos estudantes, da escola, das experiências de aprendizagem, do sistema escolar e do ambiente de aprendizagem.

1.1. Contextualização

O PISA foi criado e desenvolvido em 1997 pelo pesquisador alemão Andreas Schleicher, dando início a uma nova abordagem da OCDE na área de educação. O Brasil, assim como outros países em desenvolvimento, foi convidado a participar do PISA desde a sua primeira edição, em 2000, mesmo não sendo membro da organização [3].

Justamente pela sua proposta e objetivo de ser uma avaliação que reflete a realidade da educação mundial, ela não está restrita a países membros. A participação também comporta não-membros e, em alguns casos, avalia apenas uma parcela de um país. Este é o caso da China, que, em 2012, teve a prova aplicada apenas às províncias de Pequim, Shangai, Jangsu e Zhejiang. Há ainda participantes que são considerados apenas regiões administrativas, e não países, como Hong Kong e Singapura [3].

O Pisa avalia três domínios – leitura, matemática e ciências – em todas as edições ou ciclos. A cada edição, é avaliado um domínio principal, o que significa que os estudantes respondem a um maior número de itens no teste dessa área do conhecimento e que os questionários se concentram na coleta de informações relacionadas à aprendizagem nesse domínio. A pesquisa também avalia domínios chamados inovadores, como Resolução de Problemas, Letramento Financeiro e Competência Global [2].

O PISA é realizado a cada três anos, e cada edição se aprofunda em uma daquelas três categorias. Na edição de 2018, a categoria em foco foi a leitura, enquanto nas edições de 2015 e 2012 a avaliação concentrou-se em ciências e matemática, respectivamente. Isto ajuda a organização a obter informações mais profundas sobre uma ou outra habilidade específica dos estudantes, assim como divulgar dados mais detalhados sobre os resultados [1].

Em termos de ranking de desempenho, a primeira edição do programa, em 2000, revelou a Finlândia como a grande potência mundial em educação. Especialistas têm desenvolvido estudos sobre o desempenho dos países nórdicos e outros casos notáveis para compreender os elementos que possibilitaram esses resultados. Em 2018, por exemplo, chamou atenção a colocação da China em 1º lugar nas três categorias examinadas pelo PISA: leitura, matemática e ciências. No entanto, de maneira geral, os pesquisadores veem como ineficientes as tentativas de replicar práticas de países que obtêm bons desempenhos [2].

O PISA visa conectar fatores socioeconômicos, ambiente escolar e bem-estar dos alunos para melhor entender as variações no desempenho geral do país. O relatório de análise geral da avaliação publicado em 2019 reitera que o objetivo da prova não é gerar mais um elemento de responsabilização de diretores e professores de forma hierarquizada (de cima para baixo, apenas delegando ordens): deve funcionar como um incentivo para que os próprios educadores e profissionais da área reavaliem suas próprias práticas [3].

Além disso, segundo o relatório, tem-se por objetivo diagnosticar a capacidade dos estudantes de não apenas reproduzir o que aprenderam na escola, mas também aplicar o conhecimento de forma criativa, interdisciplinar e prática, permitindo-os efetivamente transformar seu entorno [4].

Os resultados do Pisa permitem que cada país avalie os conhecimentos e as habilidades de seus estudantes em comparação com os de outros países, aprenda com as políticas e práticas aplicadas em outros lugares e formule suas políticas e programas educacionais visando à melhora da qualidade e da equidade dos resultados de aprendizagem.

1.2. O problema proposto

O problema proposto consiste na análise de fatores socioeconômicos de cada país participante da edição do Programa Internacional de Avaliação de Estudantes de 2012, analisando diversas variáveis como por exemplo o nível de escolaridade dos pais ou quantos carros possuem em casa. Utilizando técnicas de classificação em Aprendizagem de Máquina para prever ser estes fatores impulsionam de forma positiva o aluno a obter notas acima da média em matemática.

Para melhor visão do problema em questão e da sua possível solução optou-se por utilizar a técnica dos 5-Ws, respondendo-se as perguntas a que se refere cada

Why? (Por que?): Por que esse problema é importante?

O PISA visa conectar fatores socioeconômicos, ambiente escolar e bem-estar dos alunos para melhor entender as variações no desempenho geral do país. Analisando os dados providos neste trabalho, poderemos abrir discursões sobre os reais problemas da educação brasileira focando na base dos estudos de ensino fundamental e médio.

Who? (Quem?): Os dados utilizados foram disponibilizados pelo OCDE (Organização para a Cooperação e Desenvolvimento Econômico) junto do projeto PISA (Programa Internacional de Avaliação de Estudantes)

What? (O quê?): O objetivo deste trabalho é analisar os dados disponíveis do PISA 2012, considerando amostras com um número limitado de países, e escrever um modelo de previsão para o desempenho dos alunos em Matemática.

Where? (Onde?): Os aspectos geográficos e logísticos são de abrangência mundial, com a participação de 65 países.

When? (Quando?): O período que foi analisado foi do ano de 2012.

1.3. Objetivos

O objetivo deste trabalho é analisar os dados disponíveis, considerando amostras com um número limitado de países, e escrever um modelo de previsão para o desempenho dos alunos em Matemática. Os dados a serem avaliados neste trabalho incluem, para cada aluno, informações sobre o seu desempenho no teste (já normalizados considerando idade, formação na época do teste e particularidades das amostras dos vários países) e informações associadas aos questionários respondidos. Embora vários países tenham sido avaliados em resolução criativa de problemas e educação financeira, na base explorada neste trabalho, não incluímos estes dados.

2. Coleta de Dados

O conjunto de dados PISA 2012 divididos em 4 partes *Pisa Mathematics Scale, Pisa Reading Scale, Pisa Science Scale e PISA 2012 Vardesc* foram obtidos no site OECD em 17 de junho de 2021. A soma total de dados agrupadas nessas 4 partes foi totalizada em 485.375 dados distintos. A referência a essa quantidade foi dada a cada linha do arquivo, contendo dados do tipo *string* e separados por vírgula "," .

Figura 1: Site OECD Disponível em: https://pisadataexplorer.oecd.org/ide/idepisad/

.

Figura 1.2: Obtenção dos dados. Disponível em: https://pisadataexplorer.oecd.org/ide/idepisad/

Os nomes dos conjuntos de dados utilizados assim como suas respectivas propriedades podem ser verificadas na figura 2.3 logo abaixo.

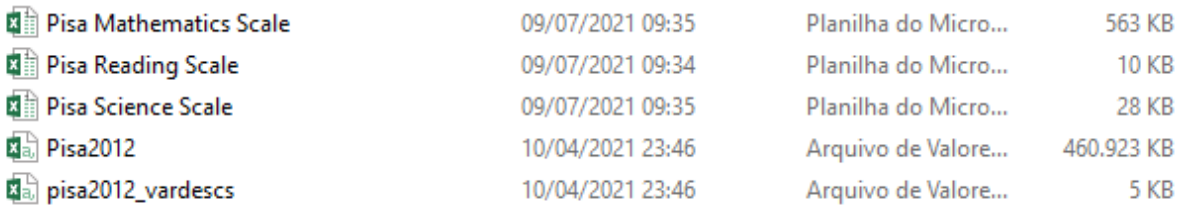


Figura 2: Nomes e propriedades dos conjuntos de dados utilizados.

Abaixo segue a tabela com a descrição de cada campo/coluna do *Dataset* agrupado.

Nome da coluna/campo	Descrição	Tipo
CNT	**Country name**	**String**
OECD	**OECD country?**	**String**
ST01Q01	**International Grade**	**Int**
ST03Q01	**Birth - Month**	**Int**
ST03Q02	**Birth - Year**	**Int**

ST04Q01	**Genre**	**String**
ST07Q01	**Repeat - ISCED1 (1o-5o ano Brasil)**	**String**
ST07Q02	**Repeat - ISCED2 (6o-9o ano Brasil)**	**String**
ST08Q01	**Truancy - Late for School**	**String**
ST09Q01	**Truancy - Skip whole school day**	**String**
ST13Q01	**Mother Highest Schooling**	**String**
ST15Q01	**Mother Current Job Status**	**String**
ST17Q01	**Father Highest Schooling**	**String**
ST19Q01	**Father Current Job Status**	**String**
ST26Q01	**Possessions - desk**	**String**
ST26Q04	**Possessions - computer**	**String**
ST26Q06	**Possessions - Internet**	**String**
ST27Q03	**How many - computers**	**String**
ST27Q04	**How many - cars**	**String**
ST27Q05	**How many - rooms bath or shower**	**String**
ST28Q01	**How many books at home**	**String**
ST42Q02	**Math Self-Concept - Not Good at Maths**	**String**
ST42Q03	**Math Anxiety - Get Very Tense**	**String**
ST42Q04	**Math Self-Concept - Get Good Grades**	**String**
ST42Q05	**Math Anxiety - Get Very Nervous**	**String**
ST42Q06	**Math Self-Concept - Learn Quickly**	**String**
ST42Q07	**Math Self-Concept - One of Best Subjects**	**String**
ST42Q08	**Math Anxiety - Feel Helpless**	**String**
ST42Q09	**Math Self-Concept - Understand Difficult Work**	**String**

ST43Q06	**Perceived Control - Perform Poorly Regardless**	**String**
ST44Q01	**Attributions to Failure - Not Good at Maths Pr..**	**String**
ST44Q08	**Attributions to Failure - Unlucky**	**String**
ST57Q01	**Out-of-School Study Time - Homework**	**Int**
ST57Q02	**Out-of-School Study Time - Guided Homework**	**Int**
ST57Q03	**Out-of-School Study Time - Personal Tutor**	**Int**
ST57Q04	**Out-of-School Study Time - Commercial Company**	**Int**
ST57Q05	**Out-of-School Study Time - With Parent**	**Int**
ST57Q06	**Out-of-School Study Time - Computer**	**Int**
ST62Q01	**Familiarity with Math Concepts - Exponential F...**	**String**
ST62Q02	**Familiarity with Math Concepts - Divisor**	**String**
ST62Q03	**Familiarity with Math Concepts - Quadratic Fun...**	**String**
ST62Q04	**Overclaiming - Proper Number**	**String**
ST62Q06	**Familiarity with Math Concepts - Linear Equation**	**String**
ST62Q07	**Familiarity with Math Concepts - Vectors**	**String**
ST62Q09	**Familiarity with Math Concepts - Rational Number**	**String**
ST62Q10	**Familiarity with Math Concepts - Radicals**	**String**

ST62Q12	**Familiarity with Math Concepts - Polygon**	**String**
ST62Q15	**Familiarity with Math Concepts - Congruent Figure**	**String**
ST62Q16	**Familiarity with Math Concepts - Cosine**	**String**
ST62Q17	**Familiarity with Math Concepts - Arithmetic Mean**	**String**
ST62Q19	**Familiarity with Math Concepts - Probability**	**String**
ST70Q01	**# class-periods/week - Language**	**String**
ST70Q02	**# class-periods/week - Maths**	**Int**
ST70Q03	**# class-periods/week - Science**	**Int**
ST71Q01	**# class-periods/week**	**Int**
ST72Q01	**Class Size in Language**	**Int**
ST79Q04	**Student Orientation - Assigns Complex Projects**	**String**
ST79Q07	**Student Orientation - Has Students Work in Sma...**	**String**
ST79Q10	**Student Orientation - Plans Classroom Activities**	**String**
ST82Q03	**Vignette Teacher Support - Homework Once a Wee...**	**String**
ST84Q03	**Vignette Classroom Management - Students Frequ...**	**String**
ST91Q06	**Perceived Control - Perform Poor Regardless**	**String**
ST93Q01	**Perseverance - Give up easily**	**String**
ST93Q03	**Perseverance - Put off difficult problems**	**String**
ST94Q06	**Openness for Problem Solving - Quick to Unders...**	**String**

ST94Q10	**Openness for Problem Solving - Can Link Facts**	**String**
ST94Q14	**Openness for Problem Solving - Like to Solve C...**	**String**
ST96Q01	**Problem Text Message - Press every button**	**String**
ST96Q03	**Problem Text Message - Manual**	**String**
ST96Q05	**Problem Text Message - Ask a friend**	**String**
ST101Q02	**Problem Route Selection - Study map**	**Int**
ST101Q03	**Problem Route Selection - Leave it to brother**	**Int**
ST101Q05	**Problem Route Selection - Just drive**	**Int**
ST104Q01	**Problem Ticket Machine - Similarities**	**Int**
ST104Q04	**Problem Ticket Machine - Try buttons**	**Int**
ST104Q05	**Problem Ticket Machine - Ask for help**	**Int**
ST104Q06	**Problem Ticket Machine - Find ticket office**	**Int**
IC01Q04	**At Home - Internet connection**	**String**
IC01Q06	**At Home - Cell phone w/o Internet**	**String**
IC01Q09	**At Home - Printer**	**String**
IC01Q10	**At Home - USB (memory) stick**	**String**
IC02Q04	**At school - Internet connection**	**String**
IC02Q07	**At school - Ebook reader**	**String**
IC03Q01	**First use of computers**	**String**
IC05Q01	**Internet at School**	**Int**

IC06Q01	**Internet out-of-school - Weekday**	**Int**
IC07Q01	**Internet out-of-school - Weekend**	**Int**
IC08Q03	**Out-of-school 8 - Use email**	**String**
IC08Q08	**Out-of-school 8 - Obtain practical information...**	**String**
IC08Q11	**Out-of-school 8 - Upload content**	**String**
IC09Q01	**Out-of-school 9 - Internet for school**	**String**
IC10Q04	**At School - Download from website**	**String**
IC10Q05	**At School - Post on website**	**String**
IC10Q07	**At School - Practice and drilling**	**String**
IC10Q08	**At School - Homework**	**String**
IC11Q03	**Maths lessons - Geometric figures**	**String**
IC11Q05	**Maths lessons - Algebra**	**String**
IC22Q08	**Attitudes - Too unreliable**	**String**
EC03Q01	**Future Orientation - Internship**	**String**
EC03Q03	**Future Orientation - Job fair**	**String**
EC03Q04	**Future Orientation - Career advisor at school**	**String**
EC03Q09	**Future Orientation - web search - ISCED 3-5 (a...**	**String**
PV1MATH	**Maths performance**	**Float**
PV1READ	**Reading performance**	**Float**
PV1SCIE	**Science performance**	**Float**

3. Processamento/Tratamento de Dados

O processamento e o tratamento dos dados foram feitos utilizando a linguagem Python, versão 3.7.0, no ambiente Jupyter Notebook, na versão 5.6.0, Dentro da linguagem, utilizou-se a biblioteca *"Pandas"* que é uma poderosa ferramenta para tratamento e análise de dados, *"Numpy"* uma biblioteca para a linguagem Python com funções para se trabalhar com computação numérica, *"Matplotlib"* é uma biblioteca abrangente para criar visualizações estáticas, animadas e interativas em Python. e *"Sklearn" Scikit-learn (Sklearn)* é a biblioteca mais útil e robusta para aprendizado de máquina em Python. Ele fornece uma seleção de ferramentas eficientes para aprendizado de máquina e modelagem estatística, incluindo classificação, regressão, clustering e redução de dimensionalidade através de uma interface de consistência em Python.

O primeiro passo dado foi a importação das bibliotecas citadas acima.

```
[ ]  #Importação das bibliotecas

     import numpy as np
     import pandas as pd
     from tqdm.notebook import tqdm

     import sklearn
     import sys
     import os

     import matplotlib.pyplot as plt
     from matplotlib import cm, colors
     %matplotlib inline

     import warnings

     warnings.filterwarnings("ignore")
```

Figura 3: Importação das bibliotecas.

Mensagens de aviso são normalmente emitidas em situações em que é útil alertar o usuário de alguma condição em um programa, onde essa condição (normalmente) não justifica levantar uma exceção e encerrar o programa. Por exemplo, pode-se querer emitir um aviso quando um programa usa um módulo obsoleto. Por esse motivo foi utilizado a função *"Warnings"*, para evitar que tivesse uma quantidade poluente de avisos nesse trabalho.

As versões das seguintes bibliotecas.

```
[ ]  print("Python", sys.version)
     print("--------")
     print("Pandas:", pd.__version__)
     print("Numpy:", np.__version__)
     print("SKLearn:", sklearn.__version__)
```

```
Python 3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]
--------
Pandas: 1.2.4
Numpy: 1.18.1
SKLearn: 0.22.1
```

Figura 4: Versões das bibliotecas utilizadas.

Em seguida, deve-se fazer a leitura e tratamento dos datasets escolhidos, os datasets foram mesclados utilizando a ferramenta Microsoft Excel, para melhorar a minha manipulação deles dentro do Jupyter Notebook. A seguir, são abertos os dois arquivos a serem usados neste trabalho: (1) os dados do PISA 2012 selecionados para uso que foram mesclados dos outros três datasets; e (2) a descrição das 105 variáveis usadas para descrever cada estudante. O uso do comando pd.read_ cria um dataframe lendo um arquivo, nesse caso usaremos o formato “CSV”, que é uma estrutura bidimensional de dados similar a uma planilha.

```
In [5]: pisa0 = pd.read_csv('data/data/pisa2012.csv')

In [6]: pisa_dict = pd.read_csv('data/pisa2012_vardescs.csv')
        with pd.option_context('display.max_rows', None):
            display(pisa_dict)
```

	varname	desc
0	CNT	Country name
1	OECD	OECD country?
2	ST01Q01	International Grade
3	ST03Q01	Birth - Month
4	ST03Q02	Birth - Year
5	ST04Q01	Genre
6	ST07Q01	Repeat - ISCED1 (1o-5o ano Brasil)
7	ST07Q02	Repeat - ISCED2 (6o-9o ano Brasil)
8	ST08Q01	Truancy - Late for School
9	ST09Q01	Truancy - Skip whole school day
10	ST13Q01	Mother Highest Schooling

Figura 5: Importação dos Datasets.

Como vamos utilizar dois datasets nesse trabalho, e bom sempre saber quais as variáveis estão sendo utilizadas, utilizando o dataset Pisa2012_vardesc, podemos descrever quaisquer variáveis do conjunto de dados.

```
In [7]: pisa_desc = dict(zip(list(pisa_dict['varname'].values), list(pisa_dict['desc'].values)))

        # ex: qual o significado da variável ST28Q01
        pisa_desc['ST28Q01']

Out[7]: 'How many books at home'
```

Figura 6: Dicionário para descrever quaisquer variável.

Utilizar a função “info()” que, neste caso, mostrou que o dataframe possui 480.443 entradas, divididas nas 105 colunas apresentadas anteriormente.

```
In [9]: #Utilizando a função info() para saber exatamente as dimensoes de colunas e os tipos de dados do meu dataframe
        pisa.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 314423 entries, 4743 to 480443
Columns: 105 entries, CNT to PV1SCIE
dtypes: float64(21), int64(6), object(78)
memory usage: 254.3+ MB
```

Figura 7: Uso da função Info().

A função shape() retorna uma tupla representando a dimensionalidade do dataframe, mostrando a quantidade de colunas e linhas que serão utilizadas.

```
In [10]: #Utilizando a função shape() para saber quantas linhas e colunas eu vou utilizar
         pisa.shape

Out[10]: (314423, 105)
```

Figura 8: Uso da Função Shape().

A função Pandas describe() é usado para visualizar alguns detalhes estatísticos básicos como percentil, média, std etc. de uma moldura de dados ou uma série de valores numéricos. Quando este método é aplicado a uma série de cordas, devolve uma saída diferente que é mostrada nos exemplos abaixo. Tipo de retorno: Resumo estatístico da moldura de dados

```
In [11]: #O função describe() me descreve um conjunto de estatisticas descritivas
         pisa.describe()
```

Out[11]:

	ST01Q01	ST03Q01	ST03Q02	ST57Q01	ST57Q02	ST57Q03	ST57Q04	ST57Q05	ST57Q06
count	205454.000000	205454.000000	205454.000000	128778.000000	113850.000000	118712.000000	116917.000000	122413.000000	122477.000000
mean	9.731186	6.530873	1996.079955	5.307545	1.642872	0.853115	0.802604	1.210043	1.627212
std	1.665197	3.409751	0.271224	5.328890	2.600242	2.203234	2.264433	2.363123	2.840401
min	7.000000	1.000000	1996.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	9.000000	4.000000	1996.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	10.000000	7.000000	1996.000000	3.000000	1.000000	0.000000	0.000000	0.000000	1.000000
75%	10.000000	9.000000	1996.000000	7.000000	2.000000	1.000000	0.000000	2.000000	2.000000
max	96.000000	12.000000	1997.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000

8 rows × 27 columns

Figura 9: Utilizando a função Describe().

Desse modo não conseguimos ter uma total certeza de que os dados estão completamente limpos no dataframe, usaremos a função Isnull() para verificar quais funções do dataframe estão nulas ou não, nessa função o dataframe irar retornar uma função booleana de false ou true, false quando o dado estiver preenchido e true quando o dado estiver ausente.

```
In [14]: #Usando a função isnull() podemos ver qual são os dados que estão ausentes no dataset,
         #quando se estiver false significa que tem dados preenchidos e quando estiver em True significa que tem dados faltantes.
         pisa.isnull()
```

Out[14]:

	CNT	OECD	ST01Q01	ST03Q01	ST03Q02	ST04Q01	ST07Q01	ST07Q02	ST08Q01	ST09Q01	...	IC11Q03	IC11Q05	IC22Q08	EC03Q01	EC03Q03
4743	False	False	False	False	False	False	False	False	False	False	...	True	True	True	True	True
4744	False	False	False	False	False	False	False	False	False	False	...	True	True	True	True	True
4745	False	False	False	False	False	False	False	False	False	False	...	True	True	True	True	True
4746	False	False	False	False	False	False	False	False	False	False	...	True	True	True	True	True
4747	False	False	False	False	False	False	False	False	False	False	...	True	True	True	True	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
480439	False	False	False	False	False	False	False	False	False	False	...	True	True	True	True	True
480440	False	False	False	False	False	False	True	True	False	False	...	True	True	True	True	True
480441	False	False	False	False	False	False	False	True	False	False	...	True	True	True	True	True
480442	False	False	False	False	False	False	False	False	False	False	...	True	True	True	True	True
480443	False	False	False	False	False	False	False	False	False	False	...	True	True	True	True	True

205454 rows × 105 columns

Figura 10: Utilizando a função IsNull().

Dessa forma ainda ficaria dificultoso de verificar quais dados estão ausentes no dataframe, utilizando a função sum() em conjunto com a função isnull() poderemos somar os dados faltantes no dataframe

```
In [15]: #Usando o sum() apos o isnull() podemos somar a quantidade de dados faltantes conforme o atributo.
         pisa.isnull().sum()
```

```
Out[15]: CNT             0
         OECD            0
         ST01Q01         0
         ST03Q01         0
         ST03Q02         0
                      ...
         EC03Q04    142756
         EC03Q09    142760
         PV1MATH         0
         PV1READ         0
         PV1SCIE         0
         Length: 105, dtype: int64
```

Figura 11: Utilização da função IsNull.Sum().

Sabendo quais os dados faltantes do dataframe, irei fazer uma substituição dos dados com valores faltantes começando com os tipos float e integer

```
In [12]: #Filtrando os dados pleo o tipo, neste caso usando o tipo float
         pisa.select_dtypes(include = ['float'])
```

Out[12]:

	ST57Q01	ST57Q02	ST57Q03	ST57Q04	ST57Q05	ST57Q06	ST70Q01	ST70Q02	ST70Q03	ST71Q01	...	ST101Q02
4743	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	2.0
4744	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
4745	NaN	NaN	NaN	NaN	NaN	NaN	8.0	8.0	2.0	35.0	...	NaN
4746	NaN	NaN	NaN	NaN	NaN	NaN	8.0	NaN	NaN	7.0	...	NaN
4747	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...
480439	1.0	1.0	1.0	1.0	1.0	1.0	6.0	6.0	6.0	NaN	...	2.0
480440	3.0	3.0	4.0	2.0	3.0	5.0	NaN	NaN	NaN	7.0	...	NaN
480441	0.0	0.0	0.0	0.0	0.0	0.0	3.0	3.0	3.0	NaN	...	2.0
480442	1.0	1.0	1.0	1.0	1.0	1.0	20.0	10.0	30.0	NaN	...	NaN
480443	0.0	NaN	16.0	0.0	2.0	3.0	7.0	7.0	7.0	NaN	...	NaN

205454 rows × 21 columns

Figura 12: Função Dtypes, traz apenas os dados do tipo escolhido, neste caso o tipo float.

Logo após isso farei a mesma coisa, com os dados do tipo integer.

```
In [13]: #Filtrando os dados pelo o tipo, neste caso usando o tipo integer
         pisa.select_dtypes(include = ['integer'])
```

Out[13]:

	ST01Q01	ST03Q01	ST03Q02	IC05Q01	IC06Q01	IC07Q01
4743	10	8	1996	97	97	97
4744	10	6	1996	97	97	97
4745	10	9	1996	97	97	97
4746	9	12	1996	97	97	97
4747	10	3	1996	97	97	97
...	...	...	...	...	...	...
480439	9	3	1997	97	97	97
480440	9	2	1997	97	97	97
480441	9	1	1997	97	97	97
480442	9	10	1996	97	97	97
480443	10	8	1996	97	97	97

205454 rows × 6 columns

Figura 13: Função Dtypes utilizando o tipo integer.

Agora que sabemos quais são as os dados das variáveis faltantes, podemos fazer algumas operações para corrigir estes dados ausentes, poderíamos utilizar uma função que excluiriam todos as linhas e colunas que não possuíssem dados, mas neste caso não gostaria

de perder informação que pode ser utilizada no algoritmo de treinamento, irei substituir os dados faltantes float pela moda geral e os dados integer pela média geral.

```
In [16]: #Criando a moda das variaveis
media_values = {'ST57Q01': pisa.ST57Q01.mean(),
                'ST57Q02': pisa.ST57Q01.mean(),
                'ST57Q03': pisa.ST57Q01.mean(),
                'ST57Q04': pisa.ST57Q01.mean(),
                'ST57Q05': pisa.ST57Q01.mean(),
                'ST57Q06': pisa.ST57Q01.mean(),
                 'ST70Q01': pisa.ST70Q01.mean(),
                 'ST70Q02': pisa.ST70Q02.mean(),
                 'ST70Q01': pisa.ST70Q03.mean(),
                 'ST71Q01': pisa.ST70Q03.mean(),
                 'ST101Q02': pisa.ST101Q02.mean(),
                 'ST101Q03': pisa.ST101Q03.mean(),
                 'ST101Q05': pisa.ST101Q05.mean(),
                 'ST104Q01': pisa.ST104Q01.mean(),
                  'ST104Q04': pisa.ST104Q04.mean(),
                 'ST104Q05': pisa.ST104Q05.mean(),
                 'ST104Q06': pisa.ST104Q06.mean()
                }
media_values

Out[16]: {'ST57Q01': 5.307544766963301,
          'ST57Q02': 5.307544766963301,
          'ST57Q03': 5.307544766963301,
          'ST57Q04': 5.307544766963301,
          'ST57Q05': 5.307544766963301,
          'ST57Q06': 5.307544766963301,
          'ST70Q01': 4.486954273293632,
          'ST70Q02': 4.584783517185369,
          'ST71Q01': 4.486954273293632,
          'ST101Q02': 2.0315849227974567,
          'ST101Q03': 2.838698427620506,
          'ST101Q05': 2.1783434184318744,
          'ST104Q01': 1.8509388781958989,
          'ST104Q04': 2.8042633304992104,
          'ST104Q05': 1.9150813260774,
          'ST104Q06': 1.9096178503417875}
```

Figura 14: Verificando a média geral das variáveis do tipo Float.

Agora vamos verificar a moda das variáveis Integer.

```
In [17]: #Criando a media das variaveis
         moda_values = {'IC05Q01': pisa.IC05Q01.mode(),
                        'IC06Q01': pisa.IC06Q01.mode(),
                        'IC07Q01': pisa.IC06Q01.mode(),
                        'ST01Q01': pisa.ST01Q01.mode(),
                        'ST03Q01': pisa.ST03Q01.mode(),
                        'ST03Q02': pisa.ST03Q02.mode(),
                       }
         moda_values

Out[17]: {'IC05Q01': 0    97
          dtype: int64,
          'IC06Q01': 0    97
          dtype: int64,
          'IC07Q01': 0    97
          dtype: int64,
          'ST01Q01': 0    10
          dtype: int64,
          'ST03Q01': 0    8
          dtype: int64,
          'ST03Q02': 0    1996
          dtype: int64}
```

Figura 15: Verificando a moda das variáveis do tipo Integer.

Agora que já sabemos qual são os valores que vamos substituir usarei um simples loop usando a função '**For'** para fazer essa substituição de forma mais automática.

```
In [18]: for r in media_values:
             if r != 'PV1MATH' or 'PV1READ' or 'PV1SCIE':
                 pisa.update(pisa[r].fillna(pisa[r].mean()))

In [19]: for i in moda_values:
             if i != 'PV1MATH' or 'PV1READ' or 'PV1SCIE':
                 pisa.update(pisa[i].fillna(pisa[i].mode()))

In [20]: #Verificando se ainda tem dados ausentes no dataset
         pisa.ST57Q01.isnull()

Out[20]: 4743      False
         4744      False
         4745      False
         4746      False
         4747      False
                   ...
         480439    False
         480440    False
         480441    False
         480442    False
         480443    False
         Name: ST57Q01, Length: 205454, dtype: bool
```

Figura 16: Substituição dos dados ausentes Float e Integer.

Depois de fazer essas pequenas substituições e conhecemos a dimensões dos nossos datasets podemos passar para a próxima etapa do projeto, a análise e exploração dos dados.

4. Análise e Exploração dos Dados

Para esta etapa de análise e exploração dos dados, responderei algumas perguntas para que tenha um contexto a ser analisado no decorrer do trabalho, gerando assim insights e uma contextualização mais clara.

Considerando o desempenho médio dos alunos em matemática, leitura e ciências respectivamente nos atributos (PV1MATH, PV1READ, PV1SCIE), como os países se saíram?

```
:  #Criando um dicionario das medias dos alunos por pais
   dic_MRS = pisa.groupby('CNT').agg({'PV1MATH': 'mean',
                                      'PV1READ': 'mean',
                                      'PV1SCIE': 'mean'})
```

Figura 17: Dicionário de Medias

Nessa etapa, foi criado um dicionário de medias para verificar quais são as medias dos países que participaram dessa edição de Pisa, na função acima conseguimos agregar as medias das três matérias por pais.

```
#Criando uma função para retornar a media das materias
#PV1(MATH, READ e SCIE) junto aos paises.
def soma_media(frame, new_col_name, dic_MRS):
    frame[new_col_name] = frame[dic_MRS].astype(float).sum(axis=1)/3
    return(frame)

result = soma_media(dic_MRS,'Media_CNT', ['PV1MATH', 'PV1READ', 'PV1SCIE'])
```

Figura 18: Função de somar as medias

Na etapa acima, foi criado uma função que irá retornar a média das matérias somadas junto aos países, resultando no seguinte dicionário.

```
import operator
sorted(result['Media_CNT'].items(), key=operator.itemgetter(1))
```

```
[('Peru', 375.3494586854454),
 ('Indonesia', 385.5707373176802),
 ('Brazil', 392.75887374505265),
 ('Colombia', 403.1093947904047),
 ('Uruguay', 414.206183411728),
 ('Mexico', 422.4684383155363),
 ('United Arab Emirates', 437.90090191304444),
 ('Bulgaria', 445.1742782342548),
 ('Chile', 456.33068016336165),
 ('Turkey', 462.7577169760732),
 ('Greece', 466.19815925853624),
 ('Slovak Republic', 476.8647432022222),
 ('Russia', 482.72483516399717),
 ('Sweden', 483.46165234375),
 ('Denmark', 484.04304866996455),
 ('Lithuania', 484.21032911794526),
 ('Portugal', 485.2419021787259),
 ('Hungary', 494.78399245322277),
 ('Italy', 495.73226971969393),
 ('United States', 496.6254376206207),
 ('Latvia', 499.09349940393275),
 ('United Kingdom', 499.3270028306611),
 ('Australia', 501.64625233064),
 ('France', 503.81596262013164),
 ('Switzerland', 507.0097615370912),
 ('Canada', 511.75927223820986),
 ('Finland', 515.4686899460107),
 ('Czech Republic', 520.1430998247921),
 ('Liechtenstein', 527.2656882821387),
 ('Taiwan', 533.854893736906),
 ('Japan', 540.3124305831096),
 ('South Korea', 542.7387614146628),
 ('Singapore', 550.8096553191477),
 ('China', 553.7326484125928)]
```

Figura 19: Dicionários dos países por media decrescente

Puxando somente os cinco melhores e os cinco piores países em questão de notas poderemos ter um insight geral de qual pais obteve a melhor e pior média geral.

```
#5 Paises de melhor Media geral
result['Media_CNT'].sort_values(ascending = False).head(5)
```

```
CNT
China          553.732648
Singapore      550.809655
South Korea    542.738761
Japan          540.312431
Taiwan         533.854894
Name: Media_CNT, dtype: float64
```

```
#5 Paises com a pior Media geral
result['Media_CNT'].sort_values(ascending = True).head(5)
```

```
CNT
Peru         375.349459
Indonesia    385.570737
Brazil       392.758874
Colombia     403.109395
Uruguay      414.206183
Name: Media_CNT, dtype: float64
```

Figura 20: 10 países com melhor e pior média geral.

Dessa forma sabemos que a China se encontra no primeiro lugar dos países que tiveram uma excelente nota geral, e o Brasil se encontra no top 5 piores países com a média geral.

Entrando em outra discursão podemos ver que nesta lista a países que participam ativamente do OECD e países que não fazem parte, como o Brasil que no momento dessa prova não fazia parte da OECD, a OECD conhecida como Organização para a Cooperação e Desenvolvimento Econômico, é uma organização intergovernamental composta de 38 países membros fundada em 1961 para estimular o professo econômico e o comercio mundial. Comparando as distribuições de probabilidade para cada uma das notas usadas anteriormente, podemos verificar quais os países pertencentes ao OECD e não pertencentes ao OECD que obtiveram as melhores medias.

```
#Quais os paises membro e nao membros do OCDE?
member_oecd = pisa.groupby(['CNT', 'OECD']).agg({'PV1MATH': 'mean',
                                                 'PV1READ': 'mean',
                                                 'PV1SCIE': 'mean'})
```

Figura 21: Países Membros e Não Membros do OECD.

Criando esse agregador, podemos criar agora uma função que irá retornar a média geral das provas por pais.

```
: #Criação de uma função que retornar a media das provas
  #por pais pertecendo ou não ao OECD.
  def member_oecd1(frame, new_col_name, member_oecd):
      frame[new_col_name] = frame[member_oecd].astype(float).sum(axis=1)/3
      return(frame)

  result1 = member_oecd1(member_oecd,'Media_CNT', ['PV1MATH', 'PV1READ', 'PV1SCIE'])

  pisa3 = pd.DataFrame(result1)
```

```
: import operator
  sorted(result1['Media_CNT'].items(), key=operator.itemgetter(1))
```

```
: [(('Peru', 'Non-OECD'), 375.3494586854454),
   (('Indonesia', 'Non-OECD'), 385.5707373176802),
   (('Brazil', 'Non-OECD'), 392.75887374505265),
   (('Colombia', 'Non-OECD'), 403.1093947904047),
   (('Uruguay', 'Non-OECD'), 414.206183411728),
   (('Mexico', 'OECD'), 422.4684383155363),
   (('United Arab Emirates', 'Non-OECD'), 437.90090191304444),
   (('Bulgaria', 'Non-OECD'), 445.1742782342548),
   (('Chile', 'OECD'), 456.33068016336165),
   (('Turkey', 'OECD'), 462.7577169760732),
   (('Greece', 'OECD'), 466.19815925853624),
   (('Slovak Republic', 'OECD'), 476.8647432022222),
   (('Russia', 'Non-OECD'), 482.72483516399717),
   (('Sweden', 'OECD'), 483.46165234375),
   (('Denmark', 'OECD'), 484.04304866996455),
   (('Lithuania', 'Non-OECD'), 484.21032911794526),
   (('Portugal', 'OECD'), 485.2419021787259),
   (('Hungary', 'OECD'), 494.78399245322277),
   (('Italy', 'OECD'), 495.73226971969393),
   (('United States', 'OECD'), 496.6254376206207),
   (('Latvia', 'Non-OECD'), 499.09349940393275),
   (('United Kingdom', 'OECD'), 499.3270028306611),
   (('Australia', 'OECD'), 501.64625233064),
   (('France', 'OECD'), 503.81596262013164),
   (('Switzerland', 'OECD'), 507.0097615370912),
   (('Canada', 'OECD'), 511.75927223820986),
   (('Finland', 'OECD'), 515.4686899460107),
   (('Czech Republic', 'OECD'), 520.1430998247921),
   (('Liechtenstein', 'Non-OECD'), 527.2656882821387),
   (('Taiwan', 'Non-OECD'), 533.854893736906),
   (('Japan', 'OECD'), 540.3124305831096),
   (('South Korea', 'OECD'), 542.7387614146628),
   (('Singapore', 'Non-OECD'), 550.8096553191477),
   (('China', 'Non-OECD'), 553.7326484125928)]
```

Figura 22: Lista de Países membros e não membros do OECD

Para criar uma organização mais limpa, podemos organizar esse dicionário como um data-frame criando uma nova visão do resultado obtido.

CNT	OECD	PV1MATH	PV1READ	PV1SCIE	Media_CNT
Australia	OECD	492.842855	500.845303	511.250599	501.646252
Brazil	Non-OECD	382.500793	400.453951	395.321878	392.758874
Bulgaria	Non-OECD	442.158361	442.452408	450.912065	445.174278
Canada	OECD	509.328982	511.235906	514.712929	511.759272
Chile	OECD	444.419644	459.999572	464.572824	456.330680
China	Non-OECD	570.007384	540.049095	551.141467	553.732648
Colombia	Non-OECD	386.185442	414.549009	408.593734	403.109395
Czech Republic	OECD	519.774978	513.008171	527.646151	520.143100
Denmark	OECD	486.185453	483.913033	482.030660	484.043049
Finland	OECD	507.525776	510.905535	527.974759	515.468690
France	OECD	499.470436	509.233472	502.743981	503.815963
Greece	OECD	453.890068	477.386100	467.318310	466.198159
Hungary	OECD	485.390840	496.648704	502.312434	494.783992
Indonesia	Non-OECD	376.071790	397.659527	382.980895	385.570737
Italy	OECD	491.904167	495.244210	500.048433	495.732270
Japan	OECD	536.064600	537.757029	547.115662	540.312431
Latvia	Non-OECD	495.449250	495.539979	506.291269	499.093499
Liechtenstein	Non-OECD	537.899774	518.354060	525.543230	527.265688
Lithuania	Non-OECD	478.680925	477.694993	496.255069	484.210329
Mexico	OECD	418.699072	429.133635	419.572607	422.468438
Peru	Non-OECD	367.810655	384.370436	373.867285	375.349459
Portugal	OECD	484.559765	484.732846	486.433096	485.241902
Russia	Non-OECD	483.826431	478.457998	485.890076	482.724835
Singapore	Non-OECD	568.359669	537.372245	546.697052	550.809655
Slovak Republic	OECD	485.644734	468.407184	476.542312	476.864743
South Korea	OECD	554.228404	536.063023	537.924858	542.738761
Sweden	OECD	479.152938	485.592069	485.639950	483.461652
Switzerland	OECD	520.672206	497.243059	503.114019	507.009762
Taiwan	Non-OECD	557.576851	521.925757	522.062074	533.854894
Turkey	OECD	448.833618	475.561779	463.877754	462.757717
United Arab Emirates	Non-OECD	431.373626	437.239951	445.089129	437.900902
United Kingdom	OECD	489.645545	498.292310	510.043154	499.327003
United States	OECD	485.632260	502.823313	501.420740	496.625438
Uruguay	Non-OECD	410.957687	413.573968	418.086895	414.206183

Figura 23: Lista de Países pertencentes ou não ao OECD e com suas medias gerais das provas.

Dessa forma, podemos perceber que nesta edição do PISA, tivemos uma quantidade maior de países que não pertenciam ao OECD tirando notas acima da média, mostrando os dois melhores qualificados, temos uma visão que novamente a China manteve seu pódio.

```
#Melhores paises pertecentes ao OECD
pisa3.sort_values('OECD', ascending = False).head(2)
```

CNT	OECD	PV1MATH	PV1READ	PV1SCIE	Media_CNT
Australia	OECD	492.842855	500.845303	511.250599	501.646252
Hungary	OECD	485.390840	496.648704	502.312434	494.783992

```
#Melhores paises não pertecentes ao OECD
pisa3.sort_values('Media_CNT', ascending = False).head(2)
```

CNT	OECD	PV1MATH	PV1READ	PV1SCIE	Media_CNT
China	Non-OECD	570.007384	540.049095	551.141467	553.732648
Singapore	Non-OECD	568.359669	537.372245	546.697052	550.809655

Figura 24: Melhores 2 países pertencentes e não pertencentes ao OECD.

Criando um histograma, podemos avaliar como a média dos países nas três áreas de atuação, matemática, leitura e ciências.

```
#Histograma das notas PV1MATH
fig, ax = plt.subplots()

ax.hist('PV1MATH', data=pisa, density=True, bins=30)
ax.set_title("Histograma da notas PV1MATH")
ax.set_ylabel("Quantidade de paises")
ax.set_xlabel("Media das notas")

plt.tight_layout()
```

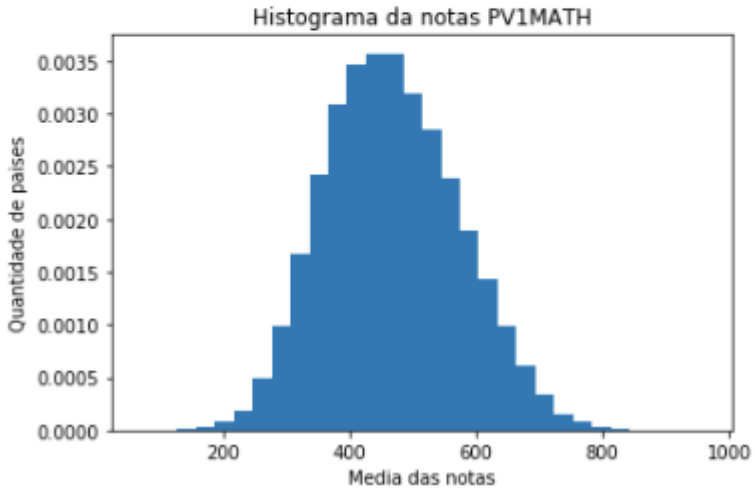


Figura 25: Histograma das medias da prova de Matemática.

Neste histograma, podemos ver quer que a grande maioria das notas de matemática ficaram divididas entre 400 e 600 pontos.

```
#Histograma das notas PV1READ
fig, ax = plt.subplots()

ax.hist('PV1READ', data=pisa, density=True, bins=30)
ax.set_title("Histograma da notas PV1READ")
ax.set_ylabel("Quantidade de paises")
ax.set_xlabel("Media das notas")

plt.tight_layout()
```

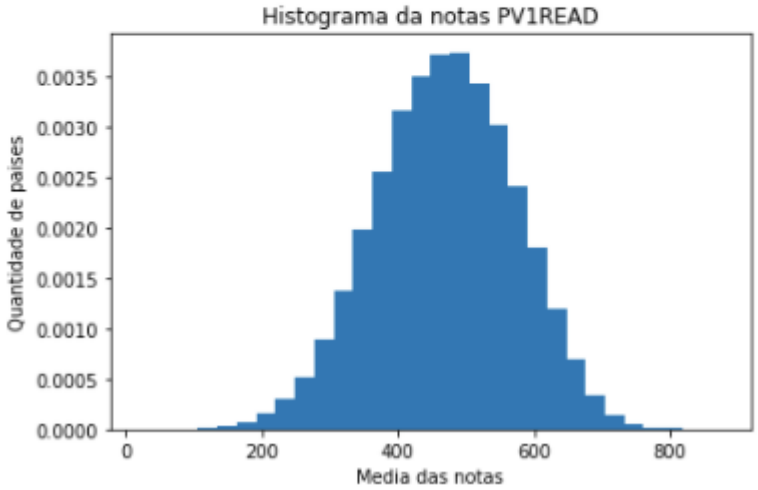


Figura 26:: Histograma das medias da prova de Leitura.

Já no histograma de Leitura, as notas têm um deslocamento para a esquerda, dando a entender que a nota tem sua média localizada mais aos 500 e 600 pontos.

```
#Histograma das notas PV1SCIE
fig, ax = plt.subplots()

ax.hist('PV1SCIE', data=pisa, density=True, bins=30)
ax.set_title("Histograma da notas PV1SCIE")
ax.set_ylabel("Quantidade de paises")
ax.set_xlabel("Media das notas")

plt.tight_layout()
```

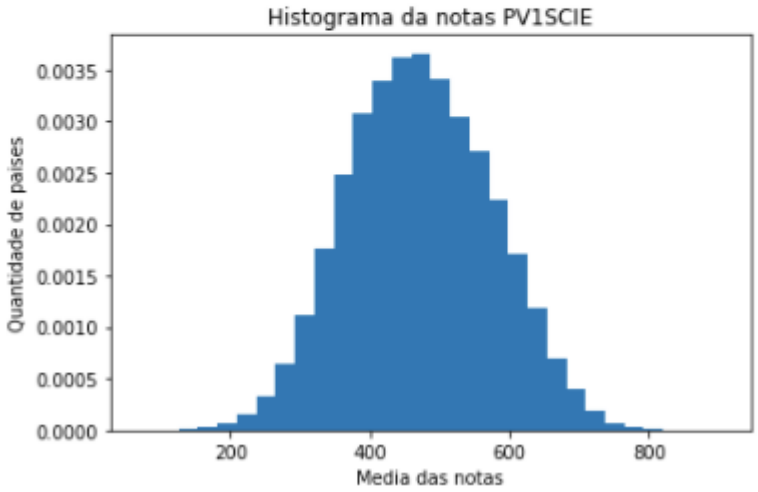


Figura 27: Histograma das medias da prova de Ciências.

Comparando aos outros dois, a nota de ciências já tem seu histograma mais centrado, podendo ver que as notas medias dessa categoria de prova se acumularam nos 500 pontos.

A idade que os alunos fazem essa prova varia entre 14 a 16 anos, mas vamos ver se isso influencia de alguma forma o desempenho deles.

```
pisa14.ST03Q02.value_counts()
```

```
1996    288022
1997     26401
Name: ST03Q02, dtype: int64
```

```
def discretize_age(age):
    if age == 1996:
        return '16'
    else:
        return '15'
```

```
pisa14['Age'] = pisa14['ST03Q02'].apply(discretize_age)
```

```
pisa14.groupby('Age').agg({'PV1MATH': 'mean',
                           'PV1READ': 'mean',
                           'PV1SCIE': 'mean'})
```

Age	PV1MATH	PV1READ	PV1SCIE
15	471.233443	480.653725	485.805244
16	473.659869	476.114151	478.322324

Figura 28: Idade dos alunos que participaram da Prova de 2012.

Dessa forma fica claro que a idade não proporcionar uma melhora nas notas obtidas pelos alunos nas três grandes áreas de interesse adotadas pelo programa em 2012.

É interessante pensar que há uma diferença entre homens e mulheres em cada uma das áreas de interesse do Pisa, verificando o desempenho de homens e mulheres em cada umas dessas áreas, podemos ver qual pais tem a menor diferença entre os sexos.

```
pisa5 = pd.DataFrame(pisa[['CNT', 'ST04Q01', 'PV1MATH', 'PV1READ', 'PV1SCIE']])
pisa5
```

	CNT	ST04Q01	PV1MATH	PV1READ	PV1SCIE
4743	United Arab Emirates	Male	328.2521	313.5423	397.2771
4744	United Arab Emirates	Male	270.7665	269.5957	301.5106
4745	United Arab Emirates	Male	314.9322	267.9116	293.7710
4746	United Arab Emirates	Male	250.2804	217.3890	329.2055
4747	United Arab Emirates	Male	310.1807	279.2190	378.1611
...	...	...	...	...	...
480439	United States	Male	311.8165	231.1825	275.7740
480440	United States	Male	311.8944	287.7196	307.3853
480441	United States	Male	296.0819	249.6272	268.3141
480442	United States	Male	296.9388	242.9711	274.6550
480443	United States	Female	406.4574	351.2479	389.9105

314423 rows × 5 columns

Figura 29: Criando um novo Dataset verificando ambos os sexos.

Plotando um histograma podemos ver a relação de homens e mulheres em cada uma das áreas de estudo.

```
#Agora um histograma levando em consideração a media de matemática entre homens e mulheres
pisa6Male = pisa[pisa['ST04Q01']=='Male']
pisa6Female = pisa[pisa['ST04Q01']=='Female']

pisa6Male.PV1MATH.hist(bins = 30, density = True, alpha=0.5, label = 'Homem')
pisa6Female.PV1MATH.hist(bins = 30, density = True, alpha=0.5, label = 'Mulher')
```

```
<AxesSubplot:>
```

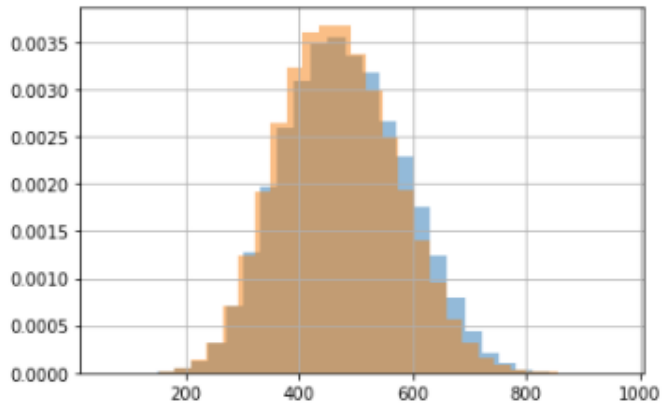


Figura 30: Histograma das medias da prova de Matemática de ambos os sexos.

```
#Agora um histograma levando em consideração a media de Leitura entre homens e mulheres
pisa6Male.PV1READ.hist(bins = 30, density = True, alpha= 0.5, label = 'Homem')
pisa6Female.PV1READ.hist(bins = 30, density = True, alpha=0.5, label = 'Mulher')

plt.legend()
```

```
<matplotlib.legend.Legend at 0x1880b61af10>
```

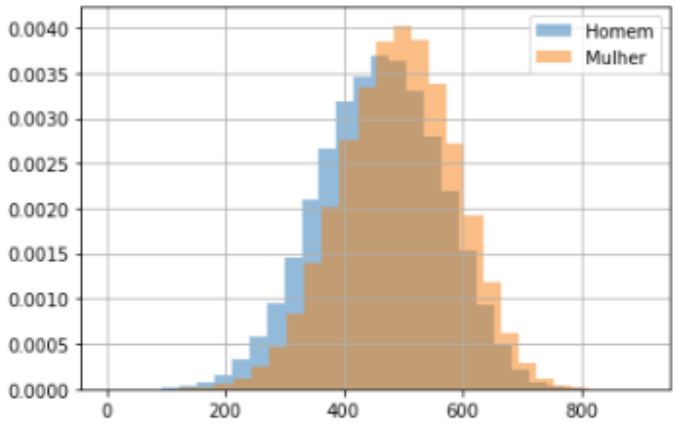


Figura 31: Histograma das medias da prova de Leitura de ambos os sexos.

```
#Agora um histograma levando em consideração a media de Ciências entre homens e mulheres
pisa6Male.PV1SCIE.hist(bins = 30, density = True, alpha= 0.5, label = 'Homem')
pisa6Female.PV1SCIE.hist(bins = 30, density = True, alpha=0.5, label = 'Mulher')

plt.legend()
```

```
<matplotlib.legend.Legend at 0x1880b84a3d0>
```

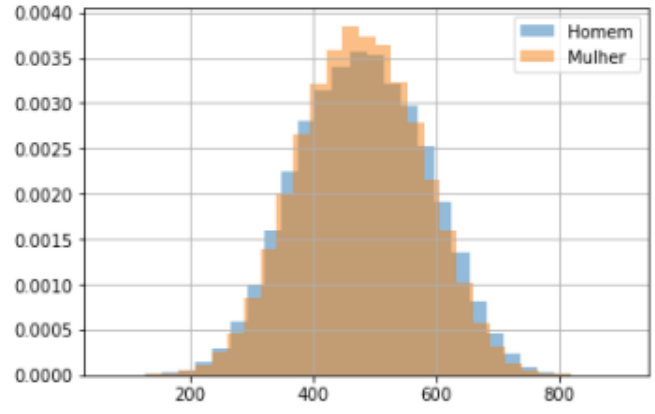


Figura 32: Histograma das medias da prova de ciência de ambos os sexos.

Analisando os três histogramas criados podemos ver que as mulheres têm medias maiores comparados aos homens nas três grandes áreas de interesse do PISA. Olhando todos os países de forma mais detalhada podemos obter um dataframe com a média por sexo de cada prova.

```
pisa.groupby(['CNT', 'ST04Q01']).agg({'PV1MATH': 'mean',
                                      'PV1READ': 'mean',
                                      'PV1SCIE': 'mean'}).head(60)
```

		PV1MATH	PV1READ	PV1SCIE
CNT	ST04Q01			
Australia	Female	486.949163	518.283852	508.833434
	Male	498.473137	484.186143	513.559733
Brazil	Female	374.125981	414.378592	394.032508
	Male	391.938571	384.761934	396.774899
Bulgaria	Female	442.465094	476.277435	459.685495
	Male	441.865922	410.203548	442.547456
Canada	Female	505.068733	529.374147	513.943460
	Male	513.726671	492.512505	515.507221
Chile	Female	430.861337	470.116341	459.661955
	Male	458.659111	449.374545	469.730412
China	Female	567.075268	555.054238	549.942772
	Male	572.797039	525.772993	552.281922
Colombia	Female	372.632528	421.589186	398.693190
	Male	401.457091	406.616020	419.749832
Czech Republic	Female	513.662622	532.175570	526.297687
	Male	525.963536	493.601812	529.011426
Denmark	Female	479.150287	499.628015	476.815832
	Male	493.359271	467.888333	487.348264
Finland	Female	507.139756	539.887589	534.479945
	Male	507.904092	482.501952	521.599414
France	Female	494.341289	530.314409	503.572312
	Male	504.913565	486.862056	501.864943
Greece	Female	449.803422	502.286718	473.578390
	Male	458.020001	462.221726	480.991921
Hungary	Female	479.523334	513.638378	499.835905
	Male	491.896286	477.811839	505.058221
Indonesia	Female	374.414599	411.634250	385.281982
	Male	377.787781	383.188959	380.598162
Italy	Female	481.951575	514.916862	498.541250
	Male	501.487702	476.301049	501.499727

Japan	Female	525.367581	548.809028	540.016058
	Male	545.769012	527.730577	553.556475
Latvia	Female	496.140442	520.512969	512.500502
	Male	494.735214	469.741632	499.876820
Liechtenstein	Female	525.321564	532.233739	515.651064
	Male	549.098439	505.996668	534.350450
Lithuania	Female	478.795747	506.119076	504.072239
	Male	478.568659	449.903909	488.611987
Mexico	Female	412.066784	441.104773	416.543260
	Male	425.860701	416.207050	422.843735
Peru	Female	357.944823	393.849495	370.495413
	Male	378.356307	374.238208	377.471501
Portugal	Female	478.338774	504.488508	487.278591
	Male	490.746062	465.087358	485.592317
Russia	Female	483.636633	498.079828	487.323970
	Male	484.014070	459.059398	484.472495
Singapore	Female	571.038626	554.128161	547.920551
	Male	565.720983	520.868206	545.491945
Slovak Republic	Female	480.733013	489.427185	473.030260
	Male	490.122890	449.242647	479.744350
South Korea	Female	544.938283	548.954007	536.473486
	Male	562.313674	524.843891	539.187998
Sweden	Female	479.807955	509.656965	488.688122
	Male	478.492366	461.323059	482.565925
Switzerland	Female	514.744382	515.746513	499.969055
	Male	526.525539	478.972126	506.219463
Taiwan	Female	555.010418	537.997923	521.938356
	Male	560.297182	504.889808	522.193211
Turkey	Female	441.790163	497.347383	467.623036
	Male	455.570095	454.725669	460.295703

Figura 33: Países divididos por sexo.

Levando em consideração que os pais atuam constantemente na criação dos seus filhos, o grau de instrução agregado dos pais deve ser colocado em conta, já que pais com maiores graus de instrução teoricamente podem dar melhores condições de ensino para seus filhos.

Pisa 2012 foram utilizados alguns parâmetros para averiguar se o nível de instrução acadêmica do pais resultariam em notas melhores por seus filhos. Foi criado uma tabela para calcular o grau de instrução.

- 0 = Ele/Ela Não completou o ISCED nível 1 (Pessoas que não terminaram o ensino fundamental no Brasil)
- 1 = ISCED nível 1 (Primário ou Ensino Fundamental no Brasil)
- 2 = ISCED nível 2 (Secundário ou Ensino Médio no Brasil)
- 3 = ISCED nível 3B, 3C (Ensino Superior, mas focado em formação profissional como Graduação ou Tecnólogo)
- 4 = ISCED nível 3A (Superior ao nível de Mestrado ou Doutorado)

O grau de instrução agregado dos pais deve então ser estimado como a soma dos valores dados para o pai e para a mãe, seguindo como exemplo se o pai tem o nível de ensino ISCED 1 e a mãe tem um ISCED 2, a soma de ambos resultaria em 3.

Criando uma cópia do dataframe para não modificar os dados reais, podemos criar uma função de agregação de ambos os pais.

```
pisa6 = pisa.copy()
```

```
def grau_instrucao_agregado_mae(grauMae):
    if grauMae == "She did not complete <ISCED level 1> ":
        valueM = 0
    elif grauMae == "<ISCED level 1> ":
        valueM = 1
    elif grauMae == "<ISCED level 2> ":
        valueM = 2
    elif grauMae == "<ISCED level 3B, 3C> ":
        valueM = 3
    else:
        valueM = 4
    return valueM

def grau_instrucao_agregado_pai(grauPai):
    if grauPai == "He did not complete <ISCED level 1> ":
        valueP = 0
    elif grauPai == "<ISCED level 1> ":
        valueP = 1
    elif grauPai == "<ISCED level 2> ":
        valueP = 2
    elif grauPai == "<ISCED level 3B, 3C> ":
        valueP = 3
    else:
        valueP = 4
    return valueP
```

Figura 34:Função de Agregação de ambos os pais.

Agora que as funções foram criadas, podemos fazer a soma de ambas para criar uma informação geral da instrução dos pais.

```
pisa6['SomaPais'] = pisa.ST13Q01.apply(grau_instrucao_agregado_mae) +
                    pisa.ST13Q01.apply(grau_instrucao_agregado_pai)

pisa6['SomaPais'].value_counts()
```

```
8    170829
4     67761
6     50206
2     25627
Name: SomaPais, dtype: int64
```

Figura 35: Total da soma dos pais.

Podemos ver que a grande maioria dos países tem acesso à educação superior de nível mestrado ou doutorado, mas dessa forma fica um pouco difícil de ter certeza se a informação está correta.

Levando em consideração que diversos fatores externos podem acabar impactando as notas dos alunos que participaram do PISA 2012, qual seria o tempo dedicado pelos alunos

para as tarefas em casa, como o desempenho das notas em leitura, matemática e ciências podem várias com este tempo dedicado a estudo pelos alunos?

```
pisa13['tempo_estudo']= pd.qcut(pisa.ST57Q01.drop_duplicates(), q=12,
        labels = ['1h semanal','3h semanais','5h semanais','7h semanais',
                  '9h semanais','11h semanais','13h semanais','16h semanais','18
                  '20h semanais','25h semanais','mais de 25h semanais'])
```

```
pisa13.groupby('tempo_estudo').agg({'PV1MATH': 'mean',
                                    'PV1READ': 'mean',
                                    'PV1SCIE': 'mean'})
```

tempo_estudo	PV1MATH	PV1READ	PV1SCIE
1h semanal	294.913533	280.983300	315.653367
3h semanais	412.974533	417.313933	459.660600
5h semanais	410.001600	448.269050	453.506150
7h semanais	537.734333	465.594500	549.179400
9h semanais	421.958250	356.125550	443.621750
11h semanais	464.514233	503.062167	456.365800
13h semanais	488.998800	570.848167	606.620633
16h semanais	463.748250	501.054800	544.237200
18h semanais	400.329733	470.505567	434.141500
20h semanais	471.303900	478.268450	494.582250
25h semanais	469.006100	485.087733	501.436033
mais de 25h semanais	412.143633	409.399933	425.997767

Figura 36: Horas estudadas semanalmente pelos alunos.

As horas utilizadas pelos alunos podem apresentar diversos fatores diferentes, mas a ideia geral que fica e que ao menos 1 hora de estudos extra por dia, pode melhorar consideravelmente a nota média de alunos do ensino médio.

Criando uma lista de países pertencentes ao OECD e não pertencentes ao OECD podemos ver quais países tem mais pontos.

```
for i in Pais6OECD:
    pais_ig_OECD.loc[len(pais_ig_OECD)] = [i, pisa6OECD[pisa6OECD['CNT']==i].SomaPais.sum()]
pais_ig_OECD = pais_ig_OECD.drop_duplicates()
pais_ig_OECD.sort_values(by=['igPais'], ascending = False)
```

	CNT	igPais
11	Italy	196560
1	Canada	166856
14	Mexico	165852
0	Australia	97746
8	United Kingdom	86260
16	United States	77508
2	Switzerland	67772
6	Finland	60498
5	Denmark	49730
12	Japan	48650
3	Chile	47672
4	Czech Republic	39130
13	South Korea	37678
9	Greece	35186
17	Slovak Republic	34330
18	Sweden	34290
10	Hungary	33482
7	France	31692
15	Portugal	29240
19	Turkey	19184

Figura 37: Países pertencentes ao OECD com pontos por instrução dos pais.

```
for i in Pais6NONOECD:
    pais_ig_NONOECD.loc[len(pais_ig_NONOECD)] = [i, pisa6NONOECD[pisa6NONOECD['CNT']==i].SomaPais.sum()
pais_ig_NONOECD = pais_ig_NONOECD.drop_duplicates()
pais_ig_NONOECD.sort_values(by=['igPais'], ascending = False)
```

	CNT	igPais
2	Brazil	103154
4	China	80724
0	United Arab Emirates	78152
3	Colombia	52670
10	Russia	47990
11	Singapore	38436
12	Taiwan	38418
9	Peru	35838
1	Bulgaria	35818
7	Lithuania	32286
8	Latvia	30572
13	Uruguay	27870
5	Indonesia	27168
6	Liechtenstein	1754

Figura 38: Países não pertencentes ao OECD com pontos e instrução dos pais.

Criando um gráfico de barras, podemos ver como varia os desempenhos dos estudantes em relação ao grau de instrução dos seus pais.

```
pisa6.groupby(['SomaPais']).agg({'PV1MATH': 'mean',
                                 'PV1READ': 'mean',
                                 'PV1SCIE': 'mean'})
```

SomaPais	PV1MATH	PV1READ	PV1SCIE
2	413.224760	423.480300	418.328274
4	434.234346	439.874228	439.864131
6	495.307268	493.528013	498.389484
8	491.627510	493.968672	497.835978

```
pisa7 = pisa6[pisa6.CNT.isin(list(pisa6['CNT'].drop_duplicates()))]
group_pais_dep = pd.crosstab(pisa6.CNT, pisa7.SomaPais)
```

```
fig = group_pais_dep.plot.bar(figsize=(18,12))
fig.figure.show()
```

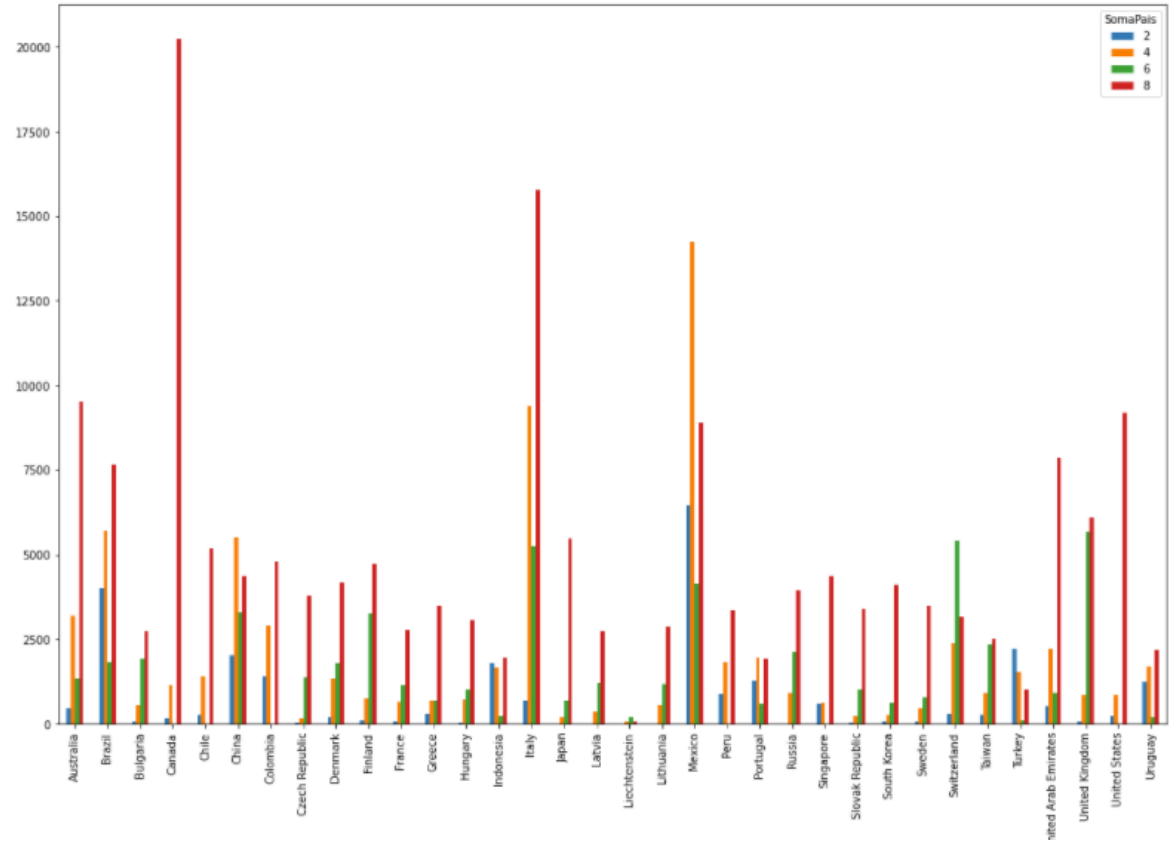


Figura 39: Grau de instrução dos Pais relacionado com a nota do alunos nas três áreas de interesse do PISA.

Dessa maneira fica bem mais claro perceber que o grau de instrução dos pais beneficia o aluno de uma forma que ele tire medias maiores do que aqueles que não tiveram acesso à educação.

Ficando claro que dessa forma, países ricos que tem os melhores desempenhos, mas a outros atributos que podem comprometer esse ensino, como por exemplo a quantidade de livros em casa ou até de acesso ao computador?

```
pisa.groupby('ST28Q01').agg({'PV1MATH': 'mean',
                             'PV1READ': 'mean',
                             'PV1SCIE': 'mean'})
```

ST28Q01	PV1MATH	PV1READ	PV1SCIE
0-10 books	413.353603	418.427756	418.198133
101-200 books	515.714763	518.092341	522.233055
11-25 books	441.457775	448.797808	447.921220
201-500 books	544.459681	544.193553	549.309377
26-100 books	488.107746	490.292003	493.366249
More than 500 books	538.638491	534.755037	543.737377

```
pisa = pisa[pisa.CNT.isin(list(pisa['CNT'].drop_duplicates()))]
group_livros_pais = pd.crosstab(pisa.CNT, pisa.ST28Q01)
```

```
fig = group_livros_pais.plot.bar(figsize=(15,8))
fig.figure.show()
```

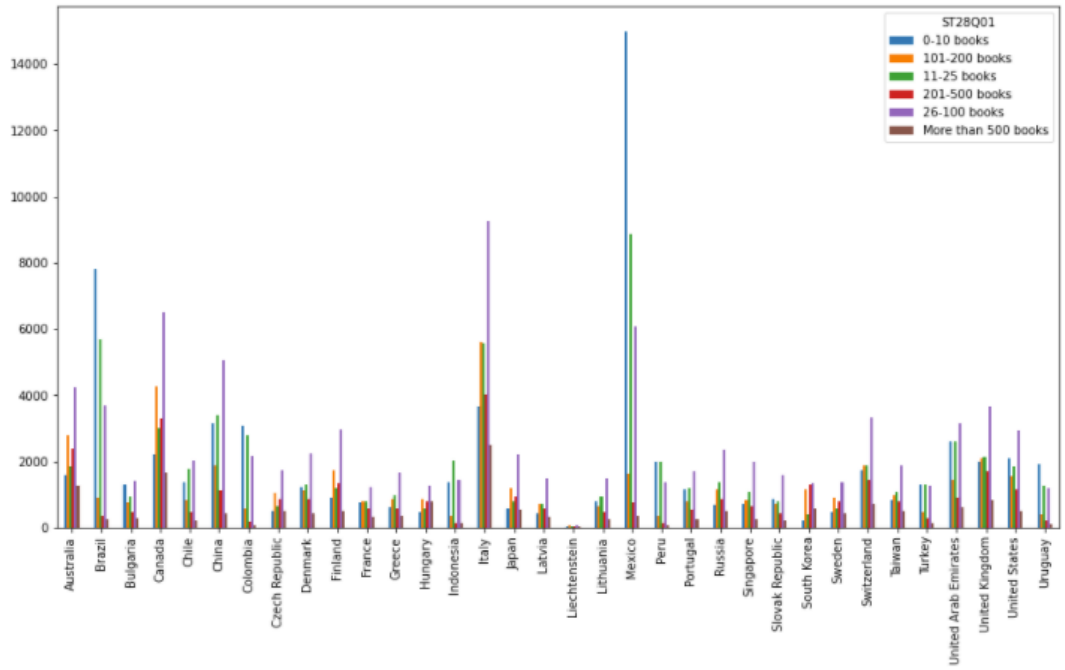


Figura 40: Quantidades de livros por família.

Claramente pessoas que tiveram acesso ao computador, e a informação rápida que a internet provém terão certa vantagem aqueles que não dispões desse artificio. A relação clara que se pode obter e que as famílias que tiveram acesso a mais livros ou tem mais livros em casa obtiveram uma média muito mais elevada do que os que não tiveram esta oportunidade.

5. Criação de Modelos de Machine Learning

O objetivo do estudo desse projeto foi analisar quais as variáveis que impulsionavam os alunos a tirar notas mais altas em Matemática, visando isso, optou-se pela utilização de algoritmos de classificação, que nada mais é do que cálculos preditivos usados para atribuir dados a categorias predefinidas, analisando um conjunto de dados de treinamento. Classificação é o processo de reconhecer, compreender e agrupar ideias e objetos em categorias predefinidas ou "subpopulações".
Nesse estudo serão utilizados ao total 6 tipos de algoritmos de classificação, entre eles: Arvore de Decisão, Naive Bayes, Regressão Logística, Gradiente Descendente, KNN (K-Nearest Neighbors) e Random Forest, logo após verificar qual o algoritmo com melhor desempenho, verificarei qual foram os atributos dos dados que ele utilizou para chegar nesta conclusão.

Uma árvore de decisão é uma representação de uma tabela de decisão sob a forma de árvore. Trata-se de uma forma alternativa de expressar as mesmas regras que são obtidas quando se constrói a tabela.

Em estatística, os classificadores Bayes ingênuos são uma família de "classificadores probabilísticos" simples baseados na aplicação do teorema de Bayes com fortes suposições de independência entre os recursos.

A regressão logística é um algoritmo que lida com questões e problemas de classificação. Quer dizer que este tipo de algoritmo de machine learning analisa diferentes aspectos ou variáveis de um objeto para depois determinar uma classe na qual ele se encaixa melhor.

A gradiente descendente é um algoritmo de otimização usado para minimizar algumas funções movendo-se iterativamente na direção da descida mais íngreme, conforme definido pelo negativo do gradiente. Nos modelos de machine learning, usamos gradiente descendente para atualizar os parâmetros do nosso modelo.

O algoritmo k-nearest neighbor, chamado apenas de KNN, é uma abordagem para classificação de dados que estima a probabilidade de um ponto de dados ser membro de um grupo ou de outro, dependendo de qual grupo os pontos de dados mais próximos a ele estão.

Florestas aleatórias ou florestas de decisão aleatória são um método de aprendizagem de conjunto para classificação, regressão e outras tarefas que operam construindo uma infinidade de árvores de decisão no momento do treinamento.

Preparando os dados para prever se um dado aluno vai receber uma nota acima ou abaixo da média na prova de matemática, para se fazer isso criarei uma nova variável chamada de ”Target” com o valor 1 para os alunos com nota maior que a média global em matemática e 0, caso o contrário. Transformando as variáveis do tipo “Object” em categóricas do tipo inteiro, padronizando as variáveis reais utilizando o “Z-score” e garantindo dessa forma que as variáveis numéricas não incluam NaN, após sua normalização.

```
: pisa15 = pisa.copy()

: #Criação da variavel TARGET.
  media_math = pisa15['PV1MATH'].mean()
  media_math

: 473.45613067526097
```

Figura 41: Criação da variável Target.

Após a criação da variável “Target”, irei definir para os alunos com maior média global em matemática o valor 1 e para aqueles que não alcançaram a média o valor 0.

```
def categoriza(s):
    if s < media_math:
        return 0
    else:
        return 1
```

```
#Variavel TARGERT.
pisa15['TARGET'] = pisa15['PV1MATH'].apply(categoriza)
```

Figura 42: Categorização da variável Target.

Após serem aplicadas as mudanças preciso garantir que os dados não estejam com ruídos para que atrapalhe o treino dos meus algoritmos.

```
pisa15a = pisa15.drop(columns=['PV1MATH', 'PV1READ', 'PV1SCIE'])
```

```
pisa15a.fillna(0, inplace = True)
```

```
pisa15a.select_dtypes(include='object')
```

	CNT	OECD	ST04Q01	ST07Q01	ST07Q02	ST08Q01	ST09Q01	ST13Q01	ST15Q01	S
4743	United Arab Emirates	Non-OECD	Male	No, never	No, never	One or two times	One or two times	<ISCED level 3A>	Other (e.g. home duties, retired)	I
4744	United Arab Emirates	Non-OECD	Male	Yes, once	Yes, once	None	One or two times	<ISCED level 3A>	Other (e.g. home duties, retired)	I
4745	United Arab Emirates	Non-OECD	Male	No, never	No, never	None	None	0	Other (e.g. home duties, retired)	
4746	United Arab Emirates	Non-OECD	Male	Yes, once	No, never	None	None	<ISCED level 3A>	Other (e.g. home duties, retired)	
4747	United Arab Emirates	Non-OECD	Male	No, never	No, never	None	None	<ISCED level 3A>	Other (e.g. home duties, retired)	I

Figura 43: Verificação das variáveis.

Agora que temos todas a variáveis do tipo ‘object’, faremos a transformação para o tipo ‘category’.

```
#Transformando as colunas object em category.
for c in pisa15a.select_dtypes(include='object'):
    pisa15a[c] = pisa15a[c].astype('category').cat.codes
```

```
pisa15a.select_dtypes(include='float64')
```

	ST57Q01	ST57Q02	ST57Q03	ST57Q04	ST57Q05	ST57Q06	ST70Q01	ST70Q02	ST70Q
4743	5.572087	1.61926	0.849429	0.827374	1.171911	1.555714	4.39898	4.502827	
4744	5.572087	1.61926	0.849429	0.827374	1.171911	1.555714	4.39898	4.502827	
4745	5.572087	1.61926	0.849429	0.827374	1.171911	1.555714	8.00000	8.000000	
4746	5.572087	1.61926	0.849429	0.827374	1.171911	1.555714	8.00000	4.502827	
4747	5.572087	1.61926	0.849429	0.827374	1.171911	1.555714	4.39898	4.502827	
...	...	...	...	...	...	...	...	...	
480439	1.000000	1.00000	1.000000	1.000000	1.000000	1.000000	6.00000	6.000000	
480440	3.000000	3.00000	4.000000	2.000000	3.000000	5.000000	4.39898	4.502827	
480441	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	3.00000	3.000000	
480442	1.000000	1.00000	1.000000	1.000000	1.000000	1.000000	20.00000	10.000000	
480443	0.000000	1.61926	16.000000	0.000000	2.000000	3.000000	7.00000	7.000000	

314423 rows × 18 columns

Figura 44: Transformação das colunas Object em Category.

Agora que os dados foram transformados. Usarei a normalização Z-Score para os dados do tipo Float. A normalização Z-score é a pontuação padrão é o número de desvios padrão pelos quais o valor de uma pontuação bruta está acima ou abaixo do valor médio do que está sendo observado ou medido. Pontuações brutas acima da média têm pontuações padrão positivas, enquanto aquelas abaixo da média têm pontuações padrão negativas.

```
# normalizacao z-score
for c in pisa15a.select_dtypes(include='float64'):
    pisa15a[c] = (pisa15a[c] - pisa15a[c].mean()) / pisa15a[c].std()
```

```
#Variaveis numericas sem dados faltantes.
pisa15a.select_dtypes(include='int64').isnull().sum()
```

```
ST01Q01    0
ST03Q01    0
ST03Q02    0
IC05Q01    0
IC06Q01    0
IC07Q01    0
TARGET     0
dtype: int64
```

Figura 45: Normalização Z-Score.

Assim dessa forma o dataframe está pronto para ser utilizado para os algoritmos de classificação.

```
pisa15a.corr()
```

	CNT	OECD	ST01Q01	ST03Q01	ST03Q02	ST04Q01	ST07Q01	ST07
CNT	1.000000	0.064481	0.036363	1.118885e-03	0.151943	0.003568	0.001066	-0.049
OECD	0.064481	1.000000	0.038065	-1.171518e-02	0.111937	0.008494	-0.082195	-0.11
ST01Q01	0.036363	0.038065	1.000000	-2.670086e-02	0.017593	0.001646	0.011678	-0.01
ST03Q01	0.001119	-0.011715	-0.026701	1.000000e+00	-0.312455	-0.000828	0.006118	-0.00
ST03Q02	0.151943	0.111937	0.017593	-3.124545e-01	1.000000	0.000764	-0.014499	-0.02
ST04Q01	0.003568	0.008494	0.001646	-8.282529e-04	0.000764	1.000000	0.003223	0.01
ST07Q01	0.001066	-0.082195	0.011678	6.118007e-03	-0.014499	0.003223	1.000000	-0.02
ST07Q02	-0.049318	-0.113119	-0.010069	-8.075509e-04	-0.026061	0.018449	-0.025013	1.00

Figura 46: Dataframe Final.

Criação dos Testes e Trainos para os Algoritmos de ML

```
|: #from sklearn.model_selection import train_teste_split
from sklearn.utils import resample
from sklearn.metrics import accuracy_score, classification_report

X = pisa15a.drop("TARGET", axis = 1)
y = pisa15a["TARGET"]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, stratify = y, test_size=0.30, random_state = 42)
```

Figura 47: Criação do Treino e Teste para o modelo de Machine Learning.

Arvore de Decisão

```
:
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
target_tree = DecisionTreeClassifier(random_state=0)
target_tree = target_tree.fit(X_train, y_train)
print("Acuracia do modelo: ", target_tree.score(X_train, y_train))
Train_predict = target_tree.predict(X_test)
print("Acuracia da Previsão: ", accuracy_score(y_test,Train_predict))
print(classification_report(y_test, Train_predict))
```

```
Acuracia do modelo:  0.9994275225356208
Acuracia da Previsão:  0.7038811793017906
              precision    recall  f1-score   support

           0       0.71      0.71      0.71     48539
           1       0.69      0.70      0.70     45788

    accuracy                           0.70     94327
   macro avg       0.70      0.70      0.70     94327
weighted avg       0.70      0.70      0.70     94327
```

Figura 48: Modelo de Arvore de Decisão.

Regressão Logistica

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()
LR = LR.fit(X_train, y_train)
print("Acuracia do modelo: ", LR.score(X_train, y_train))
LR_predict = LR.predict(X_test)
print("Acuracia da Previsão: ", accuracy_score(y_test, LR_predict))
print(classification_report(y_test, LR_predict))
```

```
Acuracia do modelo:  0.7716496437917999
Acuracia da Previsão:  0.7692177213311141
              precision    recall  f1-score   support

           0       0.78      0.77      0.77     48539
           1       0.76      0.77      0.76     45788

    accuracy                           0.77     94327
   macro avg       0.77      0.77      0.77     94327
weighted avg       0.77      0.77      0.77     94327
```

Figura 49: Algoritmo de Regressão Logística.

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
NB = GaussianNB()
NB = NB.fit(X_train, y_train)
print("Acuracia do modelo: ", NB.score(X_train, y_train))
NB_predict = NB.predict(X_test)
print("Acuracia da Previsão: ", accuracy_score(y_test, NB_predict))
print(classification_report(y_test, NB_predict))
```

```
Acuracia do modelo:  0.6904259959290492
Acuracia da Previsão:  0.6878518345754662
              precision    recall  f1-score   support

           0       0.73      0.62      0.67     48539
           1       0.65      0.76      0.70     45788

    accuracy                           0.69     94327
   macro avg       0.69      0.69      0.69     94327
weighted avg       0.69      0.69      0.69     94327
```

Figura 50: Algoritmo de Naive Bayes.

Gradiente Descendente

```
from sklearn.linear_model import SGDClassifier
SGD = SGDClassifier()
SGD = SGD.fit(X_train, y_train)
print("Acuracia do modelo: ", SGD.score(X_train, y_train))
SGD_predict = SGD.predict(X_test)
print("Acuracia da Previsão: ", accuracy_score(y_test, SGD_predict))
print(classification_report(y_test, SGD_predict))
```

```
Acuracia do modelo:  0.7684146917708636
Acuracia da Previsão:  0.7656026376329153
              precision    recall  f1-score   support

           0       0.78      0.76      0.77     48539
           1       0.75      0.77      0.76     45788

    accuracy                           0.77     94327
   macro avg       0.77      0.77      0.77     94327
weighted avg       0.77      0.77      0.77     94327
```

Figura 51: Modelo de Gradiente Descendente

KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn = knn.fit(X_train, y_train)
print("Acuracia do modelo: ", knn.score(X_train, y_train))
knn_predict = knn.predict(X_test)
print("Acuracia da Previsão: ", accuracy_score(y_test ,knn_predict))
print(classification_report(y_test, knn_predict))
```

```
Acuracia do modelo:  0.8113505015993021
Acuracia da Previsão:  0.7358020503143321
              precision    recall  f1-score   support

           0       0.82      0.62      0.71     48539
           1       0.68      0.86      0.76     45788

    accuracy                           0.74     94327
   macro avg       0.75      0.74      0.73     94327
weighted avg       0.75      0.74      0.73     94327
```

Figura 52: Modelo de KNN

Random Forest

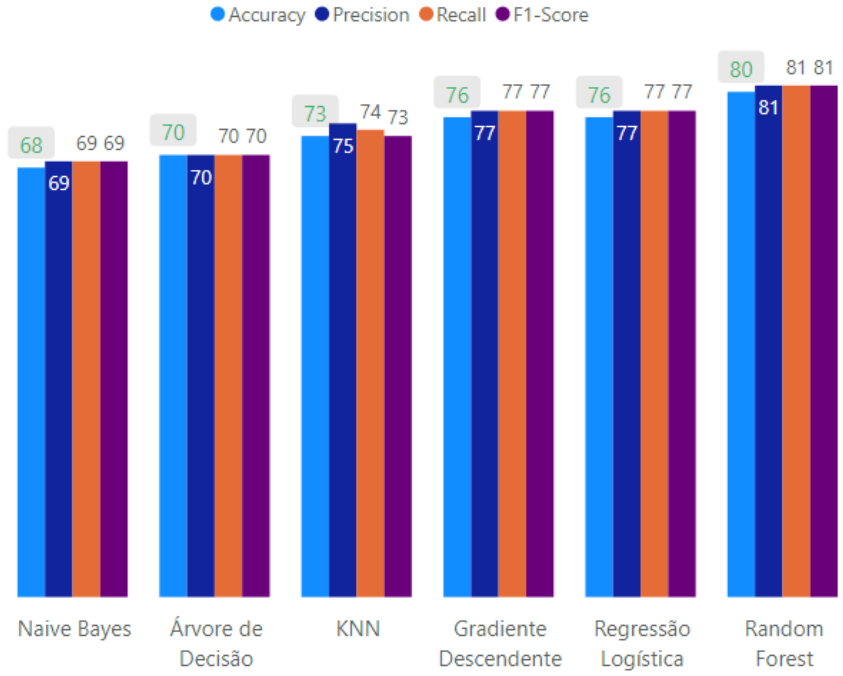
```
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier()
RF = RF.fit(X_train, y_train)
print("Acuracia do modelo: ", RF.score(X_train, y_train))
RF_predict = RF.predict(X_test)
print("Acuracia da Previsão: ", accuracy_score(y_test, RF_predict))
print(classification_report(y_test, RF_predict))
```

```
Acuracia do modelo:  0.9994229790636813
Acuracia da Previsão:  0.807563051936349
              precision    recall  f1-score   support

           0       0.82      0.80      0.81     48539
           1       0.80      0.81      0.80     45788

    accuracy                           0.81     94327
   macro avg       0.81      0.81      0.81     94327
weighted avg       0.81      0.81      0.81     94327
```

Figura 53: Modelo Random Forest

6. Interpretação dos Resultados

Algoritmo	Accuracy	Precision	Recall	F1-Score	Resultado
Árvore de Decisão	0.70	0.70	0.70	0.70	70% de acerto.
Regressão Logística	0.76	0.77	0.77	0.77	76% de acerto.
Naive Bayes	0.68	0.69	0.69	0.69	68% de acerto.
Gradiente Descendente	0.76	0.77	0.77	0.77	76% de acerto.
KNN	0.73	0.75	0.74	0.73	73% de acerto.
Random Forest	0.80	0.81	0.81	0.81	80% de acerto.



Analisando as informações da tabela e do gráfico, percebe-se que três Gradiente Descendente, Regressão Logística e Random Forest. Apresentando resultados razoáveis para todas as medidas de avaliação.

Fazendo uma análise de forma mais detalhada podemos ver a precisão dos resultados individuais, o valor gerado pela previsão foi de 80%, identificando alguns falsos positivos dentre os dados observados. A cada 100 previsões teríamos a chance de errar 20 delas alegando que alunos obtiveram a nota maior que a média global quando na verdade eles estavam tirando uma nota muito abaixo da média.

Esses resultados demonstrados que há uma grande chance de previsibilidade de resultado com os dados apresentados, apesar de se conseguir obter uma melhor acurácia organizando os atributos utilizados no treino e teste do algoritmo. Resumindo se todos os alunos seguirem

o mesmo critério de avaliação deste conjunto de dados, é possível prever, com elevado grau de acurácia se ele terá uma média acima dos demais alunos em matemática.

7. Apresentação dos Resultados

Como o melhor algoritmo de classificação foi o Random Forest, irei utiliza-lo mais uma vez para a verificação de quais foram os atributos presentes no dataframe que ele utilizou como parâmetro para definir o quão importante eles foram para a classificação.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler, MinMaxScaler

from matplotlib import rcParams
import warnings

warnings.filterwarnings("ignore")

# figure size in inches
rcParams["figure.figsize"] = 10, 6
```

Figura 54: Importação das bibliotecas.

Após a importação das bibliotecas necessárias para utilizar o Random Forest irei fazer um pré-processamento dos dados.

```
X = pisa15a.drop("TARGET", axis = 1)
y = pisa15a["TARGET"]
```

```
#Pre-processamento dos dados
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
#Dividindo o conjunto de dados em treino e teste
X_train, X_teste, y_train, y_test = train_test_split(
    X_scaled, y, stratify = y, test_size=0.30, random_state = 42)
```

```
#Criando um classificador
classificador = RandomForestClassifier(max_depth = 2, n_estimators = 100)
```

```
#Treinando o modelo usando os dados de treino
classificador.fit(X_train, y_train)
```

```
RandomForestClassifier(max_depth=2)
```

```
#predicao dos dados de treino
y_pred = classificador.predict(X_teste)
```

Figura 55: Pré-processamento dos dados para a utilização do algoritmo Random Forest.

Após o pré-processamento dos dados, podemos ver qual a acurácia e sua matriz de confusão.

```
#calculando a acuracia do modelo
print("Acuracia:", accuracy_score(y_test, y_pred))
```

```
Acuracia: 0.6900251253617734
```

```
#Valores utilizados na variavel
y.value_counts()
```

```
0    161798
1    152625
Name: TARGET, dtype: int64
```

```
print('Matriz de Confusao: ', confusion_matrix(y_test, y_pred))
```

```
Matriz de Confusao:  [[32275 16264]
 [12975 32813]]
```

Figura 56: Acurácia e Matriz de Confusão.

Agora usando o atributo “Feature_Importances_” da Random Forest, podemos descobrir os atributos mais importantes usados para a classificação. Como o dataset é muito grande, utilizarei apenas uma amostra de atributos.

```
pisa_feature_importances = pd.DataFrame(
    {"feature": list(X.columns),
     "importance": classificador.feature_importances_}).sort_values
    ("importance", ascending=False)
```

```
#Os recursos mais importantes
pisa_feature_importances_25 = pisa_feature_importances.head(15)
pisa_feature_importances_25
```

	feature	importance
16	ST26Q06	0.123890
20	ST28Q01	0.107711
17	ST27Q03	0.087417
98	EC03Q01	0.068090
101	EC03Q09	0.059948
99	EC03Q03	0.046890
18	ST27Q04	0.046365
100	EC03Q04	0.045447
10	ST13Q01	0.043317
12	ST17Q01	0.034454
15	ST26Q04	0.029360
77	IC01Q04	0.025716
89	IC08Q11	0.021289
33	ST57Q02	0.020127
83	IC03Q01	0.016745

Figura 57: Lista de atributos usados pela Random Forest.

Esse foram os 15 atributos mais importantes escolhidos pelo nosso algoritmo, plotando um gráfico, a visibilidade dessas informações fica bem mais perceptível.

```
import seaborn as sns
sns.barplot(x=pisa_feature_importances_25.feature, y=pisa_feature_importances_25

plt.xlabel("Feature Importance Score")
plt.ylabel("Features")
plt.title("Visualizing Important Features")
plt.xticks(rotation=45, horizontalalignment="right", fontweight="light", fontsiz

plt.show()
```

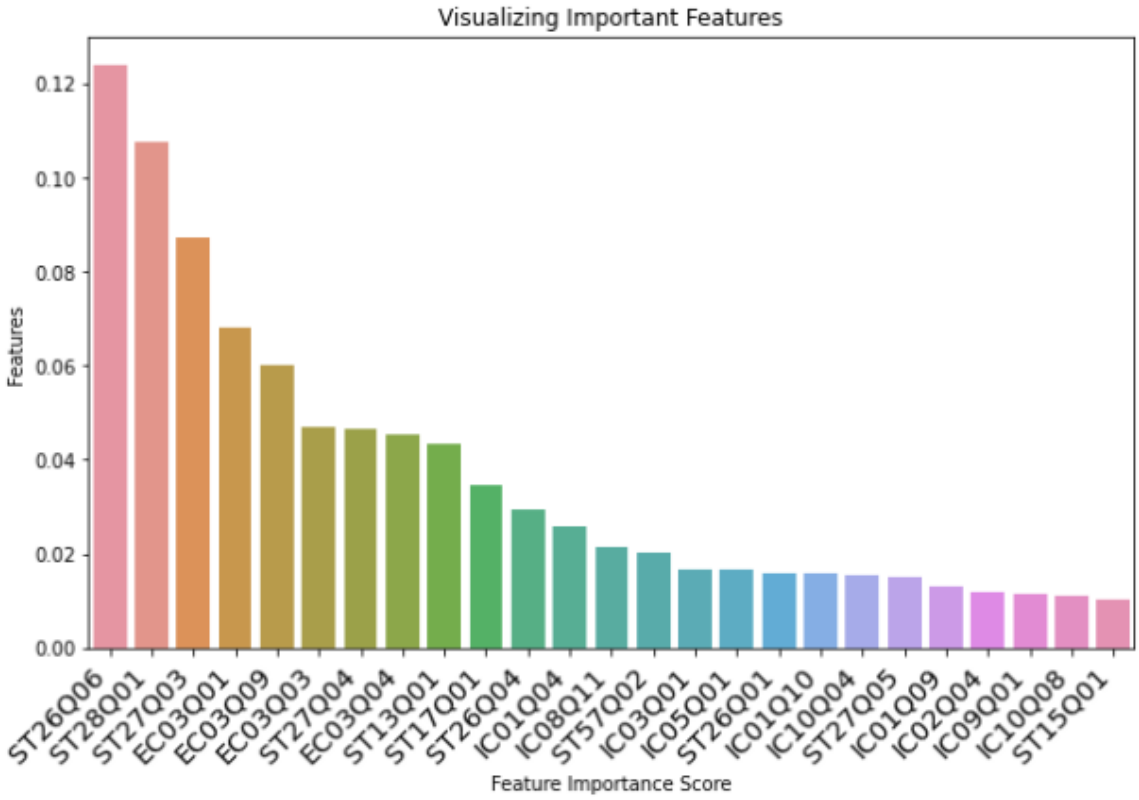


Figura 58: Gráfico dos Atributos mais Importantes.

Como utilizamos um algoritmo de aprendizagem supervisionado em que uma propriedade (rótulo) está disponível para um determinado conjunto de dados (conjunto de treinamento). Todas as entradas e saídas são conhecidas, mas precisam ser previstas para outras instancias. Desse modo os dados passados para o sistema de aprendizagem, que tem como função descobrir caminhos e ajustar seu próprio modelo, dessa forma o algoritmo de Random Forest, atribuiu mais pesos aos seguintes atributos.

```
pisa_desc['ST26Q06'], pisa_desc['ST28Q01'], pisa_desc['EC03Q09'], pisa_desc['EC0
```

```
('Possessions - Internet',
 'How many books at home',
 'Future Orientation - web search - ISCED 3-5 (apos 9o ano)',
 'Future Orientation - web search - ISCED 3-5 (apos 9o ano)',
 'Possessions - Internet',
 'Future Orientation - Career advisor at school',
 'How many - computers',
 'Future Orientation - Job fair',
 'Possessions - computer',
 'At Home - Internet connection',
 'Father Highest Schooling',
 'Mother Highest Schooling',
 'Future Orientation - Internship',
 'Out-of-School Study Time - Guided Homework',
 'How many - cars',
 'First use of computers',
 'How many - rooms bath or shower')
```

Figura 59: Nomes das Variáveis.

A conclusão deste trabalho se deu a partir da percepção de que as notas dos alunos são afetadas pelo ambiente externo, podemos ver que alguns atributos como a conexão com a internet em casa podem de fato afetar o aprendizado de crianças e adolescente. As features mais importante que posso citar neste trabalho são: Acesso à Internet, computadores em casa, quantos livros tem em casa, Orientação de futuro (Relacionado ao profissional), Feiras de trabalhos (Feiras estudantis focadas em demonstrar como funciona as carreiras de trabalho), estudos em casa e nível de instrução dos Pais.

Alguns desafios socioeconômicos como a falta de igualdade de oportunidades entre as diversas camadas da sociedade fazem com que nem todos tenham acesso à escola formal, faculdade ou qualquer outro nível de formação. Também não é igualitária a aprendizagem, uma vez que há déficit na cadeia de ensino para pessoas com deficiências físicas e intelectuais, assim como as que têm qualquer dificuldade. A evasão escolar enquanto o início do Ensino Fundamental tem boa adesão, as demais fases da educação básica sofrem com a chamada evasão escolar. A taxa de analfabetismo também surge como um dos obstáculos enfrentados pelo sistema educacional. Falta de investimentos é consensual que há déficits no que diz respeito à estrutura das escolas já existente, à quantidade de instituições, sobretudo nas periferias, à formação dos professores, aos equipamentos de tecnologia, entre tantos outros. Em resumo, se não existirem ações concretas que mudem a qualidade da educação brasileira, para que toda a população possa ter uma melhor distribuição de ensino e consequentemente

de renda, o resultado do rank brasileiro e previsível tendendo a cair mais e mais no ranking de educação mundial.

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: youtube.com/...

Link para o repositório: github.com/...

REFERÊNCIAS

[1] O que é o PISA e quem é responsável pelo seu desenvolvimento? Disponível em: < https://www.politize.com.br/pisa-educacao/> acesso em: 17/06/2020.

[2] Programa Internacional de Avaliação de Estudantes (Pisa). Disponível em: <https://www.gov.br/inep/pt-br/areas-de-atuacao/avaliacao-e-exames-educacionais/pisa> acesso em: 17/06/2020.

[3] Programa Internacional de Avaliação de Alunos. Disponível em: < https://pt.wikipedia.org/wiki/Programa_Internacional_de_Avaliação_de_Alunos> acesso em: 15/06/2021

[4] Relatório Nacional PISA 2012 Resultados brasileiros. Disponível em: <https://download.inep.gov.br/acoes_internacionais/pisa/resultados/2014/relatorio_nacional_pisa_2012_resultados_brasileiros.pdf > acesso em: 15/06/2021

[5] PYTHON. Disponível em: <https://www.python.org/>.

[6] JUPYTER. Disponível em: <https://jupyter.org/>.

[7] SEABORN. Disponível em: <https://seaborn.pydata.org/>

APÊNDICE

Programação/Scripts

```
#Importação das bibliotecas
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm

import sklearn
import sys
import os

import matplotlib.pyplot as plt
from matplotlib import cm, colors
%matplotlib inline

import warnings

warnings.filterwarnings("ignore")


#Função para acrescentar cores nos fundos dos dataframes.
def global_background_gradient(s, m, M, cmap='PuBu', low=0, high=0):
   rng = M - m
   norm = colors.Normalize(m - (rng * low),
                           M + (rng * high))
   normed = norm(s.values)
   c = [colors.rgb2hex(x) for x in plt.cm.get_cmap(cmap)(normed)]
   return ['background-color: %s' % color for color in c]

#Verificando as versões das bibliotecas utilizadas no projeto
print("Python", sys.version)
print("--------")
print("Pandas:", pd.__version__)
print("Numpy:", np.__version__)
print("SKLearn:", sklearn.__version__)

#Criação dos dicionários dos países utilizados no projeto
paises_escolhidos  =  { Paises: ['Bulgaria', 'United Arab Emirates', 'Liechtenstein', 'Singapore',
'Mexico',  'China',  'Indonesia',  'Turkey',  'Peru',  'United  States',  'Czech  Republic',  'Latvia',
'France',  'United  Kingdom',  'Lithuania','Finland',  'Switzerland',  'Canada',  'Brazil','Uruguay',
'Slovak  Republic',  'Hungary',  'Portugal',  'United  States',  'Greece',  'Australia',  'South  Korea',
'Russia',  'Japan',  'Taiwan',  'Hungary',  'Italy',  'Colombia','Chile',  'South  Korea',  'Denmark',
'Sweden']}

#Leitura do CSV
```

```
pisa0 = pd.read_csv('data/data/pisa2012.csv')

#Leitura do 2 CSV
pisa_dict = pd.read_csv('data/pisa2012_vardescs.csv')
with pd.option_context('display.max_rows', None):   display(pisa_dict)

#Verificando quais os nomes dos atributos do 2 CSV
pisa_desc = dict(zip(list(pisa_dict['varname'].values), list(pisa_dict['desc'].values)))

#Atribuindo valor ao dicionário dos países.
PS = ‘paises_escolhidos’
countries = paises_escolhidos’ [Paises]

pisa = pisa0[pisa0.CNT.isin(countries)].copy()

#Utilizando a função info() para saber exatamente as dimensoes de colunas e os tipos de dados do meu dataframe
pisa.info()

#Utilizando a função shape() para saber quantas linhas e colunas eu vou utilizar
Pisa.shape

#O função describe() me descreve um conjunto de estatísticas descritivas.
Pisa.describe()

#Filtrando os dados pleo o tipo, neste caso usando o tipo float
Pisa.select_dtypes(include=[‘float’])

#Filtrando os dados pelo o tipo, neste caso usando o tipo integer
Pisa.select_dtypes(include=[‘integer])

#Usando a função isnull() podemos ver qual são os dados que estão ausentes no dataset, quando se estiver false significa que tem dados preenchidos e quando estiver em True significa que tem dados faltantes.
Pisa.isnull()

#Usando o sum() apos o isnull() podemos somar a quantidade de dados faltantes conforme o atributo.
Pisa.isnull().sum()

#Criando a média das variáveis
media_values = {'ST57Q01': pisa.ST57Q01.mean(),
          'ST57Q02': pisa.ST57Q01.mean(),
          'ST57Q03': pisa.ST57Q01.mean(),
          'ST57Q04': pisa.ST57Q01.mean(),
          'ST57Q05': pisa.ST57Q01.mean(),
          'ST57Q06': pisa.ST57Q01.mean(),
```

```
            'ST70Q01': pisa.ST70Q01.mean(),
            'ST70Q02': pisa.ST70Q02.mean(),
            'ST70Q01': pisa.ST70Q03.mean(),
            'ST71Q01': pisa.ST70Q03.mean(),
            'ST101Q02': pisa.ST101Q02.mean(),
            'ST101Q03': pisa.ST101Q03.mean(),
            'ST101Q05': pisa.ST101Q05.mean(),
            'ST104Q01': pisa.ST104Q01.mean(),
             'ST104Q04': pisa.ST104Q04.mean(),
            'ST104Q05': pisa.ST104Q05.mean(),
            'ST104Q06': pisa.ST104Q06.mean()
           }
media_values

#Criando a moda das variaveis
moda_values = {'IC05Q01': pisa.IC05Q01.mode(),
            'IC06Q01': pisa.IC06Q01.mode(),
            'IC07Q01': pisa.IC06Q01.mode(),
            'ST01Q01': pisa.ST01Q01.mode(),
            'ST03Q01': pisa.ST03Q01.mode(),
            'ST03Q02': pisa.ST03Q02.mode(),
           }
moda_values

#Criando uma função de loop para substituir todos os dados faltantes do tipo real pela media.
for r in media_values:
   if r != 'PV1MATH' or 'PV1READ' or 'PV1SCIE':
      pisa.update(pisa[r].fillna(pisa[r].mean()))

#Criando uma função de loop para substituir todos os dados faltantes do tipo inteiro pela moda.
for i in moda_values:
   if i != 'PV1MATH' or 'PV1READ' or 'PV1SCIE':
      pisa.update(pisa[i].fillna(pisa[i].mode()))

#Verificando se ainda tem dados ausentes no dataset
pisa.ST57Q01.isnull()


#Criando um dicionario das medias dos alunos por pais
dic_MRS = pisa.groupby('CNT').agg({'PV1MATH': 'mean',
                   'PV1READ': 'mean',
                   'PV1SCIE': 'mean'})

#Criando uma função para retornar a média das matérias PV1(MATH, READ e SCIE) junto aos paises.
def soma_media(frame, new_col_name, dic_MRS):
```

```
    frame[new_col_name] = frame[dic_MRS].astype(float).sum(axis=1)/3
    return(frame)

result = soma_media(dic_MRS,'Media_CNT', ['PV1MATH', 'PV1READ', 'PV1SCIE'])

import operator
sorted(result['Media_CNT'].items(), key=operator.itemgetter(1))

#5 Paises de melhor Media geral
result['Media_CNT'].sort_values(ascending = False).head(5)

#5 Paises com a pior Media geral
result['Media_CNT'].sort_values(ascending = True).head(5)

#Quais os paises membro e nao membros do OCDE?
member_oecd = pisa.groupby(['CNT', 'OECD']).agg({'PV1MATH': 'mean',
                          'PV1READ': 'mean',
                          'PV1SCIE': 'mean'})

#Criação de uma função que retornar a média das provas por pais pertecendo ou não ao
OECD.
def member_oecd1(frame, new_col_name, member_oecd):
    frame[new_col_name] = frame[member_oecd].astype(float).sum(axis=1)/3
    return(frame)

result1 = member_oecd1(member_oecd,'Media_CNT', ['PV1MATH', 'PV1READ', 'PV1SCIE'])

pisa3 = pd.DataFrame(result1)

import operator
sorted(result1['Media_CNT'].items(), key=operator.itemgetter(1))

#Melhores paises pertecentes ao OECD
pisa3.sort_values('OECD', ascending = False).head(2)

#Melhores paises não pertecentes ao OECD
pisa3.sort_values('Media_CNT', ascending = False).head(2)

#Histograma das notas PV1MATH
fig, ax = plt.subplots()

ax.hist('PV1MATH', data=pisa, density=True, bins=30)
ax.set_title("Histograma da notas PV1MATH")
ax.set_ylabel("Quantidade de paises")
ax.set_xlabel("Media das notas")

plt.tight_layout()
```

```
#Histograma das notas PV1READ
fig, ax = plt.subplots()

ax.hist('PV1READ', data=pisa, density=True, bins=30)
ax.set_title("Histograma da notas PV1READ")
ax.set_ylabel("Quantidade de paises")
ax.set_xlabel("Media das notas")

plt.tight_layout()

#Histograma das notas PV1SCIE
fig, ax = plt.subplots()

ax.hist('PV1SCIE', data=pisa, density=True, bins=30)
ax.set_title("Histograma da notas PV1SCIE")
ax.set_ylabel("Quantidade de paises")
ax.set_xlabel("Media das notas")

plt.tight_layout()

#histograma com todos os paises que fazem parte do OECD
pisa2 = pisa.copy()

pisaOECD = pisa[pisa['OECD']=='OECD']

pisaOECD.PV1MATH.hist(bins = 30, density = True, alpha= 0.5)
pisaOECD.PV1READ.hist(bins = 30, density = True, alpha= 0.5)
pisaOECD.PV1SCIE.hist(bins = 30, density = True, alpha= 0.5)

# histograma com todos os paises que não fazem parte do OECD
pisaNONOECD = pisa[pisa['OECD']=='NON-OECD']

pisaOECD.PV1MATH.hist(bins = 30, density = True, alpha=0.5)
pisaOECD.PV1READ.hist(bins = 30, density = True, alpha= 0.5)
pisaOECD.PV1SCIE.hist(bins = 30, density = True, alpha= 0.5)


#Agrupamento dos países por sexo e nota média das 3 areas de interesse.
pisa5 = pd.DataFrame(pisa[['CNT', 'ST04Q01', 'PV1MATH', 'PV1READ', 'PV1SCIE']])


#Agora um histograma levando em consideração a média de matemática entre homens e
mulheres
pisa6Male = pisa[pisa['ST04Q01']=='Male']
pisa6Female = pisa[pisa['ST04Q01']=='Female']
```

```
pisa6Male.PV1MATH.hist(bins = 30, density = True, alpha=0.5, label = 'Homem')
pisa6Female.PV1MATH.hist(bins = 30, density = True, alpha=0.5, label = 'Mulher')


#Agora um histograma levando em consideração a média de Leitura entre homens e mulheres
pisa6Male.PV1READ.hist(bins = 30, density = True, alpha= 0.5, label = 'Homem')
pisa6Female.PV1READ.hist(bins = 30, density = True, alpha=0.5, label = 'Mulher')

plt.legend()

#Agora um histograma levando em consideração a média de Ciências entre homens e mulheres
pisa6Male.PV1SCIE.hist(bins = 30, density = True, alpha= 0.5, label = 'Homem')
pisa6Female.PV1SCIE.hist(bins = 30, density = True, alpha=0.5, label = 'Mulher')

plt.legend()

#Agrupamento por sexo te todos os países pela medias das 3 áreas de interesse.
pisa.groupby(['CNT', 'ST04Q01']).agg({'PV1MATH': 'mean',
                        'PV1READ': 'mean',
                        'PV1SCIE': 'mean'}).head(60)

#Criação de uma função para classificar o grau de instrução do pai e da mãe.
def grau_instrucao_agregado_mae(grauMae):
   if grauMae == "She did not complete <ISCED level 1> ":
      valueM = 0
   elif grauMae == "<ISCED level 1> ":
      valueM = 1
   elif grauMae == "<ISCED level 2> ":
      valueM = 2
   elif grauMae == "<ISCED level 3B, 3C> ":
      valueM = 3
   else:
      valueM = 4
   return valueM

def grau_instrucao_agregado_pai(grauPai):
   if grauPai == "He did not complete <ISCED level 1> ":
      valueP = 0
   elif grauPai == "<ISCED level 1> ":
      valueP = 1
   elif grauPai == "<ISCED level 2> ":
      valueP = 2
   elif grauPai == "<ISCED level 3B, 3C> ":
      valueP = 3
   else:
      valueP = 4
```

```
    return valueP

#Lista de países com maior grau de instrução
pisa6['SomaPais'] = pisa.ST13Q01.apply(grau_instrucao_agregado_mae) +
pisa.ST13Q01.apply(grau_instrucao_agregado_pai)
 pisa6['SomaPais'].value_counts()

#Lista de países pertencentes a OECD com maiores graus de instrução.
pisa6OECD = pisa6[pisa6['OECD']=='OECD']
Pais6OECD = list(pisa6OECD['CNT'].drop_duplicates())
pais_ig_OECD = pd.DataFrame(columns = ['CNT', 'igPais'])

for i in Pais6OECD:
    pais_ig_OECD.loc[len(pais_ig_OECD)] = [i,
pisa6OECD[pisa6OECD['CNT']==i].SomaPais.sum()]
pais_ig_OECD = pais_ig_OECD.drop_duplicates()
pais_ig_OECD.sort_values(by=['igPais'], ascending = False)

#Lista de países não pertencentes a OECD com maiores graus de instrução.
pisa6NONOECD = pisa6[pisa6['OECD']=='Non-OECD']
Pais6NONOECD = list(pisa6NONOECD['CNT'].drop_duplicates())
pais_ig_NONOECD = pd.DataFrame(columns = ['CNT', 'igPais'])
for i in Pais6NONOECD:
    pais_ig_NONOECD.loc[len(pais_ig_NONOECD)] = [i,
pisa6NONOECD[pisa6NONOECD['CNT']==i].SomaPais.sum()]
pais_ig_NONOECD = pais_ig_NONOECD.drop_duplicates()
pais_ig_NONOECD.sort_values(by=['igPais'], ascending = False)

#Valores absolutos do grau de instrução dos pais.
pisa6['SomaPais'].value_counts()

#Agrupamento das medias em matemática, leitura e ciências pelo grau de instrução dos pais.
pisa6.groupby(['SomaPais']).agg({'PV1MATH': 'mean',
                        'PV1READ': 'mean',
                        'PV1SCIE': 'mean'})

#Criando um dataset sem duplicatas de dados para criar um gráfico
pisa7 = pisa6[pisa6.CNT.isin(list(pisa6['CNT'].drop_duplicates()))]
group_pais_dep = pd.crosstab(pisa6.CNT, pisa7.SomaPais)

#Gráfico de barras com todos os países divididos pelo instrução de educação dos pais.
fig = group_pais_dep.plot.bar(figsize=(18,12))
fig.figure.show()

#Agrupamento de dados por quantidade de livros.
pisa.groupby('ST28Q01').agg({'PV1MATH': 'mean',
                        'PV1READ': 'mean',
```

```
                        'PV1SCIE': 'mean'})

#Criando um dataset sem duplicatas de dados para criar um gráfico.
pisa = pisa[pisa.CNT.isin(list(pisa['CNT'].drop_duplicates()))]
group_livros_pais = pd.crosstab(pisa.CNT, pisa.ST28Q01)

#Gráfico de barras com todos os países marcando quantos livros possuem.
fig = group_livros_pais.plot.bar(figsize=(15,8))
fig.figure.show()


#Criação da features tempo de estudo para verificação de quantas horas de estudo cada aluno possui.
pisa13['tempo_estudo']= pd.qcut(pisa.ST57Q01.drop_duplicates(), q=12,
      labels = ['1h semanal','3h semanais','5h semanais','7h semanais',
            '9h semanais','11h semanais','13h semanais','16h semanais','18h semanais',
            '20h semanais','25h semanais','mais de 25h semanais'])

#Agrupamento por hora semanal de estudo.
pisa13.groupby('tempo_estudo').agg({'PV1MATH': 'mean',
                        'PV1READ': 'mean',
                        'PV1SCIE': 'mean'})


#Copia do dataset
pisa15 = pisa.copy()

#Criação da variável media em matematica
media_math = pisa15['PV1MATH'].mean()
media_math

#com valor 1 para alunos com nota maior que a média global em Matemática; 0, em caso contrário.
def categoriza(s):
   if s < media_math:
      return 0
   else:
      return 1

#Variavel TARGERT.
pisa15['TARGET'] = pisa15['PV1MATH'].apply(categoriza)

#Excluindo as variáveis.
pisa15a = pisa15.drop(columns=['PV1MATH', 'PV1READ', 'PV1SCIE'])

#Preenchendo valores faltosos com 0.
pisa15a.fillna(0, inplace = True)
```

```
#Selecionados as features com o tipo Object.
pisa15a.select_dtypes(include='object')

#Transformando as colunas object em category.
for c in pisa15a.select_dtypes(include='object'):
   pisa15a[c] = pisa15a[c].astype('category').cat.codes

pisa15a.select_dtypes(include='float64')

# normlizacao z-score
for c in pisa15a.select_dtypes(include='float64'):
   pisa15a[c] = (pisa15a[c] - pisa15a[c].mean()) / pisa15a[c].std()

#Variaveis numericas sem dados faltantes.
pisa15a.select_dtypes(include='int64').isnull().sum()

#Função para atribuir valores as colunas com dados omissos.
def drop_nan(df, target):
   ig = 0
   tipo = df.dtypes.to_dict()
   for c in tqdm(df.columns):
      if (c not in target) and (df[c].isnull().any()):
         if tipo[c] == 'int64':
            df[c] = df[c].fillna(df[c].mode().values[0])
         elif tipo[c] == 'float64':
            df[c] = df[c].fillna(df[c].mean())
         else:
            ig = ig + 1
   print('Out from %d columns, %d had the NaNs replaced'%(len(df.columns), len(df.columns)-
ig))

pisa15a.corr()

#Importação das bibliotecas para utilizar o Random Forest
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler, MinMaxScaler

from matplotlib import rcParams
import warnings

warnings.filterwarnings("ignore")

# figure size in inches
```

```
rcParams["figure.figsize"] = 10, 6

#Dividindo as variáveis que serão utilizadas com a variável alvo.
X = pisa15a[[c for c in pisa15a.columns if c != 'TARGET']]
y = pisa15a['TARGET']

#Pre-processamento dos dados
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

#Dividindo o conjunto de dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(
   X_scaled, y, stratify = y, test_size=0.3, random_state = 42)

#Criando um classificador
classificador = RandomForestClassifier(max_depth = 2, n_estimators = 100)

#Treinando o modelo usando os dados de treino
classificador.fit(X_train, y_train)

#predicao dos dados de treino
y_pred = classificador.predict(X_test)

#calculando a acuracia do modelo
print("Acuracia:", accuracy_score(y_test, y_pred))

#Valores utilizados na variavel
y.value_counts()

#Matriz de Confusão.
print('Matriz de Confusao: ', confusion_matrix(y_test, y_pred))


#Criação de um Dataframe com as features que foram utilizadas ordenadas com o peso de importância.
pisa_feature_importances = pd.DataFrame( {"feature": list(X.columns), "importance": classificador.feature_importances_}).sort_values ("importance", ascending=False)

#Um pequeno dataframe com as features mais importantes.
pisa_feature_importances_25 = pisa_feature_importances.head(15)
pisa_feature_importances_25

#Um relatório com as features selecionadas anteriormente.
print('Relatorio de feature_importance:', pisa_feature_importances.head(15))

#Mostrando um gráfico de barras com as Features mais importantes.
import seaborn as sns
```

```
sns.barplot(x=pisa_feature_importances_25.feature,
y=pisa_feature_importances_25.importance)

plt.xlabel("Feature Importance Score")
plt.ylabel("Features")
plt.title("Visualizing Important Features")
plt.xticks(rotation=45, horizontalalignment="right", fontweight="light", fontsize="x-large")

plt.show()

#Mostrando quais Features tiveram maiore relevância no treinamento do modelo.
pisa_desc['ST26Q06'], pisa_desc['ST28Q01'], pisa_desc['EC03Q09'], pisa_desc['EC03Q09'],
pisa_desc['ST26Q06'], pisa_desc['EC03Q04'], pisa_desc['ST27Q03'],
pisa_desc['EC03Q03'],pisa_desc['ST26Q04'], pisa_desc['IC01Q04'], pisa_desc['ST17Q01'],
pisa_desc['ST13Q01'],pisa_desc['EC03Q01'], pisa_desc['ST57Q02'], pisa_desc['ST27Q04'],
pisa_desc['IC03Q01'], pisa_desc['ST27Q05']


#Criação dos testes e treinos para os algoritmos de Machine Learning.

#from sklearn.model_selection import train_teste_split
from sklearn.utils import resample
from sklearn.metrics import accuracy_score, classification_report

X = pisa15a.drop("TARGET", axis = 1)
y = pisa15a["TARGET"]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
   X_scaled, y, stratify = y, test_size=0.30, random_state = 42)


#Árvore de Decisão
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
target_tree = DecisionTreeClassifier(random_state=0)
target_tree = target_tree.fit(X_train, y_train)
print("Acuracia do modelo: ", target_tree.score(X_train, y_train))
Train_predict = target_tree.predict(X_test)
print("Acuracia da Previsão: ", accuracy_score(y_test,Train_predict))
print(classification_report(y_test, Train_predict))

#Regressão Logística
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()
LR = LR.fit(X_train, y_train)
```

```
print("Acuracia do modelo: ", LR.score(X_train, y_train))
LR_predict = LR.predict(X_test)
print("Acuracia da Previsão: ", accuracy_score(y_test, LR_predict))
print(classification_report(y_test, LR_predict))

#Naive Bayes
from sklearn.naive_bayes import GaussianNB
NB = GaussianNB()
NB = NB.fit(X_train, y_train)
print("Acuracia do modelo: ", NB.score(X_train, y_train))
NB_predict = NB.predict(X_test)
print("Acuracia da Previsão: ", accuracy_score(y_test, NB_predict))
print(classification_report(y_test, NB_predict))

#Gradiente Descendente
from sklearn.linear_model import SGDClassifier
SGD = SGDClassifier()
SGD = SGD.fit(X_train, y_train)
print("Acuracia do modelo: ", SGD.score(X_train, y_train))
SGD_predict = SGD.predict(X_test)
print("Acuracia da Previsão: ", accuracy_score(y_test, SGD_predict))
print(classification_report(y_test, SGD_predict))

#K-Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn = knn.fit(X_train, y_train)
print("Acuracia do modelo: ", knn.score(X_train, y_train))
knn_predict = knn.predict(X_test)
print("Acuracia da Previsão: ", accuracy_score(y_test ,knn_predict))
print(classification_report(y_test, knn_predict))

#Random Forest
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier()
RF = RF.fit(X_train, y_train)
print("Acuracia do modelo: ", RF.score(X_train, y_train))
RF_predict = RF.predict(X_test)
print("Acuracia da Previsão: ", accuracy_score(y_test, RF_predict))
print(classification_report(y_test, RF_predict))
```