Modelo de predição de manutenção corretiva de ativos baseada em KNN e Naive Bayes

Amanda Aparecida Machado Goulart
Universidade Federal de São Paulo
Instituto de Ciência e tecnologia
Interdiciplinar em Ciência e tecnologia
São José dos Campos - São Paulo
Email: amanda.goulart@unifesp.br

***Abstract*—Este trabalho tem como objetivo realizar uma comparação entre os algortimos de aprendizado supervizionado K-Nearest Neighbors(KNN) e Naive Bayes. Com base nesses algoritmos será utilizado para fazer previsões de que se sera necessário realizar uma manutenção corretiva ou não, com base nos valores coletados dos sensores de 1(um) ativo e o histórico de manutenção do mesmo.**

1. Introdução

A problemática levantada neste trabalho é sobre ativos, ou seja tudo que tem valor para a empresa, porém no vocabulário da manutenção ele corresponde aos maquinários. Como qualquer coisa eles tem a tendência a deteriorar, e com o decorrer do tempo necessitar de manutenção.

Existe uma divisão entre manutenção programada e manutenção não programada. Para a manutenção programada temos os seguintes tipos, preventiva que são feitas de forma períodica e visa evitar falhas, e preditiva que usa dados gerados pelo proprio ativo para agendar a manutenção.

Mas iremos nos atentar a manutenção não programada,que correspode a manutenção corretiva, que acontece quando o ativo repentivamente apresenta uma falha e é necessário o reparo, esse tipo de caso é extremamente custoso, pois intenrrompe a produção e prejudica os prazos, diante disso o objetivo é prever e minimizar este tipo de manutenção.

Para isso será utilizado aprendizado de máquina supervizionado, com os dados levantados do dia 01/08/2020 a 30/09/2020. Serão dados levantados sobre os sensores do ativo em questão e os dias que ocorreu uma manutenção corretiva. Esses dados serão aplicados nos algoritmos de Naive Bayes e KNN.

1.1. Desenvolvimento

Para cumprir o objetivo, primeiramente será feito um tratamento de dados, que foi dividido em duas partes, uma utilizando o Google Sheets e outra parte utlizando o próprio python.

Após isso, foi feita a aplicação dos algoritmos Naive Bayes e KNN e verificada a sua acurácia.

Sobre o dataset, temos a príncipio dois arquivos .csv, sendo eles ”Device 1 - manutention.csv”, que possui a data da manutenção e o tipo de menutenção que foi realizada, de acordo com as quais já foi citadas anteriormente e ”Device 1 - sensorData.csv”, que possui o dia e a hora que foi coletada o dado daquele sensor, e 3(três) tipos de sensores, caracterizados como Sensor A, Sensor B e Sensor C, que são respectivamente sensor de temperatura(coletado em ºC), sensor de umidade(coletado em g/m³) e sensor de vibração(coletado em m/s²). Com o tratamento de dados, gera o ”device1.csv” que esse realmente é utilizado para predição.

Uma obervação é que busquei o equipamento que tivesse tipo o máximo de manutenções corretivas, para que fosse possível treinar e apresentar resultados satisfatórios.

1.1.1. Tratamento de dados - Parte 1. A primeira parte do tratamento de dados foi feita com o Google Sheets, pois o conhecimento para manipular um arquivo .csv é maior do que o python para o que foi realizado.

No arquivo ”Device 1 - sensorData.csv” foi realizada a média dos dados coletados de todos os dias que possui no dataset, e foi incluido no ”device1.csv”, pois para fazer as comparações de manutenção, foi necessário enxugar o comportamento dos sensores que são coletados várias vezes por hora, e restringir a uma medida do dia, sendo assim foi utlilizada uma media aritmética simples.

$$mediaSensorDia = \frac{\sum dadosSensor}{\sum qtdDados}$$

Já no dataset ”Device 1 - manutention.csv”, foi realizada a lógica para que retornasse no no ”device1.csv” na coluna ”isCorrective” 1(um) ou 0(zero), caso tivesse realizada a manutenção corretiva naquele dia. E assim descartando os outros tipos de manutenção que não se encaixam no propósito atual.

Resultando neste dataset, esta mostrado na figura somente os 5(cinco) primeiros dados:

	date	median sensor A	median sensor B	median sensor C	isCorrective
0	01/08/2020	44	19	2	0
1	02/08/2020	50	23	4	1
2	03/08/2020	46	20	3	0
3	04/08/2020	45	20	3	0
4	05/08/2020	46	21	3	0

Figure 1. device1.csv

1.1.2. Tratamento de dados - Parte 2. O tratamento de dados nesta parte foi mais simples.

```
device =
    device.drop(columns=['typeManutention'])
```

Primeiramente retirei a coluna que possuia a marcação dos tipo de manutenção, isso não será mais necessário a partir deste ponto.

```
array = device.values
X = array[:,1:4]
Y = array[:,4]
```

Neste ponto separei os dados que iriam ser usados como parâmetros de classificação e a classe, lembrando que 1 é para manutenção corretiva e 0 para nenhuma manutenção.

```
validation_size = 0.3
seed = 7
X_train, X_validation, Y_train, Y_validation
    = model_selection.train_test_split(X, Y,
    test_size=validation_size,
    random_state=seed)
```

Neste procedimento separei 70% para treino e 30% para teste. Deixando assim o dataset totalmente preparado para ser aplicado nos algoritmos.

1.1.3. Naive Bayes. O algoritimo de Naive Bayes, é sustentado na probabilidade de determinado evento ocorrer, no caso precisar ou não de uma manutenção corretiva. Optei por ele por não necessitar de tantas obervações para obter uma boa acurácia.

Ele é definido pelo Teorema de Bayes que considera várias feaures caso necessário.

$$P(y|x1,...,xn) = \frac{P(y)P(x1,...xn|y)}{P(x1,...,xn)}$$

Utilizei também que o Naive Bayes baseado em Gaussian, utilizando a fórmula:

$$P(x1|y) = \frac{1}{\sqrt{2\pi\sigma_2^y}} exp(-\frac{(xi-\mu)^2}{2\sigma_2^y})$$

Com base nesse cálculo é possivel ver onde os dados se encontram na districuição normal.

Partindo disso, foi realizado os seguintes procedimentos:

```
clf = GaussianNB()

scoring = 'accuracy'
clf.fit(X_train, Y_train.astype(int))

pred_clf = clf.predict(X_validation)

prob_pos_clf =
    clf.predict_proba(X_validation)[:, 1]
```

Foi criada uma váriavel para o cálculo da fórmula de Gaussian, e partir disso foi treinado os dados na função de fit(), e a partir disso foi feita a predição com a função predict(), também utilizei o predict_proba() para retornar as probalididades e ter mais insigths sobre o resultado.

```
pred_clf_df =
    pd.DataFrame(pred_clf.reshape(19,1))
pred_clf_prob_df =
    pd.DataFrame(prob_pos_clf.reshape(19,1))
Y_validation_df =
    pd.DataFrame(Y_validation.reshape(19,1))

pred_clf_df.rename(columns={0:'Prediction'},
    inplace=True)
pred_clf_prob_df.rename(columns={0:'Probably
    prediction'}, inplace=True)
Y_validation_df.rename(columns={0:'Should
    be'}, inplace=True)

pred_outcome = pd.concat([pred_clf_prob_df,
    pred_clf_df, Y_validation_df], axis=1)
```

Neste trecho foi feita uma manipulação para que formasse um Dataframe, que tivesse as colunas de Previsão do Naive Bayes, a porcentagem das previsões e qual o resultado esperado.

```
accuracy_score(Y_validation.astype(int),
    pred_clf)
```

E finalmente foi verificada a acurácia do algoritmo.

1.1.4. K-Nearest Neighbors. O KKN é também outro algotimo da família de aprendizado supervizionado, ele é baseado na distância entre os vizinhos. Para isso pode ser utilizada várias medidas de distância. Neste caso, foi utilizada a distância de Minkowski, dada pela fôrmula de:

$$D(xi,xj) = (\sum_{l-1}^{d} |xil - xjl|^{\frac{1}{p}})^p$$

Para a medida de número de vizinhos, foi utilizado o número 5(cinco). Sendo assim, foi praticado o seguinte algoritmo:

```
scaler = StandardScaler()
scaler.fit(X_train)

x_train = scaler.transform(X_train)
x_test = scaler.transform(X_validation)
```

Neste caso, antes apliquei um pré-processamento, utilizando o StardartScaler(), para agir sobre as colunas, porém

seu método é diferente uma vez que este subtrai do valor em questão a média da coluna e divide o resultado pelo desvio padrão. No final temos uma distribuição de dados com desvio padrão igual a 1 e variância de 1 também.

```
classifier =
    KNeighborsClassifier(n_neighbors = 5)
classifier.fit(X_train, Y_train.astype(int))
```

Agora sim foi setado o KNN, com 5 vizinhos e e feito o treinamendo com a função fit().

```
y_pred = classifier.predict(x_test)
```

E partindo do traino, feita a previsão.

```
pred_clf_df =
    pd.DataFrame(y_pred.reshape(19,1))
Y_validation_df =
    pd.DataFrame(Y_validation.reshape(19,1))

pred_clf_df.rename(columns={0:'Prediction'},
    inplace=True)
Y_validation_df.rename(columns={0:'Should
    be'}, inplace=True)

pred_outcome = pd.concat([pred_clf_df,
    Y_validation_df], axis=1)
```

Novamente fiz a união dos Dataframes para fazer uma comparação visual.

```
accuracy_score(Y_validation.astype(int),
    y_pred)
```

E assim fazer o cálculo da acurácia.

2. Conclusão

Diante destes dois algoritmos, antecipo que os resultados foram idênticos, tanto de previsão, tanto de acurácia.

Mas primeiramente vamos analisar os resultados individualmente, a príncipio os resultados de Naive Bayes.

Para os resultados de testes tive a seguinte apresentação de resulatados, conforme compilados na junção dos dataframes citados anteriormente no desenvolvimento.

É possível perceber que teve uma váriação das porcentagens, porém ficou extremamente abaixo de 0, fazendo com que a classificação, retornasse todos os elementos em 0, indicando que não é necessária a manutenção. Mesmo em alguns casos que realemente foi feita a manutenção. Indicando que não conseguiu realizar um aprendizado completo. Mesmo aprensentando um nível de acurácia relativamente bom,que foi em 89.47%

	Probably prediction	Prediction	Should be
0	0.006254	0	0
1	0.028938	0	0
2	0.028938	0	0
3	0.119979	0	0
4	0.054308	0	0
5	0.045513	0	0
6	0.000244	0	1
7	0.050809	0	0
8	0.030980	0	0
9	0.000638	0	0
10	0.107712	0	1
11	0.005404	0	0
12	0.000377	0	0
13	0.000562	0	0
14	0.006895	0	0
15	0.001991	0	0
16	0.000199	0	0
17	0.019103	0	0
18	0.001336	0	0

Figure 2. Resultados Naive Bayes

Acredito que apesar desse resultado, é possivel analisar as porcentagens, descritas na coluna ”Probably prediction”, para se realizar a tomada de alguma decisão.

Para os resultados de KNN, a unisão dos datasets foi mais binária, não incluindo uma probabilidade.ficando somente a coluna de ”Predition” e ”Shold be”. Novamente acaba retornando o mesmo resultado de Naive Bayes, que mantém um viés direcionado para não ocorrer nenhuma manutenção.

Mais uma vez também a acurácia manteve em 89.47%.

Os resultados podem ser analisados na figura a seguir:

	Prediction	Should be
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	1
7	0	0
8	0	0
9	0	0
10	0	1
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0

Figure 3. Resultados KNN

Acredito eu que esta ocorrência de predizer os mesmos resultados, e não classificar nenhum para manutenção se deve pelo fato que apesar de reunir dados de 2(dois) meses, para esta previsão seria necessária um maior número de dados coletados com manutenção corretiva.
Porém este era o máximo de dados que já foi levantado pela empresa em relação aos sensores, por conta de ser uma nova aplicação, apesar de ter dados mais antigos de manutenção, para correlacionar com o estado atual não houve dados suficientes.

Também o fato de aplicar a média ponderada, fez com que se pegasse uma quatidade de dados e ficasse reduzida, além disso pode ter algum viés, por causa da média ponderada, visto que pode ter vários outliers.

Diante disso, fica como observação, para melhor obtenção de resultados, um maior acumulo de dados, provavelmente de 1(um) ano ou até mais, no entanto para isso irá ocorrer um maior custo de processamento. Talvez seja necessário o uso de outras ferramentas de Big Data, como o Hadoop.

Além disso, deve se ter uma maior atenção no tratamento de dados, verificando de há outliers, por meio de varificação do desvío padrão, e caso sim, aplicando a técnica de ”Interquartile Rule”, onde se encontra os quartis e elimina os extremos.

Apesar de apresentarem os mesmos resultados, acabo preferindo o Naive Bayes, por conta de conseguir estimar as probabilidades de cada evento, acho que este se adequa bem mais ao proposíto da solução. Além de ser menos custoso e permitir poucas entradas de dados.

2.1. Referências

ENGEMAN. **Tipos de Manutenção.** Disponível em: https://blog.engeman.com.br/tipos-de-manutencao/. Acesso em: 12 out. 2020.

OELHO, Caique. **Pré processamento de dados em Machine Learning.** Disponível em: https://medium.com/@caiquecoelho/um-guia-completo-para-o-pr/%C3/%A9-processamento-de-dados-em-machine-learning-f860fbadabe1. Acesso em: 12 out. 2020.

PURPLEMATH. **Interquartile Ranges Outliers.** Disponível em: https://www.purplemath.com/modules/boxwhisk3.htm. Acesso em: 12 out. 2020.

HOLT, Brian et Al. **Sklearn**. Disponível em: https://scikit-learn.org/stable/index.html. Acesso em: 12 out. 2020.