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

Alexandre Cordeiro

Aprendizado de máquina para gestão de projetos de software

Belo Horizonte
2021

Alexandre Cordeiro

Aprendizado de máquina para gestão de projetos de software

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

1.1. Contextualização

Na Receita Federal, a distribuição de recursos para desenvolvimento e manutenção de sistemas é feita a partir do rateio, entre suas diversas coordenações, do montante de capital disponível.

Esse rateio toma como base as demandas solicitadas aos prestadores de serviço e iniciadas no ano anterior ao corrente. Não são analisadas possíveis instabilidades econômicas que possam ter impactado em alguns períodos.

O valor gasto com desenvolvimento e manutenção de sistemas é aferido através da Análise de Pontos de Função (APF). Que é uma métrica internacional de software, padronizada com objetivo principal de descobrir o tamanho de um software. Ela tem como foco principal a visão do usuário, ou seja, somente são contados os requisitos funcionais (requisitos de negócio) e, por isso, não leva em consideração a linguagem de programação.

Essa é a métrica utilizada pela Receita Federal para saber o tamanho de cada demanda e/ou funcionalidade e assim, calcular seu prazo e custo. Dessa forma é possível determinar as cotas de cada coordenação no rateio do montante disponível para desenvolvimento e manutenção de sistemas.

Sendo assim, não é feito um planejamento de longo prazo. É feito um planejamento anual que vai sofrendo adaptações no decorrer do ano.

Apesar de haver um planejamento em relação às demandas solicitadas, há incertezas em relação às demandas entregues. Por estarem sujeitas a reavaliações de prioridade, embora seja possível medir quanto uma demanda custará, seu prazo de entrega pode sofrer alterações.

Finalmente, a fim de ter previsibilidade de longo prazo, o ideal seria que o gestor público pudesse realizar projeções das entregas anuais, com base no histórico de entregas, mesmo estando sujeito a instabilidades.

1.2. O problema proposto

Para se ter uma visão mais abrangente do problema e da solução, primeiro devemos responder algumas questões.

1.2.1 Por que esse problema é importante?

Uma gestão de recursos mais eficiente gera diminuição dos custos e prazos de atendimento.

1.2.2 De quem são os dados analisados?

Receita Federal do Brasil.

1.2.3 Quais os objetivos com essa análise?

Prever tendência de aumento/redução de solicitações/orçamento;

Prever capacidade de atendimento do prestador de serviço;

Prever tendência de longo prazo no atendimento de demandas.

1.2.4 O que iremos analisar?

Os dados de demandas de desenvolvimento e manutenção de sistemas.

1.2.5 Trata dos aspectos geográficos e logísticos de sua análise?

Os dados são nacionais, abordam todos os estados e o Distrito Federal.

1.2.6 Qual o período está sendo analisado?

Estão sendo analisados dados de 2005 a 2020.

2. Coleta de Dados

2.1 Onde os dados foram obtidos?

Receita Federal do Brasil;

Sistema SGPTI;

Base de dados de demandas.

2.2 Data de coleta dos coletados

Dia 16/04/2021.

2.3 Origem

https://sgpti.receita.fazenda/

2.4 Formato

Arquivos de texto, tipo “CSV”.

2.5 Estrutura

Dataset	Nome da coluna/campo	Descrição	Tipo
1	Esforço PF	Esforço calculado em pontos de função (PF) para construção da demanda	float64
	Ano	Ano de abertura da demanda	int64
2	Esforço PF	Esforço calculado em pontos de função (PF) para construção da demanda	float64
	Ano	Ano de abertura da demanda	int64
	Data de Aprovação do Titular	Dia de início da demanda	object
	Data Assinatura Anexo 4/5 Prest. Serv	Dia considerado como a data de entrega da demanda construída. Nesta data, a demanda já foi construída e homologada. Sendo assim, está disponível para implantação no ambiente produtivo.	object
	Esforço PF	Esforço calculado em pontos de função (PF) para construção da demanda	float64

3	Ano	Ano de abertura da demanda	int64
	Data de Aprovação do Titular	Dia de início da demanda.	object
	Data Assinatura Anexo 4/5 Prest. Serv	Dia considerado como a data de entrega da demanda construída. Nesta data, a demanda já foi construída e homologada. Sendo assim, está disponível para implantação no ambiente produtivo.	object
	ANO_ASSIN	Dia considerado como a data de entrega da demanda construída. Nesta data, a demanda já foi construída e homologada. Sendo assim, está disponível para implantação no ambiente produtivo.	int64
	TEMPO_ENTREGA	Tempo, em dias, gasto desde a aprovação do titular até a entrega, pelo prestador de serviço, da demanda construída.	int64

2.6 Relacionamento entre os datasets

Data-set	**Nome do arquivo**	**Descrição**
1	extracao_sgpti_01.csv	Conjunto mais abrangente de demandas Foco no ano de abertura e no esforço calculado Comporta todas as demandas solicitadas pelos usuá-rios Inclui as demandas canceladas e as não terminadas
2	extracao_datas_01.csv	Conjunto com informações mais abrangentes Contém dados brutos que serão filtrados para geração do Datset3
3	extracao_sgpti_01.csv	Conjunto mais restrito de demandas Só apresenta demandas efetivamente construídas

		Foco nas datas de início e término e no tempo de construção das demandas

3. Processamento/Tratamento de Dados

3.1 Extração dos Datasets

Foram extraídos dados da base de demandas da Receita Federal, através do sistema SGPTI.

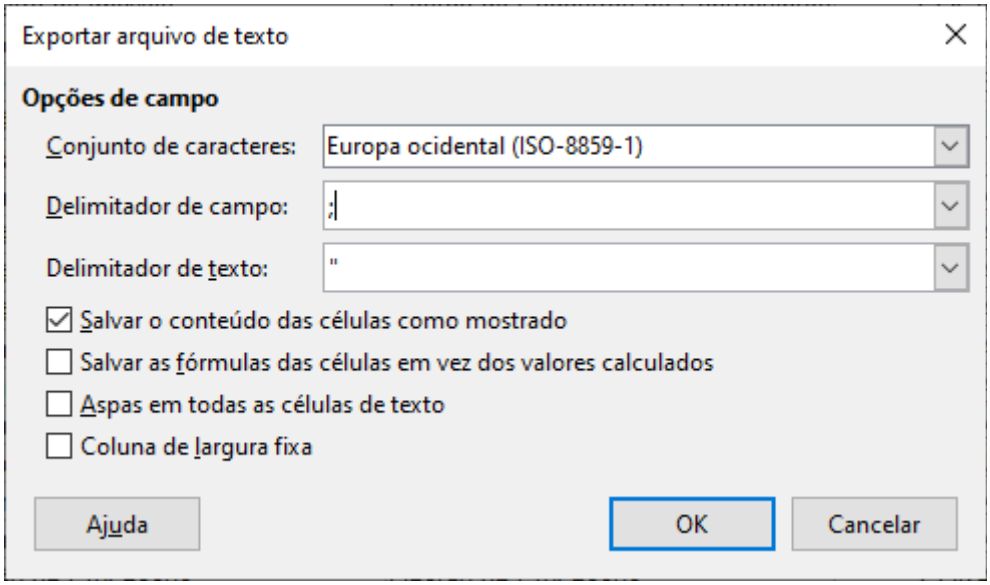
Abaixo, os Scripts usados na extração de dados do sistema SGPTI e geração dos Datasets:

Dataset	Script	Tratamento	Total Registros
1	*Select table_demanda.Ano, table_demanda.ESFORCO_PF as `Esforço PF` From SGPTI Where table_demanda.ESFORCO_PF > 0*	– Excluir registros com esforço em PF nulo, total de 2.434 registros; – Não houve registros duplicados;	11.279
2	*Select table_demanda.ESFORCO_PF as `Esforço PF`, table_demanda.Ano, Date(table_demanda.ANEXO4_5_DEACORDO_PS) as `Data Assinatura Anexo 4/5 Prest. Serv.`, Date(table_demanda.DataAprovTitular) as `Data de Aprovação do Titular` From SGPTI Where table_demanda.ESFORCO_PF > 0*	– Excluir registros com esforço em PF nulo, total de 2.434 registros; – Não houve registros duplicados;	11.279

Excluir registros com esforço em PF nulo foi necessário porque demandas que não apresentam esforço calculado em PF não possuem relevância neste estudo, visto que se encaixam em uma das 3 alternativas abaixo;

1. Demanda aberta, mas que não chegou na etapa de definição dos requisitos funcionais;
2. Demanda com ônus para o prestador de serviços;
3. Demanda cancelada.

O formato utilizado foi o CSV (Comma-separated Values), com a seguinte configuração:



3.2 Utilização dos Datasets

Os datasets utilizados para este trabalho foram submetidos aos processamentos e tratamentos abaixo:

Da-taset	**Processamento/Tratamento**	**Registros**
1	Não houve registros duplicados e nem com informações ausentes.	11.279
2	Os dados deste dataset foram usados para geração do dataset 3.	11.279
3	Este arquivo foi gerado a partir de tratamento e processamento dos dados do Dataset 2 da seguinte forma: – Exclusão de registros com os seguintes campos nulos: 1. ‘Data de Aprovação do Titular’: indica a data de início formal da demanda (excluídos 29 registros); 2. ‘Data Assinatura Anexo 4/5 Prest. Serv.’: indica a data em que a demanda foi homologada e disponibilizada para implantação (excluídos 3.291 registros); - Geração de dois novos campos: 1. ‘*TEMPO_ENTREGA*’: tempo, em dias, do início até a	7.972

	disponibilização da demanda para implantação; 2. ‘*ANO_ASSIN’*: ano de conclusão da demanda, a partir do campo “*Data Assinatura Anexo 4/5 Prest. Serv.*”.	

3.2.1 Scritp em Python para geração do Dataset 3: ”1-Gera_Dataset3.py”

```
#Importação da biblioteca Pandas
import pandas as pd

origem  = 'extracao_sgpti_datas_01.csv'
destino = 'extracao_sgpti_03.csv'

print ("Arquivo ",origem, " não foi encontrado")
try:
        print ("Lendo arquivo ", origem, "...")
        #Leitura do arquivo CSV e criação do DataFrame
        df = pd.read_csv(origem,encoding='iso-8859-1',delimiter =';', quotechar='"', thousands='.', decimal=',')

        #Converte as datas que estavam com o tipo string para o tipo datetime
        df['Data de Aprovação do Titular'] = pd.to_datetime(df['Data de Aprovação do Titular'], format='%d/%m/%Y')
        df['Data Assinatura Anexo 4/5 Prest. Serv.'] = pd.to_datetime(df['Data Assinatura Anexo 4/5 Prest. Serv.'], format='%d/%m/%Y')

        #Cria duas novas colunas:
        # 1) ANO_ASSI, contendo somente o ano da entrega da demanda;
        # 2) TEMPO_ENTREGA, contendo o número, em dias,
        df['ANO_ASSIN']    = df['Data Assinatura Anexo 4/5 Prest. Serv.'].dt.year
        df['TEMPO_ENTREGA'] = (df['Data Assinatura Anexo 4/5 Prest. Serv.']-df['Data de Aprovação do Titular']).dt.days

        #seleciona os registros os quais possuem assinatura de entrega do prestador de serviço
        df = df.loc[df['Data Assinatura Anexo 4/5 Prest. Serv.'].notnull()]
        df = df.loc[df['Data de Aprovação do Titular'].notnull()]

        try:
                df.to_csv(destino,encoding='iso-8859-1', index=False, sep=';', decimal=',')
        except IOError:
           print ("Erro ao criar arquivo: ",destino)
        finally:
             print ("Arquivo: ",destino, "gerado!")

except IOError:
   print ("Arquivo ", origem, "não foi encontrado")
```

3.2.2 Configurações de leitura dos Datasets

```
        df = pd.read_csv(origem,encoding='iso-8859-1',delimiter =';', quotechar='"', thousands='.', decimal=',')
```

Sendo:

1. “***encoding='iso-8859-1'***”: conjunto de caracteres acentuados “Europa Ocidental iso=8859-1”

2. “***thousands='.', decimal=','*** ”: reconhecimento dos padrões de ponto flutuante e milhar utilzados no Brasil

   Ex.: Padrão americano → 1,000.05; Padrão Brasileiro → 1.000,05

3. “***delimiter =';',***”: separador de campos alterado para não haver confusão com ponto decimal usado no Brasil

4. Análise e Exploração dos Dados

Os modelos quantitativos ou matemáticos de previsão partem do princípio de que dados passados são pertinentes para o futuro (GAITHER; FRAZIER, 2002), permitem controle do erro, entretanto exigem informações quantitativas preliminares (MOREIRA, 2012).

Previsões baseadas em correlações buscam prever a demanda relacionando duas ou mais variáveis. Quando a correlação entre as variáveis leva a uma equação linear ela é chamada de regressão linear. Porém, quando leva a uma equação curvilínea, chama-se regressão não linear. No caso de apenas duas variáveis estarem envolvidas, chama-se de regressão simples e quando a análise inclui mais de uma variável independente, nomeia-se de regressão múltipla (AAKER; KUMAR; DAY, 2007).

São características comumente encontradas nas técnicas de previsão (TUBINO; 2009): a) as previsões são imperfeitas, pois não se é capaz de prever todas as variações aleatórias que sucederão; b) julga-se que os fundamentos que motivaram a demanda passada continuarão a acontecer no futuro; c) a previsão para grupos de produtos é mais assertiva do que para os produtos individualmente, em razão de que no grupo os erros individuais de previsão se minimizam; d) a acuracidade das previsões diminui com a ampliação do período de tempo sondado.

Sendo assim, o objetivo deste trabalho é, com base nos dados históricos, demonstrar a possibilidade de prever as quantidades demandadas e produzidas de software para os anos futuros.

Dentro das técnicas quantitativas, as baseadas em correlações buscam modelar matematicamente a produção futura, com base no histórico produzido, através da análise de variáveis que possuam correlação significativa.

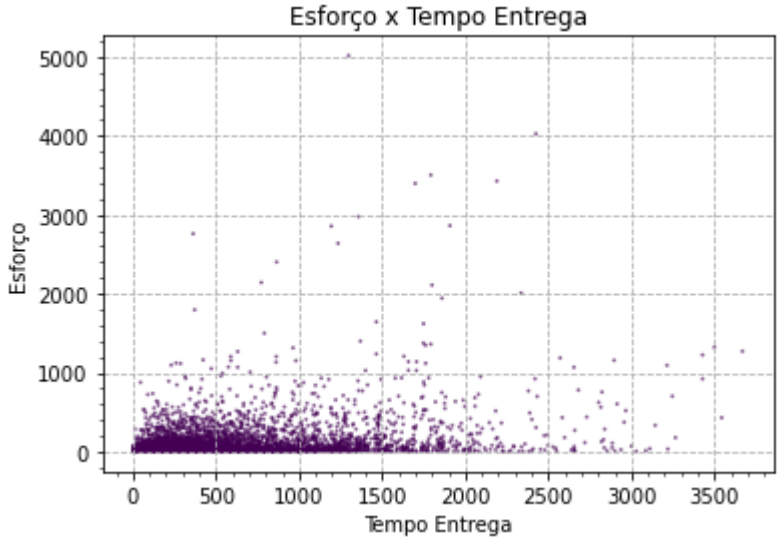
Utilizaremos o modelo de regressão chamado Regressão Linear Simples, visto que ele serve para prever comportamentos com base na associação entre duas variáveis que possuem uma boa correlação.

Iremos computar os valores de todas as demandas construídas do ano de 2012 ao de 2020. Não utilizaremos dados relativos ao ano de 2021, visto que este ano ainda não atingiu todas as suas entregas. Também não utilizaremos dados

anteriores a 2012 visto que as entregas começaram a ser computadas em 2011, com poucas demandas e processo incipiente.

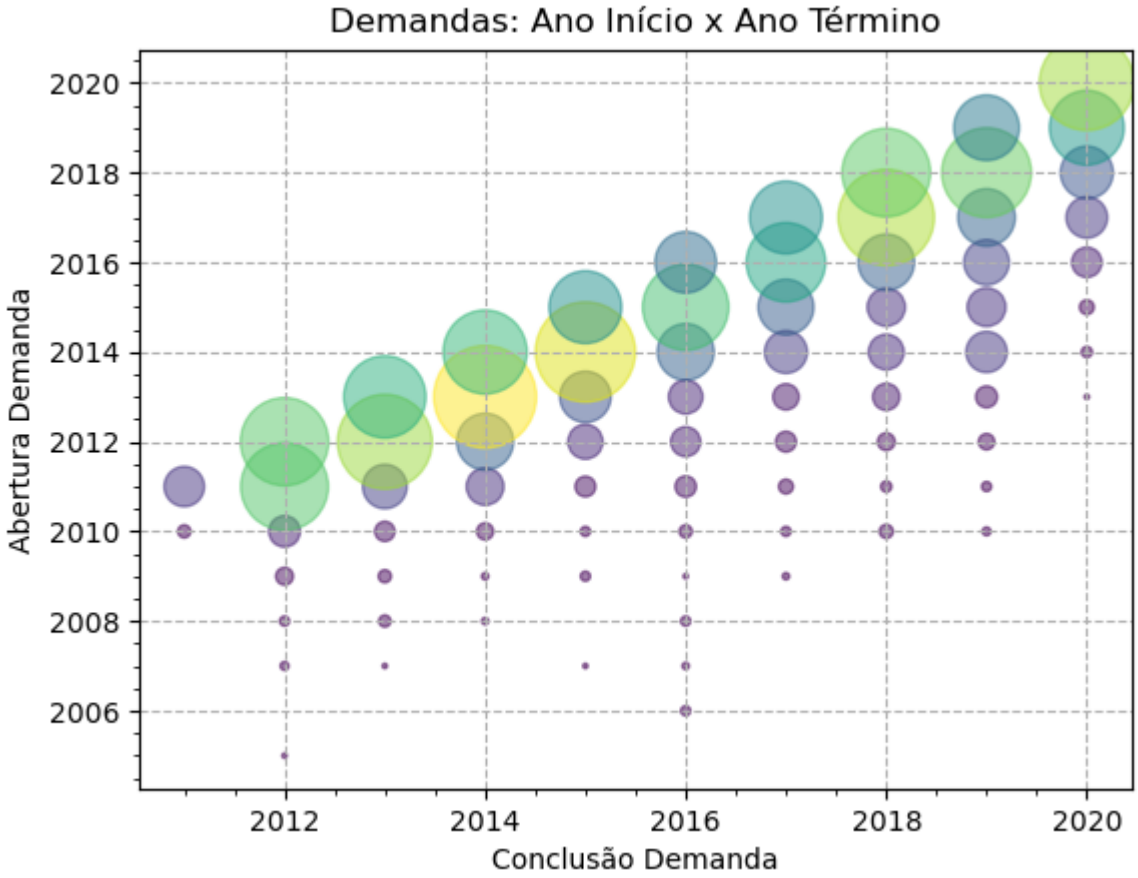
Como a gestão de recursos para desenvolvimento de sistemas na Receita Federal sofre com alguns problemas, verificaremos formas de melhorar a análise no contexto de longo prazo.

A figura abaixo exibe um gráfico de dispersão que representa, para o conjunto de demandas, o esforço em PF (eixo y) e o tempo de entrega, em dias (eixo X). Os tamanhos e cores dos marcadores refletem o número de demandas.



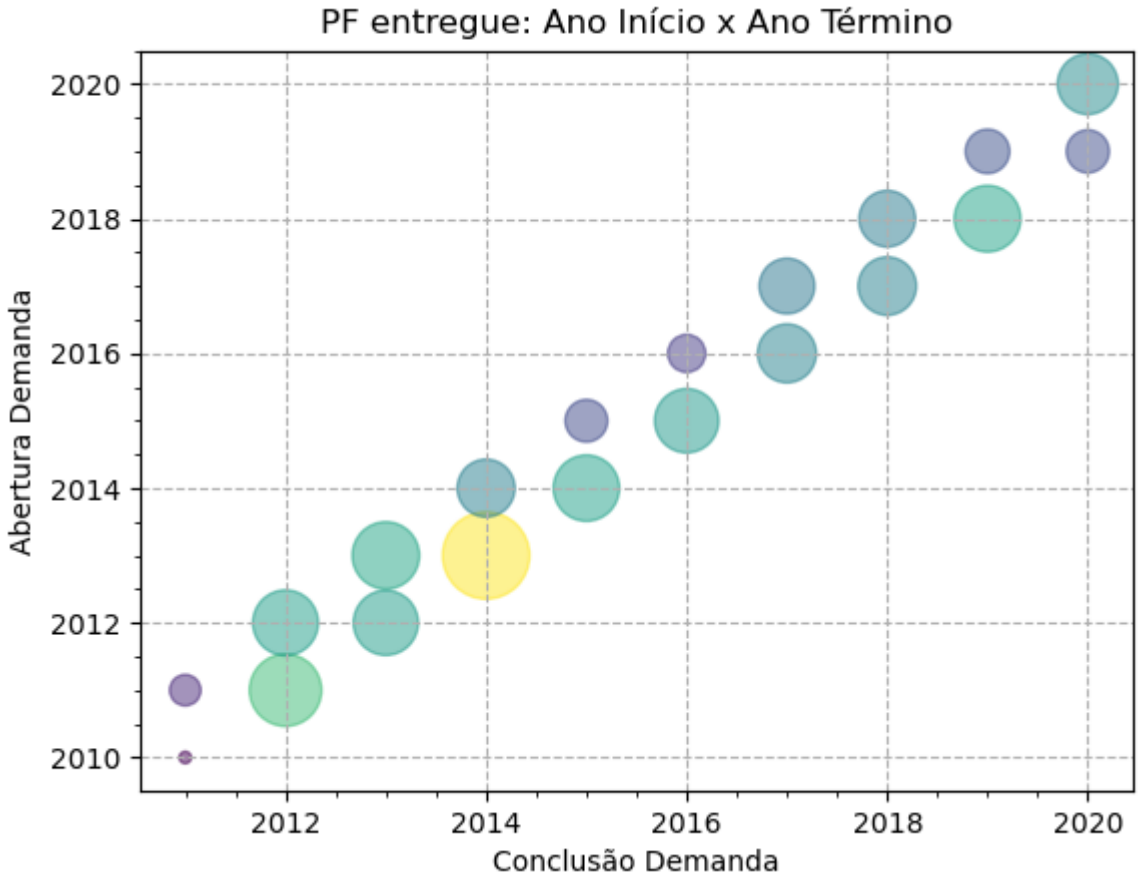
Ao observar o gráfico, não é possível observar qualquer linearidade entre o esforço para construção de uma demanda e seu tempo de entrega.

Porém, ao relacionarmos o ano de abertura (eixo y) com o ano de conclusão (eixo X), como mostra o gráfico de dispersão abaixo, podemos reparar que as entregas se relacionam com seu ano de entrega, o que poderia ser explicado pelas demandas estarem sujeitas a processos de repriorizações.



Como podemos observar na figura acima, demandas são abertas todos os anos e a maior parte delas é concluída no ano seguinte ou no mesmo ano de abertura. Porém, há demandas que levam até 10 anos para serem concluídas, o que indica que demandas mais novas alteraram a ordem de prioridade de algumas mais antigas.

Na figura abaixo, selecionamos apenas as demandas com tempo de entrega inferior a um ano. O resultado foi um gráfico bem mais organizado que o anterior.



Sendo assim, a fim de ser possível prever o andamento dos trabalhos, será analisada a base de dados de demandas. A ideia é descobrir, dentro de vasto histórico cadastrado, correlações entre os dados disponíveis e compreender o processo, para ser possível a implementação de um algoritmo de predição.

4.1 Hipóteses levantadas

A fim de testar a existência de correlação entre algumas variáveis, foram consideradas as correlações entre o esforço em PF calculado para uma demanda e as seguintes variáveis:

1. Tipo da demanda;
2. Texto do resumo da demanda;
3. Ano de abertura da demanda;
4. Ano de conclusão da demanda;
5. Subsecretaria;
6. Projeto;
7. Processo e

8. Subprocesso.

4.2 Padrões identificados

1. Correlação entre o esforço total demandado (em PF) e ano de criação da demanda;
2. Correlação entre o esforço total construído (em PF) e o ano de finalização da demanda;
3. Correlação entre o tempo de construção e ano de finalização.

4.3 Métricas

Há várias métricas para avaliação de um modelo de aprendizado de máquina. Quando tratamos das métricas de avaliação dos modelos de regressão, geralmente utilizamos as que medem a distância entre o modelo gerado e os dados, como: MSE (mean squared error), MSLE (mean squared logarithmic error), RMSE (square root mean squared error), MAE (mean_absolute_error) etc.

Em geral, seguimos a convenção de que quanto menor o valor de retorno encontrado, melhor ajustado está o modelo. Sendo que o valor zero é o valor ideal.

Porém, essas métricas sofrem variações diferentes conforme os módulos das variáveis utilizadas, o que pode nos levar a ter a impressão de que os modelos gerados não apresentam resultados satisfatórios. Para ilustrar essa situação, desenvolvemos o teste abaixo (arquivo “9-TesteDeMetricas.py”).

```
from sklearn.metrics import mean_squared_error, mean_squared_log_error, mean_absolute_error
from math import sqrt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

#Implementação de funções
def PlotaGrafico(X,y, cor, pred, cor_pred, titulo, eixo_x, eixo_y):
        #Visualizando o conjunto de resultados
        plt.scatter(X, y, color = cor)
        plt.plot(X, pred, color = cor_pred)
        plt.grid(which='major', linestyle='--')
        plt.minorticks_on()
        plt.title(titulo)
        plt.xlabel(eixo_x)
        plt.ylabel(eixo_y)
        plt.show()


y_true = [3, 5, 6.5, 9.5, 10.4, 13.5, 14.5, 17.5, 18.4, 20.6]
X_true = [[0,1], [1,2],[2,3], [3,4], [4,5], [5,6], [6,7], [7,8], [8,9], [9,10]]
```

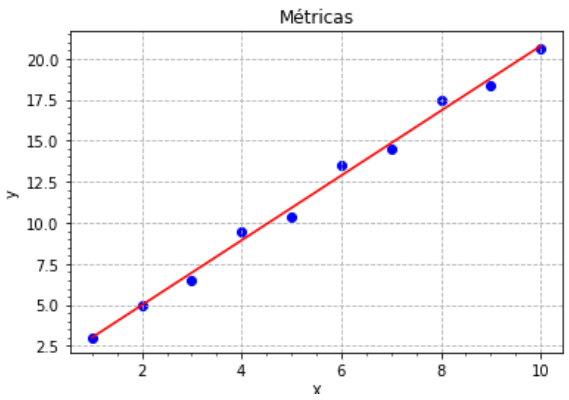
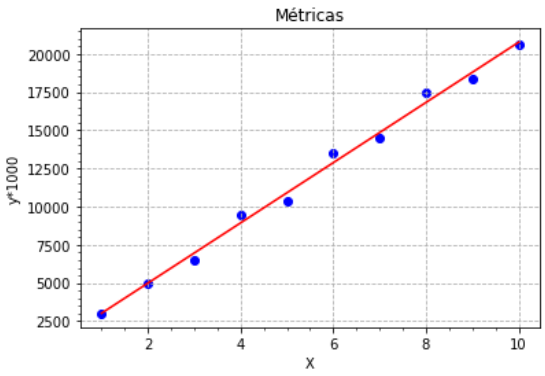
```
X = [1,2,3,4,5,6,7,8,9,10]

#Divide os arrays em subconjuntos randômicos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X_true, y_true, train_size=0.30, test_size=0.30, random_state=42)

#Fitting Simple Linear Regression to the set
regressor = LinearRegression()
regressor.fit(X_true, y_true)

#Métrica padrão, que é a mais relevante para a tarefa de Machine Learning
print('r2_train = %0.2f' %regressor.score(X_train, y_train))
print('r2_test = %0.2f'  %regressor.score(X_test, y_test))

y_pred    = regressor.predict(X_true)

print('MSE: %0.3f' %mean_squared_error(y_true, y_pred))
print('RMSE:%0.3f' %sqrt(mean_squared_error(y_true, y_pred)) )
print('MSLE:%0.3f' %mean_squared_log_error(y_true, y_pred))
print('MAE: %0.3f' %mean_absolute_error(y_true, y_pred))

PlotaGrafico(X,y_true, 'blue', y_pred, 'red', 'Métricas', 'X', 'y')

#---------------------- tudo multiplicado por 1.000
print('------- Agora com todos os dados multiplicados por 1.000 --------')
y_true = [3000, 5000, 6500, 9500, 10400, 13500, 14500, 17500, 18400, 20600]

#Divide os arrays em subconjuntos randômicos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X_true, y_true, train_size=0.30, test_size=0.30, random_state=42)

#Fitting Simple Linear Regression to the set
regressor = LinearRegression()
regressor.fit(X_true, y_true)

#Métrica padrão, que é a mais relevante para a tarefa de Machine Learning
print('r2_train = %0.2f' %regressor.score(X_train, y_train))
print('r2_test = %0.2f'  %regressor.score(X_test, y_test))

y_pred    = regressor.predict(X_true)

print('MSE: %0.3f' %mean_squared_error(y_true, y_pred))
print('RMSE:%0.3f' %sqrt(mean_squared_error(y_true, y_pred)) )
print('MSLE:%0.3f' %mean_squared_log_error(y_true, y_pred))
print('MAE: %0.3f' %mean_absolute_error(y_true, y_pred))

PlotaGrafico(X,y_true, 'blue', y_pred, 'red', 'Métricas', 'X','y*1000', )
```

Os resultados apurados foram:

Métrica	**Y = f(X)**	**Y = 1000*f(X)**
R2 treino	0,99	0,99
R2 teste	0,99	0,99

MSE	0,194	194.206,061
RMSE	0,441	440,688
MSLE	0,001	0,002
MAE	0,377	376,606

Conforme pode ser verificado pelos gráficos, esse resultado demonstra que, apesar do valor ideal para MSE continuar sendo 0, um valor alto para essa métrica não representa necessariamente falha do modelo.

Y = f(X), MSE = 0,194	**Y = 1000*f(X), MSE = 194.206,061**
	

4.3.1 Cálculo do coeficiente de correlação de Pearson

Para se verificar a associação entre as variáveis numéricas deste trabalho utilizamos a associação linear e a ferramenta que utilizamos para quantificar essa relação foi o coeficiente de correlação linear de Pearson (r).

Esse coeficiente mede o grau de relação entre conjuntos de dados. Em termos quantitativos, o seu valor varia na faixa de $-1 \leq r \leq 1$.

Abaixo temos a escala de correlação linear entre variáveis (PINHEIRO, 2019)

Pearson (r)	**Escala de referência para o grau de associação entre duas variáveis**
r = 1	Os pontos desenhados no diagrama de dispersão estão perfeitamente alinhados em uma reta que passa por eles com inclinação positiva.

0,9 < r <= 1	Relação linear forte positiva.
0,5 < r <= 0,9	Relação linear moderada positiva.
0,0 < r <= 0,5	Relação linear fraca positiva.
r = 0	Indica que não existe grau de relação linear entre as variáveis analisadas.
-0,5 <= r < 0	Relação linear fraca negativa.
-0,9 <= r < -0,5	Relação linear moderada negativa.
-1,0 <= r < -0,9	Relação linear forte negativa.
r = −1	Os pontos desenhados no diagrama de dispersão estão perfeitamente alinhados em uma reta que passa por eles com inclinação negativa.

4.3.2 Coeficiente de Determinação simples R^2

A forma de avaliar a qualidade do ajuste do modelo, usada neste trabalho, foi o coeficiente de determinação, **R^2**, que é uma medida descritiva da qualidade do ajuste obtido. Esse coeficiente indica quanto o modelo foi capaz de explicar os dados coletados (PINHEIRO, 2019).

Em geral referimo-nos ao **R^2** como a quantidade de variabilidade nos dados que é explicada pelo modelo de regressão ajustado.

Sendo assim, um $R^2 = 0,82$ significa que o modelo explica 82% da variância da variável dependente, a partir dos regressores (variáveis independentes) incluídos no modelo linear.

4.3.3 Mean Squared Error – MSE

A função *mean_squared_log_error* calcula o erro quadrático médio, uma métrica de risco correspondente ao valor esperado do erro ou perda quadrática.

4.3.4 Mean Squared Logarithmic Error – MSLE

A função *mean_squared_log_error* calcula uma métrica de risco correspondente ao valor esperado do erro ou perda logarítmica quadrática (quadrática).

A introdução do logaritmo faz com que a MSLE se preocupe apenas com a diferença relativa entre o valor verdadeiro e o previsto, ou seja, apenas se preocupe com a diferença percentual entre eles.

Isso significa que o MSLE tratará pequenas diferenças entre pequenos valores verdadeiros e preditos aproximadamente da mesma forma que grandes diferenças entre grandes valores verdadeiros e preditos.

Use MSLE ao fazer regressão, acreditando que seu alvo, condicionado na entrada, é normalmente distribuído, e você não deseja que erros grandes sejam significativamente mais penalizados do que pequenos, nos casos em que o intervalo do valor alvo é grande.

Exemplo: você deseja prever os preços futuros das casas e seu conjunto de dados inclui casas com preços muito diferentes. O preço é um valor contínuo e, portanto, queremos fazer uma regressão. MSLE pode ser usado aqui como a função de perda.

4.3.4 Square Root Mean Squared Error – RMSE

A função *root_mean_squared_log_error* calcula a raiz quadrada do erro quadrático médio, uma métrica de risco correspondente à raiz quadrada do valor esperado do erro ou perda quadrática.

4.3.5 Mean Absolute Error – MAE

A função *mean_absolute_error* calcula o erro absoluto médio, uma métrica de risco correspondente ao valor esperado da perda de erro absoluto ou perda de norma.

5. Criação de Modelos de Machine Learning

Uma vez que esperamos que grande parte da variação da variável de saída seja explicada pelas variáveis de entrada, podemos utilizar o modelo para obter valores de Y correspondentes a valores de X que não estavam entre os dados. Esse procedimento é chamado de predição e, em geral, usamos valores de X que estão dentro do intervalo de variação estudado.

Acredita-se que a predição seja a aplicação mais comum dos modelos de regressão. Dessa forma, os modelos desenvolvidos neste trabalho exploram a Predição baseada em Regressão Linear Simples.

Para o desenvolvimento deste projeto foi escolhida a distribuição Python Anaconda 3, para Windows 64 bits, porque esta distribuição disponibiliza todas as bibliotecas e ferramentas necessárias para conclusão deste TCC.

Para este TCC, criamos vários experimentos de aprendizado de máquina, mas descreveremos apenas 3. Todos eles trabalham com regressão linear:

N.	**Experimento**	**Treino/Teste**	**Descrição**
1	*EsforcoPFSolicitado*	*train_test_split*	Esforço em PF Solicitado x Ano de abertura da demanda;
2		KFold	Esforço em PF Solicitado x Ano de abertura da demanda;
3	*EsforcoPFEntregue*	*train_test_split*	Esforço em PF Entregue x Ano de abertura da demanda;
4		KFold	Esforço em PF Entregue x Ano de abertura da demanda;

5.1 Ferramentas utilizadas no desenvolvimento do trabalho

Para o Machine Learning, foi utilizada a biblioteca de aprendizado de máquina chamada **Scikit Learn**.

Para o tratamento de arquivos, manipulação e análise de dados; foi utilizada a biblioteca **Pandas**.

Para a criação de gráficos e visualizações, foi utilizada a biblioteca MatPlotLib.

A biblioteca **Numpy** foi utilizada para arrays e matrizes e funções matemáticas.

A interface de desenvolvimento, que permitiu um ambiente confortável para desenvolvimento foi o **Jupyter Notebook**.

Lembrando que todo o código desenvolvido foi implementado em **Python**.

5.2 Passos para a criação do modelo

5.2.1 Extração como Dataframe

EsforcoPFSolicitado	*location= "extracao_sgpti_01.csv"* *df = pd.read_csv(location,encoding='iso-8859-1',delimiter =';', quotechar='"', thousands='.', decimal=',')*
EsforcoPFEntregue	*location = "extracao_sgpti_03.csv"* *df = pd.read_csv(location,encoding='iso-8859-1',delimiter =';', quotechar='"', thousands='.', decimal=',')*

5.2.2 Selecionar as variáveis envolvidas

Foram testadas várias opções de variáveis para contraposição ao esforço em pontos de função, como: Processo de Trabalho, Subprocesso, Unidade de Atendimento e Tipo de demanda.

Porém, como a APF é uma métrica que tem como base os requisitos funcionais, as demais variáveis não mostraram influência nos resultados.

Dessa forma, as variáveis escolhidas foram:

- Esforço em PF;
- Ano de solicitação da demanda;
- Ano de início da demanda;
- Ano de conclusão da demanda;

Duração, em dias, para conclusão da demanda.

5.2.3 Definir o período de avaliação

Usamos valores de X que estão dentro do intervalo de variação estudado. A utilização de valores fora desse intervalo recebe o nome de extrapolação e deve ser usada com muito cuidado, pois, o modelo adotado pode não ser correto fora do intervalo estudado (PINHEIRO, 2019).

Determinação dos intervalos para seleção:

EsforcoPFSolicitado	*df = df.loc[df['Ano']<=2020]*

	df = df.loc[df['Ano']>=2012]
EsforcoPFEntregue	*df = df.loc[df['ANO_ASSIN']<=2020]* *df = df.loc[df['ANO_ASSIN']>=2012]*

5.2.4 Avaliar se há correlação

EsforcoPFSolicitado e EsforcoPFEntregue	*Kfold e train_test_split*	*#calculando a correlação entre as variáveis* *r = np.corrcoef(X, y, rowvar=False)* *pearson = r[1,0]* *#Não trabalhar com o módulo do coef. de pearson abaixo de 0,5 para não trabalhar com relação linear fraca.* *if (pearson>0.5 or pearson<-0,5):*

5.2.5 A importância da divisão de dados

O aprendizado de máquina supervisionado trata da criação de modelos que mapeiem com precisão as entradas fornecidas (variáveis independentes ou preditores) para as saídas fornecidas (variáveis dependentes ou respostas).

Como você mede a precisão do seu modelo depende do tipo de problema que você está tentando resolver. Na análise de regressão, você normalmente usa o R^2 (coeficiente de determinação), RMSE (erro quadrático médio), MAE (erro absoluto médio) ou métricas semelhantes.

O que é mais importante entender é que você geralmente precisa de uma avaliação imparcial para usar adequadamente essas medidas, avaliar o desempenho preditivo de seu modelo e validar o modelo.

Aprender os parâmetros de uma função de predição e testá-la com os mesmos dados é um erro metodológico. Isso significa que você não pode avaliar o desempenho preditivo de um modelo com os mesmos dados usados para o treinamento. Um modelo que apenas repetisse os rótulos das amostras que acabou de ver teria uma pontuação perfeita, mas não conseguiria prever nada útil. Essa situação é chamada de overfitting (sobreajuste). Já a situação na qual um modelo que não capturasse bem as dependências e os relacionamentos entre as variáveis seria chamada underfitting (subajuste).

Dessa forma, a fim de realizar uma avaliação imparcial do modelo e identificar underfitting (subajuste) ou overfitting (sobreajuste), você precisa avaliar o modelo com dados novos que não foram vistos pelo modelo antes. Você pode fazer isso dividindo seu conjunto de dados antes de usá-lo.

A separação da base de dados em base de treinamento e base de testes deve respeitar a semelhança entre elas. As bases devem ser semelhantes estatisticamente e ter a mesma cobertura do espaço de solução.

Os dados de treino são usados para treinar nosso modelo e os dados de teste são utilizados para testar o quanto nosso modelo está aprendendo.

Para bases de dados grandes, utilizamos a divisão percentual (percentage split). Caso contrário, devemos utilizar a validação cruzada (cross validation).

5.2.5.1 Validação Cruzada

Ao avaliar diferentes configurações (hiperparâmetros) para estimadores ainda há o risco de overfitting no conjunto de teste porque os parâmetros podem ser ajustados até que o estimador tenha um desempenho ideal.

Validação cruzada é uma técnica para avaliar modelos de ML (Machine Learning) por meio de treinamento de vários modelos de ML em subconjuntos de dados de entrada disponíveis e avaliação deles no subconjunto complementar dos dados, a fim de detectar sobreajuste.

A validação K-fold é a técnica mais conhecida de validação cruzada. Essa técnica é conhecida pelo seguinte:

k = o número de subdivisões

Fold = significa cada um dos blocos de cada k

5.2.5.2 Funções para divisão de dados

Neste TCC foram usados dois algoritmos para dividir os dados em dados de treinamento e dados de testes. *train_test_split*. e *Kfold*.

train_test_split: implementa divisão aleatória em conjuntos de treinamento e teste e pode ser calculada rapidamente com a função auxiliar.

Kfold: divide todas as amostras em ***k*** grupos de amostras, chamados folds (dobras),se ***k=n***, é equivalente à estratégia *Leave One Out*, de tamanhos iguais (se possível). A função de previsão (treino) é aprendida usando ***k-1*** dobras, e a dobra deixada de fora é usada para teste.

Exemplo:

	Dataset

train_test_split	**Conjunto de treinamento**				**Conjunto de teste**
Kfold k = 5	Fold1 – teste	Fold2 – treino	Fold3 – treino	Fold4 – treino	Fold5 – treino
	Fold1 – treino	Fold2 – teste	Fold3 – treino	Fold4 – treino	Fold5 – treino
	Fold1 – treino	Fold2 – treino	Fold3 – teste	Fold4 – treino	Fold5 – treino
	Fold1 – treino	Fold2 – treino	Fold3 – treino	Fold4 – teste	Fold5 – treino
	Fold1 – treino	Fold2 – treino	Fold3 – treino	Fold4 – treino	Fold5 – teste

EsforcoPFSolicitado	*train_test_split*	*tam_treino = 0.55* *tam_teste = 0.45* *#Testa todas as divisões possíveis para o período de tempo* *for num in range(1,24,1):* *tam_treino = tam_treino + 0.01* *tam_teste = tam_teste - 0.01* *#Divide os arrays em subconjuntos randômicos de treino e teste* *X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=tam_treino, test_size=tam_teste, random_state=42)*
	KFold	*#Testa todas as divisões possíveis para o período de tempo* *for num in range(2,9,1):* *kf = KFold(n_splits=num)* *for train_index, test_index in kf.split(X):* *#Não aceita conjuntos menores q 2 elementes* *if (len(train_index)>1 and len(test_index)>1):* *X_train, X_test, y_train, y_test = X[train_index], X[test_index], y[train_index], y[test_index]*
EsforcoPFEntregue	*train_test_split*	*tam_treino = 0.55* *tam_teste = 0.45* *#Testa todas as divisões possíveis para o período de tempo* *for num in range(1,24,1):* *tam_treino = tam_treino + 0.01* *tam_teste = tam_teste - 0.01* *#Divide os arrays em subconjuntos randômicos de treino e teste* *X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=tam_treino, test_size=tam_teste, random_state=42)*
	KFold	*#Testa todas as divisões possíveis para o período de tempo* *for num in range(2,9,1):* *kf = KFold(n_splits=num)* *for train_index, test_index in kf.split(X):* *#Não aceita conjuntos menores q 2 elementes* *if (len(train_index)>1 and len(test_index)>1):* *X_train, X_test, y_train, y_test = X[train_index], X[test_index], y[train_index], y[test_index]*

Nos trechos de código acima, para a divisão aleatória de subconjuntos de treinamento e testes, testamos a variação de tamanho do subconjunto de treinamento (*train_size*) de 55% a 78% da base de dados e o subconjunto de testes (*test_size*) oscilou de 22% a 45%.

O embaralhamento aplicado ao gerador de número aleatório, antes de aplicar a divisão, (parâmetro *Random_state*) sendo um inteiro, produzirá os mesmos resultados em chamadas diferentes. As sementes aleatórias de números inteiros populares são 0 e 42.

Para a validação cruzada, foi utilizado o máximo de subdivisões, ***k*** grupos de amostras, permitidas.

5.2.6 Gerar modelo de Regressão Linear Simples

EsforcoPFSolicitado e EsforcoPFEntregue	*train_test_split e KFold*	*#Fitting Simple Linear Regression to the set* *regressor = LinearRegression()*

5.2.7 Ajustar modelo linear (fit)

EsforcoPFSolicitado, EsforcoPFEntregue	*train_test_split e KFold*	*regressor.fit(X, y)*

5.2.8 Medir o quão próximos estão os dados da linha de regressão

EsforcoPFSolicitado,	*train_test_split*	*y_pred = regressor.predict(X)* *train_ind = list(X_train.index)* *test_ind = list(X_test.index)* *rmse_train = sqrt(mean_squared_error(y_train,y_pred[train_ind]))* *rmse_test = sqrt(mean_squared_error(y_test,y_pred[test_ind]))* *rmse = sqrt(mean_squared_error(y,y_pred))* *msle = mean_squared_log_error(y,y_pred)* *mae = mean_absolute_error(y,y_pred)*
	KFold	*y_pred= regressor.predict(X)* *rmse_train = sqrt(mean_squared_error(y_train,y_pred[train_index]))* *rmse_test = sqrt(mean_squared_error(y_test,y_pred[test_index]))* *rmse = sqrt(mean_squared_error(y,y_pred))* *msle = mean_squared_log_error(y,y_pred)* *mae = mean_absolute_error(y,y_pred)*
	train_test_split	*train_ind = list(X_train.index)* *test_ind = list(X_test.index)* *y_pred = regressor.predict(X)* *rmse_train = sqrt(mean_squared_error(y_train,y_pred[train_ind]))*

EsforcoPFEntregue		*rmse_test = sqrt(mean_squared_error(y_test,y_pred[test_ind]))* *msle_test = mean_squared_log_error(y_test,y_pred[test_ind])* *msle_train = mean_squared_log_error(y_train,y_pred[train_ind])* *rmse = sqrt(mean_squared_error(y,y_pred))* *msle = mean_squared_log_error(y,y_pred)* *mae = mean_absolute_error(y,y_pred)*
	KFold	*y_pred = regressor.predict(X)* *rmse_train = sqrt(mean_squared_error(y_train,y_pred[train_index]))* *rmse_test = sqrt(mean_squared_error(y_test,y_pred[test_index]))* *msle_test = mean_squared_log_error(y_test ,y_pred[test_index])* *msle_train = mean_squared_log_error(y_train,y_pred[train_index])* *rmse = sqrt(mean_squared_error(y,y_pred))* *msle = mean_squared_log_error(y,y_pred)* *mae = mean_absolute_error(y,y_pred)*

5.2.9 Gerar gráficos com os resultados

Desenha uma linha com os valores de previsão e os pontos correspondentes ao aferido na base de dados.

EsforcoPFSolicitado	*train_test_split e KFold*	*#Visualizando o conjunto de resultados* *label1 = 'Valores Reais'* *label2 = 'Predição - RMSE'* *label3 = 'Predição - R2'* *titulo = 'Esforço PF Solicitado x Ano - Split'* *eixo_x = 'Ano'* *eixo_y = 'Esforço PF'* *plt.scatter(menor_rmse_X, menor_rmse_y, color = 'green', label=label2)* *plt.plot(menor_rmse_X, menor_rmse_pred, color = 'blue', label=label1)* *plt.grid(which='major', linestyle='--')* *plt.minorticks_on()* *plt.title(titulo)* *plt.xlabel(eixo_x)* *plt.ylabel(eixo_y)* *plt.legend([label2, label1], loc=1)* *plt.show()* *#Visualizando o conjunto de resultados* *plt.scatter(maior_R2_X, maior_R2_y, color = 'green', label=label3)* *plt.plot(maior_R2_X, maior_R2_pred, color = 'blue', label=label1)* *plt.grid(which='major', linestyle='--')* *plt.minorticks_on()* *plt.title(titulo)* *plt.xlabel(eixo_x)* *plt.ylabel(eixo_y)* *plt.legend([label3, label1], loc=1)* *plt.show()*
EsforcoPFEntregue	*train_test_split e KFold*	*#Visualizando o conjunto de resultados* *label1 = 'Valores Reais'* *label2 = 'Predição - MSLE'* *label3 = 'Predição - R2'* *titulo = 'Esforço PF Entregue x Ano - Split'*

```
eixo_x = 'Ano'
eixo_y = 'Esforço PF'

plt.scatter(menor_msle_X, menor_msle_y, color = 'red', label=label1)
plt.plot(menor_msle_X, menor_msle_pred, color = 'blue', label=label2)
plt.grid(which='major', linestyle='--')
plt.minorticks_on()
plt.title(titulo)
plt.xlabel(eixo_x)
plt.ylabel(eixo_y)
plt.legend([label2, label1], loc=1)
plt.show()

#Visualizando o conjunto de resultados
plt.scatter(maior_R2_X, maior_R2_y, color = 'red', label=label1)
plt.plot(maior_R2_X, maior_R2_pred, color = 'blue', label=label3)
plt.grid(which='major', linestyle='--')
plt.minorticks_on()
plt.title(titulo)
plt.xlabel(eixo_x)
plt.ylabel(eixo_y)
plt.legend([label3, label1], loc=1)
plt.show()
```

5.3 Código dos 4 modelos exibidos

5.3.1 Script Python: "2-kfold-EsforcoPFSolicitado.py"

a) Esforço em PF Solicitado x Ano de abertura da demanda;

b) Código fonte;

```
#Demandas Solicitadas
#Método de criação de subconjuntos -> KFold
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, mean_squared_log_error, mean_absolute_error
from math import sqrt

#Importando o dataset
location= "extracao_sgpti_01.csv"

try:
```

```
menor_rmse_rmse,     menor_rmse_msle,   menor_rmse_mae = 0, 0, 0
menor_rmse_num,      menor_rmse_pearson              = 0, 0
menor_rmse_r2_train,  menor_rmse_r2_test             = 0, 0
menor_rmse_pred,     menor_rmse_X,     menor_rmse_y   = 0, 0, 0
menor_rmse_train_ind, menor_rmse_test_ind            = 0, 0
menor_rmse_rmse_train,menor_rmse_rmse_test           = 0, 0

maior_R2_rmse,      maior_R2_msle,  maior_R2_mae = 0, 0, 0
maior_R2_num,       maior_R2_pearson           = 0, 0
maior_R2_r2_train,   maior_R2_r2_test          = 0, 0
maior_R2_pred,      maior_R2_X,     maior_R2_y  = 0, 0, 0
maior_R2_train_ind,  maior_R2_test_ind         = 0, 0
maior_R2_rmse_train,maior_R2_rmse_test         = 0, 0

print ("Lendo arquivo ", location, "...")
df = pd.read_csv(location,encoding='iso-8859-1',delimiter =';', quotechar='"', thousands='.', decimal=',')

#Determina intervalos para seleção
df = df.loc[ df['Ano']<=2020]
df = df.loc[ df['Ano']>=2012]

#soma os valores em PF das demandas, agrupados por Ano
df = df.groupby(["Ano"]).sum().reset_index()

# seleciona as colunas "Ano" e "Esforço PF"
X = np.array(df[["Ano"]])
y = np.array(df[["Esforço PF"]])

#calculando a correlação entre as variáveis
r = np.corrcoef(X, y, rowvar=False)
pearson = r[1,0]

#Não trabalhar com o módulo do coef. de pearson abaixo de 0,5 para não trabalhar com relação linear fra-
ca.
if (pearson>0.5 or pearson<-0,5):
   #Testa todas as divisões possíveis para o período de tempo
   for num in range(2,9,1):
      kf = KFold(n_splits=num)
      for train_index, test_index in kf.split(X):
         #Não aceita conjuntos menores q 2 elementes
         if (len(train_index)>1 and len(test_index)>1):
            X_train, X_test, y_train, y_test = X[train_index], X[test_index], y[train_index], y[test_index]

            #Fitting Simple Linear Regression to the set
            regressor = LinearRegression()
            regressor.fit(X, y)

            #Métrica padrão, que é a mais relevante para a tarefa de Machine Learning
            r2_train = regressor.score(X_train, y_train)
            r2_test = regressor.score(X_test, y_test)
            #Exige r2_test seja positivo para ser melhor que uma reta horizontal
            if (r2_test>0.9 and r2_test<=1): #or (r2_test<-0.5 and r2_test>-1)
               y_pred= regressor.predict(X)
               rmse_train = sqrt(mean_squared_error(y_train,y_pred[train_index]) )
               rmse_test  = sqrt(mean_squared_error(y_test,y_pred[test_index]) )
               rmse  = sqrt(mean_squared_error(y,y_pred) )
```

```
            msle  = mean_squared_log_error(y,y_pred)
            mae   = mean_absolute_error(y,y_pred)

            #Seleciona o menor valor de RMSE no teste
            if (rmse_test<menor_rmse_rmse or menor_rmse_rmse==0):
               menor_rmse_rmse,    menor_rmse_msle,   menor_rmse_mae = rmse, msle, mae
               menor_rmse_num,     menor_rmse_pearson             = num, pearson
               menor_rmse_r2_train, menor_rmse_r2_test            = r2_train, r2_test
               menor_rmse_pred,    menor_rmse_X,     menor_rmse_y   = y_pred, X, y
               menor_rmse_train_ind,menor_rmse_test_ind           = train_index, test_index
               menor_rmse_rmse_train,menor_rmse_rmse_test          = rmse_train, rmse_test

               #Seleciona o Maior valor de R2 Teste
               if (r2_test>maior_R2_r2_test or maior_R2_r2_test==0):
                  maior_R2_rmse,     maior_R2_msle,  maior_R2_mae = rmse, msle, mae
                  maior_R2_num,      maior_R2_pearson           = num, pearson
                  maior_R2_r2_train,  maior_R2_r2_test          = r2_train, r2_test
                  maior_R2_pred,     maior_R2_X,     maior_R2_y   = y_pred, X, y
                  maior_R2_train_ind, maior_R2_test_ind          = train_index, test_index
                  maior_R2_rmse_train,maior_R2_rmse_test         = rmse_train, rmse_test

del [[df]]


print("---------- Melhor RMSE ----------")
print("Pearson = %0.3f" %menor_rmse_pearson)
print("Split = ", menor_rmse_num)
print('Coeficiente de determinação da predição R2:')
print('1-Set de treino: %0.3f' % menor_rmse_r2_train)
print('2-Set de teste: %0.3f'  % menor_rmse_r2_test)
print("Índice de treino: ", menor_rmse_train_ind)
print("Índice de teste: ",  menor_rmse_test_ind)
print("RMSE Treino: %0.3f" % menor_rmse_rmse_train )
print("RMSE Teste:  %0.3f" % menor_rmse_rmse_test)
print("RMSE: %0.3f" % menor_rmse_rmse)
print("MSLE: %0.3f" % menor_rmse_msle)
print("MAE : %0.3f" % menor_rmse_mae)


#Visualizando o conjunto de resultados
label1 = 'Valores Reais'
label2 = 'Predição - RMSE'
label3 = 'Predição - R2'
titulo = 'Esforço PF Solicitado x Ano - KFold'
eixo_x = 'Ano'
eixo_y = 'Esforço PF'
plt.scatter(menor_rmse_X, menor_rmse_y, color = 'green', label=label1)
plt.plot(menor_rmse_X, menor_rmse_pred, color = 'blue', label=label2)
plt.grid(which='major', linestyle='--')
plt.minorticks_on()
plt.title(titulo)
plt.xlabel(eixo_x)
plt.ylabel(eixo_y)
plt.legend([label2, label1], loc=1)
plt.show()
```

```
    print("---------- Melhor R2 ----------")
    print("Pearson = %0.3f" %maior_R2_pearson)
    print("Split = ", maior_R2_num)
    print('Coeficiente de determinação da predição R2:')
    print('1-Set de treino: %0.3f' % maior_R2_r2_train)
    print('2-Set de teste: %0.3f'  % maior_R2_r2_test)
    print("Índice de treino: ", maior_R2_train_ind)
    print("Índice de teste: ",  maior_R2_test_ind)
    print("RMSE Treino: %0.3f" % maior_R2_rmse_train )
    print("RMSE Teste:  %0.3f" % maior_R2_rmse_test)
    print("RMSE: %0.3f" % maior_R2_rmse)
    print("MSLE: %0.3f" % maior_R2_msle)
    print("MAE : %0.3f" % maior_R2_mae)


    #Visualizando o conjunto de resultados
    plt.scatter(maior_R2_X, maior_R2_y, color = 'green', label=label1)
    plt.plot(maior_R2_X, maior_R2_pred, color = 'blue', label=label3)
    plt.grid(which='major', linestyle='--')
    plt.minorticks_on()
    plt.title(titulo)
    plt.xlabel(eixo_x)
    plt.ylabel(eixo_y)
    plt.legend([label3, label1], loc=1)
    plt.show()

except IOError:
    print ("Arquivo ", location, " não foi encontrado")
```

5.3.2 Script Python: "2-TrainTestSplit-2-EsforcoPFSolicitado.py"

a) Esforço em PF Solicitado x Ano de abertura da demanda;

b) Código fonte;

```
#Importando as bibliotecas

#Demandas Solicitadas

#Método de criação de subconjuntos -> train_test_split

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error, mean_squared_log_error, mean_absolute_error

from math import sqrt



#Importando o dataset

location= "extracao_sgpti_01.csv"
```

```
try:
        menor_rmse_rmse_train,menor_rmse_rmse_test, menor_rmse_num  = 0, 0, 0
        menor_rmse_r2_train,  menor_rmse_r2_test                = 0, 0
        menor_rmse_train_ind, menor_rmse_test_ind,menor_rmse_pearson= 0, 0, 0
        menor_rmse_pred,     menor_rmse_X,      menor_rmse_y     = 0, 0, 0
        maior_rmse_tam_treino,maior_rmse_tam_teste              = 0, 0
        menor_rmse_rmse, menor_rmse_msle, menor_rmse_mae         = 0, 0, 0

        maior_R2_rmse_train, maior_R2_rmse_test, maior_R2_num  = 0, 0, 0
        maior_R2_r2_train,   maior_R2_r2_test               = 0, 0
        maior_R2_train_ind, maior_R2_test_ind,maior_R2_pearson = 0, 0, 0
        maior_R2_pred,     maior_R2_X,      maior_R2_y      = 0, 0, 0
        maior_R2_tam_treino,maior_R2_tam_teste              = 0, 0

        print ("Lendo arquivo ", location, "...")
        df    =    pd.read_csv(location,encoding='iso-8859-1',delimiter    =';',    quotechar='"',    thousands='.',
decimal=',')

        #Determina intervalos para seleção
        df = df.loc[ df['Ano']<=2020]
        df = df.loc[ df['Ano']>=2012]

        #soma os valores em PF das demandas, agrupados por Ano
        df = df.groupby(["Ano"]).sum().reset_index()

        # seleciona as colunas "Ano" e "Esforço PF"
        X = df[["Ano"]]
        y = df[["Esforço PF"]]

        #calculando a correlação entre as variáveis
        r = np.corrcoef(X, y, rowvar=False)
        pearson = r[1,0]

        #Não trabalhar com o módulo do coef. de pearson abaixo de 0,5 para não trabalhar com relação line-
ar fraca.
        if (pearson>0.5 or pearson<-0,5):

                #testar diferentes proporções entre os tamanhos de teste e treino
```

```
                tam_treino = 0.55
                tam_teste  = 0.45

                #Testa todas as divisões possíveis para o período de tempo
                for num in range(1,24,1): #9
                        tam_treino = tam_treino + 0.01
                        tam_teste  = tam_teste  - 0.01
                        #Divide os arrays em subconjuntos randômicos de treino e teste
                        X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=tam_treino,
test_size=tam_teste, random_state=42)

                        #Fitting Simple Linear Regression to the set
                        regressor = LinearRegression()
                        regressor.fit(X, y)

                        #Métrica padrão, que é a mais relevante para a tarefa de Machine Learning
                        r2_train = regressor.score(X_train, y_train)
                        r2_test  = regressor.score(X_test, y_test)

                        #Exige r2_test seja positivo para ser melhor que uma reta horizontal
                        if (r2_test>0.9 and r2_test<=1):
                                y_pred    = regressor.predict(X)
                                train_ind = list(X_train.index)
                                test_ind = list(X_test.index)

                                rmse_train = sqrt(mean_squared_error(y_train,y_pred[train_ind]) )
                                rmse_test  = sqrt(mean_squared_error(y_test,y_pred[test_ind]) )
                                rmse  = sqrt(mean_squared_error(y,y_pred) )
                                msle  = mean_squared_log_error(y,y_pred)
                                mae   = mean_absolute_error(y,y_pred)


                                #Seleciona o menor valor de RMSE no teste
                                if (rmse_test< menor_rmse_rmse or menor_rmse_rmse==0):
                                        menor_rmse_rmse_train,menor_rmse_rmse_test =
rmse_train, rmse_test,
                                        menor_rmse_num,      menor_rmse_pearson    = num, pear-
son
```

```
                                        menor_rmse_r2_train,  menor_rmse_r2_test          = r2_train,
r2_test
                                        menor_rmse_train_ind, menor_rmse_test_ind         = train_ind,
test_ind,
                                        menor_rmse_pred,    menor_rmse_X,    menor_rmse_y= y_pred,
X, y
                                        menor_rmse_tam_treino,menor_rmse_tam_teste        =
tam_treino, tam_teste
                                        menor_rmse_rmse,       menor_rmse_msle,  menor_rmse_mae =
rmse, msle, mae

                                #Seleciona o Maior valor de R2 Teste
                                if (r2_test>maior_R2_rmse_test or maior_R2_rmse_test==0):
                                        maior_R2_rmse_train,  maior_R2_rmse_test,  maior_R2_num   =
rmse_train, rmse_test, num
                                        maior_R2_r2_train,    maior_R2_r2_test            = r2_train,
r2_test
                                        maior_R2_train_ind,   maior_R2_test_ind,maior_R2_pearson   =
train_ind, test_ind, pearson
                                        maior_R2_pred,    maior_R2_X,    maior_R2_y    = y_pred, X, y
                                        maior_R2_tam_treino,maior_R2_tam_teste          = tam_trei-
no, tam_teste
                                        maior_R2_rmse,    maior_R2_msle,   maior_R2_mae    = rmse,
msle, mae

                        del [[df]]
        #Fim do loop for

        print("---------- Melhor RMSE ----------")
        print("Pearson = %0.3f" %menor_rmse_pearson)
        print("Tam. Treino = %0.3f" % menor_rmse_tam_treino,
      ", Tam. Teste = %0.3f" % menor_rmse_tam_teste)
        print('Coeficiente de determinação da predição R2:')
        print('1-Set de treino: %0.3f' % menor_rmse_r2_train)
        print('2-Set de teste: %0.3f'  % menor_rmse_r2_test)
        print("Índice de treino: ", menor_rmse_train_ind)
        print("Índice de teste: ",  menor_rmse_test_ind)
        print("RMSE Treino: %0.3f" % menor_rmse_rmse_train )
```

```
print("RMSE Test: %0.3f"   % menor_rmse_rmse_test)
print("RMSE: %0.3f" % menor_rmse_rmse)
print("MSLE: %0.3f" % menor_rmse_msle)
print("MAE : %0.3f" % menor_rmse_mae)


#Visualizando o conjunto de resultados
label1 = 'Valores Reais'
label2 = 'Predição - RMSE'
label3 = 'Predição - R2'
titulo = 'Esforço PF Solicitado x Ano - Split'
eixo_x = 'Ano'
eixo_y = 'Esforço PF'
plt.scatter(menor_rmse_X, menor_rmse_y, color = 'green', label=label2)
plt.plot(menor_rmse_X, menor_rmse_pred, color = 'blue', label=label1)
plt.grid(which='major', linestyle='--')
plt.minorticks_on()
plt.title(titulo)
plt.xlabel(eixo_x)
plt.ylabel(eixo_y)
plt.legend([label2, label1], loc=1)
plt.show()


print("---------- Melhor R2 ----------")
print("Pearson = %0.3f" %maior_R2_pearson)
print("Tam. Treino = %0.3f" % maior_R2_tam_treino, ", Tam. Teste = %0.3f" % maior_R2_tam_teste)
print('Coeficiente de determinação da predição R2:')
print('1-Set de treino: %0.3f' % maior_R2_r2_train)
print('2-Set de teste: %0.3f'  % maior_R2_r2_test)
print("Índice de treino: ", maior_R2_train_ind)
print("Índice de teste: ",  maior_R2_test_ind)
print("RMSE Treino: %0.3f" % maior_R2_rmse_train )
print("RMSE Teste:  %0.3f" % maior_R2_rmse_test)
print("RMSE: %0.3f" % maior_R2_rmse)
print("MSLE: %0.3f" % maior_R2_msle)
print("MAE : %0.3f" % maior_R2_mae)
```

```
        #Visualizando o conjunto de resultados
        plt.scatter(maior_R2_X, maior_R2_y, color = 'green', label=label3)
        plt.plot(maior_R2_X, maior_R2_pred, color = 'blue', label=label1)
        plt.grid(which='major', linestyle='--')
        plt.minorticks_on()
        plt.title(titulo)
        plt.xlabel(eixo_x)
        plt.ylabel(eixo_y)
        plt.legend([label3, label1], loc=1)
        plt.show()

except IOError:
   print ("Arquivo ", location, " não foi encontrado")
```

5.3.3 Script Python: "4-kfold-EntregasDeDemandasDuracaoCalculada.py"

a) Esforço em PF Entregue x Ano de abertura da demanda;

b) Código fonte;

```
#Demandas Entregues em tempo inferior a 2 anos
#Método de criação de subconjuntos -> KFold
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, mean_squared_log_error, mean_absolute_error
from math import sqrt


#Importando o dataset
location = "extracao_sgpti_03.csv"

try:
        menor_msle_rmse,    menor_msle_msle,   menor_msle_mae    = 0, 0, 0
        menor_msle_num,     menor_msle_dias,   menor_msle_pearson= 0, 0, 0
        menor_msle_r2_train, menor_msle_r2_test               = 0, 0
        menor_msle_pred,    menor_msle_X,     menor_msle_y     = 0, 0, 0
        menor_msle_train_ind,menor_msle_test_ind              = 0, 0
        menor_msle_msle_train,menor_msle_msle_test            = 0, 0

        maior_R2_rmse,      maior_R2_msle,  maior_R2_mae    = 0, 0, 0
        maior_R2_num,       maior_R2_dias,  maior_R2_pearson= 0, 0, 0
        maior_R2_r2_train,   maior_R2_r2_test            = 0, 0
        maior_R2_pred,      maior_R2_X,     maior_R2_y     = 0, 0, 0
        maior_R2_train_ind,  maior_R2_test_ind           = 0, 0
        maior_R2_msle_train,maior_R2_msle_test           = 0, 0
```

```
        print ("Lendo arquivo ", location, "...")

        #Testa várias durações de demandas
        for dias in range(20,3648,5):
                df = pd.read_csv(location,encoding='iso-8859-1',delimiter =';', quotechar='"', thousands='.',
decimal=',')

                #Determina intervalos para seleção
                df = df.loc[ df['ANO_ASSIN']<=2020]
                df = df.loc[ df['ANO_ASSIN']>=2012]

                #seleciona demandas com tempo de construção inferior a 1 ano
                df = df.loc[ df['TEMPO_ENTREGA']<dias]

                #soma os valores em PF das demandas, agrupados por Ano
                df = df.groupby(["ANO_ASSIN"]).sum().reset_index()

                # seleciona as colunas "Ano" e "Esforço PF"
                X = np.array(df[["ANO_ASSIN"]])
                y = np.array(df[["Esforço PF"]])

                #calculando a correlação entre as variáveis
                r = np.corrcoef(X, y, rowvar=False)
                pearson = r[1,0]

                #Não trabalhar com o módulo do coef. de pearson abaixo de 0,5 para não trabalhar com re-
lação linear fraca.
                if (pearson>0.5 or pearson<-0,5):

                        #Testa todas as divisões possíveis para o período de tempo
                        for num in range(2,9,1):

                                kf = KFold(n_splits=num)
                                for train_index, test_index in kf.split(X):

                                        #Não aceita conjuntos menores q 2 elementes
                                        if (len(train_index)>1 and len(test_index)>1):

                                                X_train, X_test, y_train, y_test = X[train_index], X[test_in-
dex], y[train_index], y[test_index]

                                                #Fitting Simple Linear Regression to the set
                                                regressor = LinearRegression()
                                                regressor.fit(X, y)

                                                #Métrica padrão, que é a mais relevante para a tarefa de
Machine Learning
                                                r2_train = regressor.score(X_train, y_train)
                                                r2_test = regressor.score(X_test, y_test)

                                                #Exige r2_test seja positivo para ser melhor que uma reta
horizontal
                                                if (r2_test>0.9 and r2_test<=1): #or (r2_test<-0.5 and
r2_test>-1)
                                                        y_pred  = regressor.predict(X)
                                                        rmse_train                                      =
```

```
sqrt(mean_squared_error(y_train,y_pred[train_index]) )
                                                    rmse_test                                   =
sqrt(mean_squared_error(y_test,y_pred[test_index]) )
                                                    msle_test     =   mean_squared_log_error(y_test
,y_pred[test_index])
                                                    msle_train                                  =
mean_squared_log_error(y_train,y_pred[train_index])
                                                    #print("msle  treino  %0.2f" %  msle_train,  "msle
teste %0.2f" % msle_test)
                                                    rmse  = sqrt(mean_squared_error(y,y_pred) )
                                                    msle  = mean_squared_log_error(y,y_pred)
                                                    mae   = mean_absolute_error(y,y_pred)

                                                    #Seleciona o menor valor de RMSE
                                                    if          (msle_test<menor_msle_msle          or
menor_msle_msle==0):
                                                                menor_msle_rmse,
menor_msle_msle,   menor_msle_mae  = rmse, msle, mae
                                                                menor_msle_num,
menor_msle_dias, menor_msle_pearson= num, dias, pearson
                                                                menor_msle_r2_train,
menor_msle_r2_test              = r2_train, r2_test
                                                                menor_msle_pred,        menor_msle_X,
menor_msle_y    = y_pred, X, y

          menor_msle_train_ind,menor_msle_test_ind           = train_index, test_index

          menor_msle_msle_train,menor_msle_msle_test          = msle_train, msle_test

          menor_msle_rmse_train,menor_msle_rmse_test          = rmse_train, rmse_test

                                                    #Seleciona o Maior valor de R2 Teste
                                                    if          (r2_test>maior_R2_r2_test           or
maior_R2_r2_test==0):
                                                                maior_R2_rmse,          maior_R2_msle,
maior_R2_mae   = rmse, msle, mae
                                                                maior_R2_num,     maior_R2_dias, mai-
or_R2_pearson= num, dias, pearson
                                                                maior_R2_r2_train,    maior_R2_r2_test
= r2_train, r2_test
                                                                maior_R2_pred,     maior_R2_X,     mai-
or_R2_y     = y_pred, X, y
                                                                maior_R2_train_ind,maior_R2_test_ind
= train_index, test_index

          maior_R2_msle_train,maior_R2_msle_test         = msle_train, msle_test

          maior_R2_rmse_train,maior_R2_rmse_test         = rmse_train, rmse_test

                    del [[df]]
          #Fim do loop for

          print("---------- Melhor RMSE ----------")
          print("Pearson %0.3f" %menor_msle_pearson)
          print("dias = ", menor_msle_dias, ", split = ", menor_msle_num)
          print('Coeficiente de determinação da predição R2:')
```

```
print('1-Set de treino: %0.3f' % menor_msle_r2_train)
print('2-Set de teste: %0.3f'  % menor_msle_r2_test)
print("Índice de treino: ", menor_msle_train_ind)
print("Índice de teste: ",  menor_msle_test_ind)
print("RMSE Treino: %0.3f" % menor_msle_msle_train )
print("RMSE Teste:  %0.3f" % menor_msle_rmse_test)
print("MSLE Treino: %0.3f" % menor_msle_msle_train)
print("MSLE Teste:  %0.3f" % menor_msle_msle_test)
print("RMSE: %0.3f" % menor_msle_rmse)
print("MSLE: %0.3f" % menor_msle_msle)
print("MAE : %0.3f" % menor_msle_mae)


#Visualizando o conjunto de resultados
label1 = 'Valores Reais'
label2 = 'Predição - MSLE'
label3 = 'Predição - R2'
titulo = 'Esforço PF Entregue x Ano - KFold'
eixo_x = 'Ano'
eixo_y = 'Esforço PF'

plt.scatter(menor_msle_X, menor_msle_y, color = 'red', label=label1)
plt.plot(menor_msle_X, menor_msle_pred, color = 'blue', label=label2)
plt.grid(which='major', linestyle='--')
plt.minorticks_on()
plt.title(titulo)
plt.xlabel(eixo_x)
plt.ylabel(eixo_y)
plt.legend([label2, label1], loc=1)
plt.show()


print("---------- Melhor R2 ----------")
print("Pearson %0.3f" %maior_R2_pearson)
print("dias = ", maior_R2_dias, ", split = ", maior_R2_num)
print('Coeficiente de determinação da predição R2:')
print('1-Set de treino: %0.3f' % maior_R2_r2_train)
print('2-Set de teste: %0.3f'  % maior_R2_r2_test)
print("Índice de treino: ", maior_R2_train_ind)
print("Índice de teste: ",  maior_R2_test_ind)
print("RMSE Treino: %0.3f" % maior_R2_rmse_train )
print("RMSE Teste:  %0.3f" % maior_R2_rmse_test)
print("MSLE Treino: %0.3f" % maior_R2_msle_train )
print("MSLE Teste:  %0.3f" % maior_R2_msle_test)
print("RMSE: %0.3f" % maior_R2_rmse)
print("MSLE: %0.3f" % maior_R2_msle)
print("MAE : %0.3f" % maior_R2_mae)


#Visualizando o conjunto de resultados
plt.scatter(maior_R2_X, maior_R2_y, color = 'red', label=label1)
plt.plot(maior_R2_X, maior_R2_pred, color = 'blue', label=label3)
plt.grid(which='major', linestyle='--')
plt.minorticks_on()
plt.title(titulo)
plt.xlabel(eixo_x)
```

```
        plt.ylabel(eixo_y)
        plt.legend([label3, label1], loc=1)
        plt.show()

except IOError:
   print ("Arquivo ", location, " não foi encontrado")
```

5.3.4 Script Python: "4-TrainTestSplit-EntregasDeDemandasDuracaoCalculada.py"

a) Esforço em PF Entregue x Ano de abertura da demanda;

b) Código fonte;

```
#Demandas Entregues em tempo inferior a 2 anos
# Split_train_test
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_squared_log_error, mean_absolute_error
from math import sqrt

#Importando o dataset
location = "extracao_sgpti_03.csv"

try:
        menor_msle_rmse,    menor_msle_msle,   menor_msle_mae    = 0, 0, 0
        menor_msle_num,     menor_msle_dias,   menor_msle_pearson= 0, 0, 0
        menor_msle_r2_train, menor_msle_r2_test                  = 0, 0
        menor_msle_pred,    menor_msle_X,      menor_msle_y      = 0, 0, 0
        menor_msle_train_ind,menor_msle_test_ind                 = 0, 0

        maior_R2_msle,      maior_R2_msle,  maior_R2_mae    = 0, 0, 0
        maior_R2_num,       maior_R2_dias,  maior_R2_pearson= 0, 0, 0
        maior_R2_r2_train,  maior_R2_r2_test               = 0, 0
        maior_R2_pred,      maior_R2_X,     maior_R2_y      = 0, 0, 0
        maior_R2_train_ind, maior_R2_test_ind              = 0, 0

        print ("Lendo arquivo ", location, "...")

        #Testa várias durações de demandas
        for dias in range(20,3648,5):
                df = pd.read_csv(location,encoding='iso-8859-1',delimiter =';', quotechar='"', thousands='.',
decimal=',')

                #Determina intervalos para seleção
                df = df.loc[ df['ANO_ASSIN']<=2020]
                df = df.loc[ df['ANO_ASSIN']>=2012]

                #seleciona demandas com tempo de construção inferior a 1 ano
                df = df.loc[ df['TEMPO_ENTREGA']<dias]

                #soma os valores em PF das demandas, agrupados por Ano
                df = df.groupby(["ANO_ASSIN"]).sum().reset_index()

                # seleciona as colunas "Ano" e "Esforço PF"
```

```
                X = df[["ANO_ASSIN"]]
                y = df[["Esforço PF"]]

                #calculando a correlação entre as variáveis
                r = np.corrcoef(X, y, rowvar=False)
                pearson = r[1,0]

                if (pearson>0.5 or pearson<-0,5):
                        tam_treino = 0.55
                        tam_teste  = 0.45

                        #Testa todas as divisões possíveis para o período de tempo
                        for num in range(1,24,1):

                                tam_treino = tam_treino + 0.01
                                tam_teste  = tam_teste  - 0.01

                                #Divide os arrays em subconjuntos randômicos de treino e teste
                                X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=tam_trei-
no, test_size=tam_teste, random_state=42)

                                train_ind = list(X_train.index)
                                test_ind = list(X_test.index)

                                #Não aceita conjuntos menores q 2 elementos
                                if (len(train_ind)>1 and len(test_ind)>1):
                                        #Fitting Simple Linear Regression to the set
                                        regressor = LinearRegression()
                                        regressor.fit(X, y)

                                        #Métrica padrão, que é a mais relevante para a tarefa de Machine
Learning
                                        r2_train = regressor.score(X_train, y_train)
                                        r2_test = regressor.score(X_test, y_test)

                                        #Exige r2_test seja positivo para ser melhor que uma reta horizon-
tal
                                        if (r2_test>0.9 and r2_test<=1): #or (r2_test<-0.5 and r2_test>-1)
                                                y_pred    = regressor.predict(X)
                                                rmse_train                                              =
sqrt(mean_squared_error(y_train,y_pred[train_ind]) )
                                                rmse_test                                               =
sqrt(mean_squared_error(y_test,y_pred[test_ind]) )
                                                msle_test                                               =
mean_squared_log_error(y_test,y_pred[test_ind])
                                                msle_train                                              =
mean_squared_log_error(y_train,y_pred[train_ind])
                                                rmse  = sqrt(mean_squared_error(y,y_pred) )
                                                msle  = mean_squared_log_error(y,y_pred)
                                                mae   = mean_absolute_error(y,y_pred)

                                                #Seleciona o menor valor de MSLE
                                                if              (msle_test<menor_msle_msle              or
menor_msle_msle==0):
                                                        menor_msle_rmse,      menor_msle_msle,    me-
nor_msle_mae  = rmse, msle, mae
```

```
                                                    menor_msle_num,          menor_msle_dias,
menor_msle_pearson= num, dias, pearson
                                                    menor_msle_r2_train,     menor_msle_r2_test
= r2_train, r2_test
                                                    menor_msle_pred,         menor_msle_X,
menor_msle_y    = y_pred, X, y
                                                    menor_msle_train_ind,menor_msle_test_ind
= train_ind, test_ind
                                                    menor_msle_rmse_train,menor_msle_rmse_test
= rmse_train, rmse_test
                                                    menor_msle_msle_train,menor_msle_msle_test
= msle_train, msle_test
                                                    menor_msle_tam_treino,menor_msle_tam_tes-
te          = tam_treino, tam_teste

                                                #Seleciona o Maior valor de R2 Teste
                                                if (r2_test>maior_R2_r2_test or maior_R2_r2_test==0):
                                                    maior_R2_rmse,           maior_R2_msle,
maior_R2_mae   = rmse, msle, mae
                                                    maior_R2_num,            maior_R2_dias,
maior_R2_pearson= num, dias, pearson
                                                    maior_R2_r2_train, maior_R2_r2_test      =
r2_train, r2_test
                                                    maior_R2_pred,    maior_R2_X,    maior_R2_y
= y_pred, X, y
                                                    maior_R2_train_ind,maior_R2_test_ind     =
train_ind, test_ind
                                                    maior_R2_rmse_train,maior_R2_rmse_test
= rmse_train, rmse_test
                                                    maior_R2_msle_train,maior_R2_msle_test
= msle_train, msle_test
                                                    maior_R2_tam_treino,maior_R2_tam_teste
= tam_treino, tam_teste


                    del [[df]]
                    #Fim do loop for

        print("---------- Melhor MSLE ----------")
        print("Pearson %0.3f" %menor_msle_pearson)
        print("dias = ", menor_msle_dias, ", split = ", menor_msle_num)
        print("Tam. Treino = %0.3f" % menor_msle_tam_treino, ", Tam. Teste = %0.3f" %
menor_msle_tam_teste)
        print('Coeficiente de determinação da predição R2:')
        print('1-Set de treino: %0.3f' % menor_msle_r2_train)
        print('2-Set de teste: %0.3f'  % menor_msle_r2_test)
        print("Índice de treino: ", menor_msle_train_ind)
        print("Índice de teste: ",  menor_msle_test_ind)
        print("RMSE Treino: %0.3f" % menor_msle_rmse_train )
        print("RMSE Teste:  %0.3f" % menor_msle_rmse_test)
        print("MSLE Treino: %0.3f" % menor_msle_msle_train )
        print("MSLE Teste:  %0.3f" % menor_msle_msle_test)
        print("RMSE: %0.3f" % menor_msle_rmse)
        print("MSLE: %0.3f" % menor_msle_msle)
        print("MAE : %0.3f" % menor_msle_mae)
```

```
        #Visualizando o conjunto de resultados
        label1 = 'Valores Reais'
        label2 = 'Predição - MSLE'
        label3 = 'Predição - R2'
        titulo = 'Esforço PF Entregue x Ano - Split'
        eixo_x = 'Ano'
        eixo_y = 'Esforço PF'

        plt.scatter(menor_msle_X, menor_msle_y, color = 'red', label=label1)
        plt.plot(menor_msle_X, menor_msle_pred, color = 'blue', label=label2)
        plt.grid(which='major', linestyle='--')
        plt.minorticks_on()
        plt.title(titulo)
        plt.xlabel(eixo_x)
        plt.ylabel(eixo_y)
        plt.legend([label2, label1], loc=1)
        plt.show()


        print("---------- Melhor R2 ----------")
        print("Pearson %0.3f" %maior_R2_pearson)
        print("dias = ", maior_R2_dias, ", split = ", maior_R2_num)
        print("Tam. Treino = %0.3f" % maior_R2_tam_treino, ", Tam. Teste = %0.3f" % maior_R2_tam_teste)
        print('Coeficiente de determinação da predição R2:')
        print('1-Set de treino: %0.3f' % maior_R2_r2_train)
        print('2-Set de teste: %0.3f'  % maior_R2_r2_test)
        print("Índice de treino: ", maior_R2_train_ind)
        print("Índice de teste: ",  maior_R2_test_ind)
        print("RMSE Treino: %0.3f" % maior_R2_rmse_train )
        print("RMSE Teste:  %0.3f" % maior_R2_rmse_test)
        print("MSLE Treino: %0.3f" % maior_R2_msle_train )
        print("MSLE Teste:  %0.3f" % maior_R2_msle_test)
        print("RMSE: %0.3f" % maior_R2_rmse)
        print("MSLE: %0.3f" % maior_R2_msle)
        print("MAE : %0.3f" % maior_R2_mae)


        #Visualizando o conjunto de resultados
        plt.scatter(maior_R2_X, maior_R2_y, color = 'red', label=label1)
        plt.plot(maior_R2_X, maior_R2_pred, color = 'blue', label=label3)
        plt.grid(which='major', linestyle='--')
        plt.minorticks_on()
        plt.title(titulo)
        plt.xlabel(eixo_x)
        plt.ylabel(eixo_y)
        plt.legend([label3, label1], loc=1)
        plt.show()

except IOError:
   print ("Arquivo ", location, " não foi encontrado")
```

6. Apresentação dos Resultados

Começaremos apresentando os resultados utilizando o modelo de Canvas proposto por Vasandani.

Data Science Workflow Canvas*

Start here. The sections below are ordered intentionally to make you state your goals first, followed by steps to achieve those goals. You're allowed to switch orders of these steps!

Title: Aprendizado de máquina para gestão de projetos de software

1 Problem Statement
What problem are you trying to solve?
What larger issues do the problem address?

1) Gestão ineficiente de recursos
2) Longos prazos de atendimento

2 Outcomes/Predictions
What prediction(s) are you trying to make?
Identify applicable predictor (X) and/or target (y) variables.

Identificar correlações entre:
1) Esforço total (PF) demandado e ano de criação das demandas
y = Esforço em PF
x = Ano de criação das demandas

2) Esforço total (PF) entregue e ano de entrega das demandas
y = Esforço em PF
x = Ano entrega

3) Tempo de construção (dias) e ano de entrega das demandas
y = Tempo de construção
X = Ano entrega

3 Data Acquisition
Where are you sourcing your data from?
Is there enough data? Can you work with it?

1) Dados extraídos de base própria
a) demandas com esforço nulo já extraídas
b) tratamentos nos dados feitos em Python

2) Volume de dados adquirido suficiente

4 Modeling
What models are appropriate to use given your outcomes?

1) Dados contínuos
2) Labeled datasets
3) Sem preocupação com outliers
4) Divisão dos dados (treino/teste)
a) aletória
b) validação cruzada
5) Regressão Linear Simples

5 Model Evaluation
How can you evaluate your model's performance?

Métricas:

a) Coeficiente de correlação de Pearson (r>0,5 e r<-0,5)

b) Coeficiente de determinaçao R2 (maior valor, R2 > 0,9)

c) RMSE: raiz do erro quadrático médio (menor valor)

d) MSLE: erro logarítmico quadrático médio (menor valor)

6 Data Preparation
What do you need to do to your data in order to run your model and achieve your outcomes?

1) Extração de dados
a) exclusão de demandas com esforços em PF nulos

2) Tratamento/Processamento
a) Exclusão de demandas sem data de início
b) Exclusão de demandas sem data de término
c) Criação de novo dataset com dados tratados

Activation

When you finish filling out the canvas above, now you can begin implementing your data science workflow in roughly this order.

1 Problem Statement → 2 Data Acquisition → 3 Data Prep → 4 Modeling → 5 Outcomes/Preds → 6 Model Eval

Análise 1: Predição de esforço demandado por ano

1. Confirmação da correlação entre esforço demandado e período

   Os gráficos abaixo exibem a relação entre o esforço em PF (y) e o ano de solicitação (X). Em verde estão os valores reais, extraídos da base de dados, e a linha azul representa o resultado da regressão linear simples.

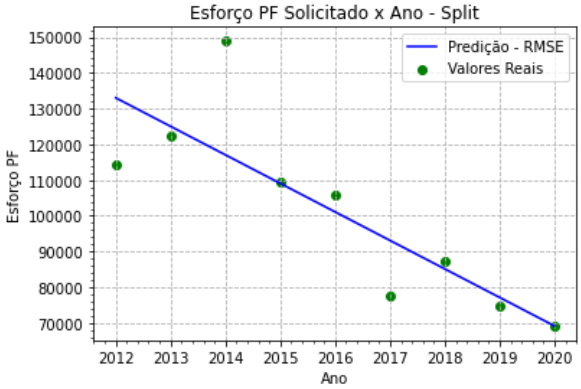


Fig 1: predição baseada no menor valor de RMSE

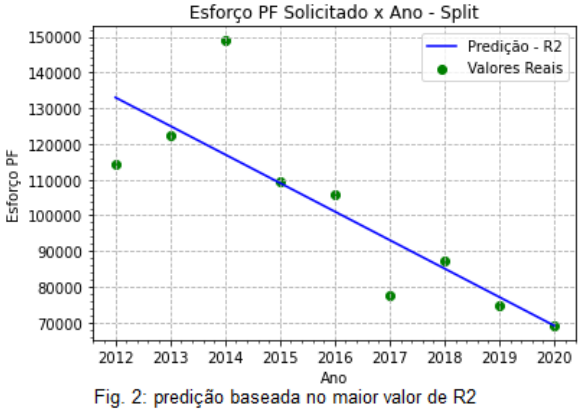


Fig. 2: predição baseada no maior valor de R2

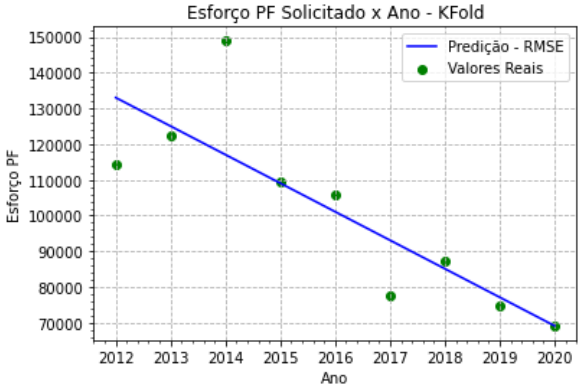


Fig 3: validação cruzada, menor RMSE

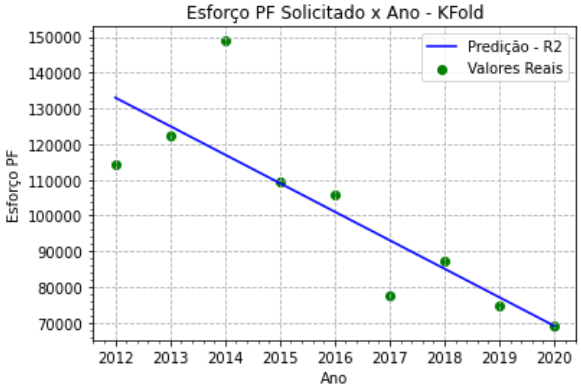


Fig. 4: validação cruzada, maior valor de R2

Demandas Solicitadas

As figuras 1 e 2 representam os dados divididos nos conjuntos de treinamento e teste, através do processo de divisão aleatória, com a função *train_test_split.*

As figuras 3 e 4 representam os dados divididos nos conjuntos de treinamento e teste através da validação cruzada, com a função *Kfold.split.*

Método para avaliação dos modelos:

1ª verificação – verificar a correlação entre as variáveis esforço PF e ano de solicitação das demandas, aceitando somente os modelos que apresentarem coeficiente de correlação de Pearson maior do que 0,5 ou menor que -0,5.

2ª verificação – aceitar somente os modelos que apresentarem o coeficiente de determinação R^2 superior a 0,90.

3ª verificação – considerar como melhores modelos os que apresentarem os melhores resultados de RMSE e R2 nos conjuntos de testes, segundo os seguintes critérios:

a) maior valor para o coeficiente de determinação R^2, visto que ele indica quanto o modelo foi capaz de explicar os dados coletados;

b) menor valor para RMSE, quanto menor o valor de RMSE, menor o risco correspondente à raiz quadrada do valor esperado do erro ou perda quadrática.

4ª verificação – avaliar qual o modelo apresentou melhores resultados – considerando, segundo os critérios abaixo, os resultados da comparação entre os valores de predição e os valores reais:

a) menor valor para RMSE, risco correspondente à raiz quadrada do valor esperado do erro ou perda quadrática.

b) menor valor para MSLE, valor esperado do erro ou perda logarítmica quadrática (quadrática), representa a diferença relativa entre o valor verdadeiro e o previsto, ou seja, a diferença percentual entre eles.

Resultados:

Métrica	**Condições**	**Kfold**		**Split**	
		Menor RMSE	**Maior R2 teste**	**Menor RMSE**	**Maior R2 teste**
Pearson	>0,5 ou < -0,5	-0,836	-0,836	-0,836	-0,836
R2 maior	Treino	0,396	0,396	0,622	0,622
	Teste (>0,9)	0,932	0,939	0,988	0,988
Split	Treino (0,22 a 0,78)			0,780	0,780
	Teste (0,78 a 0,22)			0,220	0,220
RMSE Treino	menor	16.507,560	16.507,560	15.270,283	15.270,283
RMSE Teste	menor	1.865,211	1.865,211	2.565,486	2.565,486
RMSE	menor	13.521,317	13.521,317	13.521,317	13.521,317
MSLE	menor	0,013	0,013	0,013	0,013
MAE	menor	8.738,852	8.738,852	8.738,852	8.738,852

Como podemos verificar na tabela acima e seguindo os nossos critérios de avaliação, temos que o método de validação cruzada usado com a função Kfold forneceu melhores resultados para a etapa de testes, mas os resultados finais, ao comparar a predição com os valores reais, foram semelhantes em todos os quatro modelos selecionados.

Análise 2: Predição de esforço entregue por ano

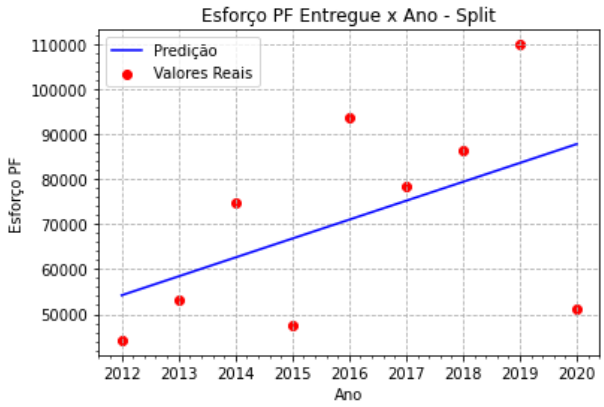


Fig. 5: somatório dos PF entregues por ano

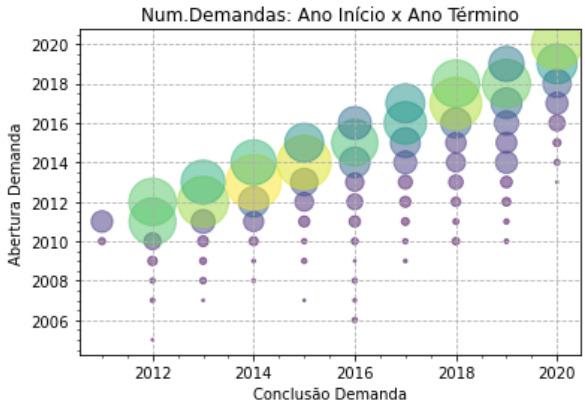


Fig. 6: número de demandas entregues por ano

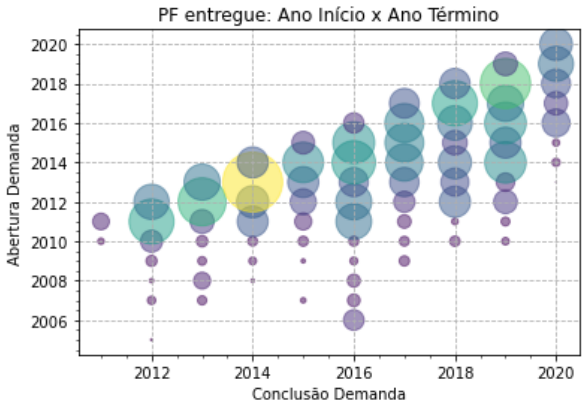


Fig. 7: PF entregues por ano

Diferentemente das solicitações, as entregas sofrem com repriorizações e adiamentos. Na Figura 1, o gráfico nos mostra uma fraca correlação entre o esforço entregue (PF) e o ano de conclusão das demandas.

Na Figura 2, no gráfico de dispersão exibido, podemos observar que – a cada ano – além de demandas abertas no ano corrente, são entregues demandas de

vários outros anos. Isso faz com que o esforço de construção de uma

Demandas Entregues:

A figura 5 representa o somatório de todas as entregas de 2012 a 2020. A qualidade do ajuste de teste do modelo, medida pelo coeficiente de determinação, R^2, apresentou 55% da variância da variável dependente, a partir dos regressores (variáveis independentes) incluídos no modelo linear. O que indicou a baixa qualidade do ajuste obtido.

A figura 6 e 7 exibem gráficos de dispersão que representam o ano de abertura (eixo y) e o ano de conclusão (eixo X) das demandas. Na figura 6,os

tamanhos e cores dos marcadores refletem o número de demandas concluídas. Mas na figura 7, os tamanhos e cores dos marcadores refletem o esforço em PF de demandas concluídas.

Podemos observar que o maior volume de entregas anuais está entre as demandas abertas no ano da própria entrega e o ano anterior. Mas demandas de vários anos são concluídas ao longo do tempo. Ou seja, demandas são solicitadas, porém dependendo da avaliação dos usuários, podem ser colocadas numa ordem de prioridade diferente da original, o que impacta em sua conclusão.

Diante do cenário exibido acima, a hipótese considerada foi a de que demandas com tempo de construção inferior a dois anos estariam sujeitas a menores efeitos das alterações nas prioridades e nos cronogramas.

Sendo assim, geramos modelos de aprendizado de máquina baseado para estabelecer o tempo de entrega abaixo do qual estaríamos menos suscetíveis a alterações que influíssem na relação entre o tempo de entrega e o número de pontos de função.

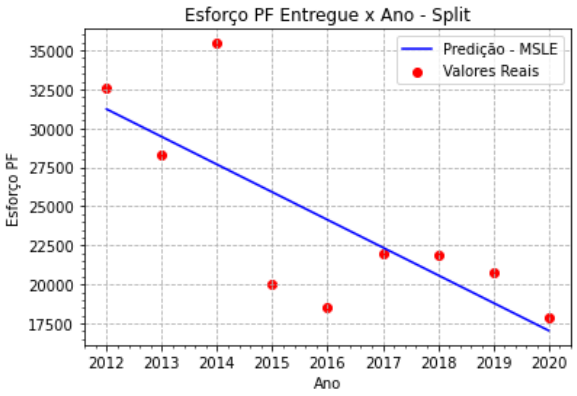


Fig. 8: demandas com tempo de entrega inferior a 375 dias

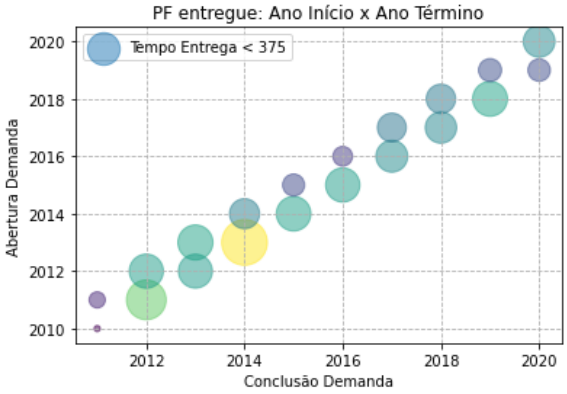


Fig. 9: PF das demandas com tempo de entrega inferior a 375 dias

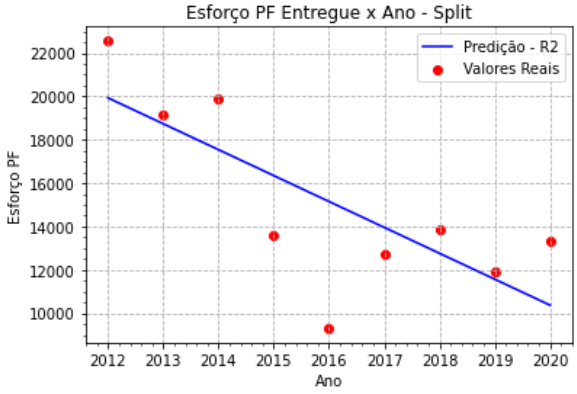


Fig. 10: demandas com tempo de entrega inferior a 245 dias

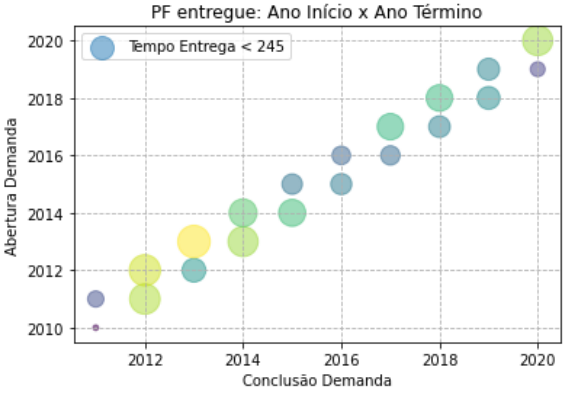


Fig. 11: PF das demandas com tempo de entrega inferior a 245 dias

As figuras 8 e 10 representam os dados divididos nos conjuntos de treinamento e teste, através do processo de divisão aleatória, com a função *train_test_split.*

A figura 9 representa o esforço em PF entregue por ano, considerando somente as demandas com tempo de construção menor que 375 dias.

A figura 11 representa o esforço em PF entregue por ano, considerando somente as demandas com tempo de construção menor que 245 dias.

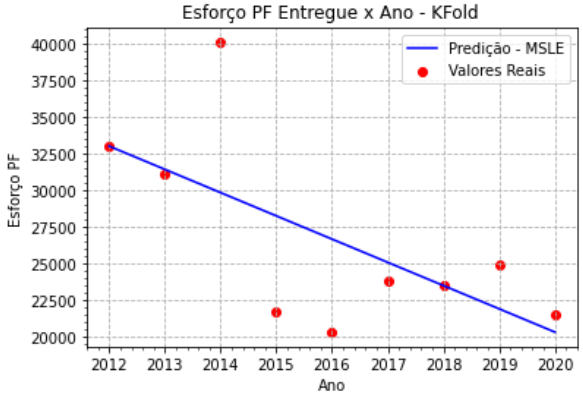


Fig. 12: demandas com tempo de entrega inferior a 415 dias

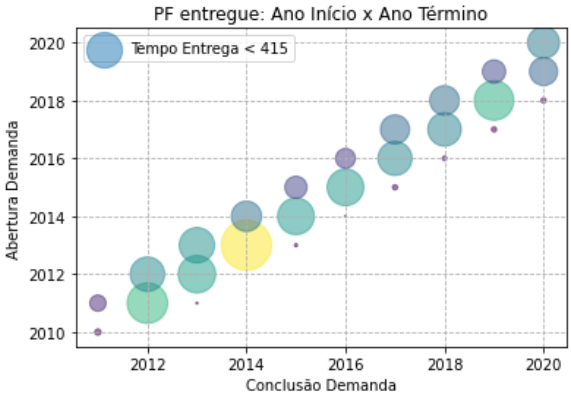


Fig. 13: PF das demandas com tempo de entrega inferior a 415 dias

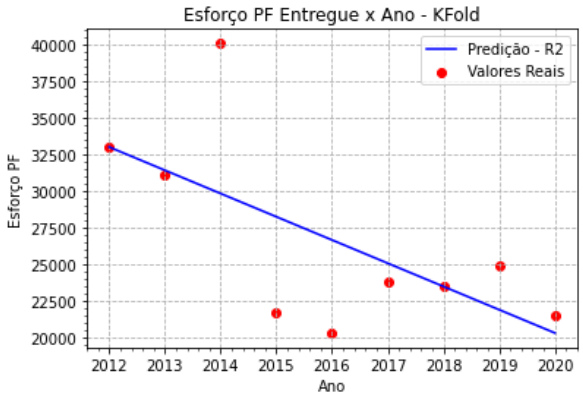


Fig. 14: validação cruzada, demandas com tempo de entrega inferior a 415 dias

As figuras 12 e 14 representam os dados divididos nos conjuntos de treinamento e teste, através do processo de validação cruzada, com a função KFold.

A figura 13 representa o esforço em PF entregue por ano, considerando somente as demandas com tempo de construção menor que 415 dias.

Método para avaliação dos modelos:

1ª verificação – verificar a correlação entre as variáveis esforço PF e ano de entrega das demandas, aceitando somente os modelos que apresentarem coeficiente de correlação de Pearson maior do que 0,5 ou menor que -0,5.

2ª verificação – aceitar somente os modelos que apresentarem o coeficiente de determinação R^2 superior a 0,90**.**

3ª verificação – testar várias durações de demandas para verificar qual faixa de tempo de construção é menos afetada pelas repriorizações;

4ª verificação – considerar como melhores modelos os que apresentarem os melhores resultados nos conjuntos de testes, segundo os seguintes critérios:

a) maior valor para R^2, visto que ele indica quanto o modelo foi capaz de explicar os dados coletados;

b) menor valor para MSLE, uma vez que esta métrica é pouco afetada pelas alterações nos números de demandas testadas.

5ª verificação – considerar como melhores modelos os que apresentarem os melhores resultados ao comparar os valores de predição e os valores reais, segundo os critérios abaixo:

a) menor valor para MSLE, valor esperado do erro ou perda logarítmica quadrática (quadrática), representa a diferença relativa entre o valor verdadeiro e o previsto, ou seja, a diferença percentual entre eles.

b) menor valor para RMSE, risco correspondente à raiz quadrada do valor esperado do erro ou perda quadrática.

Resultados:

Métrica	**Condições**	**Kfold**		**Split**	
		Menor MSLE	**Maior R2 teste**	**Menor MSLE**	**Maior R2 teste**
Pearson	>0,5 ou < -0,5	-0,655	-0,655	-0,761	-0,751
R2 maior R2	Treino	0,272	0,272	0,386	0,474
	Teste (>0,9)	0,932	0,932	0,923	0,988
Kfold	Split(2 a 8)	8	5		
	Dias (20 a 3.647)	415	415	375	245
Split	Treino (0,22 a 0,78)			0,660	0,780
	Teste (0,78 a 0,22)			0,340	0,220
MSLE Treino	menor	0,036	0,036	0,041	0,054
MSLE Teste	menor MSLE	0,0	0,0	0,003	0,001
RMSE	menor	4.727,190	4.727,190	3898.425	2.710,525
MSLE	menor MSLE	0,028	0,028	0,024	0,043
MAE	menor	3.237,236	3.237,236	2928.462	2.192,302

Como podemos verificar na tabela de resultados acima e seguindo os nossos critérios de avaliação, temos que as demandas com tempo de construção inferior a 375 dias sofrem menos os efeitos das repriorizações e, por isso, apresentam melhor relação entre esforço em PF e tempo de entrega.

7. Links

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

Link para o vídeo: https://youtu.be/1iUtw7e33pE

Link para o repositório: https://github.com/alexandrecord/repositorio-ciencia-de-dados.git

Usuário: alexandrecord

Repositório: "repositorio-ciencia-de-dados"

Arquivo	**Descrição**
1-Gera_Dataset3.py	Script em Python para geração do Dataset 3
2-kfold-EsforcoPFSolicitado.py	Exibe o somatório de pontos de função das demandas que foram solicitadas e os modelos gerados utilizam Kfolds para validação cruzada.
2-TrainTestSplit-2-EsforcoPFSolicitado.py	Exibe o somatório de pontos de função das demandas que foram solicitadas e os modelos gerados utilizam *train_test_split* divisão aleatória em conjuntos de treinamento e teste.
3-EsforcoPFEntregue.py	Exibe o somatório de pontos de função das demandas que foram concluídas.
4-kfold-EntregasDeDemandasDuracaoCalculada.py	Exibe o somatório de pontos de função das demandas que foram concluídas com tempo de construção inferior ao calculado através dos modelos gerados a partir de validação cruzada, usando Kfolds.
4-TrainTestSplit-EntregasDeDemandasDuracaoCalculada.py	Exibe o somatório de pontos de função das demandas que foram concluídas com tempo de construção inferior ao calculado através dos modelos gerados a partir de divisão aleatória em conjuntos de treinamento e teste.
5-NumDemandasSolicitEntregPorAno.py	Exibe a lista de demandas entregues com base no ano de início e término. O tamanho e a cor das bolhas representam a proporção do número de demandas na mesma situação.
6-PFSolicitEntregPorAno.py	Exibe a lista de demandas entregues com base no ano de início e término. O tamanho e a cor das bolhas representam a soma dos pontos de função das demandas na mesma situação.
7a-EntregasDurac245.py	Exibe a lista de demandas entregues e que apresentaram tempo de construção inferior 245 dias. O tamanho e a cor das bolhas representam a proporção da soma de pontos de função das demandas, por ano de conclusão.
7b-EntregasDurac375.py	Exibe a lista de demandas entregues e que apresentaram tempo de construção inferior 375 dias. O tamanho e a cor das bolhas representam a

	proporção da soma de pontos de função das demandas, por ano de conclusão.
7c-EntregasDurac415.py	Exibe a lista de demandas entregues e que apresentaram tempo de construção inferior 415 dias. O tamanho e a cor das bolhas representam a proporção da soma de pontos de função das demandas, por ano de conclusão.
8-EsforcoTempoEntrega.py	Exibe um gráfico de dispersão com os pontos de função das demandas que foram concluídas, sem agrupamento.
9-TesteDeMetricas.py	Exibe um estudo feito para melhor compreensão das métricas utilizadas para medir os modelos gerados.
Apresentacao.odp	Arquivo contendo as telas para a apresentação de 5 minutos.
extracao_sgpti_01.csv	Dataset 1
extracao_sgpti_datas_01.csv	Dataset 2
extracao_sgpti_03.csv	Dataset 3

REFERÊNCIAS

Referências relacionadas às tecnologias/metodologias usadas.

AAKER, D. A.; KUMAR, V.; DAY, G. S. **Pesquisa de marketing. 2. ed.** São Paulo: Atlas, 2007.

GAITHER, N.; FRAZIER, G. **Administração da produção e operações. 8. ed.** São Paulo: Pioneira Thomson, 2002.

TUBINO, D. F. **Planejamento e controle da produção: teoria e prática. 2. ed.** São Paulo: Atlas, 2009.

PINHEIRO, WAGNER. **Estatística Geral.** Pós-gradução Ciência de Dados: PUC/MG, 2019.

APÊNDICE

Programação/Scripts

Arquivo “8-EsforcoTempoEntrega.py”

```
#Esforço x Tempo Entrega
import matplotlib.pyplot as plt
import pandas as pd

#Importando o dataset
location = "extracao_sgpti_03.csv"

df = pd.read_csv(location,encoding='iso-8859-1',delimiter =';', quotechar='"', thousands='.', decimal=',')

#Determina intervalos para seleção
df = df.loc[ df['ANO_ASSIN']<=2020]
df = df.loc[ df['ANO_ASSIN']>=2012]

#soma os valores em PF das demandas, agrupados por Ano
df = df.groupby(["TEMPO_ENTREGA","Esforço PF"]).count().reset_index()
df = df[['TEMPO_ENTREGA','Esforço PF','ANO_ASSIN']].groupby(by=['TEMPO_ENTREGA','Esforço PF']).count().reset_index()

# seleciona as colunas "Ano" e "Esforço PF"
X = df[["TEMPO_ENTREGA"]]
y = df[["Esforço PF"]]
z = df["ANO_ASSIN"]

#Visualizando o conjunto de resultados
color= (z**2)
area = (z**2)
plt.scatter(X, y, s=area, c=color, alpha=0.5)

#Visualizando o conjunto de resultados
#plt.scatter(X, y, color = 'blue')
plt.grid(which='major', linestyle='--')
plt.minorticks_on()
plt.title('Esforço x Tempo Entrega')
plt.xlabel('Tempo Entrega')
plt.ylabel('Esforço')
plt.show()
```