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

Romulos Machado
Phillip Furtado

ANÁLISE DE DADOS DE TEMPERATURA AMBIENTE VERSUS LOGS DE ALERTA EM UM DATA CENTER

Belo Horizonte
2020

Romulos Machado
Phillip Furtado

ANÁLISE DE DADOS DE TEMPERATURA AMBIENTE VERSUS LOGS DE ALERTA EM UM DATA CENTER

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
2020

SUMÁRIO

1. Introdução 4
1.1. Contextualização 4
1.2. O problema proposto 4
2. Coleta de Dados 5
3. Processamento/Tratamento de Dados 7
4. Análise e Exploração dos Dados 11
5. Criação de Modelos de Machine Learning 14
6. Apresentação dos Resultados 18
7. Links 20

1. Introdução

1.1. Contextualização

Para que data centers e seus respectivos equipamentos (servidores, grandes unidades de armazenamento e ativos de rede) consigam atender à necessidade dos seus usuários, é imperativo que permaneçam em funcionamento constante 24 horas por dia, 7 dias por semana. Consequentemente, assim como a maioria dos equipamentos que trabalham com processamento de informações, a temperatura nos data centers está diretamente relacionada com o bom funcionamento dos servidores, sendo fundamental que eles não superaqueçam durante a execução de suas atividades. Pois, caso isso aconteça, estes equipamentos correm o risco de se auto desligarem ou queimarem, o que pode causar uma parada não programada do data center, consequentemente a interrupção dos serviços por ele provido. Com isso, se faz necessário uma análise estatística e preditiva dos eventos de elevação da temperatura de data centers e suas consequências no funcionamento dos equipamentos.

1.2. O problema proposto

No que concerne a alta disponibilidade de serviços, identificar se registros de eventos de alta temperatura dentro de um data center estão ou não relacionados a danos registrados através de logs, a partir de dados coletados sobre o próprio ambiente e usando conceitos de ciências de dados para analisá-los, ajuda não somente a entender a extensão de problemas em equipamentos, como também a definir sistematicamente que determinadas condições estabelecidas, possivelmente irão gerar impactos negativados decorrentes e, por isso, deve ser evitados ao máximo.

Para desenvolvedor este trabalho, foram utilizados dados de temperatura de ambiente registrados por sensores internos de 30 servidores Dell, de modelo R720, de um determinado data center de Manaus, entre os anos de 2014 a 2020, com o objetivo de compará-los com dados de problemas ou erros de hardware apresentados dentro desse mesmo período. Onde esta comparação determinará se existe correlação entre os eventos de superaquecimento do data center e os problemas registrados nos equipamentos, dentro de um mesmo intervalo de tempo.

2. Coleta de Dados

A base de dados é constituída por informações de 30 servidores de mesma especificação e característica de hardware e, partir deles, estudar dois tipos de logs de cada um desses servidores, que foram armazenados ao longo de aproximadamente 04 anos, através de um dispositivo chamado de *Integrated Dell Remote Access Controller* (IDRAC). Esse componente também instalado dentro de cada um desses servidores com total independência de acesso e isolado do sistema operacional, permitiu que ficassem registrados vários alertas relacionados às mais diversas falhas relativas ao hardware dessas máquinas, assim como também registrou dados de temperatura externa a estes servidores, ao longo desses 04 anos.

Sendo assim, como foi indicado no parágrafo anterior, tivemos a oportunidade de trabalhar com dois tipos de logs:

- Log de Alertas (log_servidorXX.csv): contendo dados sobre alertas de falhas do equipamento, tendo a seguinte estrutura:

Tabela 1

Nome da coluna/atributo	Descrição	Tipo
Severity	Indica qual a severidade do alerta (“Normal”, “Warning”,” Critical”)	Texto
Date	Indica a data e horário do alerta	Data/horário
Description	Indica a mensagem de alerta	Texto

- Log de Temperaturas (servidorXX_inlettemp.csv): contendo dados de temperaturas registrados de hora em hora dos equipamentos, tendo a seguinte estrutura:

Tabela 2

Nome da coluna/atributo	Descrição	Tipo
Average	Indica a temperatura média em Celsius, dentro do intervalo de tempo.	Numérico
Peak	Indica o maior pico de temperatura em Celsius, dentro do intervalo de tempo	Numérico

Time	Indica a data e hora a qual foi feito a captura de temperaturas, usada para os cálculos de pico e média	Data/horário

3. Processamento/Tratamento de Dados

Para obtermos uma base mais consistente de dados e facilitar as associações, resolvemos adicionar o atributo “Server”, de tipo “Texto”, em cada um dos logs de alertas e também nos logs de temperaturas, para preenchê-los com as respectivas informações de nome dos servidores.

Isto facilitou o processo de criação de um *dataset* unificado, contendo todos os registros de temperaturas referentes aos 30 servidores. Da mesma forma, foi criado um *dataset* unificado de registros de alertas de todos os respectivos servidores. Respectivamente, o *dataset* de temperatura foi nomeado de “servidores-inlettemp.csv” (contendo 1541376 linhas, 4 colunas e sem valores nulos ou vazios) e o *dataset* de logs de alertas foi chamado de “logs_servidores.csv” (contendo 6368 linhas, 4 colunas e sem valores nulos ou vazios).

Usando a ferramenta *Jupyter Notebook*, e com isso facilitar as nossas interações com os *datasets* através do Python, assim tratamos e processamos os mesmos conforme a sequência descrita a seguir:

Carregar arquivos originais de temperatura e log de servidores.

```
temp = pd.read_csv('datasets_consolidados/servidores-inlettemp.csv')
logs = pd.read_csv('datasets_consolidados/logs_servidores.csv')
```

Figura 1

Se avalia a ocorrência de linhas com valores nulos nos *datasets* de temperaturas e logs de alertas.

```
temp.isnull().sum()
```

```
Server         0
Severity       0
Date           0
Description    0
Time           0
DateOnly       0
LogTime        0
dtype: int64
```

```
logs.isnull().sum()
```

```
Server         0
Severity       0
Date           0
Description    0
Time           0
DateOnly       0
LogTime        0
dtype: int64
```

Figura 2

É necessário realizar alguns procedimentos de ajustes e limpeza nos dados, onde o primeiro passo é remover as não-ocorrências de temperatura no *dataset* de temperaturas, pois evidencia que o servidor estava desligado e isso é caracterizado quando o atributo *Average* indica o valor de -128. Esse tipo de registro de máquina desligada não é relevante para o nosso estudo, pois não fica viável comparar temperatura com qualquer outra variável.

```
temp.drop(temp.index[temp['Average'] == -128], inplace = True)
```

Figura 3

O segundo passo é transformar o atributo Time em tipo *date* e criar um atributo *DateOnly* para posteriores comparações entre bases. Esse novo atributo irá nos ajudar a indexar e juntar as bases. Também neste passo de ajuste é feita a ordenação da base de temperaturas em função do nome do servidor e do dia/hora.

```
temp['Time']= pd.to_datetime(temp['Time'], format="%a %b %d %H:%M:%S %Y")
temp['DateOnly'] = temp['Time'].dt.date
temp.sort_values(by=['Server', 'Time'])
```

Figura 4

Agrupar as linhas em função do nome de servidor e data, além de sumarizar através do agrupamento os atributos *Average* e *Peak* com informações de Mínimo, Máximo, Média, Mediana, Variância e Desvio Padrão. Para isso, é criado um novo *dataset* resultante chamado *temp_summary* que agora manter uma ocorrência de evento de temperatura por dia/servidor sem que os valores originais de *Average* e *Peak* sejam perdidos com a utilização das variáveis estatísticas. Essa operação de *groupby* adiciona uma nova linha de índices (nomes de colunas), pra isso é necessário realizar um ajuste para se manter uma só linha de índice sem perda de identidade das colunas.

```
temp_summary = temp.groupby(['Server','DateOnly'])[['Average','Peak']].agg(['min', 'max','mean','median','var','std']).reset_index()
temp_summary.columns = ["_".join(x) for x in temp_summary.columns.ravel()]
temp_summary.rename(columns={'Server_':'Server','DateOnly_':'DateOnly'},inplace=True)
```

Figura 5

No *dataset* de logs é necessário também realizar um processo limpeza, para isso, é necessário remover as ocorrências da base de logs aonde a data tem formato inválido pois indica um processo de reinicialização da máquina (tecnicamente

chamado de System Boot) , após desligamento, e transforma o atributo Date em tipo Date. Além de, criar o atributo DateOnly e ordernar a base de logs em função do nome do servidor e o dia do registro.

```
logs.drop(logs.index[logs['Date'] == 'System Boot'], inplace = True)
logs['Date']= pd.to_datetime(logs['Date'], format="%a %b %d %Y %H:%M:%S")
logs['DateOnly'] = logs['Date'].dt.date
logs.sort_values(by=['Server', 'Date'])
```

Figura 6

Agora com ambos *datasets* processados é necessário combinar as duas bases em uma só, usando como índice o nome do servidor e o dia do registro. Neste caso levou-se em consideração o fato de que nem todas as vezes que houve um registro de temperatura, necessariamente, houve um registro de alerta. Para isso, é criado um novo *dataset* resultante chamado *merged*.

```
merged = pd.merge(temp_summary,logs, on=['Server', 'DateOnly'], how='left')
```

Figura 7

O que se percebeu no novo *dataset* gerado no passo anterior, é a frequência de valores vazios, para é realizada a substituição dos valores vazios (ausência de registro de log) da coluna *Severity* com o valor *NoneAlert*, indicando que nessa data não houve registros de alerta. Posteriormente, isso irá nos ajudar a efetuar comparações entre dados de temperatura e severidade de alertas.

```
merged.Severity.fillna("NoneAlert", inplace=True)
```

Figura 8

Adiciona o atributo categórico *TempSala* que indica se a sala de servidores estava quente, esquentando ou fria, baseado nos valores de *Peak_max*.

```
merged.loc[merged['Peak_max'] <= 26, 'TempSala'] = 'SalaFria'
merged.loc[(merged['Peak_max'] > 26) & (merged['Peak_max'] <= 33), 'TempSala'] = 'SalaEsquentando'
merged.loc[merged['Peak_max'] > 33, 'TempSala'] = 'SalaQuente'
```

Figura 9

O atributo *Severity* será útil ao executar alguns modelos como o PCA na base *merged*, mas a maioria dos modelos requer que atributos categóricos sejam transformados em inteiros. Para isso, utilizaremos a técnica de *One-Hot Encoding*, na qual as categorias se transformaram em colunas (variáveis) onde o número 1 representa o valor afirmativo e o 0 negativo.

```
merged.Severity.unique()
```

```
array(['Normal', 'Critical', 'NoneAlert', 'Warning'], dtype=object)
```

Figura 10

Cria-se então um *dataset* com os valores possíveis e utilizamos a função *pd.get_dummies()* para gerar as colunas numéricas.

```
df = pd.DataFrame({'Severity': ['NoneAlert','Normal', 'Critical', 'Warning']})
merged = pd.concat([merged,pd.get_dummies(merged['Severity'], prefix='Severity')],axis=1)
```

Figura 11

Semelhante ao tratamento dado ao atributo *Severity*, também fizemos uso do mesmo processo no atributo *TempSala*.

```
merged.TempSala.unique()
```

```
array(['SalaFria', 'SalaEsquentando', 'SalaQuente'], dtype=object)
```

```
df = pd.DataFrame({'TempSala': ['SalaFria','SalaEsquentando','SalaQuente']})
merged = pd.concat([merged,pd.get_dummies(merged['TempSala'], prefix='TempSala')],axis=1)
```

Figura 12

Com estes passos todos executados com sucesso, exportamos as bases para arquivos do tipo CSV, os quais serão utilizados no próximo tópico (merged.csv) deste trabalho e também usados como evidências de resultados de nosso tratamento de dados (temperatura.csv e logs.csv).

```
temp.to_csv(r'datasets_consolidados/temperatura.csv', index = False)
logs.to_csv(r'datasets_consolidados/logs.csv', index = False)
merged.to_csv(r'datasets_consolidados/merged.csv', index = False)
```

Figura 13

4. Análise e Exploração dos Dados

Como primeiro passo no processo de exploração de dados verificamos a frequência de alguns atributos importantes, como é o caso do atributo *Severity* com valor **NoneAlert** o qual apresenta a maior frequência de toda. Isso era o esperado, pois, como foi mencionado na seção de tratamento de dados, a frequência de linhas de medidas de temperatura sem ocorrências de logs de alerta é muito alta.

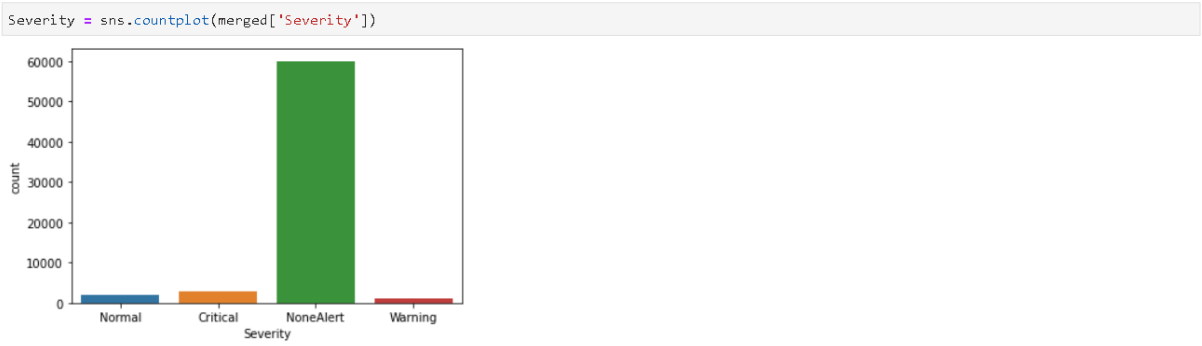

Figura 14

Podemos verificar também que a frequência de eventos de **SalaFria** é bem maior que **SalaEsquentando** e **SalaQuente**, respectivamente, o que é o esperado de um ambiente de data center com controle de temperatura adequado.

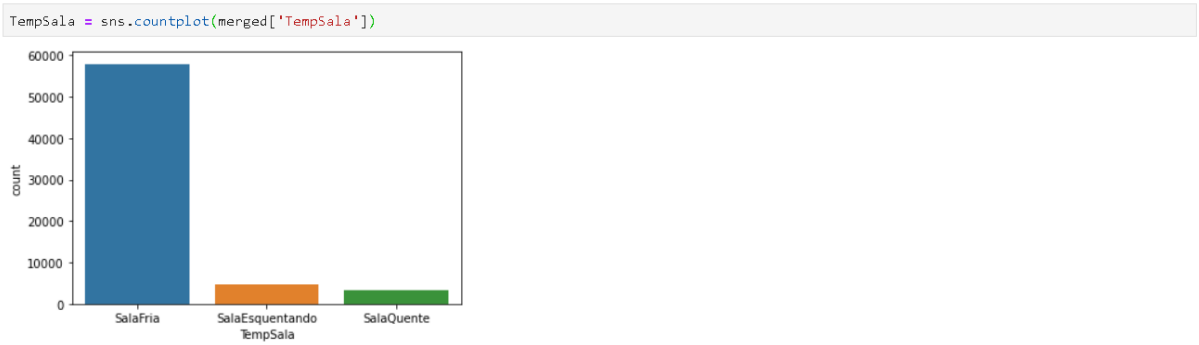

Figura 15

Nos dois gráficos seguintes (Figura 16 e Figura 17) é possível verificar a distribuição de temperaturas, em pico máximo (Peak_max) e média (Average_mean), para cada um dos servidores. O que se pode perceber é o que o pico máximo não chega na margem dos 10 a 15 graus normalmente alçando pela temperatura média. Além disso, é possível visualizar a existência de uma sobreposição densa na faixa de 15 a 25 graus entre pico e média, aonde podemos assumir que boa parte do tempo o pico máximo se mantém nesta faixa de temperatura.

```
sns.catplot(x="Peak_max", y="Server", data=merged);
```

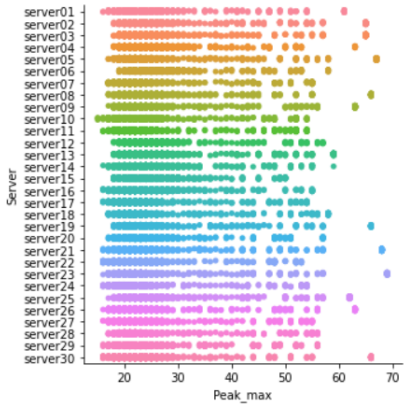

Figura 16

```
sns.catplot(x="Average_mean", y="Server", data=merged);
```

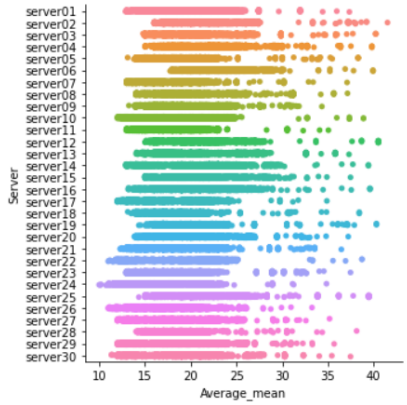

Figura 17

No gráfico mostrado na Figura 18, verifica-se a relação de picos máximos de temperatura e a severidade de um log de alerta, o qual remete à principal hipótese deste trabalho que é: **existe relação entre alta temperatura e logs de alerta com erro (Critical)?** Essa hipótese será verificada no próximo capítulo.

Ainda nesse mesmo gráfico, podemos verificar que o tipo de log de alerta *NoneAlert* permanece nas temperaturas mais baixas. Outro aspecto importante notado é que a distribuição dos tipos *Normal*, *Critical* e *Warning* possuem semelhanças gerais, como exemplifica a ocorrência dos três tipo, mesmo em altas temperaturas.

```
sns.catplot(x="Peak_max", y="Severity", data=merged);
```

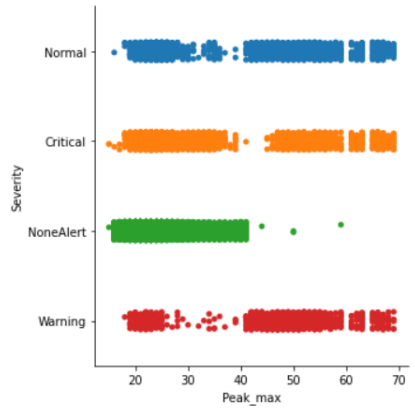

Figura 18

A matriz de correção é uma estatística muito utilizada no mercado, pois é ela ajuda a entender melhor como os atributos se relacionam e ajuda também no processo de manutenção de umas das máximas da estatística: correlação não implica em causalidade.

Observando a matriz de correlação mostrada na Figura 19, é possível notar que os principais atributos Average_mean e Peak_max tem uma correlação baixa com Severity_Critical, o que não significa que não seja causa, apesar dos gráficos anteriores indicarem que o atributo Severity_Critical é influenciado pela temperatura em diversos níveis altos e baixos.

```
corrs = merged.corr()
mask = np.zeros_like(corrs)
mask[np.triu_indices_from(mask)] = True
sns.heatmap(corrs, cmap='Spectral_r', mask=mask, square=True, vmin=-.4, vmax=.4)
plt.title('Matriz de Correlação')
```

```
Text(0.5, 1.0, 'Matriz de Correlação')
```

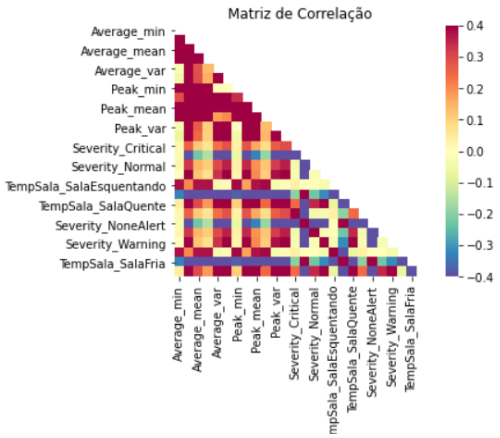

Figura 19

5. Criação de Modelos de Machine Learning

Para compreender os modelos de aprendizado de máquina aplicados ao *dataset* do trabalho, utilizamos o PCA e árvore de decisão como principais modelos neste estudo. O PCA (*principal component analisys*) é método não-supervisionado que ajuda a visualizar os dados em um plano de duas dimensões através da redução de dimensões de um conjunto de dados sem perda das características dos dados necessariamente. Diferente de outros modelos, o PCA não tem objetivo de prever o comportamento dos dados, mas sim aprender sobre a relação entre os atributos e as classes.

O modelo original tem uma oferta limitada de atributos, para executar o PCA utilizamos os atributos *Average_max*, *Average_mean* e *Peak_max*.Conforme o documento de instruções para o TCC, essa etapa não é obrigatória, mas é fortemente recomendada. Caso você crie modelos de *Machine Learn*ing em seu projeto, nessa seção você irá descrever as ferramentas utilizadas. Se você utilizou ferramentas visuais como Knime e Rapid Miner, coloque aqui um print do seu modelo. Caso você tenha escrito scripts em Python, por exemplo, coloque aqui o seu script. Explique as *features* utilizadas, justifique a escolha por determinado modelo, os parâmetros utilizados, etc. Foi criado um *dataset* X somente os atributos de teste e um *dataset* Y com as classes, neste caso utilizamos o atributo *Severity*. O PCA requer que os atributos estejam normalizados para isso utilizarmos a função *fit_transform(x)*.

```
features = ['Average_max','Average_mean','Peak_max']
# Separating out the features
x = merged.loc[:, features].values
# Separating out the target
y = merged.loc[:,['Severity']].values

x = StandardScaler().fit_transform(x)
pd.DataFrame(data = x, columns = features).head()
```

	Average_max	Average_mean	Peak_max
0	-0.020705	0.423840	-0.284171
1	0.927676	0.993522	0.882062
2	0.927676	0.993522	0.882062
3	0.927676	0.566260	0.882062
4	0.295422	-0.252658	1.027841

Figura 20

No método PCA, a relação é quantificada através da criação dos “eixos principais”, no caso utilizamos dois eixos “*principal component 1*” e “*principal*

component 2" que representam a redução dimensional dos atributos (*features*) selecionadas no passo anterior.

```
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
principalDf.head(5)
```

	principal component 1	principal component 2
0	0.037152	-0.502692
1	1.608622	-0.189491
2	1.608622	-0.189491
3	1.389178	0.167883
4	0.660630	0.793672

Figura 21

A partir do código mostrado na Figura 22, o gráfico abaixo (Figura 23) apresenta os três atributos em um espaço de duas dimensões, agrupados por tipo de log de severidade (*Severity*), podemos verificar a distribuição de pontos *Critical* agrupados em duas regiões, um grupo superior a direita mostrando que pode existir alguma relação entre valores altos e esse tipo de alerta, mas um outro grupo de *Critical* se forma em meio ao logs de *Normal* e *NoneAlert* na região inferior esquerda, o que pode representar também, que picos de temperatura são raros e os valores registrados são semelhantes a valores normais de medida de temperatura na sala de servidor.

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 Component PCA', fontsize = 20)


targets = [ 'NoneAlert','Normal', 'Warning','Critical']
colors = [ 'b','g', 'y','r']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['Severity'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```

Figura 22

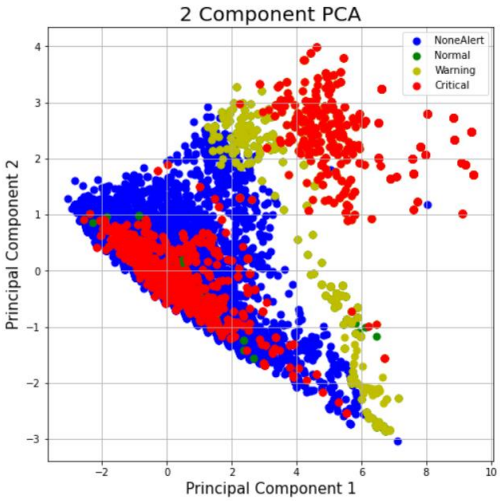

Figura 23

Para realizar o processo de predição para a hipótese principal do trabalho, que é verificar a correlação entre alta temperatura e logs de erro (*Critical*), decidimos utilizar o classificador Arvore de Decisão que é um modelo supervisionado tradicional muito utilizado em projetos de data Science. Em resumo, o classificador cria um modelo baseado em fluxo de decisões baseado na base de treino informada ao modelo.

O primeiro passo é criar uma base de treino e de teste, em nosso caso, fizemos um split com 70% de treino e 30% de teste randomizados, essa base é a mesma utilizada pelo na execução do PCA (x,y).

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics,tree
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)
```

Figura 24

O próximo passo é executar o classificador, que através da função fit(x_train,y_train) cria todo o modelo da árvore de decisão e a função predict(X_test) gera o resultado de predição classificando a base de treino.

Podemos verificar que a acurácia é bem alta, chegando a 92%, isso significa que o modelo gerado acerta bastante, mas não necessariamente representa o modelo “perfeito”, ao verificar a profundidade máxima da árvore gerada encontramos o valor 30 e o número de nós da árvore é 3739, isso pode representar um *overfitting* no

modelo, o que pode reduzir o poder de previsão quando uma nova base de teste for aplicado ao modelo. Isso é mostrado na Figura 25.

```
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
```

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9269943919567524
```

Profundidade e Número de nós da árvore

```
print(clf.tree_.max_depth, clf.tree_.node_count)
```

```
30 3739
```

Figura 25

A matriz de confusão apresentada na Figura 26 é um método comum para verificar se o modelo acertou “bem” apresentando a frequência de classificação para cada classe, é possível perceber que são apresentados muitos falsos positivos e negativos.

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

```
array([[  302,   290,   143,    71],
       [  147, 17891,    25,     3],
       [  188,   184,   101,    95],
       [  141,    16,   142,    54]])
```

Figura 26

Finalmente, na figura 26, apresentamos também o relatório de classificação com a precisão, revocação e *f1-score* para as classes, o f1-score apresenta um balanço entre precisão e revocação, observamos que eventos de *NoneAlert* tem *f1-score* alto de 98% pois a precisão e revocação foram altos, em segundo lugar os eventos *Critical* com 38% que mostra que o modelo ainda erra bastante para encontrar os verdadeiros eventos de logs de erro.

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

    Critical       0.39      0.37      0.38       806
   NoneAlert       0.97      0.99      0.98     18066
      Normal       0.25      0.18      0.21       568
     Warning       0.24      0.15      0.19       353

    accuracy                           0.93     19793
   macro avg       0.46      0.42      0.44     19793
weighted avg       0.92      0.93      0.92     19793
```

Figura 27

6. Apresentação dos Resultados

Na Figura 28, usando o modelo de Workflow Canvas proposto por Vasandani, é exibido de forma sucinta as etapas que percorremos para criar e executar abordagem de nosso trabalho, desde a definição de quais seriam as fontes de dados, assim como o tratamento necessário para adequá-los e normalizá-los de tal forma ser possível efetuar a nossa estratégia de como estudá-los, para, por fim, identificarmos ou não a existência de forte correlação entre os indicativos de temperaturas elevadas e os registros de problemas críticos de hardware nos servidores, dentro de um mesmo intervalo temporal.

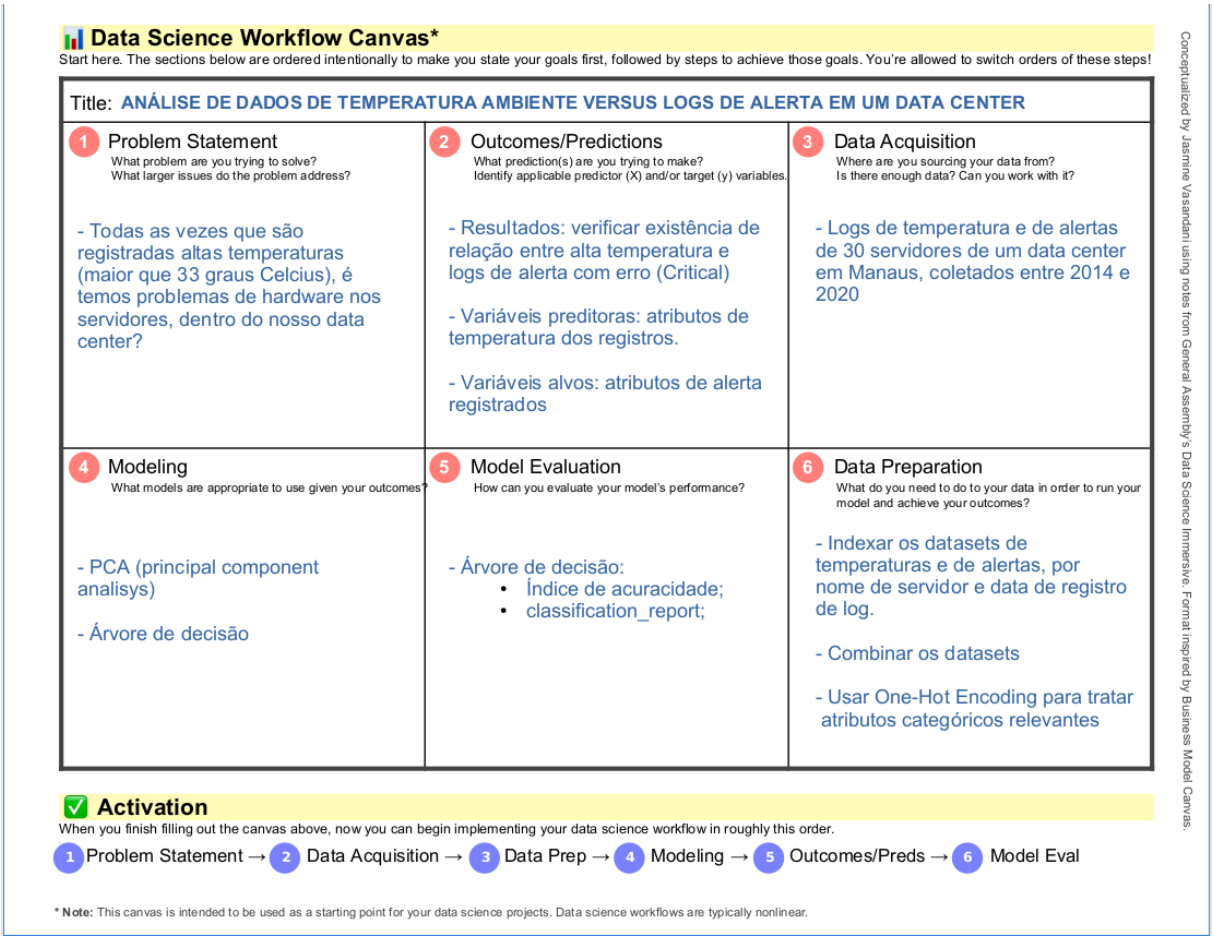

Figura 28

E, assim como ilustramos através dos nossos testes descritos com detalhes no tópico anterior (**5. Criação de Modelos de Machine Learning**), não conseguimos estabelecer que a todas as vezes que são registradas altas temperaturas (maior que 33 graus Celsius) temos problemas de hardware nos servidores, dentro do nosso data center.

Apesar disso e, portanto, o que extraímos de nosso estudo foram outras interessantes considerações, sob o ponto de vista de alta disponibilidade dos servidores:

- Dentro do data center, a sistemática hoje adotada para registrar alertas de alta temperatura não está evidenciando estes momentos, pois foram observados registros de temperatura acima de 33 graus Celsius e nenhum registro de alerta, dentro do mesmo dado de tempo. Isto implica que o monitoramento dos equipamentos pode estar recebendo a falsa informação de normalidade do ambiente.

- Existe a possibilidade de que os problemas de hardware só se tornem perceptíveis, posteriormente ao registro de elevação de temperatura.

- Como os sensores que disparam os logs tanto de temperatura, quanto de alertas são parte interna do mesmo dispositivo, o qual está instalado dentro de cada servidor, um fator que devemos considerar em um próximo estudo muito mais aprofundado, abordando outros aspectos, como por exemplo, o quanto um servidor está sendo sensível à temperatura em relação ao outro, ainda que todos tenham as mesmas características físicas, estejam localizados dentro de uma mesma sala e recebendo a mesma refrigeração.

7. Links

- **Repositório dos dados, scripts e documentos deste trabalho:** https://github.com/RomulosDS/TCC/tree/master
- Vídeo apresentação do trabalho – Romulos Machado - https://www.youtube.com/watch?v=-okatdwrq6o
  Vídeo apresentação do trabalho – Phillip Furtado – https://youtu.be/XBUdilyXoZw
- Material do curso Técnicas Estatísticas de Predição: Teoria e Aplicação da Pós em Ciência e Big Data – PUC Minas - https://pucminas.instructure.com/courses/144/wiki.
- Guia de API do Scikit Learn - https://scikit-learn.org/stable/user_guide.html